Development of the conceptual schema of the *osTicket* system by applying TDCM

Research Report

Albert Tort Pugibet
atort@essi.upc.edu

April 2011
# Contents

1. Introduction .............................................................................................................................. 1  
   1.1. The osTicket system ........................................................................................................... 1  
   1.2. The Conceptual Schema Under Development ................................................................. 2  

2. Testing strategy ........................................................................................................................ 4  
   2.1. Strategy overview ............................................................................................................. 4  

3. Source artifacts ........................................................................................................................ 6  
   3.1. Configuration and basics ................................................................................................. 6  
   3.2. Tickets management and tracking .................................................................................... 9  
   3.3. Domain rules .................................................................................................................... 12  

4. User stories .................................................................................................................................. 13  
   4.1. Configuration and basics ................................................................................................. 13  
   4.2. Tickets management and tracking .................................................................................... 20  

5. TDCM application to the case study ......................................................................................... 46  
   Iteration 1 ................................................................................................................................. 47  
   Iteration 2 ................................................................................................................................. 49  
   Iteration 3 .................................................................................................................................. 52  
   Iteration 4 .................................................................................................................................. 55  
   Iteration 5 .................................................................................................................................. 56  
   Iteration 6 .................................................................................................................................. 58  
   Iteration 7 .................................................................................................................................. 60  
   Iteration 8 .................................................................................................................................. 61  
   Iteration 9 .................................................................................................................................. 62  
   Iteration 10 ................................................................................................................................. 65  
   Iteration 11 ................................................................................................................................. 67  
   Iteration 12 ................................................................................................................................. 68  
   Iteration 13 ................................................................................................................................. 70  
   Iteration 14 ................................................................................................................................. 72  
   Configuration & basics conceptual schema .............................................................................. 75  
   About the tickets management and tracking iterations ............................................................ 76  

---

**UNIVERSITAT POLITÈCNICA DE CATALUNYA**  
**BARCELONATECH**  
**PhD in Computing**
Mock schema elements ................................................................. 78
Iteration 15 .................................................................................. 79
Iteration 16 .................................................................................. 85
Iteration 17 .................................................................................. 89
Iteration 18 .................................................................................. 90
Iteration 19 .................................................................................. 91
Iteration 20 .................................................................................. 93
Iteration 21 .................................................................................. 95
Iteration 22 .................................................................................. 96
Iteration 23 .................................................................................. 97
Iteration 24 ............................................................................... 101
Iteration 25 ............................................................................... 102
Iteration 26 ............................................................................... 103
Iteration 27 ............................................................................... 105
Iteration 28 ............................................................................... 106
Iteration 29 ............................................................................... 107
Iteration 30 ............................................................................... 109
Iteration 31 ............................................................................... 110
Iteration 32 ............................................................................... 112
Iteration 33 ............................................................................... 113
Iteration 34 ............................................................................... 114
Iteration 35 ............................................................................... 116
Iteration 36 ............................................................................... 117
Iteration 37 ............................................................................... 121
Iteration 38 ............................................................................... 123
Iteration 39 ............................................................................... 124
Iteration 40 ............................................................................... 125
Iteration 41 ............................................................................... 126
Iteration 42 ............................................................................... 130
Iteration 43 ............................................................................... 131
Iteration 44 ............................................................................... 133
DEVELOPMENT OF THE CONCEPTUAL SCHEMA OF THE osTICKET SYSTEM BY APPLYING TDCM

Iteration 76 ............................................................................................................................ 175
Iteration 77 ............................................................................................................................ 176
Iteration 78 ............................................................................................................................ 176
Iteration 79 ............................................................................................................................ 179
Iteration 80 ............................................................................................................................ 180
Iteration 81 ............................................................................................................................ 181
Iteration 82 ............................................................................................................................ 182
Iteration 83 ............................................................................................................................ 183
Iteration 84 ............................................................................................................................ 184
Iteration 85 ............................................................................................................................ 184
Iteration 86 ............................................................................................................................ 187
Iteration 87 ............................................................................................................................ 188
Iteration 88 ............................................................................................................................ 188
Iteration 89 ............................................................................................................................ 189
Iteration 90 ............................................................................................................................ 191
Iteration 91 ............................................................................................................................ 191
Iteration 92 ............................................................................................................................ 192
Iteration 93 ............................................................................................................................ 193
Iteration 94 ............................................................................................................................ 194
Iteration 95 ............................................................................................................................ 195
Iteration 96 ............................................................................................................................ 196
Iteration 97 ............................................................................................................................ 198
Iteration 98 ............................................................................................................................ 199
Iteration 99 ............................................................................................................................ 201
Iteration 100 .......................................................................................................................... 202
Resultant conceptual schema .............................................................................................. 206

6. Case study analysis ........................................................................................................... 225
  6.1. The resultant conceptual schema .................................................................................. 225
      Quality properties of the resultant conceptual schema .................................................. 226
  6.2. The test set .................................................................................................................. 227
  6.3. TDCM iterations ......................................................................................................... 228
Errors/failures categorization ........................................................................................................ 228

Errors and failures that drive conceptual modeling in the case study .................................. 229

6.4. Iterations analysis.................................................................................................................. 232

Test cases specification .............................................................................................................. 233

TDCM iterations productivity .................................................................................................... 235

Errors and failures evolution ..................................................................................................... 237

Iterations summary ................................................................................................................... 238

7. Conclusions ............................................................................................................................ 239

8. References ............................................................................................................................... 241

Appendix A. USE specification of the resultant schema ........................................................... 242

Appendix B. Methods specification .......................................................................................... 265

Appendix C. Test set ................................................................................................................... 278
1. Introduction

Conceptual schemas of information systems can be tested [6]. This essentially means that there is a testing language, in which the conceptual modeler writes programs that test the conceptual schema, and a testing environment in which test programs are executed.

TDCM is an iterative method aimed to drive the elicitation and the definition of the conceptual schema of an information system. TDCM uses test cases to drive the conceptual modeling activity. In TDCM, conceptual schemas are incrementally defined and continuously validated.

This document reports a case study application of Test-Driven Conceptual Modeling (TDCM) in the development of the conceptual schema of a well-known, open-source and widely-used online support system called osTicket [3]. In contrast with the case study application of TDCM in the development of the conceptual schema of a bowling game system [5], in this report TDCM is applied in order to perform reverse engineering of an existing system.

Chikofsky and Cross [1] defines reverse engineering as follows: “Reverse engineering is the process of analyzing a subject system to identify the system’s components and their interrelationships and create representations of the system in another form or at a higher level of abstraction”. “The purpose of reverse engineering is to understand a software system in order to facilitate enhancement, correction, documentation and redesign”.

In this report we will use the testing language called CSTL (Conceptual Schema Testing Language) [4,6] in order to specify conceptual test cases. A CSTL processor prototype will be used to execute the test cases.

In the following, we briefly present the system used as a case study in this report (Section 1.1) and the expected properties and assumptions of the Conceptual Schema Under Development (CSUD) (Section 1.2).

1.1. The osTicket system

Ticket support systems allow customers to create and keep track of support requests as tickets and allow staff members to organize, manage and respond them. A ticket contains all the information related to a customer support request.
In particular, *osTicket* is an open source support ticket system which “is designed to improve customer support efficiency by providing staff with tools they need to deliver fast, effective and measurable support” [3].

The *osTicket* system allows users to create new tickets online or by email. It also allows the staff to create tickets in behalf of customers. The system provides automatic notices by email when a ticket is created, responded, etc. Moreover, the system allows the staff members to add internal notes to tickets. Configurable help topics, assignment of staff responsibility for each ticket, due dates, departments, priorities, etc. are also offered. Finally, customers may access the system to keep track of the status of their tickets and reply them.

![osTicket's staff panel screenshot](image)

**Fig. 1. osTicket’s staff panel screenshot**

### 1.2. The Conceptual Schema Under Development

In this report, we present an experimental application of Test-Driven Conceptual Modeling (TDCM) for the reverse engineering development of the UML/OCL conceptual schema of the *osTicket* system (version 1.6.0).

You may review the main concepts and notation used to define the conceptual schema under development in [6].

In the following, we comment some assumptions about the desired properties of the Conceptual Schema Under Development (CSUD) in this case study. We defined these assumptions and properties for two different (but interrelated) areas of knowledge which constitute the *osTicket* system: 1) The basics and the configuration of the system and 2) The management and tracking of tickets.
Basics and configuration of the system

The administrators set the basics and the configuration of the system. Configuration includes knowledge about the system settings, the staff and administration users, the departments, etc.

The configuration knowledge is only updated according to changes in the customer support policy or due to changes in the organization of the company.

All configuration concepts and the relationships between them are assumed to be set by means of the occurrence of implicit structural events (entity insertion or update and relationship insertion or update) associated to each entity type or relationship type. We also assume that the occurrence of an implicit structural event is only valid if the resultant Information Base (IB) state is consistent according to the static constraints defined in the schema.

Therefore, we only focus on the structural conceptual schema for the system basics and configuration. No domain events are defined for the configuration because implicit structural events are assumed to exist.

Similarly, no predefined queries are specified because all base and derived configuration knowledge is assumed to be visible by the administrators.

Tickets management and tracking

Customers may create, comment and keep track of their tickets online by using the osTicket system. Staff members can also create tickets on behalf of customers. Staff members may manage tickets, assign them to other staff members or departments, post responses and internal notes, close them, reopen them, etc.

The knowledge about tickets management and tracking must be consistent with the structural conceptual schema of the system configuration part defined above.

The state of the domain may not change arbitrarily. Changes in the domain about tickets management are not as simple as implicit structural events for each schema element and not all changes are valid. Therefore, for the purpose of defining the general knowledge about tickets management and tracking, the conceptual schema fragments of this area of knowledge is expected to include both the structural conceptual schema and the behavioral schema.

The behavioral schema consists of the definition of the domain events (that define the valid changes of the information base) and significant predefined queries which specify commonly queried knowledge. The specification of the domain events ensures that all the changes in the Information Base are valid and they don’t lead to inconsistent states. Moreover, all knowledge defined in the conceptual schema is assumed to be visible for staff members and customers (according to the defined visibility privileges).
2. Testing strategy

The application of TDCM should be supported by a previously defined testing strategy. Defining a strategy comprises the design of a representative set of test cases, which are the input of the TDCM application. The testing strategy should also include criteria to determine the source of the test cases and its order of processing when applying TDCM.

2.1. Strategy overview

Figure 2 summarizes the strategy overview for the development of the conceptual schema of the osTicket system. In the following, we explain the strategy for each area of knowledge of the CSUD. The strategy is aimed to determine 1) the source artifacts for defining the user stories which will be formally specified as test cases (which are the input for the TDCM application) and 2) the order of testing processing (aimed to minimize the dependencies that may lead to test cases rewriting).
**Basics and configuration of the system**

The source artifacts are:

- An informal list (written in natural language) of requirements related to the basics and configuration of the system. These informal requirements are elicited from the public documentation of the *osTicket* system and by using it.
- An estimated dependency graph between the informal requirements.

Administration user stories will be written in order to cover representative configurations of the informal requirements. These stories will be formally specified as test cases.

The order of processing of test cases during the TDCM application will be determined by the following rules:

- In each TDCM iteration, the processed test case should have the minimum number of dependencies with non processed test cases. It allows minimizing the necessity of rewriting test cases and improves the efficiency of TDCM.

**Tickets management and tracking**

The source artifacts are:

- The specification of the use cases that informally describe the behavior of the system.
- The specification of the main domain rules of the system.
- An estimated dependency graph between the use cases and the domain rules.

Staff and customer user stories will be written in order to cover representative uses of the system (specified as use cases). User stories may be applied to different configurations of the system. These stories will be formally specified as test cases.

The processing order of test cases during the TDCM application will be determined by the following rules:

- In each TDCM iteration, the processed test case should have the minimum number of dependencies with non processed test cases. It allows minimizing the necessity of rewriting test cases and improves the efficiency of TDCM.
3. Source artifacts

In the following we describe the source artifacts for applying TDCM according to the testing strategy stated in Section 2.

3.1. Configuration and basics

Informal specification of features

(*) Mandatory properties

- General settings:
  - Helpdesk status: Online (customer functionalities active) / Offline (only staff functionalities are allowed) (*)
  - Helpdesk url: The root path for customers (*)
  - Helpdesk name: The name of the helpdesk (*)
  - Default email template: The default email template used when a mail is sent (*)

- Ticket settings
  - Ticket IDs mode: Sequential/Random (*)
  - Default priority: Low/Normal/High/Emergency (*)
  - Customers can change ticket priority: Yes/No (*)
  - Maximum open tickets per mail: Natural or unlimited (*)
  - Use email priority when available: Yes/No (*)
  - Ticket grace period: Hours before ticket is marked overdue (*)
  - Reopened tickets are assigned to the last respondent: Yes/No (*)

- Email settings
  - Default system email: The email address used by default for the system for sending customer notices (*)
  - Default staff alerts email: The email used by default for sending notices to staff. (*)
  - Administration email: The email for administration alerts. (*)

- Auto responses by mail to the customer that owns a ticket
  - Auto respond when a new ticket is created by a customer: Yes/No (*)
  - Auto respond when a new ticket is created by a staff member: Yes/No (*)
  - Auto respond when a new message has been appended to a ticket: Yes/no (*)
  - Auto respond when a customer violates the maximum of open tickets: Yes/no (*)
- **Alerts & notices by mail to the staff members**
  o When a new ticket is created is alerted by mail to the staff: Yes/No (*) (alerted staff may also be selected among the following options: Administration email, Department manager, department members.
  o When a new message is appended to a ticket, the message is alerted by mail to the staff: Yes/No (*) (alerted staff may also be selected among the following options: Last respondent, assigned staff, department manager.
  o When a new internal note is added to a ticket, the new note is alerted by mail to the staff: Yes/No (*) (alerted staff may also be selected among the following options: Last respondent, assigned staff, department manager.
  o When a new ticket is overdue, this fact is alerted by mail to the staff: Yes/No (*) (alerted staff may also be selected among the following options: Assigned staff, department manager, department members.

- **Email accounts**
  o Email accounts can be created/edit or delete with the following properties:
    - Email address (*)
    - Email from name (*)
    - Default new ticket priority for tickets created via this email (*)
    - Default new ticket department (*)
    - Auto responses status: enabled/disabled (overwrites department setting) (*)
  o Email addresses can be added/removed from a banned list (no tickets are allowed for banned emails)

- **Email templates**
  o Email templates can be created/edit or delete with the following properties:
    - Name of the template (*)
    - Internal notes
  o Each email template has a subject and a message for each of the following email kinds:
    - New Ticket Autoresponse (*)
    - New Message Autoresponse (*)
    - New Ticket Notice (*)
    - Over Ticket limit Notice (*)
    - Ticket Response/Reply (*)
    - New Ticket Alert to Staff (*)
    - New Message Alert to Staff (*)
    - New Internal Note Alert to Staff (*)
    - Ticket Assigned Alert/Notice to Staff (*)
    - Overdue/Stale Ticket Alert/Notice to Staff (*)

- **Staff groups**
  o Staff groups may be created/edit/delete with the following properties:
    - Group name (*)
    - Group status: active/disabled (*)
    - Departments access: A set of departments which are additionally visible by the staff members of the group
    - Privileges (Yes/no) (*) *Restrictions are non-applicable to administration staff members*
      - Can create tickets on behalf of a customer
      - Can edit tickets *(restriction is non-applicable to department managers)*
      - Can transfer tickets
      - Can delete tickets
      - Can ban emails

- **Staff members**
  o Staff members may be created/edit/delete with the following properties:
    - Username (*)
    - Department (*)
- **User group** (*
- **First Name** (*
- **Last Name** (*
- **Email address** (*
- **Office phone**
- **Phone extension**
- **Mobile phone**
- **Signature**
- **Password** (*
- **Account status**: active/locked (*
- **isAdministrator**: yes/no (*
- **Vacation mode**: no tickets assigned and no alerts are received): yes/no (*)

**Departments**

- Departments may be created/edit/delete with the following properties:
  - **Department name** (*
  - **Department email** (used for outgoing emails) (*)
  - **Department manager** (one of the staff members)
  - **Department type**: public/private (*) (when a ticket is in a private department, the name of the department is not shown to the ticket’s customer)
  - **Email template** (used for outgoing emails and notices to user and staff.) (*)
  - **Auto-response when new ticket is created**: yes/no (redefines general settings) (*)
  - **Auto-response when new message is added**: yes/no (redefines general settings) (*)
  - **Auto-response email address** (redefines general settings) (*)

**Help topics**

- Help topics may be created/edit/delete with the following properties:
  - **Help topic name** (*
  - **Topic status**: active/disabled (*)
  - **Auto response**: active/disabled. Overwrites department settings. (*)
  - **New Ticket Priority**: low, normal, high, emergency (*)
  - **New ticket Department** (*)

**Estimated dependency graph**

[Diagram showing dependencies between departments, help topics, ticket settings, and other elements]
3.2. Tickets management and tracking

**Customer use cases**

<table>
<thead>
<tr>
<th>Use case: Create a new online ticket</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Identifier:</strong> NTO.</td>
</tr>
<tr>
<td><strong>Preconditions:</strong> Online mode is enabled.</td>
</tr>
<tr>
<td><strong>Trigger:</strong> The customer wants to open a new request case as a ticket</td>
</tr>
<tr>
<td><strong>Main Success Scenario:</strong></td>
</tr>
<tr>
<td>1. The customer provides the details for creating a new ticket. [⇒NewTicketOnline]</td>
</tr>
<tr>
<td>2. The system validates that the data is correct</td>
</tr>
<tr>
<td>3. The system creates the new ticket.</td>
</tr>
<tr>
<td><strong>Extensions:</strong></td>
</tr>
<tr>
<td>2a. The ticket data is invalid.</td>
</tr>
<tr>
<td>2a1. The system asks the customer to correct the data.</td>
</tr>
<tr>
<td>2a2. The use case continues at step 1.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Use case: Create a new ticket by email</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Identifier:</strong> NTE.</td>
</tr>
<tr>
<td><strong>Preconditions:</strong> Online mode is enabled.</td>
</tr>
<tr>
<td><strong>Trigger:</strong> The customer sends an email to a ticket support</td>
</tr>
<tr>
<td><strong>Main Success Scenario:</strong></td>
</tr>
<tr>
<td>1. The system creates the new ticket with the information of the email [⇒NewTicketByEmail]</td>
</tr>
<tr>
<td><strong>Extensions:</strong></td>
</tr>
<tr>
<td>1a. The addressee is not a valid incoming email account registered in the system.</td>
</tr>
<tr>
<td>1a1. The use case ends.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Use case: Check ticket status</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Identifier:</strong> CTS.</td>
</tr>
<tr>
<td><strong>Preconditions:</strong> Online mode is enabled.</td>
</tr>
<tr>
<td><strong>Trigger:</strong> The customer wants to see the details and the status of a ticket.</td>
</tr>
<tr>
<td><strong>Main Success Scenario:</strong></td>
</tr>
<tr>
<td>1. The customer provides his/her email and the ticket identifier to log in. [⇒DisplayTicketsAssociatedToEmail]</td>
</tr>
<tr>
<td>2. The system displays a list of all the tickets associated to the provided email.</td>
</tr>
<tr>
<td>3. The customer selects a ticket.</td>
</tr>
<tr>
<td>4. The system displays the information of the ticket.</td>
</tr>
<tr>
<td>The customer repeats steps 3 and 4 as needed.</td>
</tr>
<tr>
<td><strong>Extensions:</strong></td>
</tr>
<tr>
<td>2a. The access data provided by the customer is not valid.</td>
</tr>
<tr>
<td>2a1. The system informs the customer that the provided access data is invalid.</td>
</tr>
<tr>
<td>2a2. The use case ends.</td>
</tr>
<tr>
<td>2b. There is only one ticket associated to the provided email</td>
</tr>
<tr>
<td>2b1. The use case continues at step 4.</td>
</tr>
<tr>
<td>5a. The customer wants to respond the ticket</td>
</tr>
</tbody>
</table>
5a1. The customer writes his/her response
5a2. The system saves the response.
5a3. The use case continues at step 4.
2c, 3a, 4a. The customer wants to log out
   2c1, 3a1, 4a1. The use case ends.

**Staff use cases**

Use case: **Create a new offline ticket**

**Identifier:** NTOF.
**Preconditions:** The staff member is logged in and it is allowed to create new tickets.
**Trigger:** The staff member creates a ticket on behalf of a customer
**Main Success Scenario:**
1. The staff member provides the details for creating the new ticket.
   \[ \Rightarrow \text{NewTicketOffline} \]
2. The system validates that the data is correct
3. The system creates the new ticket
**Extensions:**
2a. The ticket data is invalid.
   2a1. The system asks the staff member to correct the data.
   2a2. The use case continues at step 1.

Use case: **Staff log in**

**Identifier:** SLI.
**Preconditions:** None.
**Trigger:** An Staff member wants to log in.
**Main Success Scenario:**
1. The staff member provides his/her log in data.
   \[ \Rightarrow \text{StaffLogIn} \]
2. The system performs the log in.
**Extensions:**
2a. The log in data is not correct
   2a1. The system informs the staff member that the log in data is invalid and asks him/her to enter valid data.
   2a2. The use case continues at step 1.

Use case: **View tickets by status**

**Identifier:** VTS.
**Preconditions:** The staff member is logged in.
**Trigger:** The staff member wants to view the list of tickets which are in a particular state.
**Main Success Scenario:**
1. The staff member selects the filter status (open, assigned to the staff member, closed or overdue).
2. The system displays a list of all the tickets which are currently in the specified filter status.
   The staff member repeats the following steps as necessary
3. The staff member selects a ticket
4. Manage ticket
Extensions:
3a. The staff member does not want to manage the ticket
   3a1. The use case ends.

Use case: **Staff log out**

**Identifier:** SLO.
**Preconditions:** The staff member is logged in.
**Trigger:** An Staff member wants to log out.
**Main Success Scenario:**
1. The staff member confirms that he/she want to log out.
   \[ \Rightarrow \text{StaffLogOut} \]
2. The system performs the log out.

Use case: **Manage ticket**

**Identifier:** MT.
**Preconditions:** The staff member is logged in. The ticket is visible by the staff member. The staff member is able to manage the ticket in each case.
**Trigger:** The staff member wants to manage a ticket.
**Main Success Scenario:**
1. The staff member selects an existing ticket.
2. The system displays all the information about the ticket.
   \*The staff member may perform the following steps as necessary to manage the ticket, depending on the status of the ticket and the allowed actions of the staff member.*
3. The staff member changes the priority of the ticket
   \[ \Rightarrow \text{ChangeTicketPriority} \]
4. The staff member marks the ticket as overdue (if it is not overdue and the staff member is administrator).
   \[ \Rightarrow \text{MarkTicketOverdue} \]
5. The staff member assigns the ticket to another staff member.
   \[ \Rightarrow \text{AssignTicket} \]
6. The staff member releases (unassigning) the ticket (if it is assigned).
   \[ \Rightarrow \text{ReleaseTicket} \]
7. The staff member edits the general information of the ticket.
   \[ \Rightarrow \text{EditTicket} \]
8. The staff member posts a reply.
   \[ \Rightarrow \text{PostTicketReply} \]
9. The staff member posts an internal note.
   \[ \Rightarrow \text{PostTicketInternalNote} \]
10. The staff member transfers the responsibility of the ticket to another department.
    \[ \Rightarrow \text{TransferDepartment} \]
11. The staff member closes the ticket (if it is open).
    \[ \Rightarrow \text{CloseTicket} \]
    \[ \Rightarrow \text{CloseTicketWithResponse} \]
12. The staff member reopens the ticket (if it is closed).
    \[ \Rightarrow \text{ReopenTicket} \]
    \[ \Rightarrow \text{ReopenTicketWithResponse} \]
13. The staff member bans the email and close the ticket (if it is open).
    \[ \Rightarrow \text{BanEmailAndCloseTicket} \]

Extensions:
3a-13a. The staff member deletes the ticket.
   3a1-13a1. The system deletes the ticket
   \[ \Rightarrow \text{DeleteTicket} \]
   3a2-13a2. The use case ends.
10a, 11a. The staff member wants to close the ticket on reply (if it is open) or to reopen it (if it is closed).
10a1, 11a1. The system changes the state of the ticket.
   [⇒ CloseTicket]
   [⇒ ReopenTicket]
10a2, 11a2. The use case continues.

**System use cases**

**Use case: CheckOverdueTickets**

**Identifier:** COT.

**Preconditions:** --

**Trigger:** The system checks if any ticket is overdue

**Main Success Scenario:**
1. The systems change the status of overdue tickets to “Overdue”
2. The system sends autoresponses and alerts to staff according to the configuration
   [⇒ CheckOverdueTickets]

**Estimated dependency graph**

In order to minimize rewriting of test cases, we will process the user stories according to its associated use case and by taking into account the following order:

1. NTO
2. NTE
3. CTS
4. SLI
5. SLO
6. NTOF
7. VTS
8. MT

**3.3. Domain rules**

**DR1.** A ticket is always open or close.
**DR2.** An open ticket may be assigned to a staff member or may be unassigned.
**DR3.** An open ticket may be overdue.
4. User stories

In the following, we specify the user stories that will be processed when applying TDCM. User stories are written according to the testing strategy defined in Section 2.

4.1. Configuration and basics

We define several story fragments related to the basics of the system configured by the administrators. Each story fragment defines a representative configuration for each of the features groups defined in Section 3.1. Testable stories are combinations of some story fragments which are expected to be valid in the system. Dependent story fragments are combined in order to define testable stories to be processed by TDCM.

**Story fragments**

**Story fragment #1: Email_template_default**
- An administrator creates the email template default
  - Name: “Default”
  - Internal notes: “Email templates by default”
  - Subject “X” and message “YYY” for each email kind:
    - New Ticket Autoresponse
    - New Message Autoresponse
    - New Ticket Notice
    - Over Ticket limit Notice
    - Ticket Response/Reply
    - New Ticket Alert to Staff
    - New Message Alert to Staff
    - New Internal Note Alert to Staff
    - Ticket Assigned Alert/Notice to Staff
    - Overdue/Stale Ticket Alert/Notice to Staff

**Story fragment #2a: General_settings_online**
Dependency: Email_template_default

Helpdesk status: Online (customer functionalities active)
Helpdesk url: [http://onlinesupport.com](http://onlinesupport.com)
Helpdesk name: Online customer support
Default email template: default
DEVELOPMENT OF THE CONCEPTUAL
SCHEMA OF THE osTICKET
SYSTEM BY APPLYING TDCM

Research report
April 2011

Story fragment #2b: General_settings_offline

Dependency: Email_template_default

Helpdesk status: Offline (customer functionalities active)
Helpdesk url: http://offlinesupport.com
Helpdesk name: ---
Default email template: default

Story fragment #3a: Ticket_settings_sequential

Ticket IDs mode: Sequential
Default priority: Normal
Customers can change ticket priority: No
Use email priority when available: Yes
Maximum open tickets per mail: 2
Ticket grace period: 0
Reopened tickets are assigned to the last respondent: Yes

Story fragment #3b: Ticket_settings_random

Ticket IDs mode: Random
Default priority: High
Customers can change ticket priority: Yes
Use email priority when available: No
Maximum open tickets per mail: Unlimited
Ticket grace period: 2
Reopened tickets are assigned to the last respondent: No

Story fragment #4a: Email_account_general

Dependency: Department_general

- An administrator creates the new email account general@support.com
  - Email address: questions@support.com
  - Email from name: General questions
  - Default new ticket priority for tickets created via this email: Low
  - Default new ticket department: dptGeneral
  - Auto responses status: enabled (overwrites department setting)

Story fragment #4b: Email_account_technical

Dependency: Department_technical

- An administrator creates the new email account technical@support.com
  - Email address: b@support.com
  - Email from name: B Support
  - Default new ticket priority for tickets created via this email: High
  - Default new ticket department: dptTechnical
  - Auto responses status: disabled (overwrites department setting)

Story fragment #5a: Department_general

Dependency: Email_template_default, Email_account_general

- An administrator creates the new public department dptGeneral
  - Department name: General support
  - Department email (used for outgoing emails): support@support.com
  - Department manager (one of the staff members, maybe empty): -
o Department type: public
o Email template (Used for outgoing emails and notices to user and staff): default
o Auto-response when new ticket is created: yes
o Auto-response when new message is added: yes
o Auto-response email: general@support.com

**Story fragment #5b: Department technical**
**Dependency: Email_template_default, Email_account_technical, Staff_member_generalAdministrator**

- An administrator creates the new private department dptTechnical
  - Department name: Technical support
  - Department email (used for outgoing emails): technical@support.com
  - Department manager (one of the staff members, maybe empty): generalAdministrator
  - Department type: private
  - Email template (Used for outgoing emails and notices to user and staff): default
  - Auto-response when new ticket is created: yes
  - Auto-response when new message is added: yes
  - Auto-response email: technical@support.com

**Story fragment #6a: Staff_group_minimum_privileges**

- An administrator creates the new staff group minimumPrivilegesGroup
  - Group name: Minimum Privileges Group
  - Group status: active
  - Departments access: -
  - Privileges (Yes/no)
    - Can create tickets: No
    - Can edit tickets: No
    - Can close tickets: No
    - Can transfer tickets: No
    - Can delete tickets: No
    - Can ban emails: No

**Story fragment #6b: Staff_group_maximum_privileges**
**Dependency: Department_general, Department_technical**

- An administrator creates the new staff group maximumPrivilegesGroup
  - Group name: Maximum Privileges Group
  - Group status: active
  - Departments access: dptGeneral, dptTechnical
  - Privileges (Yes/no)
    - Can create tickets: Yes
    - Can edit tickets: Yes
    - Can close tickets: Yes
    - Can transfer tickets: Yes
    - Can delete tickets: Yes
    - Can ban emails: Yes

**Story fragment #6c: Staff_group_inactive**

- An administrator creates the new staff group inactiveGroup
  - Group name: Inactive Group
  - Group status: disabled
  - Departments access: --
Privileges (Yes/no)
- Can create tickets: Yes
- Can edit tickets: No
- Can close tickets: Yes
- Can transfer tickets: No
- Can delete tickets: Yes
- Can ban emails: No

Story fragment #7a: Staff_member_general_administrator
Dependency: Department_general, Staff_group_maximum_privileges

- An administrator creates the new staff member generalAdministrator
  - Username: john
  - Department: dptGeneral
  - User group: maximumPrivilegesGroup
  - First Name: John
  - Last Name: Johny
  - Email address: john@support.com
  - Office phone: 11111
  - Phone extension: 11
  - Mobile phone: 11111
  - Signature: John Johny
  - Password: xxx
  - Account status: active
  - isAdministrator?: yes
  - Vacation mode (no tickets assigned and no alerts are received): no

Story fragment #7b: Staff_member_general_consultant
Dependency: Department_general, Staff_group_maximum_privileges

- An administrator creates the new staff member generalConsultant
  - Username: mary
  - Department: dptGeneral
  - User group: maximumPrivilegesGroup
  - First Name: Mary
  - Last Name: Mayer
  - Email address: mary@support.com
  - Office phone: 22222
  - Phone extension: 22
  - Mobile phone: 22222
  - Signature: Mary Mayer
  - Password: yyy
  - Account status: active
  - isAdministrator?: no
  - Vacation mode (no tickets assigned and no alerts are received): no

Story fragment #7c: Staff_member_general_consultant_vacation
Dependency: Department_general, Staff_group_maximum_privileges

- An administrator creates the new staff member generalConsultantVacation
  - Username: david
  - Department: dptGeneral
  - User group: maximumPrivilegesGroup
  - First Name: David
  - Last Name: Dassel
DEVELOPMENT OF THE CONCEPTUAL SCHEMA OF THE osTICKET SYSTEM BY APPLYING TDCM

April 2011

Research report

- Email address: david@support.com
- Office phone: 33333
- Phone extension: 33
- Mobile phone: 33333
- Signature: -
- Password: zzz
- Account status: active
- isAdministrator?: no
- Vacation mode (no tickets assigned and no alerts are received): yes

**Story fragment #7d: Staff_member_technical_active**

*Dependency: Department_technical, Staff_group_minimum_privileges*

- An administrator creates the new staff member *technicalActive*
  - Username: Martin
  - Department: dptTechnical
  - User group: minimumPrivilegesGroup
  - First Name: Martin
  - Last Name: Martech
  - Email address: martin@support.com
  - Office phone: ---
  - Phone extension: --
  - Mobile phone: ---
  - Signature: --
  - Password: ttt
  - Account status: active
  - isAdministrator?: no
  - Vacation mode (no tickets assigned and no alerts are received): no

**Story fragment #7e: Staff_member_technical_inactive**

*Dependency: Department_technical*

- An administrator creates the new staff member *technicalInactive*
  - Username: Patricia
  - Department: dptTechnical
  - User group: minimumPrivilegesGroup
  - First Name: Patricia
  - Last Name: Pauls
  - Email address: patricia@support.com
  - Office phone: ---
  - Phone extension: --
  - Mobile phone: ---
  - Signature: --
  - Password: uuu
  - Account status: locked
  - isAdministrator?: no
  - Vacation mode (no tickets assigned and no alerts are received): no

**Story fragment #8a: Customer_auto_responses_active**

Auto respond when a new ticket is created by a customer: Yes
Auto respond when a new ticket is created by a staff member: Yes
Auto respond when a new message has been appended to a ticket: Yes
Auto respond when a customer violates the maximum of open tickets: Yes
**Story fragment #8b: Customer auto responses inactive**

Auto respond when a new ticket is created by a customer: No  
Auto respond when a new ticket is created by a staff member: No  
Auto respond when a new message has been appended to a ticket: No  
Auto respond when a customer violates the maximum of open tickets: No

**Story fragment #9a: Email settings**

Default system email: general@support.com  
Default staff alerts email: general@support.com  
Administration email: system@support.com

**Story fragment #10a: Staff notices alerts active**

When a new ticket is created is alerted by mail to the staff: Yes  
(Administration email, Department manager, department members).  
When a new message is appended to a ticket, the message is alerted by mail to the staff: Yes  
(Last respondent, assigned staff, department manager).  
When a new internal note is added to a ticket, the new note is alerted by mail to the staff: Yes  
(Last respondent, assigned staff, department manager).  
When a new ticket is overdue, this fact is alerted by mail to the staff: Yes  
(Assigned staff, department manager, department members).

**Story fragment #10b: Staff notices alerts inactive**

When a new ticket is created is alerted by mail to the staff: No  
When a new message is appended to a ticket, the message is alerted by mail to the staff: No  
When a new internal note is added to a ticket, the new note is alerted by mail to the staff: No  
When a new ticket is overdue, this fact is alerted by mail to the staff: No

**Story fragment #11a: Help topic Installation**

Dependency: Department_technical

- An administrator creates the new staff member installationTopic  
  o Help topic name: Installation  
  o Topic status: active  
  o Auto response: disabled  
  o New Ticket Priority: High  
  o New ticket Department: dtpTechnical

**Story fragment #11b: Help topic Use**

Dependency: Department_general

- An administrator creates the new staff member useTopic  
  o Help topic name: Use  
  o Topic status: active  
  o Auto response: active  
  o New Ticket Priority: Normal  
  o New ticket Department: dtpGeneral
Story fragment #11c: Help_topic_disabled

**Dependency: Department_general**

- An administrator creates the new staff member offersTopic
  - Help topic name: Offers
  - Topic status: disabled
  - Auto response: disabled
  - New Ticket Priority: Low
  - New ticket Department: dtpGeneral

### Testable stories

Testable stories are combinations of previously defined story fragments which represent configurations of the system which are expected to be valid. Testable stories are those that result from the combinations of the following schema:

**Compatible system basics**

- Email_template_default
- Email_account_general
- Email_account_technical
- Department_general
- Department_technical
- Staff_group_minimum_privileges
- Staff_group_maximum_privileges
- Staff_group_inactive
- Staff_member_general_administrator
- Staff_member_general_consultant
- Staff_member_general_consultant_vacation
- Staff_member_technical_active
- Staff_member_technical_inactive
- Email_settings
- Help_topic_installation
- Help_topic_use
- Help_topic_disabled

**Configuration variations**

- General_settings_online
- General_settings_offline
- Ticket_settings_sequential
- Ticket_settings_random
- Customer_auto_responses_active
- Customer_auto_responses_inactive
- Staff_notices_alerts_active
- Staff_notices_alerts_inactive

Therefore, there are 16 valid testable stories about configuration and basics of the system that contains all compatible basics:
### DEVELOPMENT OF THE CONCEPTUAL SCHEMA OF THE osTICKET SYSTEM BY APPLYING TDCM

#### 4.2. Tickets management and tracking

We define several representative stories which are based on the use cases defined in Section 3.2. These stories are designed in order to cover the general scenarios (success scenario and extensions) of the defined use cases. Stories are ordered according to the strategy defined in Section 3.2.

<table>
<thead>
<tr>
<th>#5</th>
<th>Compatible basics</th>
<th>General_settings_online</th>
<th>Ticket_settings_random</th>
<th>Customer_auto_respon ses_active</th>
<th>Staff_notices_alerts_active</th>
</tr>
</thead>
<tbody>
<tr>
<td>#6</td>
<td>Compatible basics</td>
<td>General_settings_online</td>
<td>Ticket_settings_random</td>
<td>Customer_auto_respons es_active</td>
<td>Staff_notices_alerts_inactive</td>
</tr>
<tr>
<td>#7</td>
<td>Compatible basics</td>
<td>General_settings_online</td>
<td>Ticket_settings_random</td>
<td>Customer_auto_respons es_inactive</td>
<td>Staff_notices_alerts_active</td>
</tr>
<tr>
<td>#8</td>
<td>Compatible basics</td>
<td>General_settings_online</td>
<td>Ticket_settings_random</td>
<td>Customer_auto_respons es_inactive</td>
<td>Staff_notices_alerts_inactive</td>
</tr>
<tr>
<td>#9</td>
<td>Compatible basics</td>
<td>General_settings_offline</td>
<td>Ticket_settings_sequential</td>
<td>Customer_auto_respons es_active</td>
<td>Staff_notices_alerts_active</td>
</tr>
<tr>
<td>#10</td>
<td>Compatible basics</td>
<td>General_settings_offline</td>
<td>Ticket_settings_sequential</td>
<td>Customer_auto_respons es_active</td>
<td>Staff_notices_alerts_inactive</td>
</tr>
<tr>
<td>#11</td>
<td>Compatible basics</td>
<td>General_settings_offline</td>
<td>Ticket_settings_sequential</td>
<td>Customer_auto_respons es_inactive</td>
<td>Staff_notices_alerts_active</td>
</tr>
<tr>
<td>#12</td>
<td>Compatible basics</td>
<td>General_settings_offline</td>
<td>Ticket_settings_sequential</td>
<td>Customer_auto_respons es_inactive</td>
<td>Staff_notices_alerts_inactive</td>
</tr>
<tr>
<td>#13</td>
<td>Compatible basics</td>
<td>General_settings_offline</td>
<td>Ticket_settings_random</td>
<td>Customer_auto_respons es_active</td>
<td>Staff_notices_alerts_active</td>
</tr>
<tr>
<td>#14</td>
<td>Compatible basics</td>
<td>General_settings_offline</td>
<td>Ticket_settings_random</td>
<td>Customer_auto_respons es_active</td>
<td>Staff_notices_alerts_inactive</td>
</tr>
<tr>
<td>#15</td>
<td>Compatible basics</td>
<td>General_settings_offline</td>
<td>Ticket_settings_random</td>
<td>Customer_auto_respons es_inactive</td>
<td>Staff_notices_alerts_active</td>
</tr>
<tr>
<td>#16</td>
<td>Compatible basics</td>
<td>General_settings_offline</td>
<td>Ticket_settings_random</td>
<td>Customer_auto_respons es_inactive</td>
<td>Staff_notices_alerts_inactive</td>
</tr>
</tbody>
</table>

Other variations without some basics may be also used.

When evolving the schema to include the tickets management knowledge, all valid combinations can be used as the initial state of test cases.

#### 4.2. Tickets management and tracking
Customer stories

S1: NewTicketOnline_SuccessScenario_SequENTIALTicketsNumber_StaffNotifications
Testing objectives: NTO

- Fixture: #1
  - A customer wants to create a new ticket with the following data:
    - Full Name: “Mary Marnes”
    - Email: mary@marnes.mar
    - Telephone: xxxxxxxx
    - Ext: xxxxxxx
    - Help Topic: Installation
    - Subject: “Error operating system”
    - Message: “The installation process does not finish due to the following error: incompatible operating system”

  - The creation event occurs
    - No auto-response is sent to the customer.
    - The ticket number is 1 (sequential)
    - The status of the ticket is open
    - The subject of the ticket is “Error operating system”
    - The priority of the ticket is High
    - No staff is assigned to the ticket
    - The source of the ticket is “Web”
    - The creation date is SystemDateTime.now()
    - The due date is empty
    - The last response date is empty
    - The last message date is SystemDateTime.now()
    - The ticket thread contains the customer message
    - An alert is sent to the administration email, the department manager and the department members (except for those which are in vacation mode or inactive).
    - The assigned department is dptTechnical

- TicketSettings::customersCanChangePriority becomes true

  - A customer wants to create a new ticket with the following data:
    - Full Name: “James Jordan”
    - Email: james@jordan.jam
    - Telephone: xxxxxxxx
    - Ext: xxxxxxx
    - Help Topic: Use
    - Priority: Low (redefines the default priority of the help topic “Normal”)
    - Subject: “Reopening ticket”
    - Message: “I don’t know how to reopen one of my closed tickets”

  - The creation event occurs
    - An auto-response is sent to the customer.
    - The ticket number is 2 (sequential)
    - The priority of the ticket is Normal (the default priority)
    - An alert is sent to the administration email, the department manager and the department members (except for those which are in vacation mode or inactive).
    - The assigned department is dptGeneral
S2: NewTicketOnline maximumNumberOfTicketsViolated
Testing objectives: NTO

- Fixture: #1
- A customer wants to create a new ticket with the following data:
  - Full Name: “Mary Marnes”
  - Email: mary@marnes.mar
  - Telephone: xxxxxxxx
  - Ext: xxxxxxx
  - Help Topic: Installation
  - Subject: “Error operating system”
  - Message: “The installation process does not finish due to the following error: incompatible operating system”

- The creation event occurs

- A customer wants to create a new ticket with the following data:
  - Full Name: “Mary Marnes”
  - Email: mary@marnes.mar
  - Telephone: xxxxxxxx
  - Ext: xxxxxxx
  - Help Topic: Use
  - Subject: “Reopening ticket”
  - Message: “I don’t know how to reopen one of my closed tickets”

- The creation event occurs

- A customer wants to create a new ticket with the following data:
  - Full Name: “Mary Marnes”
  - Email: mary@marnes.mar
  - Telephone: xxxxxxxx
  - Ext: xxxxxxx
  - Help Topic: Use
  - Subject: “Customize graphical interface”
  - Message: “May I change the background color?”

- The creation event cannot occur (the maximum number of tickets for this email is violated)

S3: NewTicketOnline SuccessScenario RandomNumberOfTicketsNumber StaffNotificationsDisabled
Testing objectives: NTO

- Fixture: #8
- A customer wants to create a new ticket with the following data:
  - Full Name: “Mary Marnes”
  - Email: mary@marnes.mar
  - Telephone: xxxxxxxx
  - Ext: xxxxxxx
  - Help Topic: Installation
  - Subject: “Error operating system”
  - Message: “The installation process does not finish due to the following error: incompatible operating system”

- The creation event occurs
  - No auto-response is sent to the customer.
  - An alert is not sent to the administration email, the department manager and the department members (except for those which are in vacation mode or inactive)
  - The assigned department is dptTechnical
**S4: NewTicketOnline_SuccessScenario_SequentialTicketsNumber_AutoResponsesDisabled**

**Testing objectives:** NTO

- Fixture: #3
- A customer creates a new ticket with the following data:
  - Full Name: “James Jordan”
  - Email: james@jordan.jam
  - Telephone: xxxxxxxx
  - Ext: xxxxxxx
- Priority: Normal
- Help Topic: Use
- Subject: “Reopening ticket”
- Message: “I don’t know how to reopen one of my closed tickets”
- The creation event occurs
  - No auto-response is sent to the customer.
  - An alert is sent to the administration email, the department manager and the department members (except for those which are in vacation mode or inactive)
- The assigned department is `dptGeneral`

**S5: NewTicketOnline_SuccessScenario_NoTopic**

**Testing objectives:** NTO

- Fixture: #4 (without help topics)
- A customer wants to create a new ticket with the following data:
  - Full Name: “James Jordan”
  - Email: james@jordan.jam
  - Subject: “Reopening ticket”
  - Message: “I don’t know how to reopen one of my closed tickets”
- The creation event occurs
  - No auto-response is sent to the customer.
  - No alert is sent to the administration email, the department manager and the department members.
  - The help topic of the ticket is empty
  - The priority of the ticket is “Normal”
  - The assigned department is `dptGeneral` (which is the default department)

**S6: NewTicketOnline_Extension2a**

**Testing objectives:** NTO

- Fixture: #8
- A customer wants to create a new ticket with the following data:
  - Full Name: James Jordan
  - Email: james@jordan.jam
  - Priority: Low
  - Help Topic: Offers
- Subject: “Reopening ticket”
- Message: “I don’t know how to reopen one of my closed tickets”
- The creation event cannot occur (the help topic is disabled)

**S7: NewTicketOnline_PreconditionViolation**

**Testing objectives:** NTO

- Fixture: #9
- A customer wants to create a new ticket with the following data:
  - Full Name: James Jordan
Priority: Low
Email: james@jordan.jam
Subject: “Reopening ticket”
Message: “I don’t know how to reopen one of my closed tickets”

The creation event cannot occur (online mode is disabled)

**S8: NewTicketByEmail_successScenario_generalEmail**

Testing objectives: NTE

- Fixture: #3
- A customer wants to create a new ticket by an email sent to general@support.com
  - From: james@jordan.jam
  - Subject: “Ticket priority”
  - Message: “How can I change the priority of one of my tickets?”
- The creation event occurs
  - An auto-response is sent to the customer.
  - The status of the ticket is open
  - The subject of the ticket is “Ticket Priority”
  - The priority of the ticket is Low
  - No staff is assigned to the ticket
  - The source of the ticket is “Email”
  - The creation date is SystemDateTime.now()
  - The due date is empty
  - The last response date is empty
  - The last message date is SystemDateTime.now()
  - The ticket thread contains the customer message
  - The assigned department is dptGeneral

**S9: NewTicketByEmail_successScenario_technicalEmail**

Testing objectives: NTE

- Fixture: #10
- A customer wants to create a new ticket by an email sent to technical@support.com
  - From: marta@johnes.mar
  - Subject: “See my tickets”
  - Message: “Can I see my tickets?”
- The creation event cannot occur (the online mode is disabled)
- The online mode is enabled
- The creation event occurs
  - No auto-response is sent to the customer.
  - The status of the ticket is open
  - The subject of the ticket is “See my tickets”
  - The priority of the ticket is High
  - No staff is assigned to the ticket
  - The source of the ticket is “Email”
  - The creation date is SystemDateTime.now()
  - The due date is empty
  - The last response date is empty
  - The last message date is SystemDateTime.now()
  - The ticket thread contains the customer message
  - The assigned department is dptTechnical
S10: NewTicketByEmail_extension_2a
Testing objectives: NTE

- Fixture: #8
- A customer wants to create a new ticket by an email sent to techdep@support.com
  - From: marta@johnes.mar
  - Subject: “See my tickets”
  - Message: “Can I see my tickets?”
- The creation event cannot occur (the addressee is not an incoming email)

S11: DisplayTicketsOfEmail_successScenario
Testing objectives: CTS

- Fixture: #1
- A customer creates a new ticket with the following data:
  - Full Name: James Jordan
  - Help Topic: Use
  - Email: james@jordan.jam
  - Subject: “Reopening ticket”
  - Message: “I don’t know how to reopen one of my closed tickets”
- The creation event occurs
- A customer creates a new ticket with the following data:
  - Full Name: James Jordan
  - Email: james@jordan.jam
  - Help Topic: Installation
  - Subject: “Error operating system”
  - Message: “The installation process does not finish due to the following error: incompatible operating system”
- The creation event occurs
- The customer wants to keep track of his tickets by providing the email james@jordan.jam and the number of one of his/her tickets.
- The system displays the list of his/her tickets (number, create date, status, subject, department, email).

S12: DisplayTicketsOfEmail_extension_2a_nonExisting
Testing objectives: CTS

- Fixture: #1
- The customer wants to keep track of his tickets by providing the email james@jordan.jam and the ticket number 3.
- The query cannot occur because there are no tickets associated to james@jordan.jam

S13: DisplayTicketsOfEmail_extension_2a_invalidData
Testing objectives: CTS

- Fixture: #1
- A customer creates a new ticket with the following data:
  - Full Name: James Jordan
  - Priority: Low
  - Help Topic: Use
  - Email: james@jordan.jam
  - Subject: “Reopening ticket”
  - Message: “I don’t know how to reopen one of my closed tickets”
The creation event occurs
The customer wants to keep track of his tickets by providing the email james@jordan.jam and the ticket number 2.
The query cannot occur because the pair (james@jordan.jam, 2) is not valid

S14: RespondTicket_alertsActive
Testing objectives: CTS

- Fixture: #1
- A customer creates a new ticket with the following data:
  - Full Name: James Jordan
  - Help Topic: Installation (it enables autoresponses)
  - Email: james@jordan.jam
  - Subject: “Reopening ticket”
  - Message: “I don’t know how to reopen one of my closed tickets”
- The creation event occurs
- The customer wants to respond the ticket
  - The customer provides the response
- The reply event occurs
  - The ticket thread contains the customer response
  - A notice is sent to the department manager
  - An autoresponse is sent to the customer
  - The last message date is updated with the ticket thread customer response datetime

S15: RespondTicket_alertsDisabled
Testing objectives: CTS

- Fixture: #2
- A customer creates a new ticket with the following data:
  - Full Name: James Jordan
  - Help Topic: Installation
  - Email: james@jordan.jam
  - Subject: “Reopening ticket”
  - Message: “I don’t know how to reopen one of my closed tickets”
- The creation event occurs
- The customer wants to respond the ticket
  - The customer provides the response
- The reply event occurs
  - The ticket thread contains the customer response
  - No notice is sent.
  - The last message date is updated with the ticket thread customer response datetime

Staff stories

S16: StaffLogin_successScenario
Testing objectives: NTOF

- Fixture: #1
- A staff member wants to login with the following data:
  - Username: mary
  - Password: yyy
- The login event occurs
The staff member *generalConsultant* is logged in.

**S17: StaffLogIn_PreconditionViolation**

Testing objectives: NTOF

- Fixture: #1
- A staff member wants to login with the following data:
  - Username: mary
  - Password: yyy
- The creation event occurs
  - The staff member *generalConsultant* is logged in.
- A staff member wants to login with the following data:
  - Username: mary
  - Password: yyy
- The login event cannot occur (the staff member is already logged in)

**S18: StaffLogIn_PreconditionViolation_InactiveStaffMember**

Testing objectives: NTOF

- Fixture: #1
- A staff member wants to login with the following data:
  - Username: Patricia
  - Password: uuu
- The login event cannot occur (the staff member is locked)
- The group *maximumPrivilegesGroup* becomes disabled
- A staff member wants to login with the following data:
  - Username: mary
  - Password: yyy
- The login event cannot occur (the staff member is locked)

**S19: StaffLogOut_successScenario**

Testing objectives: NTOF

- Fixture: #1
- A staff member wants to login with the following data:
  - Username: mary
  - Password: yyy
- The login event occurs
  - The staff member *generalConsultant* is logged in.
- The user logs out
  - The staff member *generalConsultant* is not logged in.

**S20: StaffLogIn_Extension2a**

Testing objectives: NTOF

- Fixture: #1
- A staff member wants to login with the following data:
  - Username: mary
  - Password: zzz
- The login event cannot occur (the login data is not valid)
S21: NewTicketOffline_SuccessScenario_Sequen...alertsAutoresponsesActive
Testing objectives: NTOF

- Fixture: #1 (minimumPrivilegesGroup allows creating tickets)
- The staff member generalConsultant logs in and wants to create a new ticket with the following data:
  - Full Name: “Mary Marnes”
  - Email: mary@marnes.mar
  - Telephone: xxxxxxxx
  - Ext: xxxxxxx
  - Ticket source: Phone
  - Department: Technical
  - Help Topic: Installation
  - Subject: “Error operating system”
  - Message: “The installation process does not finish due to the following error: incompatible operating system”
  - Internal note: “It seems that the correct installer is being used”
  - Due datetime: SystemDateTime.now() + 2 days
  - Priority: Normal
  - AssignedStaff: generalConsultant

- The creation event occurs
  - No auto-response is sent to the customer.
  - The ticket number is 1 (sequential)
  - The status of the ticket is open
  - The subject of the ticket is “Error operating system”
  - The priority of the ticket is Normal
  - Staff member generalConsultant is assigned to the ticket
  - The source of the ticket is “Phone”
  - The creation date is SystemDateTime.now()
  - The due date is SystemDateTime.now() + 2 days
  - The last response date is empty
  - The last message date is SystemDateTime.now()
  - The ticket thread contains the initial message
  - An alert is sent to the administration email, the department manager and the department members (except for those which are in vacation mode or inactive).
  - The assigned department is dptTechnical

- The staff member generalConsultant wants to create a new ticket with the following data:
  - Full Name: “James Jordan”
  - Email: james@jordan.jam
  - Help Topic: Use
  - Subject: “Reopening ticket”
  - Message: “I don’t know how to reopen one of my closed tickets”
  - Ticket source: Other
  - Department: dptGeneral
  - Internal note: (empty)
  - Due datetime: (empty)
  - Priority: Low
  - AssignedStaff: (empty)

- The creation event occurs
  - An auto-response is sent to the customer.
  - The ticket number is 2 (sequential)
  - The ticket is unassigned
  - The priority of the ticket is Low (the default priority)
  - An alert is sent to the administration email, the department manager and the department members (except for those which are in vacation mode or inactive).
  - The assigned department is dptGeneral
S22: NewTicketOffline_SuccessScenario_alertsAutoresponsesDisabled
Testing objectives: NTO

- Fixture: #2 (minimumPrivilegesGroup allows creating tickets)
- The staff member generalConsultant logs in and wants to create a new ticket with the following data:
  - Full Name: “Mary Marnes”
  - Email: mary@marnes.mar
  - Ticket source: Phone
  - Department: Technical
  - Help Topic: Installation
  - Subject: “Error operating system”
  - Message: “The installation process does not finish due to the following error: incompatible operating system”
  - Internal note: “It seems that the correct installer is being used”
  - Due datetime: SystemDateTime.now() + 2 days
  - Priority: Normal
  - AssignedStaff: generalConsultant
- The creation event occurs
  - No auto-response is sent to the customer.
  - An alert is not sent to the administration email, the department manager and the department members.
  - The assigned department is dptTechnical

S23: NewTicketOffline_SuccessScenario_NoTopic
Testing objectives: NTO

- Fixture: #4 (without help topics and minimumPrivilegesGroup allows creating tickets)
- The staff member generalAdministrator logs in and wants to create a new ticket with the following data:
  - Full Name: “James Jordan”
  - Email: james@jordan.jam
  - Subject: “Reopening ticket”
  - Message: “I don’t know how to reopen one of my closed tickets”
  - Ticket source: Other
  - Department: dptGeneral
  - Internal note: (empty)
  - Due datetime: (empty)
  - Priority: Low
  - AssignedStaff: (empty)
- The creation event occurs
  - The help topic of the ticket is empty

S24: NewTicketOffline_PreconditionViolation_cannotCreateTickets
Testing objectives: NTO

- Fixture: #9
- The staff member technicalActive logs in and wants to create a new ticket with the following data:
  - Full Name: “Mary Marnes”
  - Email: mary@marnes.mar
  - Ticket source: Phone
Department: Technical  
Help Topic: Installation  
Subject: “Error operating system”  
Message: “The installation process does not finish due to the following error: incompatible operating system”  
Internal note: (empty)  
Due datetime: (empty)  
Priority: Normal  
AssignedStaff: generalConsultant

The creation event cannot occur (the staff member cannot create tickets).

**S25: NewTicketOffline_PreconditionViolation_isNotLoggedIn**

Testing objectives: NTO

- Fixture: #9
- The staff member technicalActive wants to create a new ticket with the following data:
  - Full Name: “Mary Marnes”
  - Email: mary@marnes.mar
  - Ticket source: Phone
  - Department: Technical
  - Help Topic: Installation
  - Subject: “Error operating system”
  - Message: “The installation process does not finish due to the following error: incompatible operating system”
  - Internal note: (empty)
  - Due datetime: (empty)
  - Priority: Normal
  - AssignedStaff: generalConsultant

  The creation event cannot occur (the staff member cannot create tickets).

**S26: ViewOpenTickets**

Testing objectives: VTS

- Fixture: #11
- Fixture component: #created_tickets
  - The staff member generalConsultant logs in and wants to create a new ticket with the following data:
    - Full Name: “Mary Marnes”
    - Email: mary@marnes.mar
    - Telephone: xxxxxxxx
    - Ext: xxxxxxx
    - Ticket source: Phone
    - Department: dptTechnical
    - Help Topic: Installation
    - Subject: “Error operating system”
    - Message: “The installation process does not finish due to the following error: incompatible operating system”
    - Internal note: “It seems that the correct installer is being used”
    - Due datetime: SystemDateTime.now() + 2 days
    - Priority: Normal
    - AssignedStaff: generalConsultant

  The creation event occurs
  - The staff member generalConsultant wants to create a new ticket with the following data:
Full Name: “John Johnes”
Email: john@johnes.nes
Ticket source: Other
Department: dptGeneral
Help Topic: Use
Subject: “Can I reply a ticket?”
Message: “I don’t know how to reply a ticket”
Due datetime: (empty)
Priority: High
Assigned Staff: generalConsultant

The creation event occurs
The staff member GeneralConsultant logs out
The staff member generalAdministrator logs in and wants to create a new ticket with the following data:
- Full Name: “Martin Pope”
- Email: martin@pope.mar
- Ticket source: Phone
- Department: dptTechnical
- Help Topic: Use
- Subject: “Error while login”
- Message: “I get an error when I try to login”
- Due datetime: SystemDateTime.now() + 5 days
- Priority: Low
- Assigned Staff: technicalActive

The creation event occurs
SystemDateTime.now() +1
The staff member generalAdministrator logs out.
A customer creates a new ticket with the following data:
- Full Name: “James Jordan”
- Email: james@jordan.jam
- Help Topic: Use
- Subject: “Reopening ticket”
- Message: “I don’t know how to reopen one of my closed tickets”

The creation event occurs
A customer wants to create a new ticket by an email sent to technical@support.com
- From: marta@johnes.mar
- Subject: “See my tickets”
- Message: “Can I see my tickets?”

The creation event occurs
The staff member generalAdministrator logs in
The staff member generalAdministrator wants to view the open tickets (this staff member can see the tickets of all departments)
The result of the query is:

<table>
<thead>
<tr>
<th>Ticket</th>
<th>Date</th>
<th>Subject</th>
<th>Department</th>
<th>Priority</th>
<th>From</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>now()-1</td>
<td>Error operating system</td>
<td>dptTechnical</td>
<td>Normal</td>
<td>Mary Marnes</td>
</tr>
<tr>
<td>2</td>
<td>now()-1</td>
<td>Can I reply a ticket?</td>
<td>dptGeneral</td>
<td>High</td>
<td>John Johnes</td>
</tr>
<tr>
<td>3</td>
<td>now()-1</td>
<td>Error while login</td>
<td>dptTechnical</td>
<td>Low</td>
<td>Martin Pope</td>
</tr>
<tr>
<td>4</td>
<td>now()</td>
<td>Reopening ticket</td>
<td>dptGeneral</td>
<td>Normal</td>
<td>James Jordan</td>
</tr>
<tr>
<td>5</td>
<td>now()</td>
<td>See my tickets</td>
<td>dptTechnical</td>
<td>High</td>
<td><a href="mailto:marta@johnes.mar">marta@johnes.mar</a></td>
</tr>
</tbody>
</table>

The staff member generalAdministrator logs out
The staff member technicalActive logs in
The staff member *technicalActive* wants to view the open tickets (this staff member can only see the tickets of his/her department)

The result of the query is:

<table>
<thead>
<tr>
<th>Ticket</th>
<th>Date</th>
<th>Subject</th>
<th>Department</th>
<th>Priority</th>
<th>From</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>now()-1</td>
<td>Error operating system</td>
<td>dptTechnical</td>
<td>Normal</td>
<td>Mary Marnes</td>
</tr>
<tr>
<td>3</td>
<td>now()-1</td>
<td>Error while login</td>
<td>dptTechnical</td>
<td>Low</td>
<td>Martin Pope</td>
</tr>
<tr>
<td>5</td>
<td>now()</td>
<td>See my tickets</td>
<td>dptTechnical</td>
<td>High</td>
<td><a href="mailto:mart@johnes.mar">mart@johnes.mar</a></td>
</tr>
</tbody>
</table>

**S27: ViewOpenTickets_preconditionViolation_notLoggedIn**

Testing objectives: VTS

- Fixture: #3
- The staff member wants to view the open tickets
- The query cannot occur because the staff member is not logged in

**S28: ChangeTicketPriority**

Testing objectives: MT

- Fixture: #3
- Fixture component: #created_tickets
- The staff member *generalAdministrator* logs in.
- The staff member *generalAdministrator* wants to set the priority of the ticket #1 to *High*.
- The event for changing priority occurs
  - The ticket priority is *High*
  - The ticket thread includes the internal note with title “Ticket priority changed” and text “Ticket priority set to *High* by John Johny”

**S29: ChangeTicketPriority_TicketNotVisible**

Testing objectives: MT

- Fixture: #4
- Fixture component: #created_tickets
- The staff member *technicalActive* logs in.
- The staff member *technicalActive* wants to set the priority of the ticket #2 to *Normal*.
- The event for changing priority cannot occur (the ticket is not visible for the staff member)

**S30: ChangeTicketPriority_NotLoggedIn**

Testing objectives: MT

- Fixture: #4
- Fixture component: #created_tickets
- The staff member *generalAdministrator* logs in.
- The staff member *generalAdministrator* wants to set the priority of the ticket #2 to *Normal*.
- The event for changing priority cannot occur (the staff member is not logged in)

**S31: MarkTicketOverdue**

Testing objectives: MT

- Fixture: #3
- Fixture component: #created_tickets
- The staff member generalAdministrator logs in.
- The staff member generalAdministrator wants to mark the ticket #1 as overdue.
- The event occurs
  - The ticket is overdue
  - The ticket thread includes the internal note with title “Ticket MarkedOverdue” and text “Ticket flagged as overdue by John Johny”

**S32: MarkTicketOverdue_staffIsNotAnAdministrator**
Testing objectives: MT

- Fixture: #4
- Fixture component: #created_tickets
- The staff member technicalInactive logs in.
- The staff member technicalInactive wants to mark the ticket #1 as overdue.
- The event cannot occur (the staff member is not an administrator)

**S33: MarkTicketOverdue_notLoggedIn**
Testing objectives: MT

- Fixture: #4
- Fixture component: #created_tickets
- The staff member generalAdministrator wants to mark the ticket #1 as overdue.
- The event cannot occur (the staff member is not logged in)

**S34: AssignTicket**
Testing objectives: MT

- Fixture: #3
- Fixture component: #created_tickets
- The staff member generalConsultant logs in.
- The staff member generalConsultant wants to assign ticket #1 to generalAdministrator with the comment “This is for you”.
- The assignment event occurs
  - The ticket is assigned to generalAdministrator
  - The ticket thread includes the internal note posted by the generalConsultant with title “Ticket assigned to John Johny by Mary Mayer” and text “This is for you”.
  - An alert is sent to the generalAdministrator

**S35: AssignTicket_ticketIsNotVisible**
Testing objectives: MT

- Fixture: #3
- Fixture component: #created_tickets
- The staff member technicalActive logs in.
- The staff member technicalActive wants to assign ticket #2 to generalAdministrator with the comment “This is for you”.
- The event cannot occur (the staff member cannot manage the ticket)
S36: AssignTicket _InVacationMode
Testing objectives: MT
- Fixture: #3
- Fixture component: #created_tickets
- The staff member generalConsultant logs in.
- The staff member generalConsultant wants to assign ticket #1 to generalConsultantVacation with the comment “This is for you”.
- The assignment event cannot occur (the staff member is on vacation mode).

S37: AssignTicket _NotLoggedIn
Testing objectives: MT
- Fixture: #3
- Fixture component: #created_tickets
- The staff member generalConsultant wants to assign ticket #1 to generalAdministrator with the comment “This is for you”.
- The assignment event cannot occur (the staff member is not logged in).

S38: ReleaseTicket
Testing objectives: MT
- Fixture: #3
- Fixture component: #created_tickets
- The staff member generalConsultant logs in.
- The staff member generalConsultant wants to release ticket #1
- The event occurs
  - The ticket is not assigned
  - The ticket thread includes the internal note with title “Ticked Unassigned” and text “Ticket released (unassigned) from Mary Mayer by Mary Mayer”.

S39: ReleaseTicket _ticketIsNotVisible
Testing objectives: MT
- Fixture: #3
- Fixture component: #created_tickets
- The staff member technicalActive logs in.
- The staff member technicalActive wants to release ticket #2.
- The event cannot occur (the staff member cannot manage the ticket)

S40: ReleaseTicket _NotAssigned
Testing objectives: MT
- Fixture: #3
- Fixture component: #created_tickets
- The staff member generalConsultant logs in.
- The staff member generalConsultant wants to release ticket #4.
- The event cannot occur (the ticket is not assigned).
S41: ReleaseTicket _NotLoggedIn
Testing objectives: MT

- Fixture: #3
- Fixture component: #created_tickets
- The staff member generalConsultant wants to release ticket #2.
- The event cannot occur (the staff member is not logged in)

S42: EditTicket
Testing objectives: MT

- Fixture: #3
- Fixture component: #created_tickets
- The staff member generalConsultant logs in.
- The staff member generalConsultant wants to edit ticket #1
  - EmailAddress: mary2@marnes.mar
  - Full Name: Mary Marnes2
  - Subject: “Error operating system2”
  - Telephone: “xxx2”
  - Ext: “xx2”
  - Due datetime: SystemDateTime.now() + 3 days
  - Priority: Low
  - Help topic: Use
  - Internal note (reasons for edit): “The customer asks for this changes”
- The event occurs
  - The ticket is updated
  - The ticket thread includes the internal note with title “Ticket updated” and text “The customer asks for this changes”.

S43: EditTicket _ticketIsNotVisible
Testing objectives: MT

- Fixture: #3 (technicalActive becomes a member of the group maximumPrivilegesGroup)
- Fixture component: #created_tickets
- The staff member technicalActive logs in.
- The staff member technicalActive wants to edit ticket #2
  - EmailAddress: john2@johnes.nes
  - Full Name: John Johnes 22
  - Subject: “Can I reply a ticket? Yes or no?”
  - Telephone: “yyy2”
  - Ext: “yy2”
  - Due datetime: SystemDateTime.now() + 2 days
  - Priority: Normal
  - Help topic: Use
  - Internal note (reasons for edit): “The customer asks for this changes”
- The event cannot occur (the ticket is not visible)

S44: EditTicket _NotAllowed
Testing objectives: MT

- Fixture: #3
- Fixture component: #created_tickets
- The staff member technicalActive logs in.
- The staff member *technicalActive* wants to edit ticket #1
  - Email Address: mary2@marnes.mar
  - Full Name: Mary Marnes2
  - Subject: “Error operating system2”
  - Telephone: “xxx2”
  - Ext: “xx2”
  - Due datetime: `SystemDateTime.now() + 3 days`
  - Priority: Low
  - Help topic: Use
  - Internal note (reasons for edit): “The customer asks for these changes”

The event cannot occur (the staff member is not allowed to edit tickets)

**S45: EditTicketNotAllowedButAdministrator**

Testing objectives: MT

- Fixture: #3(*technicalActive* becomes an administrator)
- Fixture component: #created_tickets
- The staff member *technicalActive* logs in.
- The staff member *technicalActive* wants to edit ticket #1
  - Email Address: mary2@marnes.mar
  - Full Name: Mary Marnes2
  - Subject: “Error operating system2”
  - Telephone: “xxx2”
  - Ext: “xx2”
  - Due datetime: `SystemDateTime.now() + 3 days`
  - Priority: Low
  - Help topic: Use
  - Internal note (reasons for edit): “The customer asks for these changes”

The event occurs

**S46: EditTicket_NotLoggedIn**

Testing objectives: MT

- Fixture: #3
- Fixture component: #created_tickets
- The staff member *generalConsultant* wants to edit ticket #1
  - Email Address: mary2@marnes.mar
  - Full Name: Mary Marnes2
  - Subject: “Error operating system2”
  - Telephone: “xxx2”
  - Ext: “xx2”
  - Due datetime: `SystemDateTime.now() + 3 days`
  - Priority: Low
  - Help topic: Use
  - Internal note (reasons for edit): “The customer asks for these changes”

The event cannot occur (the staff member is not logged in).

**S47: PostTicketReply_alertsAndAutoresponsesActive**

Testing objectives: MT

- Fixture: #1 (help topic installation has autoresponses enabled)
- Fixture component: #created_tickets
- The staff member *generalConsultant* logs in.
The staff member generalConsultant wants to reply ticket #1:
  o “You should choose the installation executable according to your operating system”.

The event occurs
  o The last response date/time is updated.
  o The ticket thread includes the response.
  o An alert of the response is sent to the last respondent, the assigned staff and the department manager (except for those which are in vacation mode or inactive).
  o An auto-response is sent to the customer.

S48: PostTicketReply_alertsAndAutoresponsesDisabled
Testing objectives: MT

  Fixture: #4 (help topic installation has autoresponses enabled)
  Fixture component: #created_tickets
  The staff member generalConsultant logs in.
  The staff member generalConsultant wants to reply ticket #1:
    o “You should choose the installation executable according to your operating system”.
  The event occurs
    o The last response date/time is updated.
    o The ticket thread includes the response.
    o No alert of the response is sent to the last respondent, the assigned staff and the department manager (except for those which are in vacation mode or inactive).
    o No auto-response is sent to the customer.

S49: PostTicketReply_ticketIsNotVisible
Testing objectives: MT

  Fixture: #4
  Fixture component: #created_tickets
  The staff member technicalActive logs in.
  The staff member technicalActive wants to reply ticket #2:
    o “You may click on the “reply” option”.
  The event cannot occur (the ticket is not visible)

S50: PostTicketReply_NotLoggedIn
Testing objectives: MT

  Fixture: #1
  Fixture component: #created_tickets
  The staff member generalConsultant wants to reply ticket #1:
    o “You should choose the installation executable according to your operating system”.
  The event cannot occur (the staff member is not logged in).

S51: PostTicketInternalNote_staffAlertsEnabled
Testing objectives: MT

  Fixture: #1
  Fixture component: #created_tickets
  The staff member technicalActive logs in.
  The staff member technicalActive wants to add an internal note titled “No tickets?” to ticket #5
    o Note: “It seems that she does not have tickets”
  The event for changing priority occurs
The ticket thread includes the internal note
An alert of the internal note is sent to the last respondent, the assigned staff and the department manager (except for those which are in vacation mode or inactive).

S52: PostTicketInternalNote_staffAlertsDisabled
Testing objectives: MT

- Fixture: #4
- Fixture component: #created_tickets
- The staff member technicalActive logs in.
- The staff member technicalActive wants to add an internal note titled “No tickets?” to ticket #5
  - Note: “It seems that she does not have tickets”
- The event for changing priority occurs
  - The ticket thread includes the internal note
  - No alert of the internal note is sent to the last respondent, the assigned staff and the department manager (except for those which are in vacation mode or inactive).

S53: PostTicketInternalNote_TicketNotVisible
Testing objectives: MT

- Fixture: #4
- Fixture component: #created_tickets
- The staff member technicalActive logs in.
- The staff member technicalActive wants to add an internal note titled “Checked button” to ticket #4
  - Note: “Checked that the button appears”
- The event cannot occur (the ticket is not visible for the staff member)

S54: PostTicketInternalNote_NotLoggedIn
Testing objectives: MT

- Fixture: #4
- Fixture component: #created_tickets
- The staff member technicalActive logs in.
- The staff member technicalActive wants to add an internal note titled “No tickets?” to ticket #5
  - Note: “It seems that she does not have tickets”
- The event cannot occur (the staff member is not logged in)

S55: TransferTicket_staffAlertsEnabled
Testing objectives: MT

- Fixture: #4
- Fixture component: #created_tickets
- The staff member generalConsultant logs in.
- The staff member generalConsultant wants to transfer ticket #2 to dptTechnical with the comment “This is a technical question to be resolved as soon as possible”.
- The event occurs
  - The ticket is transferred to the department dptTechnical
  - The ticket thread includes the internal note posted by the generalConsultant with title “Dept. Transfer from General support to Technical Support” and text “This is a technical question to be resolved as soon as possible”.
S56: TransferTicket _ticketIsNotVisible
Testing objectives: MT

- Fixture: #3 (technicalActive becomes a member of the group maximumPrivilegesGroup and his access to the ticket department is restricted)
- Fixture component: #created_tickets
- The staff member technicalActive logs in.
- The staff member technicalActive wants to transfer ticket #2 to dptTechnical with the comment “This is a technical question to be resolved as soon as possible”.
- The event cannot occur (the ticket is not visible for the staff member).

S57: TransferTicket _SameDepartment
Testing objectives: MT

- Fixture: #3
- Fixture component: #created_tickets
- The staff member generalConsultant logs in.
- The staff member generalConsultant wants to transfer ticket #2 to dptGeneral with the comment “This is a technical question to be resolved as soon as possible”.
- The event cannot occur (the ticket is already assigned to this department).

S58: TransferTicket _NotAllowed ToTransfer
Testing objectives: MT

- Fixture: #3
- Fixture component: #created_tickets
- The staff member technicalActive logs in.
- The staff member technicalActive wants to transfer ticket #1 to dptGeneral with the comment “This is a technical question to be resolved as soon as possible”.
- The assignment event cannot occur (the staff member is not allowed to transfer tickets).

S59 TransferTicket _NotLoggedln
Testing objectives: MT

- Fixture: #3
- Fixture component: #created_tickets
- The staff member generalConsultant wants to transfer ticket #1 to dptGeneral with the comment “This is a technical question to be resolved as soon as possible”.
- The assignment event cannot occur (the staff member is not logged in).

S60: CloseTicket
Testing objectives: MT

- Fixture: #3
- Fixture component: #created_tickets
- The staff member generalConsultant logs in.
- The staff member generalConsultant wants to close ticket #1
- The event occurs
  - The ticket is closed
  - The ticket thread includes the internal note with title “Ticket closed” and text “Ticket close without response by Mary Mayer”.
- The staff member generalConsultant wants to close ticket #1
- The event cannot occur (the ticket is already closed)
S61: **CloseTicket_ticketIsNotVisibleOrNotAllowed**
Testing objectives: MT
- Fixture: #3
- Fixture component: #created_tickets
- The staff member `generalConsultant` logs in (its staff group becomes not allowed to see tickets of `dptTechnical`)
- The staff member `generalConsultant` wants to close ticket #1
- The event cannot occur (the staff member cannot manage the ticket)
- The staff group of the staff member `generalConsultant` becomes allowed to see tickets of `dptTechnical` but it loses the privilege to close tickets
- The staff member `generalConsultant` wants to close ticket #1
- The event cannot occur (the staff member cannot close tickets)

S62: **CloseTicket_NotLoggedIn**
Testing objectives: MT
- Fixture: #3
- Fixture component: #created_tickets
- The staff member `generalConsultant` wants to close ticket #1
- The event cannot occur (the staff member is not logged in)

S63: **CloseTicketWithReply_alertsAndAutoresponsesEnabled**
Testing objectives: MT
- Fixture: #3
- Fixture component: #created_tickets
- The staff member `generalConsultant` logs in.
- The staff member `generalConsultant` wants to close ticket #1 with the reply “Ticket solved”
- The event occurs
  - The ticket is closed
  - The ticket thread includes the internal note with title “Ticket closed” and text “Ticket close without response by Mary Mayer”.
  - The ticket thread includes the reply “Ticket solved”
  - An auto-response is sent to the customer
- The staff member `generalConsultant` wants to close ticket #1 with the reply “Ticket solved”
- The event cannot occur (the ticket is already closed)

S64: **CloseTicketWithReply_alertsAndAutoresponsesDisabled**
Testing objectives: MT
- Fixture: #4
- Fixture component: #created_tickets
- The staff member `generalConsultant` logs in.
- The staff member `generalConsultant` wants to close ticket #1 with the reply “Ticket solved”
- The event occurs
  - The ticket is closed
  - The ticket thread includes the internal note with title “Ticket closed” and text “Ticket close without response by Mary Mayer”.
  - The ticket thread includes the reply “Ticket solved”
  - No auto-response is sent to the customer
S65: CloseTicketWithReply_ticketIsNotVisibleOrNotAllowed
Testing objectives: MT
- Fixture: #3
- Fixture component: #created_tickets
- The staff member generalConsultant logs in (its staff group becomes not allowed to see tickets of dptTechnical)
- The staff member generalConsultant wants to close ticket #1 with reply “Ticket solved”
- The event cannot occur (the staff member cannot manage the ticket)
- The staff group of the staff member generalConsultant becomes allowed to see tickets of dptTechnical but it loses the privilege to close tickets
- The staff member generalConsultant wants to close ticket #1 with reply “Ticket solved”
- The event cannot occur (the staff member cannot close tickets)

S66: CloseTicketWithReply_NotLoggedIn
Testing objectives: MT
- Fixture: #3
- Fixture component: #created_tickets
- The staff member generalConsultant wants to close ticket #1 with the reply “Ticket solved”
- The event cannot occur (the staff member is not logged in)

S67: ReopenTicket
Testing objectives: MT
- Fixture: #3 (ticket #1 is closed)
- Fixture component: #created_tickets
- The staff member generalConsultant logs in.
- The staff member generalConsultant wants to reopen ticket #1
- The event occurs
  - The ticket is open
  - The ticket thread includes the internal note with title “Ticket reopened” and text “Ticket reopened (without comments) by Mary Mayer”.
- The staff member generalConsultant wants to reopen ticket #1
- The event cannot occur (the ticket is already open)

S68: ReopenTicket_ticketIsNotVisibleOrNotAllowed
Testing objectives: MT
- Fixture: #3 (ticket #2 is closed)
- Fixture component: #created_tickets
- The staff member technicalActive logs in.
- The staff member technicalActive wants to reopen ticket #2
- The event cannot occur (the staff member cannot manage the ticket)

S69: ReopenTicket_NotLoggedIn
Testing objectives: MT
- Fixture: #3 (ticket #1 is closed)
- Fixture component: #created_tickets
- The staff member generalConsultant wants to close ticket #1
- The event cannot occur (the staff member is not logged in)
S70: ReopenTicketWithReply_alertsAndAutoresponsesEnabled
Testing objectives: MT
- Fixture: #1 (ticket #1 is closed and helpTopicInstallation enables autoresponses)
- Fixture component: #created_tickets
- The staff member generalConsultant logs in.
- The staff member generalConsultant wants to reopen ticket #1 with the comment “The customer is not satisfied with the response”
- The event occurs
  - The ticket is open
  - The ticket thread includes the internal note with title “Ticket status changed to opened” and text “Mary Mayer reopened the ticket on reply”.
  - The ticket thread includes the reply “Ticket solved”
  - An auto-response is sent to the customer
- The staff member generalConsultant wants to close ticket #1 with the reply “Ticket solved”
- The event cannot occur (the ticket is already closed)

S71: ReopenTicketWithReply_alertsAndAutoresponsesDisabled
Testing objectives: MT
- Fixture: #4 (ticket #1 is closed)
- Fixture component: #created_tickets
- The staff member generalConsultant logs in.
- The staff member generalConsultant wants to close ticket #1 with the comment “The customer is not satisfied with the response”
- The event occurs
  - The ticket is open
  - The ticket thread includes the internal note with title “Ticket status changed to opened” and text “Mary Mayer reopened the ticket on reply”.
  - The ticket thread includes the reply “Ticket solved”
  - No auto-response is sent to the customer

S72: ReopenTicketWithReply_ticketIsNotVisible
Testing objectives: MT
- Fixture: #3 (ticket #1 is closed)
- Fixture component: #created_tickets
- The staff member technicalActive logs in.
- The staff member technicalActive wants to reopen ticket #2 with the comment “The customer is not satisfied with the response”
- The event cannot occur (the staff member cannot manage the ticket)

S73: ReopenTicketWithReply_NotLoggedIn
Testing objectives: MT
- Fixture: #3 (ticket #1 is closed)
- Fixture component: #created_tickets
- The staff member generalConsultant wants to reopen ticket #1 with the comment “The customer is not satisfied with the response”
- The event cannot occur (the staff member is not logged in)
**S74: BanTicketCloseEmail**

Testing objectives: MT

- Fixture: #3
- Fixture component: #created_tickets
- The staff member generalConsultant logs in.
- The staff member generalConsultant wants to ban the email of ticket #1 and close the ticket
- The event occurs
  - The email of ticket #1 is banned and the the ticket is closed
  - The ticket thread includes the internal note with title “Ticket closed” and text “Email (mary@marnes.mar) added to banlist & ticket status set to closed”.
- The staff member generalConsultant wants to ban the email & close again the ticket #1
- The event cannot occur (the ticket is already closed)

**S75: BanTicketCloseEmail_ticketIsNotVisible**

Testing objectives: MT

- Fixture: #3 (the technicalActive is allowed to ban emails)
- Fixture component: #created_tickets
- The staff member technicalActive logs in.
- The staff member technicalActive wants to ban the email of ticket #2 and close the ticket
- The event cannot occur (the staff member cannot manage the ticket)

**S76: BanTicketCloseEmail_notAllowed**

Testing objectives: MT

- Fixture: #3
- Fixture component: #created_tickets
- The generalConsultant cannot ban emails
- The staff member generalConsultant logs in.
- The staff member generalConsultant wants to ban the email of ticket #2 and close the ticket
- The event cannot occur (the staff member cannot ban emails of tickets)

**S77: BanTicketCloseEmail_NotLoggedIn**

Testing objectives: MT

- Fixture: #11 (ticket #1 is closed)
- Fixture component: #created_tickets
- The staff member generalConsultant wants to ban the email of ticket #1 and close the ticket
- The event cannot occur (the staff member is not logged in)

**S78: DeleteTicket**

Testing objectives: MT

- Fixture: #3
- Fixture component: #created_tickets
- The staff member generalConsultant logs in.
- The staff member generalConsultant wants to delete ticket #1
- The event occurs
  - The ticket is deleted
S79: DeleteTicket_ticketIsNotVisible
Testing objectives: MT
- Fixture: #3 (the technicalActive is not allowed to delete tickets)
- Fixture component: #created_tickets
- The staff member technicalActive logs in.
- The staff member technicalActive wants to delete ticket #2
- The event cannot occur (the staff member cannot manage the ticket)

S80: DeleteTicket_notAllowed
Testing objectives: MT
- Fixture: #3
- Fixture component: #created_tickets
- The staff member technicalActive logs in.
- The staff member technicalActive wants delete ticket #2
- The event cannot occur (the staff member cannot ban emails of tickets)

S81: DeleteTicket_NotLoggedIn
Testing objectives: MT
- Fixture: #3 (ticket #1 is closed)
- Fixture component: #created_tickets
- The staff member generalConsultant wants delete ticket #1
- The event cannot occur (the staff member is not logged in)

S82: BannedEmailsCannotCreateTickets
Testing objectives: MT
- Fixture: #4
- Fixture component: #created_tickets
- Email ‘hello_at_helloworld.hel’ is in the banlist
- No ticket can be created online, offline or by email

S83: CheckOverdueTickets_staffAlertsDisabled
Testing objectives: MT
- Fixture: #4
- Fixture component: #created_tickets
- SystemDateTime.now() + 3 days
- The system checks the overdue tickets
  - No autoresponses to the customer are sent
  - No alerts are sent to the staff

S84: CheckOverdueTickets_staffAlertsEnabled
Testing objectives: MT
- Fixture: #1
- Fixture component: #created_tickets
- SystemDateTime.now() + 2 days
- The system checks the overdue tickets
  - No autoresponses to the customer are sent
  - No alerts are sent to the staff
S85: ViewTickets open myTickets overdue closed
Testing objectives: VTS

- Fixture: #3
- Fixture component: #created_tickets
- The staff member generalAdministrator logs in
- The staff member generalAdministrator wants to view the open tickets (this staff member can see the tickets of all departments)
  - The result of the query is:

<table>
<thead>
<tr>
<th>Ticket</th>
<th>Date</th>
<th>Subject</th>
<th>Department</th>
<th>Priority</th>
<th>From</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>now()-1</td>
<td>Error operating system</td>
<td>dptTechnical</td>
<td>Normal</td>
<td>Mary Marnes</td>
</tr>
<tr>
<td>2</td>
<td>now()-1</td>
<td>Can I reply a ticket?</td>
<td>dptGeneral</td>
<td>High</td>
<td>John Johnes</td>
</tr>
<tr>
<td>3</td>
<td>now()-1</td>
<td>Error while login</td>
<td>dptTechnical</td>
<td>Low</td>
<td>Martin Pope</td>
</tr>
<tr>
<td>4</td>
<td>now()</td>
<td>Reopening ticket</td>
<td>dptGeneral</td>
<td>Normal</td>
<td>James Jordan</td>
</tr>
<tr>
<td>5</td>
<td>now()</td>
<td>See my tickets</td>
<td>dptTechnical</td>
<td>High</td>
<td><a href="mailto:marta@johnes.mar">marta@johnes.mar</a></td>
</tr>
</tbody>
</table>

- The staff member generalAdministrator wants to view his/her assigned tickets
  - The result of the query is empty
- The staff member generalAdministrator assigns ticket #4 to the generalAdministrator.
- The staff member generalAdministrator wants to view his/her assigned tickets
  - The result of the query is:

<table>
<thead>
<tr>
<th>Ticket</th>
<th>Date</th>
<th>Subject</th>
<th>Department</th>
<th>Priority</th>
<th>From</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>now()</td>
<td>Reopening ticket</td>
<td>dptGeneral</td>
<td>Normal</td>
<td>James Jordan</td>
</tr>
</tbody>
</table>

- The staff member generalAdministrator wants to view the overdue tickets
  - The result of the query is empty
- SystemDateTime.now() + 3 days
- The staff member generalAdministrator marks ticket #4 as overdue
- The staff member generalAdministrator wants to view the overdue tickets
  - The result of the query is:

<table>
<thead>
<tr>
<th>Ticket</th>
<th>Date</th>
<th>Subject</th>
<th>Department</th>
<th>Priority</th>
<th>From</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>now()-1</td>
<td>Error operating system</td>
<td>dptTechnical</td>
<td>Normal</td>
<td>Mary Marnes</td>
</tr>
<tr>
<td>4</td>
<td>now()</td>
<td>Reopening ticket</td>
<td>dptGeneral</td>
<td>Normal</td>
<td>James Jordan</td>
</tr>
</tbody>
</table>

- The staff member generalAdministrator wants to view the closed tickets
  - The result of the query is empty
- The staff member generalAdministrator closes ticket #4
- The staff member generalAdministrator wants to view the overdue tickets
  - The result of the query is:

<table>
<thead>
<tr>
<th>Ticket</th>
<th>Date</th>
<th>Subject</th>
<th>Department</th>
<th>Priority</th>
<th>From</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>now()-1</td>
<td>Error operating system</td>
<td>dptTechnical</td>
<td>Normal</td>
<td>Mary Marnes</td>
</tr>
</tbody>
</table>

- The staff member generalAdministrator wants to view the closed tickets
  - The result of the query is:

<table>
<thead>
<tr>
<th>Ticket</th>
<th>Date</th>
<th>Subject</th>
<th>Department</th>
<th>Priority</th>
<th>From</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>now()</td>
<td>Reopening ticket</td>
<td>dptGeneral</td>
<td>Normal</td>
<td>James Jordan</td>
</tr>
</tbody>
</table>
5. TDCM application to the case study

In this section, we report the results of the TDCM application for the purpose of developing the conceptual schema of the osTicket system, according to the testing strategy defined in Section 2.

We report the following information for each iteration:

- **Iteration objective**: The objective of the current iteration.

- **Current test case**: This is the test case that initiates the TDCM iteration.

- **TDCM application. Summary of changes performed in the schema.** The execution of the current test case (and the previous test cases as regression tests) provides failure and error information that drives changes in the schema. These changes are summarized in order to explain the evolution of the schema by applying TDCM.

- **Time spent**: We register the minutes spent in order to specify the initial test case and the minutes to complete the iteration.

- **Errors and failures**: We categorize the types of errors and failures revealed during the iteration as a result of the application of TDCM.

We also present the resultant conceptual schema after the development of the Configuration and basics part, and after the development of the Tickets management and tracking part.

The USEx specification of the whole resultant conceptual schema may be found in appendix A (conceptual schema in UML/OCL) and appendix B (methods defined in CSTL).

The test cases after TDCM application are collected in appendix C.
Iteration 1

Iteration objective

Email templates

Current test case

testprogram ConfigurationAndBasics{

fixturecomponent ConfigurationAndBasics{
    template_default:=new EmailTemplate(name:='Default');
    template_default.internalNotes:='Email templates by default';

    ek1.emailTemplate:=template_default;
    template_default.newTicketAutoresponse.subject:='X';
    template_default.newTicketAutoresponse.message:='Y';

    ek2:=new NewMessageAutoresponse(subject:='X',message:='Y');
    ek2.emailTemplate:=template_default;
    template_default.newMessageAutoresponse.subject:='X';
    template_default.newMessageAutoresponse.message:='Y';

    ek3.emailTemplate:=template_default;
    template_default.newTicketNotice.subject:='X';
    template_default.newTicketNotice.message:='Y';

    ek4:=new OverTicketLimitNotice(subject:='X',message:='Y');
    ek4.emailTemplate:=template_default;
    template_default.overTicketLimitNotice.subject:='X';
    template_default.overTicketLimitNotice.message:='Y';

    ek5:=new TicketResponseNotice(subject:='X',message:='Y');
    ek5.emailTemplate:=template_default;
    template_default.ticketResponseNotice.subject:='X';
    template_default.ticketResponseNotice.message:='Y';

    ek6:=new NewTicketAlertToStaff(subject:='X',message:='Y');
    ek6.emailTemplate:=template_default;
    template_default.newTicketAlertToStaff.subject:='X';
    template_default.newTicketAlertToStaff.message:='Y';

    ek7:=new NewMessageAlertToStaff(subject:='X',message:='Y');
    ek7.emailTemplate:=template_default;
    template_default.newMessageAlertToStaff.subject:='X';
    template_default.newMessageAlertToStaff.message:='Y';

    ek8:=new NewInternalNoteAlertToStaff(subject:='X',message:='Y');
    ek8.emailTemplate:=template_default;
    template_default.newInternalNoteAlertToStaff.subject:='X';
    template_default.newInternalNoteAlertToStaff.message:='Y';

    ek9:=new TicketAssignedAlertToStaff(subject:='X',message:='Y');
    ek9.emailTemplate:=template_default;
    template_default.ticketAssignedAlertToStaff.subject:='X';
    template_default.ticketAssignedAlertToStaff.message:='Y';

    ek10:=new OverdueTicketAlertToStaff(subject:='X',message:='Y');
    ek10.emailTemplate:=template_default;
    template_default.overdueTicketAlertToStaff.subject:='X';
    template_default.overdueTicketAlertToStaff.message:='Y';
}
TDCM application: Summary of changes performed in the schema

- **Added**

  class EmailTemplate
  attributes
  name:String[1]
  internalNotes:String[1]
end

class EmailKind
attributes
subject:String
message:String
end

class NewTicketAutoresponse<EmailKind
end
association newTicketAutoresponse_emailTemplate between
  NewTicketAutoresponse[1]
  EmailTemplate[1]
end

class NewMessageAutoresponse<EmailKind
end
association newMessageAutoresponse_emailTemplate between
  NewMessageAutoresponse[1]
  EmailTemplate[1]
end

class NewTicketNotice<EmailKind
end
association newTicketNotice_emailTemplate between
  NewTicketNotice[1]
  EmailTemplate[1]
end

class OverTicketLimitNotice<EmailKind
end
association overTicketLimitNotice_emailTemplate between
  OverTicketLimitNotice[1]
  EmailTemplate[1]
end

class TicketResponseNotice<EmailKind
end
association ticketResponseNotice_emailTemplate between
  TicketResponseNotice[1]
  EmailTemplate[1]
end

class NewTicketAlertToStaff<EmailKind
end
DEVELOPMENT OF THE CONCEPTUAL SCHEMA OF THE osTICKET SYSTEM BY APPLYING TDCM

Research report
April 2011

Time spent

<table>
<thead>
<tr>
<th>TIME TO WRITE TEST CASES (IN MINUTES)</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>TIME TO COMPLETE THE ITERATION (IN MINUTES)</td>
<td>7</td>
</tr>
</tbody>
</table>

Errors and failures that drive the conceptual modeling

<table>
<thead>
<tr>
<th>A basic type involved in a test case does not exist in the CSUD</th>
<th>A derived type involved in a test case does not exist in the CSUD</th>
<th>An event type involved in a test case does not exist in the CSUD</th>
</tr>
</thead>
<tbody>
<tr>
<td>The basic type is relevant and it is added to the CSUD</td>
<td>The derived type is relevant and it is added to the CSUD</td>
<td>The event type is relevant and it is added to the CSUD</td>
</tr>
<tr>
<td>Inconsistent state before the occurrence of an event</td>
<td>Inconsistent state after the occurrence of an event</td>
<td>The postcondition of an event is not satisfied.</td>
</tr>
<tr>
<td>Some static constraint is invalid and it is modified.</td>
<td>Some initial integrity constraint is invalid and it is modified.</td>
<td>The event postcondition/method is incorrect and it is modified.</td>
</tr>
<tr>
<td>The event postcondition/method is incorrect and it is modified.</td>
<td>Some constraint is invalid and it is modified.</td>
<td>The method is not correct and it is modified.</td>
</tr>
<tr>
<td>The postcondition is not satisfied.</td>
<td>The expression is corrected.</td>
<td>The CSUD is changed.</td>
</tr>
</tbody>
</table>

Iteration 2

Iteration objective

Departments and Email Accounts (and their dependencies)

Current test case

testprogram ConfigurationAndBasics{

fixturecomponent CompatibleConfigurationAndBasics{

...}

dptGeneral := new Department(name:="General support");
dptGeneral.type := #Public;
dptGeneral.emailTemplate := template_default;
dptGeneral.newTicketAutoresponsesSent := true;
dptGeneral.newAddedMessagesNotified := true;

dptTechnical := new Department(name:="Technical support");
dptTechnical.type := #Private;
dptTechnical.emailTemplate := template_default;
dptTechnical.newTicketAutoresponsesSent := true;
dptTechnical.newAddedMessagesNotified := true;

generalSupportEmailAccount:=new EmailAccount(address:='general@support.com');
generalSupportEmailAccount.fromName:='General questions';
generalSupportEmailAccount.defaultNewPriority:=#Low;
generalSupportEmailAccount.defaultNewTicketDepartment:=dptGeneral;
generalSupportEmailAccount.autoresponsesStatus:=#Disabled;


DEVELOPMENT OF THE CONCEPTUAL SCHEMA OF THE osTICKET SYSTEM BY APPLYING TDCM

Technical Support Email Account

technicalSupportEmailAccount:=new EmailAccount(address:='technical@support.com');
technicalSupportEmailAccount.fromName:='B Support';
technicalSupportEmailAccount.defaultNewPriority:=#High;
technicalSupportEmailAccount.defaultNewTicketDepartment:=dptTechnical;
technicalSupportEmailAccount.autoresponsesStatus:=#Disabled;

dptGeneral.outgoingEmail:=generalSupportEmailAccount;
dptGeneral.autoresponseEmail:=generalSupportEmailAccount;

dptTechnical.outgoingEmail:=technicalSupportEmailAccount;
dptTechnical.autoresponseEmail:=technicalSupportEmailAccount;

generalAdministrator:=new StaffMember(username:='john');
generalAdministrator.department:=dptGeneral;
generalAdministrator.firstName:='John';
generalAdministrator.lastName:='Johny';
generalAdministrator.emailAddress:='john@support.com';
generalAdministrator.officePhone:='111111';
generalAdministrator.phoneExtension:='11';
generalAdministrator.mobilePhone:='111111';
generalAdministrator.signature:='John Johny';
generalAdministrator.password:='xxx';
generalAdministrator.status:=#Enabled;
generalAdministrator.isAdministrator:=true;
generalAdministrator.isInVacationMode:=false;

dptTechnical.departmentManager:=generalAdministrator;

maximumPrivilegesGroup:=new StaffGroup(name:='Maximum Privileges Group');
maximumPrivilegesGroup.status:=#Enabled;
maximumPrivilegesGroup.departmentsAccess:=Set{dptGeneral,dptTechnical};
maximumPrivilegesGroup.canCreateTickets:=true;
maximumPrivilegesGroup.canEditTickets:=true;
maximumPrivilegesGroup.canCloseTickets:=true;
maximumPrivilegesGroup.canTransferTickets:=true;
maximumPrivilegesGroup.canDeleteTickets:=true;
maximumPrivilegesGroup.canBanEmails:=true;

generalAdministrator.staffGroup:=maximumPrivilegesGroup;
}

TDCM application: Summary of changes performed in the schema

- Added

enum DepartmentType{Public, Private}
enum Priority{Low, High}
enum Status{Enabled, Disabled}

...
EmailTemplate[1]
end

association department_departmentManager between
   Department[*] role departmentOfManager
   StaffMember[0..1] role departmentManager
end

association department_autoresponseEmail between
   Department[*] role departmentOfAutoresponseEmail
   EmailAccount[1] role autoresponseEmail
end

association department_outgoingEmail between
   Department[*] role departmentOfAutoresponseEmail
   EmailAccount[1] role outgoingEmail
end

class EmailAccount
   attributes
   address:String
   fromName:String
   defaultNewPriority:Priority
   autoresponsesStatus:Status
end

association EmailAccount_defaultNewTicketDepartment between
   EmailAccount[*]
   Department[1] role defaultNewTicketDepartment
end

class StaffMember
   attributes
   username:String
   firstName:String
   lastName:String
   emailAddress:String
   officePhone:String
   phoneExtension:String
   mobilePhone:String
   signature:String
   password:String
   status:Status
   isAdministrator:Boolean
   isInVacationMode:Boolean
end

association staffMember_department between
   StaffMember[*]
   Department[1]
end

association staffMember_staffGroup between
   StaffMember[*]
   StaffGroup[1]
end

class StaffGroup
   attributes
   name:String
   status:Status
   canCreateTickets:Boolean
   canEditTickets:Boolean
   canCloseTickets:Boolean
   canTransferTickets:Boolean
   canDeleteTickets:Boolean
   canBanEmails:Boolean
end
The multiplicity of some association ends and attributes has been changed thanks to the following failure information:

- The state is inconsistent: Multiplicity constraint violation in association 'department_departmentManager': Object 'oid12' of class 'Department' is connected to 0 objects of class 'StaffMember' but the multiplicity is specified as '1'.
- The state is inconsistent: Instances of Department violate the invariant minMultiplicityOfAttributeAutoresponseEmail

**Time spent**

| TIME TO WRITE TEST CASES (IN MINUTES) | 18 |
| TIME TO COMPLETE THE ITERATION (IN MINUTES) | 27 |

**Errors and failures that drive the conceptual modeling**

<table>
<thead>
<tr>
<th>A basic type involved in a test case does not exist in the CSUD</th>
<th>An event type involved in a test case does not exist in the CSUD</th>
<th>A derived type involved in a test case does not exist in the CSUD</th>
</tr>
</thead>
<tbody>
<tr>
<td>The basic type is relevant and it is added to the CSUD</td>
<td>The event type is relevant and it is added to the CSUD</td>
<td>The derived type is relevant and it is added to the CSUD</td>
</tr>
</tbody>
</table>

**Inconsistent state before the occurrence of an event**

<table>
<thead>
<tr>
<th>Some static constraint is invalid and it is modified.</th>
<th>Some initial integrity constraint is invalid and it is modified.</th>
<th>Some constraint is invalid and it is modified.</th>
</tr>
</thead>
<tbody>
<tr>
<td>The event postcondition/method is incorrect and it is modified.</td>
<td>The event postcondition/method is not correct and it is modified.</td>
<td>The event postcondition is not correct and it is modified.</td>
</tr>
</tbody>
</table>

**An assertion about the IB state fails or contains an error**

<table>
<thead>
<tr>
<th>The effect of an event type is not correct</th>
<th>A derivation rule is incorrect</th>
<th>A precondition is added/updated</th>
</tr>
</thead>
<tbody>
<tr>
<td>Assert non-occurrence fails</td>
<td>Assert non-occurrence fails</td>
<td>The expression is corrected</td>
</tr>
</tbody>
</table>

**Semantic error in an expression**

<table>
<thead>
<tr>
<th>The expression is corrected</th>
<th>The CSUD is changed</th>
</tr>
</thead>
</table>

**Assert consistency fails**

<table>
<thead>
<tr>
<th>A static constraint needs to be changed</th>
</tr>
</thead>
</table>

2

---

**Iteration 3**

**Iteration objective**

Other staff members and staff groups
Current test case

testprogram ConfigurationAndBasics{
fixturecomponent CompatibleConfigurationAndBasics{

    minimumPrivilegesGroup:=new StaffGroup(name:='Minimum Privileges Group');
    minimumPrivilegesGroup.status:=#Enabled;
    minimumPrivilegesGroup.departmentsAccess:=Set{};
    minimumPrivilegesGroup.canCreateTickets:=false;
    minimumPrivilegesGroup.canEditTickets:=false;
    minimumPrivilegesGroup.canCloseTickets:=false;
    minimumPrivilegesGroup.canTransferTickets:=false;
    minimumPrivilegesGroup.canDeleteTickets:=false;
    minimumPrivilegesGroup.canBanEmails:=false;

    inactiveGroup:=new StaffGroup(name:='Inactive Group');
    inactiveGroup.status:=#Disabled;
    inactiveGroup.departmentsAccess:=Set{};
    inactiveGroup.canCreateTickets:=true;
    inactiveGroup.canEditTickets:=false;
    inactiveGroup.canCloseTickets:=true;
    inactiveGroup.canTransferTickets:=false;
    inactiveGroup.canDeleteTickets:=true;
    inactiveGroup.canBanEmails:=false;

    generalConsultant:=new StaffMember(username:='mary');
    generalConsultant.department:=dptGeneral;
    generalConsultant.firstName:='Mary';
    generalConsultant.lastName:='Mayer';
    generalConsultant.emailAdress:='mary@support.com';
    generalConsultant.officePhone:='22222';
    generalConsultant.phoneExtension:='22';
    generalConsultant.mobilePhone:='22222';
    generalConsultant.signature:='Mary Mayer';
    generalConsultant.password:='yyy';
    generalConsultant.status:=#Enabled;
    generalConsultant.isAdministrator:=false;
    generalConsultant.isInVacationMode:=false;
    generalConsultant.staffGroup:=maximumPrivilegesGroup;

    generalConsultantVacation:=new StaffMember(username:='david');
    generalConsultantVacation.department:=dptGeneral;
    generalConsultantVacation.firstName:='David';
    generalConsultantVacation.lastName:='Dassel';
    generalConsultantVacation.emailAdress:='david@support.com';
    generalConsultantVacation.officePhone:='33333';
    generalConsultantVacation.phoneExtension:='33';
    generalConsultantVacation.mobilePhone:='33333';
    generalConsultantVacation.signature:='David Dassel';
    generalConsultantVacation.password:='zzz';
    generalConsultantVacation.status:=#Enabled;
    generalConsultantVacation.isAdministrator:=false;
    generalConsultantVacation.isInVacationMode:=true;
    generalConsultantVacation.staffGroup:=maximumPrivilegesGroup;

    technicalActive:=new StaffMember(username:='martin');
    technicalActive.department:=dptTechnical;
    technicalActive.firstName:='Martin';
    technicalActive.lastName:='Martech';
    technicalActive.emailAdress:='martin@support.com';
    technicalActive.password:='ttt';
    technicalActive.status:=#Enabled;
    technicalActive.isAdministrator:=false;
    technicalActive.isInVacationMode:=false;
    technicalActive.staffGroup:=minimumPrivilegesGroup;

    ...
TDCM application: Summary of changes performed in the schema

- Updated

```
class StaffMember
attributes
  username: String
  firstName: String
  lastName: String
  emailAddress: String
  officePhone: String[0..1]
  phoneExtension: String[0..1]
  mobilePhone: String[0..1]
  signature: String[0..1]
  password: String
  status: Status
  isAdministrator: Boolean
  isInVacationMode: Boolean
end
```

The multiplicity of the attributes StaffMember::mobilePhone, StaffMember::officePhone, StaffMember::phoneExtension, StaffMember::signature have been changed as a response to the following failure information:

- The state is inconsistent: Instances of StaffMember violate the invariant minMultiplicityOfAttributeMobilePhone
- The state is inconsistent: Instances of StaffMember violate the invariant minMultiplicityOfAttributeOfficePhone
- The state is inconsistent: Instances of StaffMember violate the invariant minMultiplicityOfAttributePhoneExtension
- The state is inconsistent: Instances of StaffMember violate the invariant minMultiplicityOfAttributeSignature

**Time spent**

<table>
<thead>
<tr>
<th>TIME TO WRITE TEST CASES (IN MINUTES)</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>TIME TO COMPLETE THE ITERATION (IN MINUTES)</td>
<td>3</td>
</tr>
</tbody>
</table>

**Errors and failures that drive the conceptual modeling**
A basic type involved in a test case does not exist in the CSUD | A derived type involved in a test case does not exist in the CSUD | An event type involved in a test case does not exist in the CSUD
---|---|---
The basic type is relevant and it is added to the CSUD | The derived type is relevant and it is added to the CSUD | The event type is relevant and it is added to the CSUD

Inconsistent state before the occurrence of an event | Inconsistent state after the occurrence of an event | The postcondition of an event is not satisfied.
---|---|---
Some static constraint is invalid and it is modified. | Some initial integrity constraint is invalid and it is modified. | The event precondition/method is incorrect and it is modified. | The method is not correct and it is modified. | The postcondition is not correct and it is modified.

An assertion about the IB state fails or contains an error | Assert non-occurrence fails | Semantic error in an expression
---|---|---
The effect of an event type is not correct | A derivation rule is incorrect | A precondition is added/updated | The expression is corrected | The CSUD is changed

---

**Iteration 4**

**Iteration objective**

Email settings

**Current test case**

testprogram ConfigurationAndBasics{
  fixturecomponent CompatibleConfigurationAndBasics{
...    emailSettings:=new EmailSettings;
    emailSettings.defaultSystemEmail:=generalSupportEmailAccount;
    emailSettings.defaultStaffAlertsEmail:=generalSupportEmailAccount;
    emailSettings.administrationEmail:='system@support.com';
  }
  test testConfiguration1{
    load CompatibleConfigurationAndBasics;
    assert consistency;
  }
}

**TDCM application: Summary of changes performed in the schema**

- Added

  class EmailSettings
  attributes
    administrationEmail: String
  end

  association emailSettings_emailAccount between
Time spent

<table>
<thead>
<tr>
<th>TIME TO WRITE TEST CASES (IN MINUTES)</th>
<th>0.6</th>
</tr>
</thead>
<tbody>
<tr>
<td>TIME TO COMPLETE THE ITERATION (IN MINUTES)</td>
<td>2</td>
</tr>
</tbody>
</table>

Errors and failures that drive the conceptual modeling

<table>
<thead>
<tr>
<th>A basic type involved in a test case does not exist in the CSUD</th>
<th>A derived type involved in a test case does not exist in the CSUD</th>
<th>An event type involved in a test case does not exist in the CSUD</th>
</tr>
</thead>
<tbody>
<tr>
<td>The basic type is relevant and it is added to the CSUD</td>
<td>The derived type is relevant and it is added to the CSUD</td>
<td>The event type is relevant and it is added to the CSUD</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Inconsistent state before the occurrence of an event</th>
<th>Inconsistent state after the occurrence of an event</th>
<th>The postcondition of an event is not satisfied.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Some static constraint is invalid and it is modified.</td>
<td>Some initial integrity constraint is invalid and it is modified.</td>
<td>The event postcondition/method is incorrect and it is modified.</td>
</tr>
<tr>
<td>The event postcondition/method is incorrect and it is modified.</td>
<td>Some constraint is invalid and it is modified.</td>
<td>The method is not correct and it is modified.</td>
</tr>
<tr>
<td>The postcondition is not correct and it is modified.</td>
<td></td>
<td>The postcondition is not correct and it is modified.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>An assertion about the IB state fails or contains an error</th>
<th>Assert non-occurrence fails</th>
<th>Semantic error in an expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>The effect of an event type is not correct</td>
<td>A derivation rule is incorrect</td>
<td>The expression is corrected</td>
</tr>
<tr>
<td>A precondition is added/updated</td>
<td>The CSUD is changed</td>
<td></td>
</tr>
</tbody>
</table>

Assert consistency fails

A static constraint needs to be changed
Current test case

testprogram ConfigurationAndBasics{
    fixturecomponent CompatibleConfigurationAndBasics{
        ...
        helpTopicUse:=new HelpTopic(name:='Use');
        helpTopicUse.status:=#Enabled;
        helpTopicUse.autoresponse:=#Enabled;
        helpTopicUse.newTicketPriority:=#Normal;
        helpTopicUse.newTicketDepartment:=dptGeneral;
        helpTopicInstallation:=new HelpTopic(name:='Installation');
        helpTopicInstallation.status:=#Enabled;
        helpTopicInstallation.autoresponse:=#Disabled;
        helpTopicInstallation.newTicketPriority:=#High;
        helpTopicInstallation.newTicketDepartment:=dptTechnical;
        helpTopicDisabled:=new HelpTopic(name:='Offers');
        helpTopicDisabled.status:=#Disabled;
        helpTopicDisabled.autoresponse:=#Disabled;
        helpTopicDisabled.newTicketPriority:=#Low;
        helpTopicDisabled.newTicketDepartment:=dptGeneral;
    }
    test testConfiguration1{
        load CompatibleConfigurationAndBasics;
        assert consistency;
    }
}

TDCM application: Summary of changes performed in the schema

- Added
  enum Priority{Low,Normal,High}

  class HelpTopic
  attributes
  name:String
  status:Status
  autoresponse:Status
  newTicketPriority:Priority
  end

  association helpTopic_newTicketDepartment between
      HelpTopic[*]
      Department[1] role newTicketDepartment
  End

The type of the attribute HelpTopic::autoresponse has been changed from Boolean to Status in order to solve the following error:

- [ConfigurationAndBasics.cstl] <line 199> Incompatible types: Enumeration value expression cannot be assigned to the property oid24.autoresponse

A new enumeration literal is defined for the enumeration Priority, because this user story reveals that it is relevant for the domain:

Time spent

<table>
<thead>
<tr>
<th>TIME TO WRITE TEST CASES (IN MINUTES)</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>TIME TO COMPLETE THE ITERATION (IN MINUTES)</td>
<td>3.5</td>
</tr>
</tbody>
</table>

Errors and failures that drive the conceptual modeling

<table>
<thead>
<tr>
<th>A basic type involved in a test case does not exist in the CSUD</th>
<th>A derived type involved in a test case does not exist in the CSUD</th>
<th>An event type involved in a test case does not exist in the CSUD</th>
</tr>
</thead>
<tbody>
<tr>
<td>The basic type is relevant and it is added to the CSUD</td>
<td>The derived type is relevant and it is added to the CSUD</td>
<td>The event type is relevant and it is added to the CSUD</td>
</tr>
<tr>
<td>Inconsistent state before the occurrence of an event</td>
<td>Inconsistent state after the occurrence of an event</td>
<td>The postcondition of an event is not satisfied.</td>
</tr>
<tr>
<td>Some static constraint is invalid and it is modified.</td>
<td>Some initial integrity constraint is invalid and it is modified.</td>
<td>The event postcondition/method is incorrect and it is modified.</td>
</tr>
<tr>
<td>Some constraint is invalid and it is modified.</td>
<td>The event postcondition/method is incorrect and it is modified.</td>
<td>Some constraint is invalid and it is modified.</td>
</tr>
<tr>
<td>The effect of an event type is not correct</td>
<td>A derivation rule is incorrect</td>
<td>The expression is corrected</td>
</tr>
<tr>
<td>Assert non-occurrence fails</td>
<td>A precondition is added/updated</td>
<td>The method is not correct and it is modified.</td>
</tr>
<tr>
<td>Assert consistency fails</td>
<td></td>
<td>The postcondition is not correct and it is modified.</td>
</tr>
<tr>
<td>A static constraint needs to be changed</td>
<td></td>
<td>1</td>
</tr>
</tbody>
</table>

Iteration 6

Iteration objective

General settings online

Current test case

```plaintext
testprogram ConfigurationAndBasics{
...

fixturecomponent GeneralSettingsOnline{
    generalSettings:=new GeneralSettings;
    generalSettings.status:=#Online;
    generalSettings.helpdeskURL:='http://onlinesupport.com';
    generalSettings.helpdeskName:='Online customer support';
    generalSettings.defaultEmailTemplate:=template_default;
}

test testConfiguration1{
    load CompatibleConfigurationAndBasics;
    load GeneralSettingsOnline;
    assert consistency;
}
}
```
TDCM application: Summary of changes performed in the schema

- Added

```java
enum HelpDeskStatus { Online }

class GeneralSettings
    attributes
        status: HelpDeskStatus
        helpdeskURL: String
        helpdeskName: String
    end

association generalSettings_defaultEmailTemplate between
    GeneralSettings[*]
        EmailTemplate[1] role defaultEmailTemplate
    end

context GeneralSettings inv hasOnlyOneInstance:
    GeneralSettings.allInstances().size() = 1
```

Time spent

| TIME TO WRITE TEST CASES (IN MINUTES) | 3 |
| TIME TO COMPLETE THE ITERATION (IN MINUTES) | 5 |

Errors and failures that drive the conceptual modeling

<table>
<thead>
<tr>
<th>A basic type involved in a test case does not exist in the CSUD</th>
<th>A derived type involved in a test case does not exist in the CSUD</th>
<th>An event type involved in a test case does not exist in the CSUD</th>
</tr>
</thead>
<tbody>
<tr>
<td>The basic type is relevant and it is added to the CSUD</td>
<td>The derived type is relevant and it is added to the CSUD</td>
<td>The event type is relevant and it is added to the CSUD</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Inconsistent state before the occurrence of an event</th>
<th>Inconsistent state after the occurrence of an event</th>
<th>The postcondition of an event is not satisfied.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Some static constraint is invalid and it is modified.</td>
<td>Some initial integrity constraint is invalid and it is modified.</td>
<td>The event postcondition/method is incorrect and it is modified.</td>
</tr>
<tr>
<td>The event postcondition/method is incorrect and it is modified.</td>
<td>Some constraint is invalid and it is modified.</td>
<td>The method is not correct and it is modified.</td>
</tr>
<tr>
<td>Some constraint is invalid and it is modified.</td>
<td>The postcondition is not correct and it is modified.</td>
<td>The postcondition is not correct and it is modified.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>An assertion about the IB state fails or contains an error</th>
<th>Assert non-occurrence fails</th>
<th>Semantic error in an expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>The effect of an event type is not correct</td>
<td>A derivation rule is incorrect</td>
<td>A precondition is added/updated</td>
</tr>
<tr>
<td>A precondition is added/updated</td>
<td>The expression is corrected</td>
<td>The CSUD is changed</td>
</tr>
<tr>
<td>Assert consistency fails</td>
<td>A static constraint needs to be changed</td>
<td></td>
</tr>
</tbody>
</table>

5
Iteration 7

Iteration objective

General settings offline

Current test case

testprogram ConfigurationAndBasics{
...

fixturecomponent GeneralSettingsOffline{
    generalSettings:=new GeneralSettings;
    generalSettings.status:=#Offline;
    generalSettings.defaultEmailTemplate:=template_default;
}

test testConfiguration1{
    load CompatibleConfigurationAndBasics;
    load GeneralSettingsOnline;
    assert consistency;
}

test testConfiguration9{
    load CompatibleConfigurationAndBasics;
    load GeneralSettingsOffline;
    assert consistency;
}
}

TDCM application: Summary of changes performed in the schema

- Added
- Updated

definition enum HelpDeskStatus{Online,Offline}
definition class GeneralSettings
    attributes
        status:HelpDeskStatus
        helpdeskName:String[0..1]
        helpdeskURL:String
end

The multiplicity of attributes GeneralSettings::helpdeskName and GeneralSettings::helpdeskURL is changed according the following failures:

- The state is inconsistent: Instances of GeneralSettings violate the invariant minMultiplicityOfAttributeHelpdeskName

Time spent

| Time to write test cases (in minutes) | 2.5 |
| Time to complete the iteration (in minutes) | 1 |
Errors and failures that drive the conceptual modeling

<table>
<thead>
<tr>
<th>Error Type</th>
<th>CSUD Impact</th>
</tr>
</thead>
<tbody>
<tr>
<td>A basic type involved in a test case does not exist in the CSUD</td>
<td>The basic type is relevant and it is added to the CSUD</td>
</tr>
<tr>
<td>A derived type involved in a test case does not exist in the CSUD</td>
<td>The derived type is relevant and it is added to the CSUD</td>
</tr>
<tr>
<td>An event type involved in a test case does not exist in the CSUD</td>
<td>The event type is relevant and it is added to the CSUD</td>
</tr>
<tr>
<td>Inconsistent state before the occurrence of an event</td>
<td>Some static constraint is invalid and it is modified.</td>
</tr>
<tr>
<td>Inconsistent state after the occurrence of an event</td>
<td>Some initial integrity constraint is invalid and it is modified.</td>
</tr>
<tr>
<td>The postcondition of an event is not satisfied.</td>
<td>The event postcondition/method is incorrect and it is modified.</td>
</tr>
<tr>
<td>An assertion about the IB state fails or contains an error</td>
<td>Semantic error in an expression</td>
</tr>
<tr>
<td>The effect of an event type is not correct</td>
<td>A derivation rule is incorrect</td>
</tr>
<tr>
<td>A precondition is added/updated</td>
<td>The expression is corrected</td>
</tr>
<tr>
<td>The postcondition is not correct and it is modified.</td>
<td>The CSUD is changed</td>
</tr>
</tbody>
</table>

Iteration 8

Iteration objective

Ticket settings sequential (in combination with existing compatible configuration and basics)

Current test case

testprogram ConfigurationAndBasics{
...

fixturecomponent TicketSettingsSequential{
  ticketSettings:=new TicketSettings;
  ticketSettings.mode:=#Sequential;
  ticketSettings.priority:=#Normal;
  ticketSettings.customersCanChangePriority:=false;
  ticketSettings.useEmailPriorityWhenAvailable:=true;
  ticketSettings.maximumOpenTicketsPerMail:=2;
  ticketSettings.ticketGracePeriod:=0;
  ticketSettings.reopenedTicketsAreAssignedToLastRespondent:=true;
}

test testConfiguration1{
  load CompatibleConfigurationAndBasics;
  load GeneralSettingsOnline;
  load TicketSettingsSequential;
  assert consistency;
}

test testConfiguration9{
  load CompatibleConfigurationAndBasics;
  load GeneralSettingsOffline;
  load TicketSettingsSequential;
  assert consistency;
}
TDCM application: Summary of changes performed in the schema

- **Added**
- **Updated**

```cpp
enum TicketsMode { Sequential }

class TicketSettings {
    attributes
    mode: TicketsMode
    priority: Priority
    customersCanChangePriority: Boolean
    useEmailPriorityWhenAvailable: Boolean
    maximumOpenTicketsPerMail: Integer
    ticketGracePeriod: Integer
    reopenedTicketsAreAssignedToLastRespondent: Boolean
}

context TicketSettings inv hasOnlyOneInstance:
    TicketSettings.allInstances().size() = 1
```

Time spent

<table>
<thead>
<tr>
<th>Time to write test cases (in minutes)</th>
<th>3.5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time to complete the iteration (in minutes)</td>
<td>5</td>
</tr>
</tbody>
</table>

Errors and failures that drive the conceptual modeling

<table>
<thead>
<tr>
<th>A basic type involved in a test case does not exist in the CSUD</th>
<th>A derived type involved in a test case does not exist in the CSUD</th>
<th>An event type involved in a test case does not exist in the CSUD</th>
</tr>
</thead>
<tbody>
<tr>
<td>The basic type is relevant and it is added to the CSUD</td>
<td>The derived type is relevant and it is added to the CSUD</td>
<td>The event type is relevant and it is added to the CSUD</td>
</tr>
</tbody>
</table>

9

- Inconsistent state before the occurrence of an event
  
  - Some static constraint is invalid and it is modified.
  - Some initial integrity constraint is invalid and it is modified.
  - The event postcondition/method is incorrect and it is modified.
  - Some constraint is invalid and it is modified.
  - The method is not correct and it is modified.

- Inconsistent state after the occurrence of an event
  
  - Some initial integrity constraint is invalid and it is modified.
  - The event postcondition/method is incorrect and it is modified.
  - Some constraint is invalid and it is modified.
  - The method is not correct and it is modified.

- The postcondition of an event is not satisfied.

- The method is not correct and it is modified.

- The expression is corrected.

- The CSUD is changed.

Iteration 9

Iteration objective

Ticket settings random (in combination with existing compatible configuration and basics)
Current test case

testprogram ConfigurationAndBasics{

fixturecomponent TicketSettingsRandom{
ticketSettings := new TicketSettings;
ticketSettings.mode := #Random;
ticketSettings.priority := #High;
ticketSettings.customersCanChangePriority := true;
ticketSettings.useEmailPriorityWhenAvailable := false;
//ticketSettings.maximumOpenTicketsPerMail := #Unlimited;
ticketSettings.openTicketsPerMailAreLimited := false;
ticketSettings.ticketGracePeriod := 2;
ticketSettings.reopenedTicketsAreAssignedToLastRespondent := false;
}

test testConfiguration1{
    load CompatibleConfigurationAndBasics;
    load GeneralSettingsOnline;
    load TicketSettingsSequential;
    assert consistency;
}

test testConfiguration5{
    load CompatibleConfigurationAndBasics;
    load GeneralSettingsOnline;
    load TicketSettingsRandom;
    assert consistency;
}

test testConfiguration9{
    load CompatibleConfigurationAndBasics;
    load GeneralSettingsOffline;
    load TicketSettingsSequential;
    assert consistency;
}

test testConfiguration13{
    load CompatibleConfigurationAndBasics;
    load GeneralSettingsOffline;
    load TicketSettingsRandom;
    assert consistency;
}
}

TDCM application: Summary of changes performed in the schema

- Added
- Updated

We add the following enumeration value:

enum TicketsMode {Sequential, Random}

Initially, the current test case specifies the line:

ticketSettings.maximumOpenTicketsPerMail := #Unlimited;

The test case execution reveals that this attribute was a number in the previous test cases. However, the test case expects to indicate that, in this case, there is no maximum of open tickets per mail. In order to deal with the two expectations, we modify the CSUD as follows:
When we execute again, all test cases (the current one and the previous test cases) fail, with the following failure:

- The state is inconsistent: Instances of TicketSettings violate the invariant 
  minMultiplicityOfAttributeOpenTicketsPerMailAreLimited

This failure reveals that there is an error in the CSUD: the multiplicity of the attribute
maximumOpenTicketsPerMail should be 0..1

After this change, the test cases testConfiguration1 and testConfiguration9 (those that limit
the number of tickets per mail) also fail (they are not consistent) because they do not specify a
value for the attribute openTicketsPerMailAreLimited. We set the default value for the attribute
openTicketsPerMailAreLimited to false.

<table>
<thead>
<tr>
<th>TIME TO WRITE TEST CASES (IN MINUTES)</th>
<th>2,5</th>
</tr>
</thead>
<tbody>
<tr>
<td>TIME TO COMPLETE THE ITERATION (IN MINUTES)</td>
<td>7,5</td>
</tr>
</tbody>
</table>
Errors and failures that drive the conceptual modeling

<table>
<thead>
<tr>
<th>Error Type</th>
<th>CSUD Implication</th>
</tr>
</thead>
<tbody>
<tr>
<td>A basic type involved in a test case does not exist in the CSUD</td>
<td>The basic type is relevant and it is added to the CSUD</td>
</tr>
<tr>
<td>A derived type involved in a test case does not exist in the CSUD</td>
<td>The derived type is relevant and it is added to the CSUD</td>
</tr>
<tr>
<td>An event type involved in a test case does not exist in the CSUD</td>
<td>The event type is relevant and it is added to the CSUD</td>
</tr>
</tbody>
</table>

| Inconsistent state before the occurrence of an event                      | The event postcondition/method is incorrect and it is modified.                  |
| Inconsistent state after the occurrence of an event                       | The method is not correct and it is modified.                                    |
| The postcondition of an event is not satisfied                            | The postcondition is not correct and it is modified.                             |

| An assertion about the IB state fails or contains an error                | Assert non-occurrence fails                                                      |
| Semantic error in an expression                                           |                                                                                   |

| The effect of an event type is not correct                               | A derivation rule is incorrect                                                    |
| A derivation rule is incorrect                                            | A precondition is added/updated                                                   |
| The expression is corrected                                              | The expression is corrected                                                      |
| The CSUD is changed                                                      |                                                                                   |

| Assert consistency fails                                                  |                                                                                   |
| A static constraint needs to be changed                                   |                                                                                   |

1

INCONSISTENCY BETWEEN REQUIREMENTS FIXED

Iteration 10

Iteration objective

Customer auto responses active (in combination with existing compatible configuration and basics)

Current test case

testprogram ConfigurationAndBasics{

...}

fixturecomponent CustomerAutoresponsesActive{
  customerAutoresponsesSettings:=new CustomerAutoresponsesSettings;
  customerAutoresponsesSettings.autorespondWhenNewTicketCreatedByCustomer:=true;
  customerAutoresponsesSettings.autorespondWhenNewTicketCreatedByStaff:=true;
  customerAutoresponsesSettings.autorespondWhenNewMessageAppendedToTicket:=true;
  customerAutoresponsesSettings.autorespondWhenMaximumOpenTicketsOfCustomer:=true;
}

test testConfiguration1{
  load CompatibleConfigurationAndBasics;
  load GeneralSettingsOnline;
  load TicketSettingsSequential;
  load CustomerAutoresponsesActive;
  assert consistency;
}

test testConfiguration5{
  load CompatibleConfigurationAndBasics;
  load GeneralSettingsOnline;
  load TicketSettingsRandom;
  load CustomerAutoresponsesActive;
}
assert consistency;
}

test testConfiguration9{
  load CompatibleConfigurationAndBasics;
  load GeneralSettingsOffline;
  load TicketSettingsSequential;
  load CustomerAutoresponsesActive;
  assert consistency;
}

test testConfiguration13{
  load CompatibleConfigurationAndBasics;
  load GeneralSettingsOffline;
  load TicketSettingsRandom;
  load CustomerAutoresponsesActive;
  assert consistency;
}
}

TDCM application: Summary of changes performed in the schema

- Added
  class CustomerAutoresponsesSettings
  attributes
    autorespondWhenNewTicketCreatedByCustomer:Boolean
    autorespondWhenNewTicketCreatedByStaff:Boolean
    autorespondWhenNewMessageAppendedToTicket:Boolean
    autorespondWhenMaximumOpenTicketsOfCustomer:Boolean

  context CustomerAutoresponsesSettings inv hasOnlyOneInstance:
    CustomerAutoresponsesSettings.allInstances().size()=1

Time spent

| Time to Write Test Cases (in minutes) | 2.5 |
| Time to Complete the Iteration (in minutes) | 3 |

Errors and failures that drive the conceptual modeling

<table>
<thead>
<tr>
<th>A basic type involved in a test case does not exist in the CSUD</th>
<th>A derived type involved in a test case does not exist in the CSUD</th>
<th>An event type involved in a test case does not exist in the CSUD</th>
</tr>
</thead>
<tbody>
<tr>
<td>The basic type is relevant and it is added to the CSUD</td>
<td>The derived type is relevant and it is added to the CSUD</td>
<td>The event type is relevant and it is added to the CSUD</td>
</tr>
</tbody>
</table>

Inconsistent state before the occurrence of an event

<table>
<thead>
<tr>
<th>Inconsistent state after the occurrence of an event</th>
</tr>
</thead>
<tbody>
<tr>
<td>Some static constraint is invalid and it is modified.</td>
</tr>
<tr>
<td>Some initial integrity constraint is invalid and it is modified.</td>
</tr>
<tr>
<td>The event postcondition/method is incorrect and it is modified.</td>
</tr>
<tr>
<td>Some constraint is invalid and it is modified.</td>
</tr>
<tr>
<td>The method is not correct and it is modified.</td>
</tr>
</tbody>
</table>

The postcondition of an event is not satisfied

<table>
<thead>
<tr>
<th>An assertion about the IB state fails or contains an error</th>
</tr>
</thead>
<tbody>
<tr>
<td>Assert non-occurrence fails</td>
</tr>
<tr>
<td>Semantic error in an expression</td>
</tr>
<tr>
<td>The effect of an event type is not correct</td>
</tr>
<tr>
<td>A derivation rule is incorrect</td>
</tr>
<tr>
<td>A precondition is added/updated</td>
</tr>
<tr>
<td>The expression is corrected</td>
</tr>
<tr>
<td>The CSUD is changed</td>
</tr>
</tbody>
</table>

Assert consistency fails

| A static constraint needs to be changed |
Iteration 11

Iteration objective

Customer auto responses inactive (in combination with existing compatible configuration and basics)

Current test case

testprogram ConfigurationAndBasics{  
  ...
  fixturecomponent CustomerAutoresponsesInactive{    
    customerAutoresponsesSettings:=new CustomerAutoresponsesSettings;    
    customerAutoresponsesSettings.autorespondWhenNewTicketCreatedByCustomer:=false;    
    customerAutoresponsesSettings.autorespondWhenNewTicketCreatedByStaff:=false;    
    customerAutoresponsesSettings.autorespondWhenNewMessageAppendedToTicket:=false;    
    customerAutoresponsesSettings.autorespondWhenMaximumOpenTicketsOfCustomer:=false;  
  }
  test testConfiguration1{    
    load CompatibleConfigurationAndBasics;    
    load GeneralSettingsOnline;    
    load TicketSettingsSequential;    
    load CustomerAutoresponsesActive;    
    assert consistency;  
  }
  test testConfiguration3{    
    load CompatibleConfigurationAndBasics;    
    load GeneralSettingsOnline;    
    load TicketSettingsSequential;    
    load CustomerAutoresponsesInactive;    
    assert consistency;  
  }
  test testConfiguration5{    
    load CompatibleConfigurationAndBasics;    
    load GeneralSettingsOnline;    
    load TicketSettingsRandom;    
    load CustomerAutoresponsesActive;    
    assert consistency;  
  }
  test testConfiguration9{    
    load CompatibleConfigurationAndBasics;    
    load GeneralSettingsOffline;    
    load TicketSettingsSequential;    
    load CustomerAutoresponsesActive;    
    assert consistency;  
  }
  test testConfiguration13{    
    load CompatibleConfigurationAndBasics;    
    load GeneralSettingsOffline;    
    load TicketSettingsRandom;    
    load CustomerAutoresponsesActive;    
    assert consistency;  
  }
}
TDCM application: Summary of changes performed in the schema

No changes in the CSUD.

Time spent

<table>
<thead>
<tr>
<th>Time to write test cases (in minutes)</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time to complete the iteration (in minutes)</td>
<td>0</td>
</tr>
</tbody>
</table>

Iteration 12

Iteration objective

Staff notices alerts inactive (in combination with existing compatible configuration and basics)

Current test case

```plaintext
testprogram ConfigurationAndBasics{
    ...
    fixturecomponent StaffNoticesAlertsInactive{
        staffNoticesAlertsSettings:=new staffNoticesAlertsSettings;
        staffNoticesAlertsSettings.alertWhenNewTicketCreated:=false;
        staffNoticesAlertsSettings.alertWhenNewMessage:=false;
        staffNoticesAlertsSettings.alertWhenInternalNote:=false;
        staffNoticesAlertsSettings.alertWhenTicketOverdue:=false;
    }

test testConfiguration1{
    load CompatibleConfigurationAndBasics;
    load GeneralSettingsOnline;
    load TicketSettingsSequential;
    load CustomerAutoresponsesActive;
    assert consistency;
}

test testConfiguration3{
    load CompatibleConfigurationAndBasics;
    load GeneralSettingsOnline;
    load TicketSettingsSequential;
    load CustomerAutoresponsesInactive;
    assert consistency;
}

test testConfiguration5{
    load CompatibleConfigurationAndBasics;
    load GeneralSettingsOnline;
    load TicketSettingsRandom;
    load CustomerAutoresponsesActive;
    assert consistency;
}

test testConfiguration9{
    load CompatibleConfigurationAndBasics;
    load GeneralSettingsOffline;
    load TicketSettingsSequential;
    load CustomerAutoresponsesActive;
```
assert consistency;
}

test testConfiguration13{
    load ConfigurationsAndBasics;
    load GeneralSettingsOffline;
    load TicketSettingsRandom;
    load CustomerAutoresponsesActive;
    assert consistency;
}
}

TDCM application: Summary of changes performed in the schema

- Added
  class StaffNoticesAlertsSettings
  attributes
  alertWhenNewTicketCreated: Boolean
  alertWhenNewMessage: Boolean
  alertWhenInternalNote: Boolean
  alertWhenTicketOverdue: Boolean
end

context StaffNoticesAlertsSettings inv hasOnlyOneInstance:
    StaffNoticesAlertsSettings.allInstances()->size()=1

Time spent

| TIME TO WRITE TEST CASES (IN MINUTES) | 2 |
| TIME TO COMPLETE THE ITERATION (IN MINUTES) | 2,5 |

Errors and failures that drive the conceptual modeling

<table>
<thead>
<tr>
<th>A basic type involved in a test case does not exist in the CSUD</th>
<th>A derived type involved in a test case does not exist in the CSUD</th>
<th>An event type involved in a test case does not exist in the CSUD</th>
</tr>
</thead>
<tbody>
<tr>
<td>The basic type is relevant and it is added to the CSUD</td>
<td>The derived type is relevant and it is added to the CSUD</td>
<td>The event type is relevant and it is added to the CSUD</td>
</tr>
</tbody>
</table>

5

<table>
<thead>
<tr>
<th>Inconsistent state before the occurrence of an event</th>
<th>Inconsistent state after the occurrence of an event</th>
<th>The postcondition of an event is not satisfied</th>
</tr>
</thead>
<tbody>
<tr>
<td>Some static constraint is invalid and it is not modified.</td>
<td>Some initial integrity constraint is invalid and it is not modified.</td>
<td>The event postcondition/method is incorrect and it is not modified.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>An assertion about the IB state fails or contains an error</th>
<th>Assert non-occurrence fails</th>
<th>Semantic error in an expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>The effect of an event type is not correct</td>
<td>A derivation rule is incorrect</td>
<td>The expression is corrected</td>
</tr>
<tr>
<td>A precondition is added/updated</td>
<td>The CSUD is changed</td>
<td></td>
</tr>
</tbody>
</table>

Assert consistency fails
| A static constraint needs to be changed |
Iteration 13

Iteration objective

Staff notices alerts active (in combination with existing compatible configuration and basics)

Current test case

testprogram ConfigurationAndBasics{

    ... 
    fixturecomponent StaffNoticesAlertsActive {
        staffNoticesAlertsSettings := new StaffNoticesAlertsSettings;
        staffNoticesAlertsSettings.alertWhenNewTicketCreated := true;
        staffNoticesAlertsSettings.alertWhenNewTicketCreatedStaff := Set{#Administrator, #DepartmentManager, #DepartmentMembers};
        staffNoticesAlertsSettings.alertWhenNewMessage := true;
        staffNoticesAlertsSettings.alertWhenNewMessageStaff := Set{#LastRespondent, #AssignedStaff, #DepartmentManager};
        staffNoticesAlertsSettings.alertWhenInternalNote := true;
        staffNoticesAlertsSettings.alertWhenInternalNoteStaff := Set{#LastRespondent, #AssignedStaff, #DepartmentManager};
        staffNoticesAlertsSettings.alertWhenTicketOverdue := true;
        staffNoticesAlertsSettings.alertWhenTicketOverdueStaff := Set{#AssignedStaff, #DepartmentManager, #DepartmentMembers};
    }

    test testConfiguration1 {
        load CompatibleConfigurationAndBasics;
        load GeneralSettingsOnline;
        load TicketSettingsSequential;
        load CustomerAutoresponsesActive;
        load StaffNoticesAlertsActive;
        assert consistency;
    }

    test testConfiguration2 {
        load CompatibleConfigurationAndBasics;
        load GeneralSettingsOnline;
        load TicketSettingsSequential;
        load CustomerAutoresponsesActive;
        load StaffNoticesAlertsInactive;
        assert consistency;
    }

    test testConfiguration3 {
        load CompatibleConfigurationAndBasics;
        load GeneralSettingsOnline;
        load TicketSettingsSequential;
        load CustomerAutoresponsesInactive;
        load StaffNoticesAlertsActive;
        assert consistency;
    }

    test testConfiguration5 {
        load CompatibleConfigurationAndBasics;
        load GeneralSettingsOnline;
        load TicketSettingsRandom;
        load CustomerAutoresponsesActive;
        load StaffNoticesAlertsActive;
        assert consistency;
    }

    test testConfiguration9 {
        load CompatibleConfigurationAndBasics;
        load GeneralSettingsOffline;
    }
}
Development of the Conceptual Schema of the osTicket System by Applying TDCM

Research report
April 2011

Department of Service and Information System Engineering

load TicketSettingsSequential;
load CustomerAutoresponsesActive;
load StaffNoticesAlertsActive;
assert consistency;
}

test testConfiguration13{
    load CompatibleConfigurationAndBasics;
    load GeneralSettingsOffline;
    load TicketSettingsRandom;
    load CustomerAutoresponsesActive;
    load StaffNoticesAlertsActive;
    assert consistency;
}
}

TDCM application: Summary of changes performed in the schema

- Added
- Updated

enum StaffRole{Administrator, DepartmentManager, DepartmentMembers, LastRespondent, AssignedStaff}

class StaffNoticesAlertsSettings
    attributes
        alertWhenNewTicketCreated:Boolean
        alertWhenNewTicketCreatedStaff:Set(StaffRole)
        alertWhenNewMessage:Boolean
        alertWhenNewMessageStaff:Set(StaffRole)
        alertWhenInternalNote:Boolean
        alertWhenInternalNoteStaff:Set(StaffRole)
        alertWhenTicketOverdue:Boolean
        alertWhenTicketOverdueStaff:Set(StaffRole)
    end

Time spent

<table>
<thead>
<tr>
<th>TIME TO WRITE TEST CASES (IN MINUTES)</th>
<th>3.5</th>
</tr>
</thead>
<tbody>
<tr>
<td>TIME TO COMPLETE THE ITERATION (IN MINUTES)</td>
<td>4</td>
</tr>
</tbody>
</table>

Errors and failures that drive the conceptual modeling

<table>
<thead>
<tr>
<th>A basic type involved in a test case does not exist in the CSUD</th>
<th>A derived type involved in a test case does not exist in the CSUD</th>
<th>An event type involved in a test case does not exist in the CSUD</th>
</tr>
</thead>
<tbody>
<tr>
<td>The basic type is relevant and it is added to the CSUD</td>
<td>The derived type is relevant and it is added to the CSUD</td>
<td>The event type is relevant and it is added to the CSUD</td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Inconsistent state before the occurrence of an event</td>
<td>Inconsistent state after the occurrence of an event</td>
<td>The postcondition of an event is not satisfied.</td>
</tr>
<tr>
<td>Some static constraint is invalid and it is modified.</td>
<td>Some initial integrity constraint is invalid and it is modified.</td>
<td>Some constraint is invalid and it is modified.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The method is not correct and it is modified.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The postcondition is not correct and it is modified.</td>
</tr>
<tr>
<td>An assertion about the IB state fails or contains an error</td>
<td>Assert non-occurrence fails</td>
<td>Semantic error in an expression</td>
</tr>
<tr>
<td>The effect of an event type is not correct</td>
<td>A derivation rule is incorrect</td>
<td>The expression is changed</td>
</tr>
<tr>
<td></td>
<td>A precondition is added/updated</td>
<td>The CSUD is changed</td>
</tr>
</tbody>
</table>
DEVELOPMENT OF THE CONCEPTUAL SCHEMA OF THE osTICKET SYSTEM BY APPLYING TDCM

Iteration 14

Iteration objective

All configuration and basics combinations

Current test case

testprogram ConfigurationAndBasics{
    ...
    test testConfiguration1{
        load CompatibleConfigurationAndBasics;
        load GeneralSettingsOnline;
        load TicketSettingsSequential;
        load CustomerAutoresponsesActive;
        load StaffNoticesAlertsActive;
        assert consistency;
    }
    ...
}

...
load CustomerAutoresponsesActive;
load StaffNoticesAlertsActive;
assert consistency;
}

test testConfiguration7{
    load CompatibleConfigurationAndBasics;
    load GeneralSettingsOnline;
    load TicketSettingsRandom;
    load CustomerAutoresponsesActive;
    load StaffNoticesAlertsInactive;
    assert consistency;
}

test testConfiguration8{
    load CompatibleConfigurationAndBasics;
    load GeneralSettingsOnline;
    load TicketSettingsRandom;
    load CustomerAutoresponsesInactive;
    load StaffNoticesAlertsActive;
    assert consistency;
}

test testConfiguration9{
    load CompatibleConfigurationAndBasics;
    load GeneralSettingsOffline;
    load TicketSettingsSequential;
    load CustomerAutoresponsesActive;
    load StaffNoticesAlertsActive;
    assert consistency;
}

test testConfiguration10{
    load CompatibleConfigurationAndBasics;
    load GeneralSettingsOffline;
    load TicketSettingsSequential;
    load CustomerAutoresponsesActive;
    load StaffNoticesAlertsInactive;
    assert consistency;
}

test testConfiguration11{
    load CompatibleConfigurationAndBasics;
    load GeneralSettingsOffline;
    load TicketSettingsSequential;
    load CustomerAutoresponsesInactive;
    load StaffNoticesAlertsActive;
    assert consistency;
}

test testConfiguration12{
    load CompatibleConfigurationAndBasics;
    load GeneralSettingsOffline;
    load TicketSettingsSequential;
    load CustomerAutoresponsesInactive;
    load StaffNoticesAlertsInactive;
    assert consistency;
}

test testConfiguration13{
    load CompatibleConfigurationAndBasics;
    load GeneralSettingsOffline;
    load TicketSettingsRandom;
    load CustomerAutoresponsesActive;
    load StaffNoticesAlertsActive;
    assert consistency;
}

test testConfiguration14{
    load CompatibleConfigurationAndBasics;
    load GeneralSettingsOffline;
    load TicketSettingsRandom;
    load CustomerAutoresponsesActive;
    load StaffNoticesAlertsActive;
    assert consistency;
}
load CustomerAutoresponsesActive;
load StaffNoticesAlertsInactive;
assert consistency;
}

test testConfiguration15{
load CompatibleConfigurationAndBasics;
load GeneralSettingsOffline;
load TicketSettingsRandom;
load CustomerAutoresponsesInactive;
load StaffNoticesAlertsActive;
assert consistency;
}

test testConfiguration16{
load CompatibleConfigurationAndBasics;
load GeneralSettingsOffline;
load TicketSettingsRandom;
load CustomerAutoresponsesInactive;
load StaffNoticesAlertsInactive;
assert consistency;
}

TDCM application: Summary of changes performed in the schema

No changes in the CSUT

Time spent

<table>
<thead>
<tr>
<th>Time to Write Test Cases (in minutes)</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time to Complete the Iteration (in minutes)</td>
<td>0</td>
</tr>
</tbody>
</table>
We present the graphical form of the structural schema related to the configuration & basics of the osTicket system. This structural schema is the result of the incremental development performed by applying TDCM:
About the tickets management and tracking iterations

In the previous iterations, we developed the structural schema of the configuration options and the basics of the system.

The following iterations use the test cases that validate the configurations & basics schema as fixtures to test the tickets management and tracking stories.

These are the valid fixtures that will be used in the following iterations:

```plaintext
fixturecomponent testConfiguration1{
  load CompatibleConfigurationAndBasics;
  load GeneralSettingsOnline;
  load TicketSettingsSequential;
  load CustomerAutoresponsesActive;
  load StaffNoticesAlertsActive;
}

fixturecomponent testConfiguration2{
  load CompatibleConfigurationAndBasics;
  load GeneralSettingsOnline;
  load TicketSettingsSequential;
  load CustomerAutoresponsesActive;
  load StaffNoticesAlertsInactive;
}

fixturecomponent testConfiguration3{
  load CompatibleConfigurationAndBasics;
  load GeneralSettingsOnline;
  load TicketSettingsSequential;
  load CustomerAutoresponsesInactive;
  load StaffNoticesAlertsActive;
}

fixturecomponent testConfiguration4{
  load CompatibleConfigurationAndBasics;
  load GeneralSettingsOnline;
  load TicketSettingsSequential;
  load CustomerAutoresponsesInactive;
  load StaffNoticesAlertsInactive;
}

fixturecomponent testConfiguration5{
  load CompatibleConfigurationAndBasics;
  load GeneralSettingsOnline;
  load TicketSettingsRandom;
  load CustomerAutoresponsesActive;
  load StaffNoticesAlertsActive;
}

fixturecomponent testConfiguration6{
  load CompatibleConfigurationAndBasics;
  load GeneralSettingsOnline;
  load TicketSettingsRandom;
  load CustomerAutoresponsesActive;
  load StaffNoticesAlertsInactive;
}
DEVELOPMENT OF THE CONCEPTUAL SCHEMA OF THE osTICKET SYSTEM BY APPLYING TDCM

Department of Service and Information System Engineering

fixturecomponent testConfiguration7{
load CompatibleConfigurationAndBasics;
load GeneralSettingsOnline;
load TicketSettingsRandom;
load CustomerAutoresponsesInactive;
load StaffNoticesAlertsActive;
}

fixturecomponent testConfiguration8{
load CompatibleConfigurationAndBasics;
load GeneralSettingsOnline;
load TicketSettingsRandom;
load CustomerAutoresponsesInactive;
load StaffNoticesAlertsInactive;
}

fixturecomponent testConfiguration9{
load CompatibleConfigurationAndBasics;
load GeneralSettingsOffline;
load TicketSettingsSequential;
load CustomerAutoresponsesActive;
load StaffNoticesAlertsActive;
}

fixturecomponent testConfiguration10{
load CompatibleConfigurationAndBasics;
load GeneralSettingsOffline;
load TicketSettingsSequential;
load CustomerAutoresponsesActive;
load StaffNoticesAlertsInactive;
}

fixturecomponent testConfiguration11{
load CompatibleConfigurationAndBasics;
load GeneralSettingsOffline;
load TicketSettingsSequential;
load CustomerAutoresponsesInactive;
load StaffNoticesAlertsActive;
}

fixturecomponent testConfiguration12{
load CompatibleConfigurationAndBasics;
load GeneralSettingsOffline;
load TicketSettingsSequential;
load CustomerAutoresponsesInactive;
load StaffNoticesAlertsInactive;
}

fixturecomponent testConfiguration13{
load CompatibleConfigurationAndBasics;
load GeneralSettingsOffline;
load TicketSettingsRandom;
load CustomerAutoresponsesActive;
load StaffNoticesAlertsActive;
}

fixturecomponent testConfiguration14{
load CompatibleConfigurationAndBasics;
load GeneralSettingsOffline;
load TicketSettingsRandom;
load CustomerAutoresponsesActive;
load StaffNoticesAlertsInactive;
Mock schema elements

Mock objects [2] are simulated objects that mimic the behavior of real objects for testing purposes. In conceptual schema testing, we can also define schema elements which simulate knowledge for testing purposes. In this case study, we assume the following conceptual schema fragment in order to keep track of the emails sent by the system:

We also assume that there is an instance of the entity type System and the following attributes:

- currentDateTime: Datetime. It is a value to specify the current date time as an Integer. For testing purposes, its value may be changed as desired to simulate the evolution of time.
- aleat: By this attribute we may specify an integer for testing aleatory values.
Iteration 15

Iteration objective

S1: NewTicketOnline_SuccessScenario_SequentialTicketsNumber_StaffNotifications (first ticket)

Current test case

testprogram TicketsManagementAndTracking{

test S1{
  load testConfiguration1;

  nt:=new NewTicketOnline;
  nt.fullName:=‘Mary Marnes’;
  nt.email:=‘mary_at_marnes.mar’;
  nt.telephone:='xxxxxxxx';
  nt.ext:='xxxxxxxx';
  nt.helpTopic:=helpTopicInstallation;
  nt.subject:='Error operating system';
  nt.message:='The installation process does not finish....';
  assert occurrence nt;

  ticket1:=nt.createdTicket;
  assert equals ticket1.number 1;
  assert equals ticket1.ticketStatus #Open;
  assert equals ticket1.ticketSubject ‘Error operating system’;
  assert equals ticket1.priority #High;
  assert true ticket1.assignedStaff->isEmpty();
  assert equals ticket1.source #Web;
  assert equals ticket1.creationDatetime sys.currentDateTime;
  assert true ticket1.dueDateTime.isUndefined();
  assert true ticket1.lastResponseDatetime.isUndefined();
  assert equals ticket1.assignedDepartment dptTechnical;

  assert true [ticket1.ticketThreadMessage->one(m|m.datetime=sys.currentDateTime and 
    m.text='The installation process does not finish....' and
    m.author='Mary Marnes')];

  assert equals ticket1.lastMessageDateTime sys.currentDateTime;

  //no autoreponses
  assert true [not(EMail.allInstances()->exists(e|e.emailKind.oclIsTypeOf(NewTicketAutoresponse)))];

  //notice to administrator
  assert true [EMail.allInstances()->exists(e|e.emailKind.oclIsTypeOf(NewTicketAlertToStaff) and
      e.fromAddress='general_at_support.com' and
      e.toAddress='system_at_support.com' and
      e.ticketNumber=1)];

  //notice to department manager
  assert true [EMail.allInstances()->exists(e|e.emailKind.oclIsTypeOf(NewTicketAlertToStaff) and
      e.fromAddress='general_at_support.com' and
      e.toAddress='john_at_support.com' and
      e.ticketNumber=1)];

  //notice to department members
  assert true [EMail.allInstances()->exists(e|e.emailKind.oclIsTypeOf(NewTicketAlertToStaff) and
      e.fromAddress='general_at_support.com' and
      e.toAddress='martin_at_support.com' and
      e[toAddress='mary_at_marnes.mar']];
}
TDCM application: Summary of changes performed in the schema

- Added
- Updated

```
enum TicketsMode{Sequential, Random}
enum StaffRole{Administrator, DepartmentManager, DepartmentMembers, LastRespondent, AssignedStaff}
enum TicketStatus{Open, Closed}
enum TicketSource{Web}
```

```
-- TICKETS MANAGEMENT
class Ticket
attributes
/number:Integer=if TicketSettings.allInstances()->any(true).mode=#Sequential then
  Ticket.allInstances()->size()
else System.allInstances()->any(true).aleat
endif constant
ticketStatus:TicketStatus
fullName:String
email:String
telephone:String
ext:String
subject:String
message:String
priority:Priority
source:TicketSource
/creationDatetime:Datetime=System.allInstances()->any(true).currentDateTime constant
dueDatetime:Datetime[0..1]
lastResponseDatetime:Datetime[0..1]
lastMessageDatetime:Datetime[0..1]
end

association ticket_assignedStaff between
  Ticket[*]
  StaffMember[0..1] role assignedStaff
end

association ticket_ticketThreadMessage between
  Ticket[1]
  TicketThreadMessage[*]
end

class TicketThreadMessage
attributes
datetime:Datetime
text:String
author:String
end

association ticket_assignedDepartment between
  Ticket[*]
  Department[1] role assignedDepartment
end

event NewTicketOnline
attributes
fullName:String
email:String
telephone:String
ext:String
subject:String
message:String
createdTicket:Ticket[0..1]
operations
effect()
```
DEVELOPMENT OF THE CONCEPTUAL SCHEMA OF THE osTICKET SYSTEM BY APPLYING TDGM

April 2011

context NewTicketOnline::effect() post:
let defaultPriority:Priority=
if self.helpTopic->notEmpty() then
  self.helpTopic.newTicketPriority
else
  TicketSettings.allInstances()->any(true).priority
endif

let defaultDepartment:Department=
if self.helpTopic->notEmpty() then
  self.helpTopic.newTicketDepartment
else
  Department.allInstances()->any(d|d.isDefault)
endif

let sendNewTicketAlertToAdministrator:Boolean=
StaffNoticesAlertsSettings.allInstances()->any(true).alertWhenNewTicketCreated and
StaffNoticesAlertsSettings.allInstances()->any(true).alertWhenNewTicketCreatedStaff->includes(#Administrator)

let sendNewTicketAlertToDepartmentManager:Boolean=
StaffNoticesAlertsSettings.allInstances()->any(true).alertWhenNewTicketCreated and
StaffNoticesAlertsSettings.allInstances()->any(true).alertWhenNewTicketCreatedStaff->includes(#DepartmentManager)

let sendNewTicketAlertToDepartmentMembers:Boolean=
StaffNoticesAlertsSettings.allInstances()->any(true).alertWhenNewTicketCreated and
StaffNoticesAlertsSettings.allInstances()->any(true).alertWhenNewTicketCreatedStaff->includes(#DepartmentMembers)

let staffAlertsFromEMailAddress:String=
EmailSettings.allInstances()->any(true).defaultStaffAlertsEmail.address

let autoresponsesEMailAddress:String=
EmailSettings.allInstances()->any(true).defaultStaffAlertsEmail.address

let sendAutoresponse:Boolean=
if CustomerAutoresponsesSettings.allInstances()->any(true).autorespondWhenNewTicketCreatedByCustomer then
  if self.helpTopic->notEmpty() then
    if self.helpTopic.autoresponse=#Enabled then true
    else false
  endif
else
  defaultDepartment.newTicketAutoresponsesSent
endif
else false
endif

(Ticket.allInstances- Ticket.allInstances@pre)
->one(t | t.oclIsNew() and self.createdTicket=t
  and t.fullName=self.fullName
  and t.email=self.email
  and t.telephone=self.telephone
  and t.ext=self.ext
  and t.subject=self.subject
  and t.message=self.message
  and t.ticketStatus=#Open
  and t.priority=defaultPriority
  and t.source=#Web
  and t.assignedDepartment=defaultDepartment
  and (TicketThreadMessage.allInstances- TicketThreadMessage.allInstances@pre)
  ->one(tdm | tdm.oclIsNew() 
  and tdm.datetime=System.allInstances()->any(true).currentDateTime
  and tdm.text=self.message
  and tdm.author=self.fullName
  and tdm.ticket=t

end
The added knowledge is the result of evolving the schema in order to solve the errors and failures told by the CSTL Processor until the test case passes.

These are some of the obtained errors (some repetitive errors about undefined knowledge are not shown):

- `[TicketsManagementAndTracking.cstl] <line 435>` NewTicketOnline is not defined in the CSUT as a class or an association
- `[TicketsManagementAndTracking.cstl] <line 443>` The effect of NewTicketOnline is not defined as a method (specify/review first its postconditions)
- CSUT error: osTicketCSUT.use:362:1: Expected object type, found 'Ticket'
- CSUT error: osTicketCSUT.use:374:21: Undefined operation named `createdTicket` in expression `NewTicketOnline.createdTicket()`
- `[TicketsManagementAndTracking.cstl] <Line 447>:1:6: Undefined operation named `status` in expression `{Ticket}.status()`
- `[TicketsManagementAndTracking.cstl] <Line 448>:1:6: Undefined operation named `number` in expression `{Ticket}.number()`
- CSUT error: osTicketCSUT.use:356:16: Branches of if expression have different type, found `Set(Ticket)` and `Integer`
- It detects an error in the derivation rule
- `[TicketsManagementAndTracking.cstl] <Line 448>:1:6: Undefined operation named `status` in expression `{Ticket}.status()`
[TicketsManagementAndTracking.cstl] <line 443> Inconsistent state after nt:NewTicketOnline event execution: Instances of Ticket violate the invariant minMultiplicityOfAttributeStatus

- The event needs to specify an status of the ticket

CSUT error: osTicketCSUT.use:388:26: Undefined enumeration literal 'Open'.

CSUT error: osTicketCSUT.use:389:18: Undefined operation named 'ticketStatus' in expression 'Ticket.ticketStatus()'.

[TicketsManagementAndTracking.cstl] postcondition 'post114' of NewTicketOnline is false

- The method has to be updated in order to specify the ticketStatus

[TicketsManagementAndTracking.cstl] <Line 450>:1:6: Undefined operation named 'priority' in expression '{Ticket}.priority()'.

[TicketsManagementAndTracking.cstl] <line 443> Inconsistent state after nt:NewTicketOnline event execution: Instances of Ticket violate the invariant minMultiplicityOfAttributePriority

[TicketsManagementAndTracking.cstl] postcondition 'post25' of NewTicketOnline is false (tickets priority is not included in the method)

[TicketsManagementAndTracking.cstl] <line 443> Inconsistent state after nt:NewTicketOnline event execution: Multiplicity constraint violation in association 'ticket_assignedStaff': Object 'oid35' of class 'Ticket' is connected to 0 objects of class 'StaffMember' but the multiplicity is specified as '1'.

- We realize that there may be unasigned tickets (we change the CSUT)

[TicketsManagementAndTracking.cstl] <Line 452>:1:6: Undefined operation named 'source' in expression '{Ticket}.source()'

[TicketsManagementAndTracking.cstl] postcondition 'post62' of NewTicketOnline is false

[TicketsManagementAndTracking.cstl] <Line 454>:1:6: Undefined operation named 'creationDatetime' in expression '{Ticket}.dueDatetime()'.

[TicketsManagementAndTracking.cstl] <Line 454>:1:6: Undefined operation named 'dueDatetime' in expression '{Ticket}.dueDatetime()'.

[TicketsManagementAndTracking.cstl] <Line 454>:1:6: Undefined operation named 'creationDatetime' in expression '{Ticket}.dueDatetime()'.

[TicketsManagementAndTracking.cstl] <Line 455>:1:6: Undefined operation named 'lastResponseDatetime' in expression '{Ticket}.lastResponseDatetime()'.

[TicketsManagementAndTracking.cstl] <Line 456>:1:6: Undefined operation named 'assignedDepartment' in expression '{Ticket}.assignedDepartment()'.

[TicketsManagementAndTracking.cstl] <Line 443> Inconsistent state after nt:NewTicketOnline event execution: Multiplicity constraint violation in association 'ticket_assignedDepartment': Object 'oid35' of class 'Ticket' is connected to 0 objects of class 'Department' but the multiplicity is specified as '1'.

- During event effect specification we realize that the system needs a default department

```cstl
class Department
attributes
    name:String
    type:DepartmentType
    newTicketAutoresponseIsSent:Boolean
    newAddedMessageIsNotified:Boolean
    isDefault:Boolean=false
end

class Department inv hasAlwaysOneDefault:
    Department.allInstances() ->select(d|d.isDefault)->size()=1
```
[TicketsManagementAndTracking.cstl] <line 443> Inconsistent state before nt:NewTicketOnline event execution: Instances of Department violate the invariant hasAlwaysOneDefault

The fixture needs to be changed to ensure that there is always one default department

[TicketsManagementAndTracking.cstl] <line 444> Inconsistent state after nt:NewTicketOnline event execution: Multiplicity constraint violation in association 'ticket_assignedDepartment': Object 'oid35' of class 'Ticket' is connected to 0 objects of class 'Department' but the multiplicity is specified as '1'.

Tickets need to be assigned to departments

[TicketsManagementAndTracking.cstl] <postcondition 'post200' of NewTicketOnline is false

[TicketsManagementAndTracking.cstl] <line 459>:1:6: Undefined operation named 'lastMessageDatetime' in expression '{Ticket}.lastMessageDatetime()'.

[TicketsManagementAndTracking.cstl] <line 444> Inconsistent state after nt:NewTicketOnline event execution: Instances of Ticket violate the invariant minMultiplicityOfAttributeLastMessageDatetime

[TicketsManagementAndTracking.cstl] <Line 461>:1:6: Undefined operation named 'message' in expression '{Ticket}.message()'.

[TicketsManagementAndTracking.cstl] <Line 464>:1:27: Undefined operation `Set(TicketThreadMessage)>includes(Set(OclVoid))'

There is no message assigned to the ticket

[TicketsManagementAndTracking.cstl] <postcondition 'post384' of NewTicketOnline is false

CSUT error: osTicketCSUT.use:446:31: Undefined operation `Set(Ticket)->=(Ticket)'

The result is Undefined but it is expected to be an object  (assert equals ticket1.lastMessageDatetime sys.currentDateTime)

The lastMessageDatetime needs to be specified by the event NewTicketOnline

[TicketsManagementAndTracking.cstl] <postcondition 'post457' of NewTicketOnline is false

Assert expression is false and it is expected to be true:
assert true [EMail.allInstances()]->exists(e|e.emailKind.oclIsTypeOf(NewTicketAlertToStaff) and e.fromAddress='general_at_support.com' and e.toAddress='system_at_support.com' and e.ticketNumber=1)];

The postcondition and the method of NewTicketOnline is changed to include autoresponses and alerts until the assertions are true

**Time spent**

| TIME TO WRITE TEST CASES (IN MINUTES) | 26 |
| TIME TO COMPLETE THE ITERATION (IN MINUTES) | 83 |
Errors and failures that drive the conceptual modeling

<table>
<thead>
<tr>
<th>A basic type involved in a test case does not exist in the CSUD</th>
<th>A derived type involved in a test case does not exist in the CSUD</th>
<th>An event type involved in a test case does not exist in the CSUD</th>
</tr>
</thead>
<tbody>
<tr>
<td>The basic type is relevant and it is added to the CSUD</td>
<td>The derived type is relevant and it is added to the CSUD</td>
<td>The event type is relevant and it is added to the CSUD</td>
</tr>
<tr>
<td>32</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>Inconsistent state before the occurrence of an event</td>
<td>Inconsistent state after the occurrence of an event</td>
<td>The postcondition of an event is not satisfied</td>
</tr>
<tr>
<td>1</td>
<td>8</td>
<td>1</td>
</tr>
<tr>
<td>Some static constraint is invalid and it is modified.</td>
<td>Some initial integrity constraint is invalid and it is modified.</td>
<td>The method is not correct and it is modified.</td>
</tr>
<tr>
<td>1</td>
<td>8</td>
<td>1</td>
</tr>
<tr>
<td>An assertion about the IB state fails or contains an error</td>
<td>Assert non-occurrence fails</td>
<td>Semantic error in an expression</td>
</tr>
<tr>
<td>5</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>Assert consistency fails</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Iteration 16

Iteration objective

S1: NewTicketOnline_SuccessScenario_SequentialTicketsNumber_StaffNotifications (complete)

Current test case

```plaintext
test S1{
load testConfiguration1;
nt:=new NewTicketOnline;
tt.fullName:='Mary Marnes';
tt.email:='mary_at_marnes.mar';
tt.telephone:='xxxxxxxx';
tt.leaf:='xxxxxxxx';
tt.helpTopic:=helpTopicInstallation;
tt.subject:='Error operating system';
tt.message:='The installation process does not finish....';
assert occurrence nt;
ticket1:=nt.createdTicket;
assert equals ticket1.number 1;
assert equals ticket1.ticketStatus #Open;
assert equals ticket1.subject 'Error operating system';
assert equals ticket1.priority #High;
assert true ticket1.assignedStaff->isEmpty();
assert equals ticket1.source #Web;
assert equals ticket1.creationDatetime sys.currentDateTime;
assert true ticket1.dueDatetime.isUndefined();
```
assert true ticket1.lastResponseDateTime.isUndefined();
assert equals ticket1.assignedDepartment dptTechnical;

assert true (ticket1.ticketThreadMessage->one(m|m.datetime=sys.currentDateTime and m.text="The installation process does not finish...." and m.author="Mary Marnes"));
assert equals ticket1.lastMessageDateTime sys.currentDateTime;

//no autotemplates
assert true (not(EMail.allInstances()->|exists(e|e.emailKind.oclIsTypeOf(NewTicketAutoresponse))));

//notice to administrator
assert true (EMail.allInstances()->|exists(e|e.emailKind.oclIsTypeOf(NewTicketAlertToStaff) and e.fromAddress='general_at_support.com' and e.toAddress='system_at_support.com' and e.ticketNumber=1));

//notice to department manager
assert true (EMail.allInstances()->|exists(e|e.emailKind.oclIsTypeOf(NewTicketAlertToStaff) and e.fromAddress='general_at_support.com' and e.toAddress='john_at_support.com' and e.ticketNumber=1));

//notice to department members
assert true (EMail.allInstances()->|exists(e|e.emailKind.oclIsTypeOf(NewTicketAlertToStaff) and e.fromAddress='general_at_support.com' and e.toAddress='martin_at_support.com' and e.ticketNumber=1));

//TICKET 2
ticketSettings.customersCanChangePriority:=true;

nt2:=new NewTicketOnline;
nt2.fullName:='James Jordan';
nt2.email:='james_at_jordan.jam';
nt2.telephone:='xxxxxxxx';
nt2.ext:='xxxxxxxx';
nt2.priority:=#Low;
nt2.helpTopic:=helpTopicUse;
nt2.subject:='Reopening ticket';
nt2.message:="I do not know how to reopen one of my closed tickets';
assert occurrence nt2;

ticket2:=nt2.createdTicket;
assert equals ticket2.number 2;
assert equals ticket2.priority #Low;
assert equals ticket2.assignedDepartment dptGeneral;

//autotemplates
assert true (EMail.allInstances()->|exists(e|e.emailKind.oclIsTypeOf(NewTicketAutoresponse) and e.fromAddress='general_at_support.com' and e.toAddress='james_at_jordan.jam' and e.ticketNumber=2));

//notice to administrator
assert true (EMail.allInstances()->|exists(e|e.emailKind.oclIsTypeOf(NewTicketAlertToStaff) and e.fromAddress='general_at_support.com' and e.toAddress='system_at_support.com' and e.ticketNumber=2));

//notice to department members
assert true (EMail.allInstances()->|exists(e|e.emailKind.oclIsTypeOf(NewTicketAlertToStaff) and e.fromAddress='general_at_support.com' and e.toAddress='john_at_support.com' and e.ticketNumber=2));
TDCM application: Summary of changes performed in the schema

- [TicketsManagementAndTracking.cstl] <line 497> Binary property priority does not exist for this object

- We add the property `priority` to the event `NewTicketOnline`

- [TicketsManagementAndTracking.cstl] <line 444> Inconsistent state before nt:NewTicketOnline event execution: Instances of NewTicketOnline violate the invariant minMultiplicityOfAttributePriority

- A previous passing line causes this error. The reason is that the priority is only specified if the ticket settings allow it.

  event NewTicketOnline
  attributes
  fullName: String
  email: String
  telephone: String
  ext: String
  subject: String
  message: String
  priority: Priority[0..1]
  createdTicket: Ticket[0..1]
  operations
  effect()
  end

  context NewTicketOnline ini inv priorityMayBeSetWhenAllowed:
    if TicketSettings.allInstances()->any(true).customersCanChangePriority then
      self.priority.isDefined()
    else
      self.priority.isUndefined()
    endif

  context NewTicketOnline::effect()
  post:
    let defaultPriority: Priority=
      if self.helpTopic->notEmpty() then
        self.helpTopic.newTicketPriority
      else
        TicketSettings.allInstances()->any(true).priority
      endif
    in
    let assignedPriority: Priority=
      if TicketSettings.allInstances()->any(true).customersCanChangePriority then
        self.priority
      else
        defaultPriority
    endif

  The result is #Normal but it is expected to be #Low (assert equals ticket2.priority #Low;)

  The priority, if specified by the user, replaces the default priority

  context NewTicketOnline::effect()
  post:
    let defaultPriority: Priority=
      if self.helpTopic->notEmpty() then
        self.helpTopic.newTicketPriority
      else
        TicketSettings.allInstances()->any(true).priority
      endif
    in
    let assignedPriority: Priority=
      if TicketSettings.allInstances()->any(true).customersCanChangePriority then
        self.priority
      else
        defaultPriority

endif

in

(Ticket.allInstances- Ticket.allInstances@pre)

->one( t | t.isNew()

and self.createdTicket=t

and t.firstName=self.firstName

and t.lastName=self.lastName

and t.email=self.email

and t.telephone=self.telephone

and t.ext=self.ext

and t.subject=self.subject

and t.message=self.message

and t.ticketStatus=#Open

and t.priority=assignedPriority


...  

- Assert expression is false and it is expected to be true (assert true [EMail.allInstances()-
  >exists(e|e.emailKind.oclIsTypeOf(NewTicketAutoresponse) and e.fromAddress='technical_at_support.com' and
  e.toAddress='james_at_jordan.jam' and e.ticketNumber=2]);

- Autoresponses are not send correctly according to the expectations.

- We add a new assertions:

  assert equals EMail.allInstances->any(ticketNumber=2).fromAddress 'technical_at_support.com';

- We realize that toAddress and fromAddress are interchanged

  context NewTicketOnline::effect()

  post:

  (Ticket.allInstances- Ticket.allInstances@pre)

  ->one( t | t.isNew()

  ...  

  -- autoresponses

  and (sendAutoresponse implies

  EMail.allInstances()->exists(e | e.fromAddress=t.assignedDepartment.autoresponseEmail.address and

  e.toAddress=t.email and

  e.ticketNumber=t.number))

  ...

)

Time spent

| TIME TO WRITE TEST CASES (IN MINUTES) | 8 |
| TIME TO COMPLETE THE ITERATION (IN MINUTES) | 15 |

Errors and failures that drive the conceptual modeling

<table>
<thead>
<tr>
<th>A basic type involved in a test case does not exist in the CSUD</th>
<th>A derived type involved in a test case does not exist in the CSUD</th>
<th>An event type involved in a test case does not exist in the CSUD</th>
</tr>
</thead>
<tbody>
<tr>
<td>The basic type is relevant and it is added to the CSUD</td>
<td>The derived type is relevant and it is added to the CSUD</td>
<td>The event type is relevant and it is added to the CSUD</td>
</tr>
<tr>
<td>Inconsistent state before the occurrence of an event</td>
<td>Inconsistent state after the occurrence of an event</td>
<td>The postcondition of an event is not satisfied.</td>
</tr>
<tr>
<td>Some static constraint is invalid and it is modified.</td>
<td>Some initial integrity constraint is invalid and it is modified.</td>
<td>Some constraint is invalid and it is modified.</td>
</tr>
<tr>
<td>The event postcondition/method is incorrect and it is modified.</td>
<td>The method is not correct and it is modified.</td>
<td>The postcondition is not correct and it is modified.</td>
</tr>
</tbody>
</table>
An assertion about the IB state fails or contains an error | Assert non-occurrence fails | Semantic error in an expression
--- | --- | ---
The effect of an event type is not correct | A derivation rule is incorrect | A precondition is added/updated

| 2 | Assert consistency fails | A static constraint needs to be changed |

**Iteration 17**

**Iteration objective**

S2: NewTicketOnline_maximumNumberOfTicketsViolated

**Current test case**

```plaintext
test S2{
    load testConfiguration1;
    nt1:=new NewTicketOnline;
    nt1.fullName:= 'Mary Marnes';
    nt1.email:= 'mary_at_marnes.mar';
    nt1.telephone:= 'xxxxxxxx';
    nt1.ext:= 'xxxxxxxx';
    nt1.helpTopic:= helpTopicInstallation;
    nt1.subject:= 'Error operating system';
    nt1.message:= 'The installation process does not finish....';
    assert occurrence nt1;
    ticket1:=nt1.createdTicket;
    nt2:=new NewTicketOnline;
    nt2.fullName:= 'Mary Marnes';
    nt2.email:= 'mary_at_marnes.mar';
    nt2.telephone:= 'xxxxxxxx';
    nt2.ext:= 'xxxxxxxx';
    nt2.helpTopic:= helpTopicInstallation;
    nt2.subject:= 'Reopening ticket';
    nt2.message:= 'I do not know how to reopen one of my closed tickets';
    assert occurrence nt2;
    ticket2:=nt2.createdTicket;
    nt3:=new NewTicketOnline;
    nt3.fullName:= 'Mary Marnes';
    nt3.email:= 'mary_at_marnes.mar';
    nt3.telephone:= 'xxxxxxxx';
    nt3.ext:= 'xxxxxxxx';
    nt3.helpTopic:= helpTopicInstallation;
    nt3.subject:= 'Customize graphical interface';
    nt3.message:= 'May I change the background color?';
    assert non-occurrence nt3;
}
```
TDCM application: Summary of changes performed in the schema

- Preconditions of the domain event nt3:NewTicketOnline are satisfied and consequently, the event can occur
- We need to add an initial integrity constraint

Time spent

| TIME TO WRITE TEST CASES (IN MINUTES) | 2 |
| TIME TO COMPLETE THE ITERATION (IN MINUTES) | 6 |

Errors and failures that drive the conceptual modeling

<table>
<thead>
<tr>
<th>A basic type involved in a test case does not exist in the CSUD</th>
<th>A derived type involved in a test case does not exist in the CSUD</th>
<th>An event type involved in a test case does not exist in the CSUD</th>
</tr>
</thead>
<tbody>
<tr>
<td>The basic type is relevant and it is added to the CSUD</td>
<td>The derived type is relevant and it is added to the CSUD</td>
<td>The event type is relevant and it is added to the CSUD</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Inconsistent state before the occurrence of an event</th>
<th>Inconsistent state after the occurrence of an event</th>
<th>The postcondition of an event is not satisfied</th>
</tr>
</thead>
<tbody>
<tr>
<td>Some static constraint is invalid and it is modified.</td>
<td>Some initial integrity constraint is invalid and it is modified.</td>
<td>Some constraint is invalid and it is modified.</td>
</tr>
<tr>
<td>The event postcondition/method is incorrect and it is modified.</td>
<td>Some constraint is invalid and it is modified.</td>
<td>The method is not correct and it is modified.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>An assertion about the IB state fails or contains an error</th>
<th>Assert non-occurrence fails</th>
<th>Semantic error in an expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>The effect of an event type is not correct</td>
<td>A derivation rule is incorrect</td>
<td>The expression is corrected</td>
</tr>
<tr>
<td>A precondition is added/updated</td>
<td>The postcondition is not correct and it is modified.</td>
<td>The CSUD is changed</td>
</tr>
</tbody>
</table>

Iteration 18

Iteration objective

S3: NewTicketOnline_successScenario_RandomTicketsNumber_StaffNotificationsDisabled

Current test case

test S3{
  load testConfiguration8;
  nt1:=new NewTicketOnline;
  nt1.fullName:='Mary Marnes';
}
nt1.email := 'mary_at_marnes.mar';
nt1.telephone := 'xxxxxxxx';
nt1.ext := 'xxxxxxxx';
nt1.priority := #Normal;
nt1.helpTopic := helpTopicInstallation;
nt1.subject := 'Error operating system';
t1.message := 'The installation process does not finish....';
assert occurrence nt1;
ticket1 := nt1.createdTicket;
assert equals ticket1.number 5;
//5 is the aleatory number specified for testing purposes
assert equals ticket1.assignedDepartment dptTechnical;

//no autoresponses
assert true [not(EEmail.allInstances()->exists(e|e.emailKind.oclIsTypeOf(NewTicketAutoresponse)))];

//notice to administrator
assert false [EEmail.allInstances()->exists(e|e.emailKind.oclIsTypeOf(NewTicketAlertToStaff) and
  e.fromAddress='general_at_support.com' and
  e.toAddress='system_at_support.com' and
  e.ticketNumber=5)];

//notice to department manager
assert false [EEmail.allInstances()->exists(e|e.emailKind.oclIsTypeOf(NewTicketAlertToStaff) and
  e.fromAddress='general_at_support.com' and
  e.toAddress='john_at_support.com' and
  e.ticketNumber=5)];

//notice to department members
assert false [EEmail.allInstances()->exists(e|e.emailKind.oclIsTypeOf(NewTicketAlertToStaff) and
  e.fromAddress='general_at_support.com' and
  e.toAddress='martin_at_support.com' and
  e.ticketNumber=5)];

}

**TDCM application: Summary of changes performed in the schema**

The CSUD has not been changed

**Time spent**

<table>
<thead>
<tr>
<th>TIME TO WRITE TEST CASES (IN MINUTES)</th>
<th>2,5</th>
</tr>
</thead>
<tbody>
<tr>
<td>TIME TO COMPLETE THE ITERATION (IN MINUTES)</td>
<td>0,5</td>
</tr>
</tbody>
</table>

**Iteration 19**

**Iteration objective**

S4: NewTicketOnline_SuccessScenario_SequentialTicketsNumber_AutoresponsesDisabled
Current test case

test S4{
load testConfiguration3;

nt1:=new NewTicketOnline;
nt1.fullName:='James Jordan';
nt1.email:='james_at_jordan.jam';
nt1.telephone:='xxxxxxxx';
nt1.ext:='xxxxxxxx';
nt1.helpTopic:=helpTopicUse;
nt1.subject:='Reopening ticket';
nt1.message:='I do not know how to reopen one of my closed tickets';
assert occurrence nt1;

ticket1:=nt1.createdTicket;

assert equals ticket1.assignedDepartment dptGeneral;

//no autoreponses
assert true [not(EMail.allInstances()->exists(e|e.emailKind.oclIsTypeOf(NewTicketAutoresponse)))];

//notice to administrator
assert true [EMail.allInstances()->exists(e|e.emailKind.oclIsTypeOf(NewTicketAlertToStaff) and
  e.fromAddress='general_at_support.com' and
  e.toAddress='system_at_support.com' and
  e.ticketNumber=1)];

//notice to department members
assert true [EMail.allInstances()->exists(e|e.emailKind.oclIsTypeOf(NewTicketAlertToStaff) and
  e.fromAddress='general_at_support.com' and
  e.toAddress='john_at_support.com' and
  e.ticketNumber=1)];

assert true [EMail.allInstances()->exists(e|e.emailKind.oclIsTypeOf(NewTicketAlertToStaff) and
  e.fromAddress='general_at_support.com' and
  e.toAddress='mary_at_support.com' and
  e.ticketNumber=1)];

assert false [EMail.allInstances()->exists(e|e.emailKind.oclIsTypeOf(NewTicketAlertToStaff) and
  e.fromAddress='general_at_support.com' and
  e.toAddress='david_at_support.com' and
  e.ticketNumber=1)];
}

TDCM application: Summary of changes performed in the schema

The CSUD has not been changed

Time spent

| TIME TO WRITE TEST CASES (IN MINUTES) | 2 |
| TIME TO COMPLETE THE ITERATION (IN MINUTES) | 0.5 |
DEVELOPMENT OF THE CONCEPTUAL SCHEMA OF THE osTICKET SYSTEM BY APPLYING TDCM

Iteration 20

Iteration objective

S5: NewTicketOnline_SuccessScenario_NoTopic (assignment to the default department and with the default priority)

Current test case

```plaintext
test S5{
    load testConfiguration4;
    helpTopicUse.status:=#Disabled;
    helpTopicInstallation.status:=#Disabled;
    nt1:=new NewTicketOnline;
    nt1.fullName:='James Jordan';
    nt1.email:='james_at_jordan.jam';
    nt1.subject:='Reopening ticket';
    nt1.message:='I do not know how to reopen one of my closed tickets';
    assert occurrence nt1;
    ticket1:=nt1.createdTicket;
    assert equals ticket1.assignedDepartment dptGeneral;
    assert true ticket1.helpTopic->isEmpty();
    assert equals ticket1.priority #Normal;
    // no autoresponses
    assert false [EMail.allInstances()->exists(e|e.emailKind.oclIsTypeOf(NewTicketAutoresponse))];
    // notice to administrator
    assert false [EMail.allInstances()->exists(e|e.emailKind.oclIsTypeOf(NewTicketAlertToStaff) and
        e.fromAddress='general_at_support.com' and
        e.toAddress='system_at_support.com' and
        e.ticketNumber=1)];
    // notice to department members
    assert false [EMail.allInstances()->exists(e|e.emailKind.oclIsTypeOf(NewTicketAlertToStaff) and
        e.fromAddress='general_at_support.com' and
        e.toAddress='john_at_support.com' and
        e.ticketNumber=1)];
    assert false [EMail.allInstances()->exists(e|e.emailKind.oclIsTypeOf(NewTicketAlertToStaff) and
        e.fromAddress='general_at_support.com' and
        e.toAddress='mary_at_support.com' and
        e.ticketNumber=1)];
    assert false [EMail.allInstances()->exists(e|e.emailKind.oclIsTypeOf(NewTicketAlertToStaff) and
        e.fromAddress='general_at_support.com' and
        e.toAddress='david_at_support.com' and
        e.ticketNumber=1)];
}
```

TDCM application: Summary of changes performed in the schema

- Added
- Updated

- [TicketsManagementAndTracking.cstl] <line 683> Inconsistent state before nt1:NewTicketOnline event execution: Multiplicity constraint violation in association 'newTicketOnline_helpTopic'. Object 'oid202' of class 'NewTicketOnline' is connected to 0 objects of class 'HelpTopic' but the multiplicity is specified as '1'.

Department of Service and Information System Engineering
Help topics must be indicated only when there are available help topics

```plaintext
association newTicketOnline_helpTopic between
    NewTicketOnline[*]
    HelpTopic[0..1]
end

context NewTicketOnline ini inv helpTopicSpecifiedIfAvailable:
    if HelpTopic.allInstances()->select(hp|hp.status=#Enabled)->size()>0 then
        self.helpTopic->size()==1
    else
        self.helpTopic->size()==0
    endif
```

- [TicketsManagementAndTracking.cstl] <line 683> Inconsistent state before nt1.NewTicketOnline event execution: Instances of NewTicketOnline violate the invariant minMultiplicityOfAttributeExt

- We realize that the attribute Ext is optional

- The same happens for the attributes telephone

```plaintext
class Ticket
attributes
...
    telephone:String [0..1]
    ext:String[0..1]
...
end
```

- [TicketsManagementAndTracking.cstl] <Line 689>:1:7: Undefined operation named 'helpTopic' in expression '{Ticket}.helpTopic()'.

- We realize that the help topic is not assigned to the ticket

```plaintext
association ticket_helpTopic between
    Ticket[*]
    HelpTopic[0..1]
end

context NewTicketOnline::effect()
post:
...
(Ticket.allInstances- Ticket.allInstances@pre)
->one(t | t.oclIsNew() 
    and t.helpTopic=self.helpTopic 
    ... 
```

- [TicketsManagementAndTracking.cstl] postcondition 'post337' of NewTicketOnline is false

- The method needs to be changed according to the new postcondition

**Time spent**

| TIME TO WRITE TEST CASES (IN MINUTES) | 3,5 |
| TIME TO COMPLETE THE ITERATION (IN MINUTES) | 8 |
Errors and failures that drive the conceptual modeling

<table>
<thead>
<tr>
<th>A basic type involved in a test case does not exist in the CSUD</th>
<th>A derived type involved in a test case does not exist in the CSUD</th>
<th>An event type involved in a test case does not exist in the CSUD</th>
</tr>
</thead>
<tbody>
<tr>
<td>The basic type is relevant and it is added to the CSUD.</td>
<td>The derived type is relevant and it is added to the CSUD.</td>
<td>The event type is relevant and it is added to the CSUD.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Inconsistent state before the occurrence of an event</th>
<th>Inconsistent state after the occurrence of an event</th>
<th>The postcondition of an event is not satisfied.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Some static constraint is invalid and it is modified.</td>
<td>Some initial integrity constraint is invalid and it is modified.</td>
<td>The event postcondition/method is incorrect and it is modified.</td>
</tr>
<tr>
<td>The event postcondition/method is incorrect and it is modified.</td>
<td>Some constraint is invalid and it is modified.</td>
<td>The method is not correct and it is modified.</td>
</tr>
<tr>
<td>The postcondition is not correct and it is modified.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>2</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>An assertion about the IB state fails or contains an error</td>
<td>Assert non-occurrence fails</td>
</tr>
<tr>
<td>The effect of an event type is not correct</td>
<td>A derivation rule is incorrect</td>
</tr>
<tr>
<td>A precondition is added/updated</td>
<td>The CSUD is changed</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Assert consistency fails</td>
</tr>
<tr>
<td>A static constraint needs to be changed</td>
</tr>
</tbody>
</table>

---

### Iteration 21

**Iteration objective**

S6: NewTicketOnline extension (disabled topic)

**Current test case**

```java
test S6{
    load testConfiguration5;
    nt1:=new NewTicketOnline;
    nt1.fullName:='James Jordan';
    nt1.email:='james_at_jordan.jam';
    nt1.subject:='Reopening ticket';
    nt1.message:='I do not know how to reopen one of my closed tickets';
    nt1.helpTopic:=helpTopicDisabled;
    nt1.priority:=#Low;
    assert non-occurrence nt1;
}
```

**TDCM application: Summary of changes performed in the schema**

- Added
- Updated

- Preconditions of the domain event nt1:NewTicketOnline are satisfied and consequently, the event can occur

```java
context NewTicketOnline ini inv helpTopicsEnabled;
self.helpTopic->notEmpty() implies self.helpTopic.status=#Enabled
```
Time spent

<table>
<thead>
<tr>
<th>Errors and failures that drive the conceptual modeling</th>
</tr>
</thead>
<tbody>
<tr>
<td>A basic type involved in a test case does not exist in the CSUD</td>
</tr>
<tr>
<td>The basic type is relevant and it is added to the CSUD</td>
</tr>
<tr>
<td>Inconsistent state before the occurrence of an event</td>
</tr>
<tr>
<td>Some static constraint is invalid and it is modified.</td>
</tr>
<tr>
<td>The event postcondition/method is incorrect and it is modified.</td>
</tr>
<tr>
<td>The postcondition is not satisfied.</td>
</tr>
<tr>
<td>An assertion about the IB state fails or contains an error</td>
</tr>
<tr>
<td>The effect of an event type is not correct</td>
</tr>
<tr>
<td>Assert consistency fails</td>
</tr>
<tr>
<td>A static constraint needs to be changed</td>
</tr>
</tbody>
</table>

Iteration 22

Iteration objective

S7: NewTicketOnline_PreconditionViolation (online mode is disabled)

Current test case

```java
test S7{
load testConfiguration9;
nt1:=new NewTicketOnline;
n1.fullName:='James Jordan';
n1.email:='james_at_jordan.jam';
n1.subject:='Reopening ticket';
n1.message:='I do not know how to reopen one of my closed tickets';
n1.helpTopic:=helpTopicDisabled;
n1.priority:=#Low;
assert non-occurrence nt1;
}
```
TDCM application: Summary of changes performed in the schema

- Preconditions of the domain event nt1:NewTicketOnline are satisfied and consequently, the event can occur

```java
context NewTicketOnline ini inv helpDeskStatusIsOnline:
    GeneralSettings.allInstances()->any(true).status=#Online
```

Time spent

<table>
<thead>
<tr>
<th>TIME TO WRITE TEST CASES (IN MINUTES)</th>
<th>1.5</th>
</tr>
</thead>
<tbody>
<tr>
<td>TIME TO COMPLETE THE ITERATION (IN MINUTES)</td>
<td>3</td>
</tr>
</tbody>
</table>

Errors and failures that drive the conceptual modeling

<table>
<thead>
<tr>
<th>A basic type involved in a test case does not exist in the CSUD</th>
<th>A derived type involved in a test case does not exist in the CSUD</th>
<th>An event type involved in a test case does not exist in the CSUD</th>
</tr>
</thead>
<tbody>
<tr>
<td>The basic type is relevant and it is added to the CSUD</td>
<td>The derived type is relevant and it is added to the CSUD</td>
<td>The event type is relevant and it is added to the CSUD</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Inconsistent state before the occurrence of an event</th>
<th>Inconsistent state after the occurrence of an event</th>
<th>The postcondition of an event is not satisfied.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Some static constraint is invalid and it is modified.</td>
<td>Some initial integrity constraint is invalid and it is modified.</td>
<td>The event postcondition/method is incorrect and it is modified.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Some constraint is invalid and it is modified.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The method is not correct and it is modified.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The postcondition is not correct and it is modified.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>An assertion about the IB state fails or contains an error</th>
<th>Assert non-occurrence fails</th>
<th>Semantic error in an expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>The effect of an event type is not correct</td>
<td>A derivation rule is incorrect</td>
<td>The expression is corrected</td>
</tr>
<tr>
<td></td>
<td>A precondition is added/updated</td>
<td>The CSUD is changed</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Assert consistency fails</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>A static constraint needs to be modified</td>
<td></td>
</tr>
</tbody>
</table>

Iteration 23

Iteration objective

S8: NewTicketByEmail_successScenario_generalEmail

Current test case

```java
test S8{
    load testConfiguration3;
    nt1:=new NewTicketByEmail;
    nt1.toAddress:='general_at_support.com';
    nt1.fromName:='James Jordan';
    nt1.fromAddress:='james_at_jordan.jam';
    nt1.subject:='Ticket priority';
    nt1.message:='How can I change the priority of one of my tickets?';
```
assert occurrence nt1;

ticket1:=nt1.createdTicket;

assert equals ticket1.number 1;
assert equals ticket1.subject 'Ticket priority';
assert equals ticket1.priority #Low;
assert true ticket1.assignedStaff->isEmpty();
assert equals ticket1.source #EMail;
assert equals ticket1.creationDatetime sys.currentDateTime;
assert true ticket1.dueDatetime.isUndefined();
assert true ticket1.lastResponseDatetime.isUndefined();
assert equals ticket1.assignedDepartment dptGeneral;

assert true [ticket1.ticketThreadMessage->one(m|m.datetime=sys.currentDateTime and m.text='How can I change the priority of one of my tickets?' and m.author='James Jordan')];

//autoresponses
assert true [EMail.allInstances()->exists(e|e.emailKind.oclIsTypeOf(NewTicketAutoresponse) and e.fromAddress='general_at_support.com' and e.toAddress='james_at_jordan.jam' and e.ticketNumber=1)];

//notice to department members
assert true [EMail.allInstances()->exists(e|e.emailKind.oclIsTypeOf(NewTicketAlertToStaff) and e.fromAddress='general_at_support.com' and e.toAddress='john_at_support.com' and e.ticketNumber=1)];
assert true [EMail.allInstances()->exists(e|e.emailKind.oclIsTypeOf(NewTicketAlertToStaff) and e.fromAddress='general_at_support.com' and e.toAddress='mary_at_support.com' and e.ticketNumber=1)];
assert false [EMail.allInstances()->exists(e|e.emailKind.oclIsTypeOf(NewTicketAlertToStaff) and e.fromAddress='general_at_support.com' and e.toAddress='david_at_support.com' and e.ticketNumber=1)];

TDCM application: Summary of changes performed in the schema

- Added
- Updated

- [TicketsManagementAndTracking.cstl] <line 746> <line 746> NewTicketByEmail is not defined in the CSUT as a class or an association

    event NewTicketByEmail
    attributes
        toAddress: String
        fromName: String
        fromAddress: String
        subject: String
        message: String
    operations
        effect()
    end

- [TicketsManagementAndTracking.cstl] <line 752> The effect of NewTicketByEmail is not defined as a method (specify/review first its postconditions)event NewTicketByEmail

    context NewTicketByEmail::effect()
    post:
        let incomingEmailAccount: EmailAccount =
            EmailAccount.allInstances() -> any(e|e.address = self.fromAddress)
        in
let assignedPriority:Priority=
  if TicketSettings.allInstances()->any(true).useEmailPriorityWhenAvailable then
    incomingEmailAccount.defaultNewPriority
  else
    TicketSettings.allInstances()->any(true).priority
  endif
in
let defaultDepartment:Department=
  incomingEmailAccount.defaultNewTicketDepartment
in
let sendNewTicketAlertToAdministrator:Boolean=
  StaffNoticesAlertsSettings.allInstances()->any(true).alertWhenNewTicketCreated and
  StaffNoticesAlertsSettings.allInstances()->any(true).alertWhenNewTicketCreatedStaff->includes(#Administrator)
in
let sendNewTicketAlertToDepartmentManager:Boolean=
  StaffNoticesAlertsSettings.allInstances()->any(true).alertWhenNewTicketCreated and
  StaffNoticesAlertsSettings.allInstances()->any(true).alertWhenNewTicketCreatedStaff->includes(#DepartmentManager)
in
let sendNewTicketAlertToDepartmentMembers:Boolean=
  StaffNoticesAlertsSettings.allInstances()->any(true).alertWhenNewTicketCreated and
  StaffNoticesAlertsSettings.allInstances()->any(true).alertWhenNewTicketCreatedStaff->>includes(#DepartmentMembers)
in
let staffAlertsFromEMailAddress:String=
  EmailSettings.allInstances()->any(true).defaultStaffAlertsEmail.address
in
let autoresponsesEMailAddress:String=
  EmailSettings.allInstances()->any(true).defaultStaffAlertsEmail.address
in
let sendAutoresponse: Boolean=
  if (incomingEmailAccount.autoresponsesStatus=#Enabled) then true
  else false
  endif
in
(Ticket.allInstances- Ticket.allInstances@pre)
  ->one( t | t.oclIsNew() and self.createdTicket=t and t.fullName=self.fromName and t.email=self.fromAddress and t.subject=self.subject and t.message=self.message and t.ticketStatus=#Open and t.priority=assignedPriority and t.source=#EMail and t.assignedDepartment=incomingEmailAccount.defaultNewTicketDepartment and (TicketThreadMessage.allInstances- TicketThreadMessage.allInstances@pre)
  ->one(tdm | tdm.oclIsNew() and tdm.datetime=System.allInstances()->any(true).currentDateTime and tdm.text=self.message and tdm.author=self.fromName and tdm.ticket=t and t.lastMessageDatetime=System.allInstances()->any(true).currentDateTime)
  -- autoresponses and (sendAutoresponse implies EMail.allInstances()->exists(e|e.fromAddress=t.assignedDepartment.autoresponseEmail.address and e.toAddress=t.email and e.ticketNumber=t.number))
  -- staff notices and (sendNewTicketAlertToAdministrator implies EMail.allInstances()->exists(e|e.fromAddress=staffAlertsFromEMailAddress and e.toAddress=EmailSettings.allInstances()->any(true).administrationEmail and e.ticketNumber=t.number))
and (sendNewTicketAlertToDepartmentManager and t.assignedDepartment.departmentManager->notEmpty() and t.assignedDepartment.departmentManager.status=#Enabled and not(t.assignedDepartment.departmentManager.isInVacationMode) implies
EMail.allInstances()->exists(e|e.fromAddress=staffAlertsFromEMailAddress and
Development of the Conceptual Schema of the osTicket System by Applying TDCM

Department of Service and Information System Engineering

Research report
April 2011

\[ e.toAddress=t.assignedDepartment.departmentManager.emailAddress \text{ and } e.ticketNumber=t.number \]

\[ \text{and (sendNewTicketAlertToDepartmentMembers} \]  
\[ \text{implies t.assignedDepartment.staffMember->forAll(m|} \]  
\[ \text{(m.status=#Enabled and not(m.isInVacationMode))} \]  
\[ \text{implies EMail.allInstances()->exists(e|e.fromAddress=staffAlertsFromEMailAddress and} \]  
\[ \text{e.toAddress=m.emailAddress and} \]  
\[ \text{e.ticketNumber=t.number)})} \]

\[
\text{context NewTicketByEmail ini inv maximumOpenTicketsLimitIsNotViolated:}
\begin{align*}
\text{if TicketSettings.allInstances()->any(true).openTicketsPerMailAreLimited then} \\
\text{Ticket.allInstances->select(t|t.email=self.fromAddress)->size() not(TicketSettings.allInstances())} \\
\text{-any(true).maximumOpenTicketsPerMail}
\end{align*}
\text{else true}
\]

- osTicketCSUT.use:586:26: Undefined enumeration literal `EMail'.

- enum TicketSource{Web,Email}

- [TicketsManagementAndTracking.cstl] <line 752> The effect of NewTicketByEmail is not defined as a method (specify/review first its postconditions)

Time spent

| TIME TO WRITE TEST CASES (IN MINUTES) | 7 |
| TIME TO COMPLETE THE ITERATION (IN MINUTES) | 26 |

Errors and failures that drive the conceptual modeling

<table>
<thead>
<tr>
<th>A basic type involved in a test case does not exist in the CSUD</th>
<th>A derived type involved in a test case does not exist in the CSUD</th>
<th>An event type involved in a test case does not exist in the CSUD</th>
</tr>
</thead>
<tbody>
<tr>
<td>The basic type is relevant and it is added to the CSUD</td>
<td>The derived type is relevant and it is added to the CSUD</td>
<td>The event type is relevant and it is added to the CSUD</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Inconsistent state before the occurrence of an event</th>
<th>Inconsistent state after the occurrence of an event</th>
<th>The postcondition of an event is not satisfied.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Some static constraint is invalid and it is modified.</td>
<td>Some initial integrity constraint is invalid and it is modified.</td>
<td>The event precondition/method is incorrect and it is modified.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Some constraint is invalid and it is modified.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The method is not correct and it is modified.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The postcondition is not correct and it is modified.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The postcondition is not satisfied.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The postcondition is not satisfied.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The postcondition is not satisfied.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The postcondition is not satisfied.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The postcondition is not satisfied.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The postcondition is not satisfied.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The postcondition is not satisfied.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The postcondition is not satisfied.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The postcondition is not satisfied.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The postcondition is not satisfied.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The postcondition is not satisfied.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The postcondition is not satisfied.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The postcondition is not satisfied.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The postcondition is not satisfied.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The postcondition is not satisfied.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The postcondition is not satisfied.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The postcondition is not satisfied.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The postcondition is not satisfied.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The postcondition is not satisfied.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The postcondition is not satisfied.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The postcondition is not satisfied.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The postcondition is not satisfied.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The postcondition is not satisfied.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The postcondition is not satisfied.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The postcondition is not satisfied.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The postcondition is not satisfied.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The postcondition is not satisfied.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The postcondition is not satisfied.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The postcondition is not satisfied.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The postcondition is not satisfied.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The postcondition is not satisfied.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The postcondition is not satisfied.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The postcondition is not satisfied.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The postcondition is not satisfied.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>An assertion about the IB state fails or contains an error</th>
<th>Assert non-occurrence fails</th>
<th>Semantic error in an expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>The effect of an event type is not correct</td>
<td>A derivation rule is incorrect</td>
<td>A precondition is added/updated</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The expression is corrected</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The CSUD is changed</td>
</tr>
</tbody>
</table>

| Assert consistency fails | A static constraint needs to be changed | |
|---|---|

Errors and failures that drive the conceptual modeling
Iteration 24

Iteration objective

S9: NewTicketByEmail_successScenario_technicalEmail

Current test case

test S9{
    load testConfiguration10;
    nt1:=new NewTicketByEmail;
    nt1.toAddress:='technical_at_support.com';
    nt1.fromName:='Marta Johnes';
    nt1.fromAddress:='marta_at_johnes.mar';
    nt1.subject:='See my tickets';
    nt1.message:='Can I see my tickets?';
    assert non-occurrence nt1;

    generalSettings.status:=#Online;
    assert occurrence nt1;

    ticket1:=nt1.createdTicket;
    assert equals ticket1.number 1;
    assert equals ticket1.ticketStatus #Open;
    assert equals ticket1.assignedDepartment dptTechnical;
    assert equals ticket1.subject 'See my tickets';
    assert equals ticket1.priority #High;
    assert true ticket1.assignedStaff->isEmpty();
    assert equals ticket1.source #EMail;
    assert equals ticket1.creationDateTime sys.currentDateTime;
    assert true ticket1.dueDateTime.isUndefined();
    assert true ticket1.lastResponseDateTime.isUndefined();

    assert true [ticket1.ticketThreadMessage->one(m|m.datetime=sys.currentDateTime and
    m.text='Can I see my tickets?' and
    m.author='Marta Johnes')];

    //autoresponses
    assert false [EMail.allInstances()->exists(e|e.emailKind.oclIsTypeOf(NewTicketAutoresponse) and
    e.fromAddress='technical_at_support.com' and
    e.toAddress='marta_at_johnes.mar' and
    e.ticketNumber=1)];

    //notice to department members
    assert false [EMail.allInstances()->exists(e|e.emailKind.oclIsTypeOf(NewTicketAlertToStaff) and
    e.fromAddress='general_at_support.com' and
    e.toAddress='john_at_support.com' and
    e.ticketNumber=1)];

    assert false [EMail.allInstances()->exists(e|e.emailKind.oclIsTypeOf(NewTicketAlertToStaff) and
    e.fromAddress='general_at_support.com' and
    e.toAddress='mary_at_support.com' and
    e.ticketNumber=1)];

    assert false [EMail.allInstances()->exists(e|e.emailKind.oclIsTypeOf(NewTicketAlertToStaff) and
    e.fromAddress='general_at_support.com' and
    e.toAddress='david_at_support.com' and
    e.ticketNumber=1)];

    assert false [EMail.allInstances()->exists(e|e.emailKind.oclIsTypeOf(NewTicketAlertToStaff) and
    e.fromAddress='general_at_support.com' and
    e.toAddress='david_at_support.com' and
    e.ticketNumber=1)];
}

}
TDCM application: Summary of changes performed in the schema

- Added
  - Preconditions of the domain event nt1: NewTicketByEmail are satisfied and consequently, the event can occur

  ```
  context NewTicketByEmail ini inv helpDeskStatusIsOnline:
  GeneralSettings.allInstances()->any(true).status=#Online
  ```

**Time spent**

<table>
<thead>
<tr>
<th>TIME TO WRITE TEST CASES (IN MINUTES)</th>
<th>4.5</th>
</tr>
</thead>
<tbody>
<tr>
<td>TIME TO COMPLETE THE ITERATION (IN MINUTES)</td>
<td>6</td>
</tr>
</tbody>
</table>

**Errors and failures that drive the conceptual modeling**

<table>
<thead>
<tr>
<th>A basic type involved in a test case does not exist in the CSUD</th>
<th>A derived type involved in a test case does not exist in the CSUD</th>
<th>An event type involved in a test case does not exist in the CSUD</th>
</tr>
</thead>
<tbody>
<tr>
<td>The basic type is relevant and it is added to the CSUD</td>
<td>The derived type is relevant and it is added to the CSUD</td>
<td>The event type is relevant and it is added to the CSUD</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Inconsistent state before the occurrence of an event</th>
<th>Inconsistent state after the occurrence of an event</th>
<th>The postcondition of an event is not satisfied</th>
</tr>
</thead>
<tbody>
<tr>
<td>Some static constraint is invalid and it is modified.</td>
<td>Some initial integrity constraint is invalid and it is modified.</td>
<td>Some constraint is incorrect and it is modified.</td>
</tr>
<tr>
<td>The event postcondition/method is incorrect and it is modified.</td>
<td>The method is not correct and it is modified.</td>
<td>The postcondition is not correct and it is modified.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>An assertion about the IB state fails or contains an error</th>
<th>Assert non-occurrence fails</th>
<th>Semantic error in an expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>The effect of an event type is not correct</td>
<td>A derivation rule is incorrect</td>
<td>The expression is corrected</td>
</tr>
<tr>
<td>A precondition is added/updated</td>
<td>The postcondition is not correct and it is modified.</td>
<td>The CSUD is changed</td>
</tr>
</tbody>
</table>

**Assertion consistency fails**

- A static constraint needs to be changed

**Iteration 25**

**Iteration objective**

S10: NewTicketByEmail_extension2a (invalid incoming email address)

**Current test case**

```java
// Code block for the test case
```

Department of Service and Information System Engineering
TDCM application: Summary of changes performed in the schema

- Added

  Preconditions of the domain event nt1:NewTicketByEmail are satisfied and consequently, the event can occur

  context NewTicketByEmail ini inv theIncomingEmailsValid:
  EmailAccount.allInstances().one(address=self.toAddress)

Time spent

| TIME TO WRITE TEST CASES (IN MINUTES) | 3 |
| TIME TO COMPLETE THE ITERATION (IN MINUTES) | 5 |

Errors and failures that drive the conceptual modeling

<table>
<thead>
<tr>
<th>A basic type involved in a test case does not exist in the CSUD</th>
<th>A derived type involved in a test case does not exist in the CSUD</th>
<th>An event type involved in a test case does not exist in the CSUD</th>
</tr>
</thead>
<tbody>
<tr>
<td>The basic type is relevant and it is added to the CSUD</td>
<td>The derived type is relevant and it is added to the CSUD</td>
<td>The event type is relevant and it is added to the CSUD</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Inconsistent state before the occurrence of an event</th>
<th>Inconsistent state after the occurrence of an event</th>
<th>The postcondition of an event is not satisfied</th>
</tr>
</thead>
<tbody>
<tr>
<td>Some static constraint is invalid and it is modified.</td>
<td>Some initial integrity constraint is invalid and it is modified.</td>
<td>The postcondition/method is incorrect and it is modified.</td>
</tr>
<tr>
<td>The event postcondition/method is incorrect and it is modified.</td>
<td>Some constraint is invalid and it is modified.</td>
<td>The method is not correct and it is modified.</td>
</tr>
<tr>
<td>The postcondition is not correct.</td>
<td>The expression is incorrect.</td>
<td>The CSUD is changed.</td>
</tr>
</tbody>
</table>

An assertion about the IB state fails or contains an error

<table>
<thead>
<tr>
<th>Assert non-occurrence fails</th>
<th>Semantic error in an expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>The effect of an event type is not correct.</td>
<td>A derivation rule is incorrect.</td>
</tr>
<tr>
<td>A precondition is added/updated.</td>
<td>The expression is corrected.</td>
</tr>
<tr>
<td>The CSUD is changed.</td>
<td></td>
</tr>
</tbody>
</table>

Assert consistency fails

<table>
<thead>
<tr>
<th>A static constraint needs to be changed</th>
</tr>
</thead>
</table>

Iteration 26

Iteration objective

S11:DisplayTicketsOfEmail_successScenario

Current test case

```java
    test S11{
        load testConfiguration1;
        nt1:=new NewTicketOnline;
        nt1.fullName:='James Jordan';
        nt1.email:='james_at_jordan.jam';
    }
```
nt1.helpTopic:=helpTopicUse;
nt1.message:="I do not know how to reopen one of my closed tickets";
assert occurrence nt1;
ticket1:=nt1.createdTicket;

nt2:=new NewTicketOnline;
nt2.fullName:=James Jordan';
nt2.email:=james_at_jordan.jam';
nt2.helpTopic:=helpTopicInstallation;
nt2.subject:="Error operating system";
nt2.message:="The installation process does not finish...";
assert occurrence nt2;
ticket2:=nt2.createdTicket;

cts:=new DisplayTicketsAssociatedToEmail(email:='james_at_jordan.jam', ticketNumber:=2);
assert occurrence cts;
assert equals cts.answer()
[Set{Tuple{createDate=1,department=General support, email='james_at_jordan.jam', number=1, status=Open, subject='Reopening ticket'}], Tuple{createDate=1, department=Technical support, email='james_at_jordan.jam', number=2, status=Open, subject='Error operating system'}}];
}

TDCM application: Summary of changes performed in the schema

- [TicketsManagementAndTracking.cstl] <line 872> <line 872> DisplayTicketsAssociatedToEmail is not defined in the CSUT as a class or an association
- We define the query event until the verdict is Pass

```cstl
query DisplayTicketsAssociatedToEmail
attributes
ticketNumber:Integer
operations
answer():Set(Tuple(number:Integer,createDate:Integer,status:TicketStatus,subject:String,department:String, email:String))=
    Ticket.allInstances
    -> sortedBy(number)
    -> collect (t | Tuple {number : t.number , createDate : t.creationDatetime.value, status : t.ticketStatus , subject : t.subject , department : t.assignedDepartment.name, email : t.email})
End
```

Time spent

<table>
<thead>
<tr>
<th>Time to Write Test Cases (in minutes)</th>
<th>4.5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time to Complete the Iteration (in minutes)</td>
<td>11</td>
</tr>
</tbody>
</table>

Errors and failures that drive the conceptual modeling

<table>
<thead>
<tr>
<th>A basic type involved in a test case does not exist in the CSUD</th>
<th>A derived type involved in a test case does not exist in the CSUD</th>
<th>An event type involved in a test case does not exist in the CSUD</th>
</tr>
</thead>
<tbody>
<tr>
<td>The basic type is relevant and it is added to the CSUD</td>
<td>The derived type is relevant and it is added to the CSUD</td>
<td>The event type is relevant and it is added to the CSUD</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>
Inconsistent state before the occurrence of an event | Inconsistent state after the occurrence of an event | The postcondition of an event is not satisfied.
--- | --- | ---
Some static constraint is invalid and it is modified. | Some initial integrity constraint is invalid and it is modified. | The event postcondition/method is incorrect and it is modified.

An assertion about the IB state fails or contains an error | Assert non-occurrence fails | Semantic error in an expression
--- | --- | ---
The effect of an event type is not correct | A derivation rule is incorrect | A precondition is added/updated
A static constraint needs to be changed | | The expression is corrected

Iteration 27

Iteration objective

S12: DisplayTicketsOfEmail extension_2a (non existing tickets)

Current test case

```java
test S12{
    load testConfiguration1;

    cts:=new DisplayTicketsAssociatedToEmail(email:='james_at_jordan.jam', ticketNumber:=2);
    assert non-occurrence cts;
}
```

TDCM application: Summary of changes performed in the schema

- Added
- Updated

- Preconditions of the domain event cts:DisplayTicketsAssociatedToEmail are satisfied and consequently, the event can occur

  ```java
  context DisplayTicketsAssociatedToEmail ini inv thereAreTicketsAssociatedToTheEmail:
  Ticket.allInstances()->select(t|t.email=self.email)->size()>0
  ```

Time spent

- **TIME TO WRITE TEST CASES (IN MINUTES)** 0,5
- **TIME TO COMPLETE THE ITERATION (IN MINUTES)** 1,5

Errors and failures that drive the conceptual modeling

- A basic type involved in a test case does not exist in the CSUD
- A derived type involved in a test case does not exist in the CSUD
- An event type involved in a test case does not exist in the CSUD

- The basic type is relevant and it is added to the CSUD
- The derived type is relevant and it is added to the CSUD
- The event type is relevant and it is added to the CSUD
DEVELOPMENT OF THE CONCEPTUAL SCHEMA OF THE osTICKET SYSTEM BY APPLYING TDCM

Research report
April 2011

<table>
<thead>
<tr>
<th>Inconsistent state before the occurrence of an event</th>
<th>Inconsistent state after the occurrence of an event</th>
<th>The postcondition of an event is not satisfied.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Some static constraint is invalid and it is modified.</td>
<td>Some initial integrity constraint is invalid and it is modified.</td>
<td>The method is not correct and it is modified.</td>
</tr>
<tr>
<td></td>
<td>For event postcondition/method is incorrect and it is modified.</td>
<td>The postcondition of an event is not satisfied.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>An assertion about the IB state fails or contains an error</th>
<th>Assert non-occurrence fails</th>
<th>Semantic error in an expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>The effect of an event type is not correct</td>
<td>A derivation rule is incorrect</td>
<td>The expression is corrected</td>
</tr>
<tr>
<td></td>
<td>A precondition is added/updated</td>
<td>The CSUD is changed</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Assert consistency fails</th>
<th>A static constraint needs to be changed</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
</tr>
</tbody>
</table>

Iteration 28

Iteration objective

S13: DisplayTicketsOfEmail extension_2a Invalid access data

Current test case

- Preconditions of the domain event cts:DisplayTicketsAssociatedToEmail are satisfied and consequently, the event can occur

  context DisplayTicketsAssociatedToEmail ini inv thereAreTicketsAssociatedToTheEmail: Ticket.allInstances().select(t|t.email=self.email)->size()>0

TDCM application: Summary of changes performed in the schema

- Added
- Updated

context DisplayTicketsAssociatedToEmail ini inv accessDataIsValid: Ticket.allInstances().select(t|t.email=self.email).number->includes(self.ticketNumber)

Time spent

<table>
<thead>
<tr>
<th>TIME TO WRITE TEST CASES (IN MINUTES)</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>TIME TO COMPLETE THE ITERATION (IN MINUTES)</td>
<td>2</td>
</tr>
</tbody>
</table>

Errors and failures that drive the conceptual modeling

- A basic type involved in a test case does not exist in the CSUD
- A derived type involved in a test case does not exist in the CSUD
- An event type involved in a test case does not exist in the CSUD

- The basic type is relevant and it is added to the CSUD
- The derived type is relevant and it is added to the CSUD
- The event type is relevant and it is added to the CSUD
### Inconsistent state before the occurrence of an event

<table>
<thead>
<tr>
<th>Some static constraint is invalid and it is modified.</th>
<th>The event postcondition/method is incorrect and it is modified.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Some initial integrity constraint is invalid and it is modified.</td>
<td></td>
</tr>
</tbody>
</table>

### Inconsistent state after the occurrence of an event

<table>
<thead>
<tr>
<th>Some constraint is invalid and it is modified.</th>
</tr>
</thead>
<tbody>
<tr>
<td>The method is not correct and it is modified.</td>
</tr>
</tbody>
</table>

### The postcondition of an event is not satisfied.

<table>
<thead>
<tr>
<th>The postcondition is not satisfied.</th>
</tr>
</thead>
<tbody>
<tr>
<td>The method is not correct and it is modified.</td>
</tr>
</tbody>
</table>

### An assertion about the IB state fails or contains an error

<table>
<thead>
<tr>
<th>Assert non-occurrence fails</th>
</tr>
</thead>
<tbody>
<tr>
<td>A derivation rule is incorrect</td>
</tr>
</tbody>
</table>

### Semantic error in an expression

<table>
<thead>
<tr>
<th>The expression is corrected</th>
</tr>
</thead>
<tbody>
<tr>
<td>The CSUD is changed</td>
</tr>
</tbody>
</table>

### Assertion consistency fails

<table>
<thead>
<tr>
<th>A static constraint needs to be changed</th>
</tr>
</thead>
</table>

#### Iteration 29

### Iteration objective

S14: DisplayTicketInformationForCustomer

### Current test case

```
test S14{
    load testConfiguration1;
    nt1:=new NewTicketOnline;
    nt1.fullName:='James Jordan';
    nt1.email:='james_at_jordan.jam';
    nt1.helpTopic:=helpTopicUse;
    nt1.subject:='Reopening ticket';
    nt1.message:='I do not know how to reopen one of my closed tickets';
    assert occurrence nt1;
    
    ticket1:=nt1.createdTicket;
    cr:=new ReplyTicketByCustomer(ticket:=ticket1,replyText:='Please help me');
    assert occurrence cr;

    assert true [EMail.allInstances()->exists(e|e.emailKind.oclIsTypeOf(NewMessageAlertToStaff) and e.fromAddress='general_at_support.com' and e.toAddress='john_at_support.com' and e.ticketNumber=1)];

    assert true [EMail.allInstances()->exists(e|e.emailKind.oclIsTypeOf(NewMessageAutoresponse) and e.fromAddress='technical_at_support.com' and e.toAddress='james_at_jordan.jam' and e.ticketNumber=1)];
}
```
TDCM application: Summary of changes performed in the schema

- Added
- Updated

[TicketsManagementAndTracking.cstl] <line 919> ReplyTicketByCustomer is not defined in the CSUT as a class or an association

```plaintext
event ReplyTicketByCustomer
  attributes
  replyText: String
  operations
effect()
end

association replyTicketByCustomer_ticket between
  ReplyTicketByCustomer[*]
  Ticket[1]
end

class ReplyTicketByCustomer::effect()
  post:
  let sendNewMessageAlertToLastRespondent:Boolean=
    StaffNoticesAlertsSettings.allInstances()->any(true).alertWhenNewMessage and
    StaffNoticesAlertsSettings.allInstances()->any(true).alertWhenNewMessageStaff->includes(#LastRespondent)
  in
  let sendNewMessageAlertToAssignedStaff:Boolean=
    StaffNoticesAlertsSettings.allInstances()->any(true).alertWhenNewMessage and
    StaffNoticesAlertsSettings.allInstances()->any(true).alertWhenNewMessageStaff->includes(#AssignedStaff)
  in
  let sendNewMessageAlertToDepartmentManager:Boolean=
    StaffNoticesAlertsSettings.allInstances()->any(true).alertWhenNewMessage and
    StaffNoticesAlertsSettings.allInstances()->any(true).alertWhenNewMessageStaff->includes(#DepartmentManager)
  in
  let staffAlertsFromEMailAddress:String=
    EmailSettings.allInstances()->any(true).defaultStaffAlertsEmail.address
  in
  let sendAutoresponse:Boolean=
    if CustomerAutoresponsesSettings.allInstances()->any(true).autorespondWhenNewTicketCreatedByCustomer then
      if self.ticket.helpTopic->notEmpty then
        if self.ticket.helpTopic.autoresponse=#Enabled then true
        else false
      end
    else false
  endif
end

self.ticket.ticketThreadMessage->one(tdm | tdm.oclIsNew() and tdm.datetime=System.allInstances()->any(true).currentDateTime and tdm.text=self.replyText and tdm.author=self.ticket.fullName and self.ticket.lastMessageDatetime=System.allInstances()->any(true).currentDateTime)

-- autoresponses
and (sendAutoresponse implies
  EMail.allInstances()->exists(e|e.fromAddress=self.ticket.assignedDepartment.autoresponseEmail.address and e.toAddress=self.ticket.email and e.ticketNumber=self.ticket.number))

-- staff notices
and (sendNewMessageAlertToLastRespondent and
  self.ticket.lastRespondent->notEmpty() and
  not(self.ticket.lastRespondent.isInVacationMode or self.ticket.lastRespondent.status=#Disabled)
) implies
  EMail.allInstances()->exists(e|e.fromAddress=staffAlertsFromEMailAddress and e.toAddress=self.ticket.lastRespondent and
```
Development of the Conceptual Schema of the osTicket System by Applying TDCM

April 2011

Time spent

| Time to Write Test Cases (in Minutes) | 4 |
| Time to Complete the Iteration (in Minutes) | 35 |

Errors and failures that drive the conceptual modeling

<table>
<thead>
<tr>
<th>A basic type involved in a test case does not exist in the CSUD</th>
<th>A derived type involved in a test case does not exist in the CSUD</th>
<th>An event type involved in a test case does not exist in the CSUD</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

Inconsistent state before the occurrence of an event

<table>
<thead>
<tr>
<th>Some static constraint is invalid and it is modified.</th>
<th>Some initial integrity constraint is invalid and it is modified.</th>
<th>Some event postcondition/method is incorrect and it is modified.</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Inconsistent state after the occurrence of an event

<table>
<thead>
<tr>
<th>Some constraint is invalid and it is modified.</th>
<th>The method is not correct and it is modified.</th>
<th>The postcondition is not correct and it is modified.</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>2</td>
<td>1</td>
</tr>
</tbody>
</table>

An assertion about the IB state fails or contains an error

<table>
<thead>
<tr>
<th>The effect of an event type is incorrect</th>
<th>A derivation rule is incorrect</th>
<th>A precondition is added/updated</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
</tbody>
</table>

Semantic error in an expression

<table>
<thead>
<tr>
<th>The expression is corrected</th>
<th>The CSUD is changed</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>1</td>
</tr>
</tbody>
</table>

Iteration 30

Iteration objective

S15: RespondTicket_alertsDisabled
Current test case

test S15{
load testConfiguration2;

nt1:=new NewTicketOnline;
nt1.fullName:='James Jordan';
nt1.email:='james_at_jordan.jam';
nt1.helpTopic:=helpTopicInstallation;
nt1.subject:='Reopening ticket';
nt1.message:='I do not know how to reopen one of my closed tickets';
assert occurrence nt1;

ticket1:=nt1.createdTicket;

cr:=new ReplyTicketByCustomer(ticket:=ticket1,replyText:='Please help me');
assert occurrence cr;

assert false [EMail.allInstances()->exists(e|e.emailKind.oclIsTypeOf(NewMessageAlertToStaff) and
    e.fromAddress='general_at_support.com' and
    e.toAddress='john_at_support.com' and
    e.ticketNumber=1)];
}

TDCM application: Summary of changes performed in the schema

The CSUD has not been changed

Time spent

| TIME TO WRITE TEST CASES (IN MINUTES) | 2          |
| TIME TO COMPLETE THE ITERATION (IN MINUTES) | 0,5        |

Iteration 31

Iteration objective

S16: StaffLogin_successScenario

Current test case

test S16{
load testConfiguration1;

assert false generalConsultant.isLoggedIn;
li := new LogIn(username:='mary', password:='yyy');
assert occurrence li;
assert true generalConsultant isLoggedIn;
TDCM application: Summary of changes performed in the schema

- [TicketsManagementAndTracking.cstl] <Line 956>:1:7: Undefined operation named 'isLoggedIn' in expression '{StaffMember}.isLoggedIn()'
  
  class StaffMember
  attributes
  username:String
  firstName:String
  lastName:String
  emailAddress:String
  officePhone:String[0..1]
  phoneExtension:String[0..1]
  mobilePhone:String[0..1]
  signature:String[0..1]
  password:String
  status:Status
  isAdministrator:Boolean
  isInVacationMode:Boolean
  isLoggedIn:Boolean=false
  end

- TicketsManagementAndTracking.cstl] <line 957> LogIn is not defined in the CSUT as a class or an association

  event LogIn
  attributes
  username:String
  password:String
  operations
effect()
end

  context LogIn::effect()
  post:
  StaffMember.allInstances()->any(sm|sm.username=self.username and sm.password=self.password).isLoggedIn=true

  context StaffMember inv usernameIsUnique:
  StaffMember.allInstances()->isUnique(username)

Time spent

| TIME TO WRITE TEST CASES (IN MINUTES) | 1.5 |
| TIME TO COMPLETE THE ITERATION (IN MINUTES) | 7 |

Errors and failures that drive the conceptual modeling

<table>
<thead>
<tr>
<th>A basic type involved in a test case does not exist in the CSUD</th>
<th>A derived type involved in a test case does not exist in the CSUD</th>
<th>An event type involved in a test case does not exist in the CSUD</th>
</tr>
</thead>
<tbody>
<tr>
<td>The basic type is relevant and it is added to the CSUD</td>
<td>The derived type is relevant and it is added to the CSUD</td>
<td>The event type is relevant and it is added to the CSUD</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Inconsistent state before the occurrence of an event</th>
<th>Inconsistent state after the occurrence of an event</th>
<th>The postcondition of an event is not satisfied</th>
</tr>
</thead>
<tbody>
<tr>
<td>Some static constraint is invalid and it is modified.</td>
<td>Some initial integrity constraint is invalid and it is modified.</td>
<td>Some constraint is invalid and it is modified.</td>
</tr>
<tr>
<td></td>
<td>The event postcondition/method is incorrect and it is modified.</td>
<td>The method is not correct and it is modified.</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

| The method is not correct and it is modified. | The postcondition is not correct and it is modified. | |
DEVELOPMENT OF THE CONCEPTUAL SCHEMA OF THE osTICKET SYSTEM BY APPLYING TDCM

An assertion about the IB state fails or contains an error  Assert non-occurrence fails  Semantic error in an expression

<table>
<thead>
<tr>
<th>The effect of an event type is not correct</th>
<th>A derivation rule is incorrect</th>
<th>A precondition is added/updated</th>
<th>The expression is corrected</th>
<th>The CSUD is changed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Assert consistency fails</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A static constraint needs to be changed</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Iteration 32

Iteration objective

S17: StaffLogIn_PreconditionViolation

Current test case

```java

test S17{
load testConfiguration1;

li := new LogIn(username:='mary', password:='yyy');
asst occurrence li;
asst true generalConsultant.isLoggedIn;

li := new LogIn(username:='mary', password:='yyy');
asst non-occurrence li;
}
```

TDCM application: Summary of changes performed in the schema

- **Added**
- **Updated**

- Preconditions of the domain event li:LogIn are satisfied and consequently, the event can occur

```java
context LogIn inv isNotLoggedIn:
StaffMember.allInstances()->select(sm|sm.username=self.username and sm.password=self.password)->size()>0
implies
StaffMember.allInstances()->any(sm|sm.username=self.username and sm.password=self.password).isLoggedIn=false
```

Time spent

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>TIME TO WRITE TEST CASES (IN MINUTES)</td>
<td>2</td>
</tr>
<tr>
<td>TIME TO COMPLETE THE ITERATION (IN MINUTES)</td>
<td>5</td>
</tr>
</tbody>
</table>
Errors and failures that drive the conceptual modeling

<table>
<thead>
<tr>
<th>A basic type involved in a test case does not exist in the CSUD</th>
<th>A derived type involved in a test case does not exist in the CSUD</th>
<th>An event type involved in a test case does not exist in the CSUD</th>
</tr>
</thead>
<tbody>
<tr>
<td>The basic type is relevant and it is added to the CSUD</td>
<td>The derived type is relevant and it is added to the CSUD</td>
<td>The event type is relevant and it is added to the CSUD</td>
</tr>
</tbody>
</table>

Inconsistent state before the occurrence of an event

<table>
<thead>
<tr>
<th>Inconsistent state after the occurrence of an event</th>
<th>The postcondition of an event is not satisfied.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Some static constraint is invalid and it is modified.</td>
<td>The event postcondition/method is incorrect and it is modified.</td>
</tr>
<tr>
<td>Some initial integrity constraint is invalid and it is modified.</td>
<td>Some constraint is invalid and it is modified.</td>
</tr>
<tr>
<td>The event is relevant and it is added to the CSUD</td>
<td>The method is not correct and it is modified.</td>
</tr>
</tbody>
</table>

An assertion about the IB state fails or contains an error

<table>
<thead>
<tr>
<th>Assert non-occurrence fails</th>
<th>Semantic error in an expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>The effect of an event type is not correct</td>
<td>A derivation rule is incorrect</td>
</tr>
<tr>
<td>A precondition is added/updated</td>
<td>The expression is corrected</td>
</tr>
<tr>
<td>The CSUD is changed</td>
<td></td>
</tr>
</tbody>
</table>

Assert consistency fails

A static constraint needs to be changed

Iteration 33

Iteration objective

S18: StaffLogIn_PreconditionViolation_InactiveStaffMember

Current test case

```java
   test S18{
      load testConfiguration1;
      li := new LogIn(username:='patricia', password:='uuu');
      assert non-occurrence li;
      maximumPrivilegesGroup.status:=#Disabled;
      li := new LogIn(username:='mary', password:='yyy');
      assert non-occurrence li;
   }
```

TDCM application: Summary of changes performed in the schema

- Added
- Updated

- First assert non-occurrence: Preconditions of the domain event li:LogIn are satisfied and consequently, the event can occur

```java
context LogIn in inv staffMembersEnabled:
   StaffMember.allInstances()->select(sm|sm.username=self.username and sm.password=self.password)->size()>0
   implies
   StaffMember.allInstances()->any(sm|sm.username=self.username and sm.password=self.password).status=#Enabled
   and StaffMember.allInstances()->any(sm|sm.username=self.username and sm.password=self.password).status=#Enabled
   and StaffMember.allInstances()->any(sm|sm.username=self.username and sm.password=self.password).staffGroup.status=#Enabled
```
Time spent

| TIME TO WRITE TEST CASES (IN MINUTES) | 1.5 |
| TIME TO COMPLETE THE ITERATION (IN MINUTES) | 4 |

Errors and failures that drive the conceptual modeling

<table>
<thead>
<tr>
<th>A basic type involved in a test case does not exist in the CSUD</th>
<th>A derived type involved in a test case does not exist in the CSUD</th>
<th>An event type involved in a test case does not exist in the CSUD</th>
</tr>
</thead>
<tbody>
<tr>
<td>The basic type is relevant and it is added to the CSUD</td>
<td>The derived type is relevant and it is added to the CSUD</td>
<td>The event type is relevant and it is added to the CSUD</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Inconsistent state before the occurrence of an event</th>
<th>Inconsistent state after the occurrence of an event</th>
<th>The postcondition of an event is not satisfied.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Some static constraint is invalid and it is modified.</td>
<td>Some initial integrity constraint is invalid and it is modified.</td>
<td>The event postcondition/method is incorrect and it is modified.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Some constraint is invalid and it is modified.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The method is not correct and it is modified.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The postcondition is not correct and it is modified.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>An assertion about the IB state fails or contains an error</th>
<th>Assert non-occurrence fails</th>
<th>Semantic error in an expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>The effect of an event type is not correct</td>
<td>A derivation rule is incorrect</td>
<td>The expression is corrected</td>
</tr>
<tr>
<td></td>
<td>A precondition is added/updated</td>
<td>The CSUD is changed</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Iteration 34

Iteration objective

S19: StaffLogOut_successScenario

Current test case

test S19{
    load testConfiguration1;
    assert false generalConsultant.isLoggedIn;
    li := new LogIn(username:="mary", password:="yyy");
    assert occurrence li;
    assert true generalConsultant.isLoggedIn;
    lo := new LogOut(staffMember:=generalConsultant),
    assert occurrence lo;
    assert false generalConsultant.isLoggedIn;
}

TDCM application: Summary of changes performed in the schema

- Added
- Updated

- [TicketsManagementAndTracking.cstl] <line 1000> LogOut is not defined in the CSUT as a class or an association

```plaintext
event LogOut 
operations 
effect() 
end

association logOut_staffMember between 
LogOut[*] 
StaffMember[1] 
end

custom LogOut::effect() 
post: 
self.staffMember.isLoggedIn=false

custom LogOut ini inv isNotLoggedIn: 
self.staffMember.isLoggedIn=true
```

Time spent

<table>
<thead>
<tr>
<th>Time to Write Test Cases (in minutes)</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time to Complete the Iteration (in minutes)</td>
<td>6.5</td>
</tr>
</tbody>
</table>

Errors and failures that drive the conceptual modeling

<table>
<thead>
<tr>
<th>A basic type involved in a test case does not exist in the CSUD</th>
<th>An event type involved in a test case does not exist in the CSUD</th>
<th>An event type involved in a test case does not exist in the CSUD</th>
</tr>
</thead>
<tbody>
<tr>
<td>The basic type is relevant and it is added to the CSUD</td>
<td>The derived type is relevant and it is added to the CSUD</td>
<td>The event type is relevant and it is added to the CSUD</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Inconsistent state before the occurrence of an event</th>
<th>Inconsistent state after the occurrence of an event</th>
<th>The postcondition of an event is not satisfied</th>
</tr>
</thead>
<tbody>
<tr>
<td>Some static constraint is invalid and it is modified.</td>
<td>Some initial integrity constraint is invalid and it is modified.</td>
<td>Some constraint is invalid and it is modified.</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>An assertion about the DB state fails or contains an error</th>
<th>Assert non-occurrence fails</th>
<th>Semantic error in an expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>The effect of an event type is not correct</td>
<td>A derivation rule is incorrect</td>
<td>The expression is incorrect.</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Assert consistency fails</th>
</tr>
</thead>
<tbody>
<tr>
<td>A static constraint needs to be changed</td>
</tr>
</tbody>
</table>
Iteration 35

Iteration objective

S20: StaffLogIn extension_2a (invalid login data)

Current test case

```java
test S20{
    load testConfiguration1;
    li := new LogIn(username:='mary', password:='zzz');
    assert non-occurrence li;
}
```

TDCM application: Summary of changes performed in the schema

- Added
- Updated

- Preconditions of the domain event li:LogIn are satisfied and consequently, the event can occur

  ```
  context LogIn
  inv accessDataIsValid:
  StaffMember.allInstances()->select(sm|sm.username=self.username and sm.password=self.password)->size()>0
  ```

Time spent

| Time to write test cases (in minutes) | 0.5 |
| Time to complete the iteration (in minutes) | 1 |

Errors and failures that drive the conceptual modeling

<table>
<thead>
<tr>
<th>A basic type involved in a test case does not exist in the CSUD</th>
<th>A derived type involved in a test case does not exist in the CSUD</th>
<th>An event type involved in a test case does not exist in the CSUD</th>
</tr>
</thead>
<tbody>
<tr>
<td>The basic type is relevant and it is added to the CSUD</td>
<td>The derived type is relevant and it is added to the CSUD</td>
<td>The event type is relevant and it is added to the CSUD</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Inconsistent state before the occurrence of an event</th>
<th>Inconsistent state after the occurrence of an event</th>
<th>The postcondition of an event is not satisfied</th>
</tr>
</thead>
<tbody>
<tr>
<td>Some static constraint is invalid and it is modified.</td>
<td>Some initial integrity constraint is invalid and it is modified.</td>
<td>The event postcondition/method is incorrect and it is modified.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Some constraint is invalid and it is modified.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The method is not correct and it is modified.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The postcondition is not correct and it is modified.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>An assertion about the IB state fails or contains an error</th>
<th>Assert non-occurrence fails</th>
<th>Semantic error in an expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>The effect of an event type is not correct</td>
<td>A derivation rule is incorrect</td>
<td>The expression is corrected</td>
</tr>
<tr>
<td></td>
<td>A precondition is added/updated</td>
<td>The CSUD is changed</td>
</tr>
</tbody>
</table>

1

Assert consistency fails

A static constraint needs to be changed
Iteration 36

Iteration objective

S21:
NewTicketOffline_successScenario_SequentialTicketsNumber_alertsAutoresponsesActive

Current test case

test S21{
  load testConfiguration1;
  minimumPrivilegesGroup.canCreateTickets:=true;

  nt:=new NewTicketOffline;
  nt.fullName:='Mary Marnes';
  nt.email:='mary_at_marnes.mar';
  nt.telephone:='xxxxxxxx';
  nt.ext:='xxxxxxxx';
  nt.source:='Phone';
  nt.assignedDepartment:='dptTechnical';
  nt.helpTopic:='helpTopicInstallation';
  nt.subject:='Error operating system';
  nt.message:='The installation process does not finish....';
  nt.internalNote:='It seems that the correct installer is being used';

  dt2:=new Datetime(value:=[(sys.currentDateTime.value+2)]);
  nt.dueDatetime:=dt2;
  nt.priority:='Normal';
  nt.assignedStaff:='generalConsultant';
  assert occurrence nt;

  ticket1:=nt.createdTicket;

  assert equals ticket1.number 1;
  assert equals ticket1.ticketStatus '#Open';
  assert equals ticket1.subject 'Error operating system';
  assert equals ticket1.priority '#Normal';
  assert equals ticket1.assignedStaff generalConsultant;
  assert equals ticket1.source '#Phone';
  assert equals ticket1.creationDatetime sys.currentDateTime;
  assert equals ticket1.lastMessageDatetime sys.currentDateTime;
  assert equals ticket1.dueDatetime.value 3;
  assert true ticket1.lastResponseDatetime.isUndefined();
  assert equals ticket1.assignedDepartment 'dptTechnical';

  assert true [ticket1.ticketThreadMessage->one(m|m.datetime=sys.currentDateTime and
    m.text='The installation process does not finish....' and
    m.author='Mary Marnes')];

  assert equals ticket1.lastMessageDatetime sys.currentDateTime;

  //no autoresponses
  assert true [not(EMail.allInstances()->exists(e|e.emailKind.oclIsTypeOf(NewTicketNotice)))];

  //notice to administrator
  assert true [EMail.allInstances()->exists(e|e.emailKind.oclIsTypeOf(NewTicketAlertToStaff) and
    e.fromAddress='general_at_support.com' and
    e.toAddress='system_at_support.com' and
    e.ticketNumber=1)];

  //notice to department manager
  assert true [EMail.allInstances()->exists(e|e.emailKind.oclIsTypeOf(NewTicketAlertToStaff) and
    e.fromAddress='general_at_support.com' and
    e.toAddress='system_at_support.com' and
    e.ticketNumber=1)];
}
assert true [EMail.allInstances()->exists(e|e.emailKind.oclIsTypeOf(NewTicketAlertToStaff) and e.fromAddress='general_at_support.com' and e.toAddress='john_at_support.com' and e.ticketNumber=1)];

//notice to department members
assert true [EMail.allInstances()->exists(e|e.emailKind.oclIsTypeOf(NewTicketAlertToStaff) and e.fromAddress='general_at_support.com' and e.toAddress='martin_at_support.com' and e.ticketNumber=1)];

//TICKET 2
ticketSettings.customersCanChangePriority:=true;

nt2:=new NewTicketOffline;
nt2.fullName:='James Jordan';
nt2.email:='james_at_jordan.jam';
nt2.telephone:='xxxxxxxx';
nt2.ext:='xxxxxxxx';
nt2.source:=#Other;
nt2.assignedDepartment:=dptGeneral;
nt2.priority:=#Low;
nt2.helpTopic:=helpTopicUse;
nt2.subject:='Reopening ticket';
nt2.message:='I do not know how to reopen one of my closed tickets';
assert occurrence nt2;

ticket2:=nt2.createdTicket;
assert equals ticket2.number 2;
assert equals ticket2.priority #Low;
assert true ticket2.assignedStaff->isEmpty();

//autoresponses
assert true [EMail.allInstances()->exists(e|e.emailKind.oclIsTypeOf(NewTicketNotice) and e.fromAddress='general_at_support.com' and e.toAddress='james_at_jordan.jam' and e.ticketNumber=2)];

//notice to administrator
assert true [EMail.allInstances()->exists(e|e.emailKind.oclIsTypeOf(NewTicketAlertToStaff) and e.fromAddress='general_at_support.com' and e.toAddress='system_at_support.com' and e.ticketNumber=2)];

//notice to department members
assert true [EMail.allInstances()->exists(e|e.emailKind.oclIsTypeOf(NewTicketAlertToStaff) and e.fromAddress='general_at_support.com' and e.toAddress='john_at_support.com' and e.ticketNumber=2)];

assert true [EMail.allInstances()->exists(e|e.emailKind.oclIsTypeOf(NewTicketAlertToStaff) and e.fromAddress='general_at_support.com' and e.toAddress='mary_at_support.com' and e.ticketNumber=2)];

assert false [EMail.allInstances()->exists(e|e.emailKind.oclIsTypeOf(NewTicketAlertToStaff) and e.fromAddress='general_at_support.com' and e.toAddress='david_at_support.com' and e.ticketNumber=2)];
TDCM application: Summary of changes performed in the schema

- [TicketsManagementAndTracking.cstl] <line 1012> NewTicketOffline is not defined in the CSUT as a class or an association

  ```
  event NewTicketOffline
  attributes
  fullName: String
  email: String
  telephone: String[0..1]
  ext: String[0..1]
  source: TicketSource
  subject: String
  message: String
  priority: Priority[0..1]
  createdTicket: Ticket[0..1]
  internalNote: String[0..1]
  dueDatetime: Datetime[0..1]
  operations
  effect()
  end

  association newTicketOffline_department between
  NewTicketOffline[*]
  Department[1] role assignedDepartment
  end

  association newTicketOffline_helpTopic between
  NewTicketOffline[*]
  HelpTopic[0..1]
  end

  association newTicketOffline_assignedStaff between
  NewTicketOffline[*]
  StaffMember[0..1] role assignedStaff
  end

  context NewTicketOffline ini inv helpTopicSpecifiedIfAvailable:
  if HelpTopic.allInstances()->select(hp|hp.status=#Enabled)->size()>0 then
    self.helpTopic->size()=1
  else
    self.helpTopic->size()=0
  endif

  context NewTicketOffline ini inv helpTopicIsEnabled:
  self.helpTopic->notEmpty() implies self.helpTopic.status=#Enabled

  context NewTicketOffline ini inv maximumOpenTicketsLimitNotViolated:
  if TicketSettings.allInstances()->any(true).openTicketsPerMailAreLimited then
    Ticket.allInstances()->select(t|t.email=self.email)->size()<TicketSettings.allInstances()->any(true).maximumOpenTicketsPerMail
  else true
  endif

  context NewTicketOffline::effect()
  post:
  let sendNewTicketAlertToAdministrator:Boolean=
    StaffNoticesAlertsSettings.allInstances()->any(true).alertWhenNewTicketCreated and
    StaffNoticesAlertsSettings.allInstances()->any(true).alertWhenNewTicketCreatedStaff->includes(#Administrator)
  in
  let sendNewTicketAlertToDepartmentManager:Boolean=
    StaffNoticesAlertsSettings.allInstances()->any(true).alertWhenNewTicketCreated and
    StaffNoticesAlertsSettings.allInstances()->any(true).alertWhenNewTicketCreatedStaff->includes(#DepartmentManager)
  in
  ```
let sendNewTicketAlertToDepartmentMembers:Boolean=
  StaffNoticesAlertsSettings.allInstances()-»any(true).alertWhenNewTicketCreated and
  StaffNoticesAlertsSettings.allInstances()-»any(true).alertWhenNewTicketCreatedStaff->
  includes(#DepartmentMembers)
in
let staffAlertsFromEMailAddress:String=
  EmailSettings.allInstances()-»any(true).defaultStaffAlertsEmail.address
in
let sendAutoresponse:Boolean=
  if CustomerAutoresponsesSettings.allInstances()-»any(true).autorespondWhenNewTicketCreatedByCustomer then
    if self.helpTopic->notEmpty then
      if self.helpTopic.autoresponse=#Enabled then true
      else false
    endif
    else false
  endif
endif

in
(Ticket.allInstances- Ticket.allInstances@pre)
-»one(t | t.oclIsNew() and self.createdTicket=t
  and t.fullName=self.fullName
  and t.email=self.email
  and t.telephone=self.telephone
  and t.ext=self.ext
  and t.subject=self.subject
  and t.message=self.message
  and t.ticketStatus=#Open
  and t.priority=self.priority
  and t.source=self.source
  and t.dueDatetime=self.dueDateTime
  and (TicketThreadMessage.allInstances- TicketThreadMessage.allInstances@pre)
  -»one(tdm | tdm.oclIsNew() and tdm.datetime=System.allInstances()-»any(true).currentDateTime
  and tdm.text=self.message
  and tdm.author=self.fullName
  and tdm.ticket=t
  and t.lastMessageDatetime=System.allInstances()-»any(true).currentDateTime)
  -- autoresponses
  and (sendAutoresponse implies
    EMail.allInstances()-»exists(e|e.fromAddress=t.assignedDepartment.autoresponseEmail.address and
    e.toAddress=t.email and
    e.ticketNumber=t.number))
  -- staff notices
  and (sendNewTicketAlertToAdministrator implies
    EMail.allInstances()-»exists(e|e.fromAddress=staffAlertsFromEMailAddress and
    e.toAddress=EmailSettings.allInstances()-»any(true).administrationEmail and
    e.ticketNumber=t.number))
  and (sendNewTicketAlertToDepartmentManager
    and t.assignedDepartment.departmentManager->notEmpty()
    and t.assignedDepartment.departmentManager.status=#Enabled
    and not(t.assignedDepartment.departmentManager.isInVacationMode)
    implies
    EMail.allInstances()-»exists(e|e.fromAddress=staffAlertsFromEMailAddress and
    e.toAddress=t.assignedDepartment.departmentManager.emailAddress and
    e.ticketNumber=t.number))
  and (sendNewTicketAlertToDepartmentMembers
    implies
    t.assignedDepartment.staffMember->forAll(m|
    (m.status=#Enabled and not(m.isInVacationMode))
    implies
    EMail.allInstances()-»exists(e|e.fromAddress=staffAlertsFromEMailAddress and
    e.toAddress=m.email and
    e.ticketNumber=t.number)
e.toAddress=m.emailAddress and
e.ticketNumber=t.number))

- [TicketsManagementAndTracking.cstl] <Line 1017>:1:1: Undefined enumeration literal 'Phone' and 'Other'
  
  ```
  enum TicketSource{Web,EMail,Phone,Other}
  ```

- Several method and postcondition errors and failures until the verdict becomes Pass

### Time spent

<table>
<thead>
<tr>
<th>Description</th>
<th>Time (in minutes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time to write test cases</td>
<td>17</td>
</tr>
<tr>
<td>Time to complete the iteration</td>
<td>41</td>
</tr>
</tbody>
</table>

### Errors and failures that drive the conceptual modeling

<table>
<thead>
<tr>
<th>Error Description</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>A basic type involved in a test case does not exist in the CSUD</td>
<td>16</td>
</tr>
<tr>
<td>A derived type involved in a test case does not exist in the CSUD</td>
<td>1</td>
</tr>
<tr>
<td>An event type involved in a test case does not exist in the CSUD</td>
<td></td>
</tr>
<tr>
<td>Inconsistent state before the occurrence of an event</td>
<td></td>
</tr>
<tr>
<td>Inconsistent state after the occurrence of an event</td>
<td></td>
</tr>
<tr>
<td>The postcondition of an event is not satisfied</td>
<td></td>
</tr>
<tr>
<td>The basic type is relevant and it is added to the CSUD</td>
<td></td>
</tr>
<tr>
<td>The derived type is relevant and it is added to the CSUD</td>
<td></td>
</tr>
<tr>
<td>The event type is relevant and it is added to the CSUD</td>
<td></td>
</tr>
<tr>
<td>Some static constraint is invalid and it is modified</td>
<td></td>
</tr>
<tr>
<td>Some initial integrity constraint is invalid and it is modified</td>
<td></td>
</tr>
<tr>
<td>Some constraint is invalid and it is modified</td>
<td></td>
</tr>
<tr>
<td>The event postcondition/method is incorrect and it is modified</td>
<td></td>
</tr>
<tr>
<td>The postcondition/method is incorrect and it is modified</td>
<td></td>
</tr>
<tr>
<td>The method is not correct and it is modified</td>
<td></td>
</tr>
<tr>
<td>The postcondition is not correct and it is modified</td>
<td></td>
</tr>
<tr>
<td>The basic type is relevant and it is added to the CSUD</td>
<td></td>
</tr>
<tr>
<td>The derived type is relevant and it is added to the CSUD</td>
<td></td>
</tr>
<tr>
<td>The event type is relevant and it is added to the CSUD</td>
<td></td>
</tr>
<tr>
<td>Inconsistent state before the occurrence of an event</td>
<td></td>
</tr>
<tr>
<td>Inconsistent state after the occurrence of an event</td>
<td></td>
</tr>
<tr>
<td>The postcondition of an event is not satisfied</td>
<td></td>
</tr>
<tr>
<td>Inconsistent state before the occurrence of an event</td>
<td></td>
</tr>
<tr>
<td>Inconsistent state after the occurrence of an event</td>
<td></td>
</tr>
<tr>
<td>The postcondition of an event is not satisfied</td>
<td></td>
</tr>
<tr>
<td>Some static constraint is invalid and it is modified</td>
<td></td>
</tr>
<tr>
<td>Some initial integrity constraint is invalid and it is modified</td>
<td></td>
</tr>
<tr>
<td>Some constraint is invalid and it is modified</td>
<td></td>
</tr>
<tr>
<td>The event postcondition/method is incorrect and it is modified</td>
<td></td>
</tr>
<tr>
<td>The postcondition/method is incorrect and it is modified</td>
<td></td>
</tr>
<tr>
<td>The method is not correct and it is modified</td>
<td></td>
</tr>
<tr>
<td>The postcondition is not correct and it is modified</td>
<td></td>
</tr>
<tr>
<td>The basic type is relevant and it is added to the CSUD</td>
<td></td>
</tr>
<tr>
<td>The derived type is relevant and it is added to the CSUD</td>
<td></td>
</tr>
<tr>
<td>The event type is relevant and it is added to the CSUD</td>
<td></td>
</tr>
<tr>
<td>Inconsistent state before the occurrence of an event</td>
<td></td>
</tr>
<tr>
<td>Inconsistent state after the occurrence of an event</td>
<td></td>
</tr>
<tr>
<td>The postcondition of an event is not satisfied</td>
<td></td>
</tr>
<tr>
<td>Inconsistent state before the occurrence of an event</td>
<td></td>
</tr>
<tr>
<td>Inconsistent state after the occurrence of an event</td>
<td></td>
</tr>
<tr>
<td>The postcondition of an event is not satisfied</td>
<td></td>
</tr>
<tr>
<td>Some static constraint is invalid and it is modified</td>
<td></td>
</tr>
<tr>
<td>Some initial integrity constraint is invalid and it is modified</td>
<td></td>
</tr>
<tr>
<td>Some constraint is invalid and it is modified</td>
<td></td>
</tr>
<tr>
<td>The event postcondition/method is incorrect and it is modified</td>
<td></td>
</tr>
<tr>
<td>The postcondition/method is incorrect and it is modified</td>
<td></td>
</tr>
<tr>
<td>The method is not correct and it is modified</td>
<td></td>
</tr>
<tr>
<td>The postcondition is not correct and it is modified</td>
<td></td>
</tr>
<tr>
<td>The basic type is relevant and it is added to the CSUD</td>
<td></td>
</tr>
<tr>
<td>The derived type is relevant and it is added to the CSUD</td>
<td></td>
</tr>
<tr>
<td>The event type is relevant and it is added to the CSUD</td>
<td></td>
</tr>
<tr>
<td>Inconsistent state before the occurrence of an event</td>
<td></td>
</tr>
<tr>
<td>Inconsistent state after the occurrence of an event</td>
<td></td>
</tr>
<tr>
<td>The postcondition of an event is not satisfied</td>
<td></td>
</tr>
<tr>
<td>Inconsistent state before the occurrence of an event</td>
<td></td>
</tr>
<tr>
<td>Inconsistent state after the occurrence of an event</td>
<td></td>
</tr>
<tr>
<td>The postcondition of an event is not satisfied</td>
<td></td>
</tr>
<tr>
<td>Some static constraint is invalid and it is modified</td>
<td></td>
</tr>
<tr>
<td>Some initial integrity constraint is invalid and it is modified</td>
<td></td>
</tr>
<tr>
<td>Some constraint is invalid and it is modified</td>
<td></td>
</tr>
<tr>
<td>The event postcondition/method is incorrect and it is modified</td>
<td></td>
</tr>
<tr>
<td>The postcondition/method is incorrect and it is modified</td>
<td></td>
</tr>
<tr>
<td>The method is not correct and it is modified</td>
<td></td>
</tr>
<tr>
<td>The postcondition is not correct and it is modified</td>
<td></td>
</tr>
<tr>
<td>An assertion about the IB state fails or contains an error</td>
<td>2</td>
</tr>
<tr>
<td>Assert non-occurrence fails</td>
<td></td>
</tr>
<tr>
<td>Semantic error in an expression</td>
<td></td>
</tr>
<tr>
<td>The effect of an event type is incorrect</td>
<td></td>
</tr>
<tr>
<td>A derivation rule is incorrect</td>
<td></td>
</tr>
<tr>
<td>A precondition is added/updated</td>
<td></td>
</tr>
<tr>
<td>The expression is corrected</td>
<td></td>
</tr>
<tr>
<td>The CSUD is changed</td>
<td></td>
</tr>
<tr>
<td>Assert consistency fails</td>
<td></td>
</tr>
<tr>
<td>A static constraint needs to be changed</td>
<td></td>
</tr>
</tbody>
</table>

### Iteration 37

### Iteration objective

S22: NewTicketOffline_SuccessScenario_alertsAutoresponsesDisabled

### Current test case

```
```
DEVELOPMENT OF THE CONCEPTUAL SCHEMA OF THE osTICKET SYSTEM BY APPLYING TDCM

TDCM application: Summary of changes performed in the schema

The CSUD has not been changed
### Time spent

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>TIME TO WRITE TEST CASES (IN MINUTES)</strong></td>
<td>1</td>
</tr>
<tr>
<td><strong>TIME TO COMPLETE THE ITERATION (IN MINUTES)</strong></td>
<td>0.5</td>
</tr>
</tbody>
</table>

### Iteration 38

#### Iteration objective

S23: NewTicketOffline_SuccessScenario_NoTopic

#### Current test case

```java
test S23{
    load testConfiguration4;
    minimumPrivilegesGroup.canCreateTickets:=true;
    helpTopicUse.status:=#Disabled;
    helpTopicInstallation.status:=#Disabled;

    nt:=new NewTicketOffline;
    nt.fullName:='Mary Marnes';
    nt.email:='mary_at_marnes.mar';
    nt.telephone:='xxxxxxxx';
    nt.ext:='xxxxxxxx';
    nt.source:='#Phone;
    nt.assignedDepartment:=dptTechnical;
    nt.subject:='Error operating system';
    nt.message:='The installation process does not finish....';

    nt.priority:=#Normal;
    assert occurrence nt;

    ticket1:=nt.createdTicket;
                
    assert true ticket1.helpTopic->isEmpty();
}
```

#### TDCM application: Summary of changes performed in the schema

The CSUD has not been changed

### Time spent

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>TIME TO WRITE TEST CASES (IN MINUTES)</strong></td>
<td>2</td>
</tr>
<tr>
<td><strong>TIME TO COMPLETE THE ITERATION (IN MINUTES)</strong></td>
<td>0.5</td>
</tr>
</tbody>
</table>
DEVELOPMENT OF THE CONCEPTUAL SCHEMA OF THE osTICKET SYSTEM BY APPLYING TDCM

Iteration 39

Iteration objective

S24: NewTicketOffline_PreconditionViolation_cannotCreateTickets

Current test case

```java
test S24{
load testConfiguration9;

li := new LogIn(username:='martin', password:='ttt');
assert occurrence li;

nt:=new NewTicketOffline;
nt.fullName:='Mary Marnes';
nt.email:='mary_at_marnes.mar';
nt.telephone:='xxxxxxxx';
nt.ext:='xxxxxxxx';
nt.source:=#Phone;
nt.helpTopic:=helpTopicInstallation;
nt.assignsedDepartment:=dptTechnical;
nt.assignsedStaff:=generalConsultant;
nt.subject:='Error operating system';
nt.message:='The installation process does not finish....';
nt.priority:=#Normal;
nt.creator:=technicalActive;
assert non-occurrence nt;
}
```

TDCM application: Summary of changes performed in the schema

- Added
- Updated

- Preconditions of the domain event nt:NewTicketOffline are satisfied and consequently, the event can occur

```
context NewTicketOffline ini inv creatorIsAllowedToCreateTickets:
self.creator.staffGroup.canCreateTickets
```

Time spent

| TIME TO WRITE TEST CASES (IN MINUTES) | 2 |
| TIME TO COMPLETE THE ITERATION (IN MINUTES) | 3,5 |

Errors and failures that drive the conceptual modeling

<table>
<thead>
<tr>
<th>A basic type involved in a test case does not exist in the CSUD</th>
<th>A derived type involved in a test case does not exist in the CSUD</th>
<th>An event type involved in a test case does not exist in the CSUD</th>
</tr>
</thead>
<tbody>
<tr>
<td>The basic type is relevant and it is added to the CSUD</td>
<td>The derived type is relevant and it is added to the CSUD</td>
<td>The event type is relevant and it is added to the CSUD</td>
</tr>
</tbody>
</table>
Inconsistent state before the occurrence of an event | Inconsistent state after the occurrence of an event | The postcondition of an event is not satisfied.
--- | --- | ---
Some static constraint is invalid and it is modified. | Some static constraint is invalid and it is modified. | The event postcondition/method is incorrect and it is modified.
Some initial integrity constraint is invalid and it is modified. | The event postcondition/method is incorrect and it is modified. | Some constraint is invalid and it is modified.
The postcondition of an event is not satisfied. | The postcondition is not correct and it is modified. | The postcondition is not correct and it is modified.

An assertion about the IB state fails or contains an error | Assert non-occurrence fails | Semantic error in an expression
--- | --- | ---
The effect of an event type is not correct | A derivation rule is incorrect | A precondition is added/updated
The method is not correct and it is modified. | The expression is corrected | The CSUD is changed

**Iteration 40**

**Iteration objective**

S25: NewTicketOffline_PreconditionViolation_isNotLoggedIn

**Current test case**

```plaintext
test S24{
    load testConfiguration9;
    nt:=new NewTicketOffline;
    nt.fullName:='Mary Marnes';
    nt.email:='mary_at_marnes.mar';
    nt.telephone:='xxxxxxxx';
    nt.ext:='xxxxxxxx';
    nt.source:='#Phone';
    nt.helpTopic:=helpTopicInstallation;
    nt.assignedDepartment:=dptTechnical;
    nt.subject:='Error operating system';
    nt.message:='The installation process does not finish....';
    nt.priority:=#Normal;
    assert non-occurrence nt;
}
```

**TDCM application: Summary of changes performed in the schema**

- Added
- Updated

- Preconditions of the domain event nt:NewTicketOffline are satisfied and consequently, the event can occur.
- We realize that we do not check that we need to know the staff member that creates a ticket and checking if he/she is logged in.
association newTicketOffline_creator between
NewTicketOffline[*] role newTicketOfflineOfCreator
StaffMember[1] role creator
End

countx NewTicketOffline ini inv creatorIsLoggedIn:
self.creator.isLoggedIn

- The verdict of previous passing test cases that exercise the NewTicketOffline event is now error:
[TicketsManagementAndTracking.cstl] <line 1036> Inconsistent state before nt:NewTicketOffline event execution:
Multiplicity constraint violation in association ‘newTicketOffline_creator’: Object ‘oid775’ of class ‘NewTicketOffline’ is
connected to 0 objects of class ‘StaffMember’ but the multiplicity is specified as ‘1’.

- We realize that in the current story and in previous stories, we did not consider that the staff member that creates a ticket
offline needs to be known. We change the previous stories to include this expected knowledge.

Time spent

| TIME TO WRITE TEST CASES (IN MINUTES) | 0,5 |
| TIME TO COMPLETE THE ITERATION (IN MINUTES) | 14 |

Errors and failures that drive the conceptual modeling

<table>
<thead>
<tr>
<th>A basic type involved in a test case does not exist in the CSUD</th>
<th>A derived type involved in a test case does not exist in the CSUD</th>
<th>An event type involved in a test case does not exist in the CSUD</th>
</tr>
</thead>
<tbody>
<tr>
<td>The basic type is relevant and it is added to the CSUD</td>
<td>The derived type is relevant and it is added to the CSUD</td>
<td>The event type is relevant and it is added to the CSUD</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Inconsistent state before the occurrence of an event</th>
<th>Inconsistent state after the occurrence of an event</th>
<th>The postcondition of an event is not satisfied</th>
</tr>
</thead>
<tbody>
<tr>
<td>Some static constraint is invalid and it is modified.</td>
<td>Some initial integrity constraint is invalid and it is modified.</td>
<td>The event postcondition/method is incorrect and it is modified.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>An assertion about the IB state fails or contains an error</th>
<th>Assert non-occurrence fails</th>
<th>Semantic error in an expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>The effect of an event type is not correct</td>
<td>A derivation rule is incorrect</td>
<td>The expression is corrected</td>
</tr>
<tr>
<td>A precondition is added/updated</td>
<td>1</td>
<td>The CSUD is changed</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Assert consistency fails</th>
</tr>
</thead>
<tbody>
<tr>
<td>A static constraint needs to be changed</td>
</tr>
</tbody>
</table>

INCONSISTENCY BETWEEN REQUIREMENTS

Iteration 41

Iteration objective

S26: ViewOpenTickets
fixturecomponent created_tickets{
   li := new Login(username:='mary', password:='yyy');
   assert occurrence li;

   nt1:=new NewTicketOffline;
   nt1.fullName:='Mary Marnes';
   nt1.email:='mary_at_marnes.mar';
   nt1.telephone:='xxxxxxxx';
   nt1.ext:='xxxxxxxx';
   nt1.source:=#Phone;
   nt1.assignedDepartment:=dptTechnical;
   nt1.helpTopic:=helpTopicInstallation;
   nt1.subject:='Error operating system';
   nt1.message:='The installation process does not finish...';
   nt1.internalNote:='It seems that the correct installer is being used';
   dt2:=new Datetime(value:=(sys.currentDateTime.value+2));
   nt1.dueDatetime:=dt2;
   nt1.priority:=#Normal;
   nt1.assignedStaff:=generalConsultant;
   nt1.creator:=generalConsultant;
   assert occurrence nt1;
   ticket1 := nt1.createdTicket;

   nt2:=new NewTicketOffline;
   nt2.fullName:='John Johnes';
   nt2.email:='mary_at_marnes.mar';
   nt2.source:=#Other;
   nt2.assignedDepartment:=dptGeneral;
   nt2.helpTopic:=helpTopicUse;
   nt2.subject:='Can I reply a ticket?';
   nt2.message:='I do not know how to reply a ticket';
   nt2.priority:=#High;
   nt2.assignedStaff:=generalConsultant;
   nt2.creator:=generalConsultant;
   assert occurrence nt2;
   ticket2 := nt2.createdTicket;

   lo := new LogOut(staffMember:=generalConsultant);
   assert occurrence lo;

   li := new Login(username:='john', password:='xxx');
   assert occurrence li;

   nt3:=new NewTicketOffline;
   nt3.fullName:='Martin Pope';
   nt3.email:='martin_at_pope.mar';
   nt3.source:=#Phone;
   nt3.assignedDepartment:=dptTechnical;
   nt3.helpTopic:=helpTopicUse;
   nt3.subject:='Error while login';
   nt3.message:='I get an error when I try to login';
   nt3.priority:=#Low;
   dt3:=new Datetime(value:=(sys.currentDateTime.value+5));
   nt3.dueDatetime:=dt3;
   nt3.assignedStaff:=technicalActive;
   nt3.creator:=generalAdministrator;
   assert occurrence nt3;
   ticket3 := nt3.createdTicket;

   dt4:=new Datetime(value:=(sys.currentDateTime.value+1));
   sys.currentDateTime:=dt4;

   lo := new LogOut(staffMember:=generalAdministrator);
   assert occurrence lo;

   nt4:=new NewTicketOnline;
   nt4.fullName:='James Jordan';
}
nt4.email:='james_at_jordan.jam';
nt4.helpTopic:=helpTopicUse;
nt4.subject:="Reopening ticket";
nt4.message:="I do not know how to reopen one of my closed tickets';
assert occurrence nt4;
ticket4 := nt4.createdTicket;

nt5:=new NewTicketByEmail;
nt5.toAddress:='technical_at_support.com';
nt5.fromName:='Marta Johnes';
nt5.fromAddress:='marta_at_johnes.mar';
nt5.subject:="See my tickets";
nt5.message:="Can I see my tickets?";
assert occurrence nt5;
ticket5 := nt5.createdTicket;
}

TDCM application: Summary of changes performed in the schema

- Added

  - [TicketsManagementAndTracking.cstl] <line 1335> Initial IC 'helpDeskStatusIsOnline ' of nt4:NewTicketOnline event fails

- Updated

  - We detect that the story is incorrectly defined. We need that the help desk status is online in order to create tickets online. We change the fixture of the story from #11 to #3

- Added

  - [TicketsManagementAndTracking.cstl] <line 1348> DisplayTicketsByStatus is not defined in the CSUT as a class or an association
event DisplayTicketsByStatus
attributes
status:TicketStatus
operations
answer():Sequence(Tuple(number:Integer,createDate:Integer,subject:String,department:String,priority:Priority,email:String))=
let visibleDepartments:Set(Department)=
self.consultant.staffGroup.departmentsAccess->including(self.consultant.department) in
Ticket.allinstances->select(t|t.ticketStatus=self.status and visibleDepartments->includes(t.assignedDepartment))
->collect(t | Tuple {number : t.number, createDate:t.creationDatetime.value, subject:t.subject,department:t.assignedDepartment.name, priority:t.priority, email:t.email})->sortedBy(number)
effect()
end

association displayTicketsByStatus_consultant between
  DisplayTicketsByStatus[*] role newTicketOfflineOfConsultant
  StaffMember[1] role consultant
end

<table>
<thead>
<tr>
<th>Errors and failures that drive the conceptual modeling</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>A basic type involved in a test case does not exist in the CSUD</strong></td>
</tr>
<tr>
<td>The basic type is relevant and it is added to the CSUD</td>
</tr>
<tr>
<td><strong>Inconsistent state before the occurrence of an event</strong></td>
</tr>
<tr>
<td>Some static constraint is invalid and it is modified.</td>
</tr>
<tr>
<td><strong>An assertion about the IB state fails or contains an error</strong></td>
</tr>
<tr>
<td>The effect of an event type is not correct</td>
</tr>
<tr>
<td><strong>Assert consistency fails</strong></td>
</tr>
</tbody>
</table>
| A static constraint needs to be changed | }
Iteration 42

Iteration objective

S27: ViewOpenTickets_preconditionViolation_notLoggedIn

Current test case

test S27
load testConfiguration11;

dts:=new DisplayTicketsByStatus(consultant:=generalAdministrator, status:=#Open);
assert non-occurrence dts;
}

TDCM application: Summary of changes performed in the schema

- Preconditions of the domain event dts:DisplayTicketsByStatus are satisfied and consequently, the event can occur

context DisplayTicketsByStatus ini inv consultantIsLoggedIn:
self.consultant.isLoggedIn

Time spent

| TIME TO WRITE TEST CASES (IN MINUTES) | 1.5 |
| TIME TO COMPLETE THE ITERATION (IN MINUTES) | 3 |

Errors and failures that drive the conceptual modeling

<table>
<thead>
<tr>
<th>A basic type involved in a test case does not exist in the CSUD</th>
<th>A derived type involved in a test case does not exist in the CSUD</th>
<th>An event type involved in a test case does not exist in the CSUD</th>
</tr>
</thead>
<tbody>
<tr>
<td>The basic type is relevant and it is added to the CSUD</td>
<td>The derived type is relevant and it is added to the CSUD</td>
<td>The event type is relevant and it is added to the CSUD</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Inconsistent state before the occurrence of an event</th>
<th>Inconsistent state after the occurrence of an event</th>
<th>The postcondition of an event is not satisfied</th>
</tr>
</thead>
<tbody>
<tr>
<td>Some static constraint is invalid and it is modified.</td>
<td>The event postcondition/method is incorrect and it is modified.</td>
<td>Some constraint is invalid and it is modified.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>An assertion about the IB state fails or contains an error</th>
<th>Assert non-occurrence fails</th>
<th>Semantic error in an expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>The effect of an event type is not correct</td>
<td>A derivation rule is incorrect</td>
<td>The expression is corrected</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Assert consistency fails</th>
<th>A static constraint needs to be changed</th>
</tr>
</thead>
</table>

1
Iteration 43

Iteration objective

S28: ChangeTicketPriority

Current test case

test S28{
    load testConfiguration3;
    load created_tickets;
    li := new LogIn(username:='john', password:='xxx');
    assert occurrence li;
    stp:=new ChangeTicketPriority(staffMember:=generalAdministrator, ticket:=ticket1, newPriority:=#High);
    assert occurrence stp;
    assert equals ticket1.priority #High;
    assert true [ticket1.internalNote->one(li.datetime=sys.currentDateTime and li.text='The ticket priority has been changed' and li.subject='Ticket priority changed' and li.author='John')];
}

TDCM application: Summary of changes performed in the schema

- Added

[TicketsManagementAndTracking.cstl] <line 1419> ChangeTicketPriority is not defined in the CSUT as a class or an association

```cstl
    event ChangeTicketPriority
        attributes
            newPriority:Priority
        operations
            effect()
        end

    association changeTicketPriority_staffMember between
        ChangeTicketPriority[*]
        StaffMember[1]
    end

    association changeTicketPriority_ticket between
        ChangeTicketPriority[*]
        Ticket[1]
    end

    context ChangeTicketPriority::effect()
        post:
            self.ticket.priority=self.newPriority
```

- Updated

[TicketsManagementAndTracking.cstl] <Line 1423>:1:8: Undefined operation named `internalNote' in expression `{Ticket}.internalNote()'.

- Added

[TicketsManagementAndTracking.cstl] <line 1423>:1:8: Undefined operation named `internalNote' in expression `{Ticket}.internalNote()'.

- Updated

[TicketsManagementAndTracking.cstl] <Line 1423>:1:8: Undefined operation named `internalNote' in expression `{Ticket}.internalNote()'.
class InternalNote
attributes
datetime: Datetime
subject: String
text: String
author: String
end

association ticket_internalNote between
  Ticket[*]
  InternalNote[*]
end

- Assert expression is false and it is expected to be true: assert true [ticket1.internalNote->one(i | i.datetime=sys.currentDateTime
and i.text='Ticket priority changed' and i.subject='Ticket priority changed' and i.author='Martin')];

context ChangeTicketPriority::effect()
post:
  self.ticket.priority=self.newPriority
and
  (InternalNote.allInstances- InternalNote.allInstances@pre)
  ->one(i | i.oclIsNew()
and i.datetime=System.allInstances()->any(true).currentDateTime
and i.subject='Ticket priority changed'
and i.text='The ticket priority has been changed'
and i.author=self.staffMember.firstName
and i.ticket=self.ticket)
and self.ticket.lastMessageDateTime=System.allInstances()->any(true).currentDateTime

---

Time spent

<table>
<thead>
<tr>
<th>TIME TO WRITE TEST CASES (IN MINUTES)</th>
<th>6.5</th>
</tr>
</thead>
<tbody>
<tr>
<td>TIME TO COMPLETE THE ITERATION (IN MINUTES)</td>
<td>25</td>
</tr>
</tbody>
</table>

Errors and failures that drive the conceptual modeling

<table>
<thead>
<tr>
<th>A basic type involved in a test case does not exist in the CSUD</th>
<th>A derived type involved in a test case does not exist in the CSUD</th>
<th>An event type involved in a test case does not exist in the CSUD</th>
</tr>
</thead>
<tbody>
<tr>
<td>The basic type is relevant and it is added to the CSUD</td>
<td>The derived type is relevant and it is added to the CSUD</td>
<td>The event type is relevant and it is added to the CSUD</td>
</tr>
<tr>
<td>9</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Inconsistent state before the occurrence of an event</td>
<td>Inconsistent state after the occurrence of an event</td>
<td>The postcondition of an event is not satisfied</td>
</tr>
<tr>
<td>Some static constraint is invalid and it is modified.</td>
<td>Some initial integrity constraint is invalid and it is modified.</td>
<td>Some constraint is invalid and it is modified.</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>An assertion about the IB state fails or contains an error</td>
<td>Assert non-occurrence fails</td>
<td>Semantic error in an expression</td>
</tr>
<tr>
<td>The effect of an event type is not correct</td>
<td>A derivation rule is incorrect</td>
<td>The expression is corrected</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Assert consistency fails</td>
<td></td>
<td></td>
</tr>
<tr>
<td>A static constraint needs to be changed</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Iteration 44

Iteration objective

S29: ChangeTicketPriority_TicketNotVisible

Current test case

test S29{
  load testConfiguration3;
  load created_tickets;
  li := new LogIn(username:=‘martin’, password:=‘ttt’);
  assert occurrence li;
  stp:=new ChangeTicketPriority(staffMember:=technicalActive, ticket:=ticket2, newPriority:=#High);
  assert non-occurrence stp;
}

TDCM application: Summary of changes performed in the schema

- Added
- Updated

- Preconditions of the domain event stp:ChangeTicketPriority are satisfied and consequently, the event can occur

context ChangeTicketPriority ini inv theTicketsVisible:
  self.staffMember.isAdministrator or
  (self.staffMember.staffGroup.departmentsAccess->including(self.staffMember.department))
  ->includes(self.ticket.assignedDepartment)

Time spent

| TIME TO WRITE TEST CASES (IN MINUTES) | 5 |
| TIME TO COMPLETE THE ITERATION (IN MINUTES) | 7 |

Errors and failures that drive the conceptual modeling

<table>
<thead>
<tr>
<th>A basic type involved in a test case does not exist in the CSUD</th>
<th>A derived type involved in a test case does not exist in the CSUD</th>
<th>An event type involved in a test case does not exist in the CSUD</th>
</tr>
</thead>
<tbody>
<tr>
<td>The basic type is relevant and it is added to the CSUD</td>
<td>The derived type is relevant and it is added to the CSUD</td>
<td>The event type is relevant and it is added to the CSUD</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Inconsistent state before the occurrence of an event</th>
<th>Inconsistent state after the occurrence of an event</th>
<th>The postcondition of an event is not satisfied.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Some static constraint is invalid and it is modified.</td>
<td>Some initial integrity constraint is invalid and it is modified.</td>
<td>The event postcondition/method is incorrect and it is modified.</td>
</tr>
<tr>
<td>Some constraint is invalid and it is modified.</td>
<td>The method is not correct and it is modified.</td>
<td>The postcondition is not correct and it is modified.</td>
</tr>
</tbody>
</table>
An assertion about the IB state fails or contains an error

<table>
<thead>
<tr>
<th>Effect of an event type is not correct</th>
<th>Assert non-occurrence fails</th>
<th>Semantic error in an expression</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>A derivation rule is incorrect</td>
<td>The expression is corrected</td>
</tr>
<tr>
<td></td>
<td>A precondition is added/updated</td>
<td>The CSUD is changed</td>
</tr>
</tbody>
</table>

**Assert consistency fails**

<table>
<thead>
<tr>
<th>A static constraint needs to be changed</th>
</tr>
</thead>
</table>

---

**Iteration 45**

**Iteration objective**

S30: ChangeTicketPriority_NotLoggedIn

**Current test case**

```java
test S30{
  load testConfiguration4;
  load created_tickets;

  stp:=new ChangeTicketPriority(staffMember:=generalAdministrator, ticket:=ticket1, newPriority:=#High);
  assert non-occurrence stp;
}
```

**TDCM application: Summary of changes performed in the schema**

- Added
- Updated

- Preconditions of the domain event stp:ChangeTicketPriority are satisfied and consequently, the event can occur

  ```java
  context ChangeTicketPriority ini inv theStaffMemberIsLoggedIn:
  self.staffMember.isLoggedIn
  ```

**Time spent**

<table>
<thead>
<tr>
<th>TIME TO WRITE TEST CASES (IN MINUTES)</th>
<th>0,5</th>
</tr>
</thead>
<tbody>
<tr>
<td>TIME TO COMPLETE THE ITERATION (IN MINUTES)</td>
<td>2</td>
</tr>
</tbody>
</table>

**Errors and failures that drive the conceptual modeling**

<table>
<thead>
<tr>
<th>A basic type involved in a test case does not exist in the CSUD</th>
<th>A derived type involved in a test case does not exist in the CSUD</th>
<th>An event type involved in a test case does not exist in the CSUD</th>
</tr>
</thead>
<tbody>
<tr>
<td>The basic type is relevant and it is added to the CSUD</td>
<td>The derived type is relevant and it is added to the CSUD</td>
<td>The event type is relevant and it is added to the CSUD</td>
</tr>
</tbody>
</table>
Inconsistent state before the occurrence of an event | Inconsistent state after the occurrence of an event | The postcondition of an event is not satisfied.
---|---|---
Some static constraint is invalid and it is modified. | Some initial integrity constraint is invalid and it is modified. | The event postcondition/method is incorrect and it is modified. |

An assertion about the IB state fails or contains an error | Assert non-occurrence fails | Semantic error in an expression
---|---|---
The effect of an event type is not correct | A derivation rule is incorrect | A precondition is added/updated |
A static constraint needs to be changed | The expression is corrected | The CSUD is changed

### Iteration 46

**Iteration objective**

S31: MarkTicketOverdue

**Current test case**

```java
test S31{
load testConfiguration3;
load created_tickets;

li := new LogIn(username:='john', password:='xxx');
assert occurrence li;

mto:=new MarkTicketOverdue(staffMember:=generalAdministrator, ticket:=ticket1);
assert occurrence mto;
assert true ticket1.isOverdue;
assert true [ticket1.internalNote->one(i|i.datetime=sys.currentDateTime and i.text='Ticket flagged as overdue' and i.subject='Ticket Marked Overdue' and i.author='John')];
}
```

**TDCM application: Summary of changes performed in the schema**

- Added
- Updated

- [TicketsManagementAndTracking.cstl] <line 1466> MarkTicketOverdue is not defined in the CSUT as a class or an association
DEVELOPMENT OF THE CONCEPTUAL SCHEMA OF THE osTICKET SYSTEM BY APPLYING TDCM

Association markTicketOverdue_staffMember between
MarkTicketOverdue[*]
StaffMember[*]
end

Association markTicketOverdue_ticket between
MarkTicketOverdue[*]
Ticket[*]
end

class MarkTicketOverdue::effect()
post:
self.ticket.isOverdue
and 

(Time spent

<table>
<thead>
<tr>
<th>TIME TO WRITE TEST CASES (IN MINUTES)</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>TIME TO COMPLETE THE ITERATION (IN MINUTES)</td>
<td>7,5</td>
</tr>
</tbody>
</table>

Errors and failures that drive the conceptual modeling

<table>
<thead>
<tr>
<th>A basic type involved in a test case does not exist in the CSUD</th>
<th>A derived type involved in a test case does not exist in the CSUD</th>
<th>An event type involved in a test case does not exist in the CSUD</th>
</tr>
</thead>
<tbody>
<tr>
<td>The basic type is relevant and it is added to the CSUD</td>
<td>The derived type is relevant and it is added to the CSUD</td>
<td>The event type is relevant and it is added to the CSUD</td>
</tr>
</tbody>
</table>

3

<table>
<thead>
<tr>
<th>Inconsistent state before the occurrence of an event</th>
<th>Inconsistent state after the occurrence of an event</th>
<th>The postcondition of an event is not satisfied.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Some static constraint is invalid and it is modified.</td>
<td>Some initial integrity constraint is invalid and it is modified.</td>
<td>The event postcondition/method is incorrect and it is modified.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Some constraint is invalid and it is modified.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The method is not correct and it is modified.</td>
</tr>
</tbody>
</table>

1

<table>
<thead>
<tr>
<th>An assertion about the IB state fails or contains an error</th>
<th>Assert non-occurrence fails</th>
<th>Semantic error in an expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>The effect of an event type is not correct</td>
<td>A derivation rule is incorrect</td>
<td>The expression is incorrect</td>
</tr>
<tr>
<td>A precondition is added/updated</td>
<td>The CSUD is changed</td>
<td></td>
</tr>
</tbody>
</table>

1

1

<table>
<thead>
<tr>
<th>Assert consistency fails</th>
<th>A static constraint needs to be changed</th>
</tr>
</thead>
<tbody>
<tr>
<td>A constraint needs to be changed</td>
<td></td>
</tr>
</tbody>
</table>

osTicketCSUT.use:979:12: Undefined operation named 'isOverdue' in expression 'Ticket.isOverdue()'.
Iteration 47

Iteration objective

S32: MarkTicketOverdue_staffIsNotAnAdministrator

Current test case

test S32{
    load testConfiguration3;
    load created_tickets;
    li := new LogIn(username:='martin', password:='ttt');
    assert occurrence li;
    mto:=new MarkTicketOverdue(staffMember:=technicalActive, ticket:=ticket1);
    assert non-occurrence mto;
}

TDCM application: Summary of changes performed in the schema

- Preconditions of the domain event mto:MarkTicketOverdue are satisfied and consequently, the event can occur

context MarkTicketOverdue ini inv staffMemberIsAnAdministrator:
    self.staffMember.isAdministrator

Time spent

| TIME TO WRITE TEST CASES (IN MINUTES) | 1,5 |
| TIME TO COMPLETE THE ITERATION (IN MINUTES) | 1 |

Errors and failures that drive the conceptual modeling

<table>
<thead>
<tr>
<th>A basic type involved in a test case does not exist in the CSUD</th>
<th>A derived type involved in a test case does not exist in the CSUD</th>
<th>An event type involved in a test case does not exist in the CSUD</th>
</tr>
</thead>
<tbody>
<tr>
<td>The basic type is relevant and it is added to the CSUD</td>
<td>The derived type is relevant and it is added to the CSUD</td>
<td>The event type is relevant and it is added to the CSUD</td>
</tr>
<tr>
<td>Inconsistent state before the occurrence of an event</td>
<td>Inconsistent state after the occurrence of an event</td>
<td>The postcondition of an event is not satisfied.</td>
</tr>
<tr>
<td>Some static constraint is invalid and it is modified.</td>
<td>Some initial integrity constraint is invalid and it is modified.</td>
<td>Some constraint is invalid and it is modified.</td>
</tr>
<tr>
<td>The event postcondition/method is incorrect and it is modified.</td>
<td>The method is not correct and it is not modified.</td>
<td>The postcondition is not correct and it is not modified.</td>
</tr>
</tbody>
</table>
DEVELOPMENT OF THE CONCEPTUAL SCHEMA OF THE osTICKET SYSTEM BY APPLYING TDCM

An assertion about the IB state fails or contains an error | Assert non-occurrence fails | Semantic error in an expression
---|---|---
The effect of an event type is not correct | A derivation rule is incorrect | The expression is corrected
Assert consistency fails | A precondition is added/updated | The CSUD is changed
A static constraint needs to be changed

**Iteration 48**

**Iteration objective**

S33: MarkTicketOverdue_staffIsNotAnAdministrator

**Current test case**

```plaintext
test S33{
    load testConfiguration4;
    load created_tickets;
    mto:=new MarkTicketOverdue(staffMember:=generalAdministrator, ticket:=ticket1);
    assert non-occurrence mto;
}
```

**TDCM application: Summary of changes performed in the schema**

- Preconditions of the domain event mto:MarkTicketOverdue are satisfied and consequently, the event can occur

  context MarkTicketOverdue ini inv staffMemberIsLoggedIn:
  self.staffMember.isLoggedIn

**Time spent**

| TIME TO WRITE TEST CASES (IN MINUTES) | 0,5 |
| TIME TO COMPLETE THE ITERATION (IN MINUTES) | 1,5 |

**Errors and failures that drive the conceptual modeling**

<table>
<thead>
<tr>
<th>A basic type involved in a test case does not exist in the CSUD</th>
<th>A derived type involved in a test case does not exist in the CSUD</th>
<th>An event type involved in a test case does not exist in the CSUD</th>
</tr>
</thead>
<tbody>
<tr>
<td>The basic type is relevant and it is added to the CSUD</td>
<td>The derived type is relevant and it is added to the CSUD</td>
<td>The event type is relevant and it is added to the CSUD</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Inconsistent state before the occurrence of an event</th>
<th>Inconsistent state after the occurrence of an event</th>
<th>The postcondition of an event is not satisfied</th>
</tr>
</thead>
<tbody>
<tr>
<td>Some static constraint is invalid and it is modified</td>
<td>Some initial integrity constraint is invalid and it is modified</td>
<td>Some constraint is invalid and it is modified</td>
</tr>
<tr>
<td>The event postcondition/method is incorrect and it is modified</td>
<td>The method is not correct and it is modified</td>
<td>The postcondition is not correct and it is modified</td>
</tr>
</tbody>
</table>
DEVELOPMENT OF THE CONCEPTUAL SCHEMA OF THE osTICKET SYSTEM BY APPLYING TDCM

An assertion about the IB state fails or contains an error

<table>
<thead>
<tr>
<th>An assertion about the IB state fails or contains an error</th>
<th>Assert non-occurrence fails</th>
<th>Semantic error in an expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>The effect of an event type is not correct</td>
<td>A derivation rule is incorrect</td>
<td>The expression is added/upated</td>
</tr>
<tr>
<td>A precondition is added/upated</td>
<td>The expression is corrected</td>
<td>The CSUD is changed</td>
</tr>
</tbody>
</table>

Assert consistency fails

<table>
<thead>
<tr>
<th>Assert consistency fails</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>A static constraint needs to be changed</td>
<td></td>
</tr>
</tbody>
</table>

Iteration 49

Iteration objective

S34: AssignTicket

Current test case

test S34{
    load testConfiguration3;
    load created_tickets;
    at:=new AssignTicket(staffMember:=generalConsultant, ticket:=ticket1, assignee:=generalAdministrator, assignmentText:='This is for you');
    assert occurrence at;
    assert equals ticket1.assignedStaff generalAdministrator;

    assert true [ticket1.internalNote->one(i|i.datetime=sys.currentDateTime and i.text='This is for you' and i.subject='Ticket Reassigned' and i.author='Mary')];

    //notice sent to the assignee
    assert true [EMail.allInstances()->exists(e|e.emailKind.oclIsTypeOf(TicketAssignedAlertToStaff) and e.fromAddress='general_at_support.com' and e.toAddress='john_at_support.com' and e.ticketNumber=1)];
}

TDCM application: Summary of changes performed in the schema

- TicketsManagementAndTracking.cstl] <line 1508> AssignTicket is not defined in the CSUT as a class or an association

```
    event AssignTicket
    attributes
        assignmentText: String
    operations
        effect()
    end

    association assignTicket_staffMember between
        AssignTicket[*]
        StaffMember[1]
    end
```
**Time spent**

<table>
<thead>
<tr>
<th>Description</th>
<th>Time (in minutes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>TIME TO WRITE TEST CASES</td>
<td>9</td>
</tr>
<tr>
<td>TIME TO COMPLETE THE ITERATION</td>
<td>31</td>
</tr>
</tbody>
</table>

**Errors and failures that drive the conceptual modeling**

<table>
<thead>
<tr>
<th>Error Type</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>A basic type involved in a test case does not exist in the CSUD</td>
<td>4</td>
</tr>
<tr>
<td>An event type involved in a test case does not exist in the CSUD</td>
<td>1</td>
</tr>
<tr>
<td>An inconsistent state before the occurrence of an event</td>
<td></td>
</tr>
<tr>
<td>Some static constraint is invalid and it is modified</td>
<td></td>
</tr>
<tr>
<td>Some initial integrity constraint is invalid and it is modified</td>
<td></td>
</tr>
<tr>
<td>The event postcondition/method is incorrect and it is modified</td>
<td></td>
</tr>
<tr>
<td>The postcondition of an event is not satisfied</td>
<td>1</td>
</tr>
<tr>
<td>An assertion about the IB state fails or contains an error</td>
<td>2</td>
</tr>
<tr>
<td>Assert non-occurrence fails</td>
<td></td>
</tr>
<tr>
<td>Semantic error in an expression</td>
<td></td>
</tr>
<tr>
<td>The effect of an event type is not correct</td>
<td>1</td>
</tr>
<tr>
<td>A derivation rule is incorrect</td>
<td></td>
</tr>
<tr>
<td>A precondition is added/updated</td>
<td></td>
</tr>
<tr>
<td>The expression is corrected</td>
<td></td>
</tr>
<tr>
<td>The CSUD is changed</td>
<td></td>
</tr>
<tr>
<td>Assert consistency fails</td>
<td></td>
</tr>
<tr>
<td>A static constraint needs to be changed</td>
<td></td>
</tr>
</tbody>
</table>
Iteration 50

Iteration objective

S35: AssignTicket_ticketIsNotVisible

Current test case

test S35{
    load testConfiguration3;
    load created_tickets;
    li := new LogIn(username:='martin', password:='ttt');
    assert occurrence li;
    at:=new AssignTicket(staffMember:=technicalActive, ticket:=ticket2, assignee:=generalAdministrator,
        assignmentText:='This is for you');
    assert non-occurrence at;
}

TDCM application: Summary of changes performed in the schema

- Added
- Updated

Preconditions of the domain event at:AssignTicket are satisfied and consequently, the event can occur

context AssignTicket ini inv theTicketsVisible:
    self.staffMember.isAdministrator or
    (self.staffMember.staffGroup.departmentsAccess->including(self.staffMember.department))
    ->includes(self.ticket.assignedDepartment)

Time spent

| TIME TO WRITE TEST CASES (IN MINUTES) | 2.5 |
| TIME TO COMPLETE THE ITERATION (IN MINUTES) | 4 |

Errors and failures that drive the conceptual modeling

<table>
<thead>
<tr>
<th>A basic type involved in a test case does not exist in the CSUD</th>
<th>A derived type involved in a test case does not exist in the CSUD</th>
<th>An event type involved in a test case does not exist in the CSUD</th>
</tr>
</thead>
<tbody>
<tr>
<td>The basic type is relevant and it is added to the CSUD</td>
<td>The derived type is relevant and it is added to the CSUD</td>
<td>The event type is relevant and it is added to the CSUD</td>
</tr>
</tbody>
</table>

Inconsistent state before the occurrence of an event

<table>
<thead>
<tr>
<th>Inconsistent state after the occurrence of an event</th>
<th>The postcondition of an event is not satisfied.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Some static constraint is invalid and it is modified.</td>
<td>The event postcondition/method is incorrect and it is modified.</td>
</tr>
<tr>
<td>Some initial integrity constraint is invalid and it is modified.</td>
<td>Some constraint is invalid and it is modified.</td>
</tr>
<tr>
<td>The method is not correct and it is modified.</td>
<td>The postcondition is not correct and it is modified.</td>
</tr>
</tbody>
</table>
DEVELOPMENT OF THE CONCEPTUAL SCHEMA OF THE osTICKET SYSTEM BY APPLYING TDCM

Iteration 51

Iteration objective

S36: AssignTicket_InVacationMode

Current test case

```java
test S36{
    load testConfiguration3;
    load created_tickets;
    li := new LogIn(username:='mary', password:='yyy');
    assert occurrence li;
    at:=new AssignTicket(staffMember:=generalConsultant, ticket:=ticket1, assignee:=generalConsultantVacation,
                          assignmentText:='This is for you');
    assert non-occurrence at;
}
```

TDCM application: Summary of changes performed in the schema

- Added
- Updated

- Preconditions of the domain event at:AssignTicket are satisfied and consequently, the event can occur

```java
context AssignTicket ini inv assigneeIsNotInVacationMode:
    not(self.assignee.isInVacationMode)
```

Time spent

<table>
<thead>
<tr>
<th>TIME TO WRITE TEST CASES (IN MINUTES)</th>
<th>1,5</th>
</tr>
</thead>
<tbody>
<tr>
<td>TIME TO COMPLETE THE ITERATION (IN MINUTES)</td>
<td>2</td>
</tr>
</tbody>
</table>

Errors and failures that drive the conceptual modeling

<table>
<thead>
<tr>
<th>A basic type involved in a test case does not exist in the CSUD</th>
<th>A derived type involved in a test case does not exist in the CSUD</th>
<th>An event type involved in a test case does not exist in the CSUD</th>
</tr>
</thead>
<tbody>
<tr>
<td>The basic type is relevant and it is added to the CSUD</td>
<td>The derived type is relevant and it is added to the CSUD</td>
<td>The event type is relevant and it is added to the CSUD</td>
</tr>
</tbody>
</table>
DEVELOPMENT OF THE CONCEPTUAL SCHEMA OF THE osTICKET SYSTEM BY APPLYING TDCM

Research report
April 2011

Inconsistent state before the occurrence of an event | Inconsistent state after the occurrence of an event | The postcondition of an event is not satisfied.
---|---|---
Some static constraint is invalid and it is modified. | The postcondition/method is incorrect and it is modified. | The method is not correct and it is modified.
Some initial integrity constraint is invalid and it is modified. | Some constraint is invalid and it is modified. | The postcondition is not correct and it is modified.

An assertion about the IB state fails or contains an error | Assert non-occurrence fails | Semantic error in an expression
---|---|---
The effect of an event type is not correct | A derivation rule is incorrect | A precondition is added/updated
A assertion about the IB state fails or contains an error | The expression is corrected | The CSUD is changed

Iteration 52

Iteration objective

S37: AssignTicket_NotLoggedIn

Current test case

```java
test S37{
    load testConfiguration3;
    load created_tickets;
    at:=new AssignTicket(staffMember:=generalConsultant, ticket:=ticket1, assignee:=generalAdministrator,
        assignmentText:='This is for you');
    assert non-occurrence at;
}
```

TDCM application: Summary of changes performed in the schema

- Added
- Updated

- Preconditions of the domain event at:AssignTicket are satisfied and consequently, the event can occur

  context AssignTicket ini inv staffMemberIsLoggedIn:
  self.staffMember.isLoggedin

Time spent

<table>
<thead>
<tr>
<th>Description</th>
<th>Time (in minutes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>TIME TO WRITE TEST CASES</td>
<td>1.5</td>
</tr>
<tr>
<td>TIME TO COMPLETE THE ITERATION</td>
<td>1</td>
</tr>
</tbody>
</table>
Errors and failures that drive the conceptual modeling

<table>
<thead>
<tr>
<th>Errors and failures</th>
<th>A basic type involved in a test case does not exist in the CSUD</th>
<th>A derived type involved in a test case does not exist in the CSUD</th>
<th>An event type involved in a test case does not exist in the CSUD</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>The basic type is relevant and it is added to the CSUD</td>
<td>The derived type is relevant and it is added to the CSUD</td>
<td>The event type is relevant and it is added to the CSUD</td>
</tr>
<tr>
<td>Inconsistent state before the occurrence of an event</td>
<td>Inconsistent state after the occurrence of an event</td>
<td>The postcondition of an event is not satisfied</td>
<td></td>
</tr>
<tr>
<td>Some static constraint is invalid and it is modified.</td>
<td>Some initial integrity constraint is invalid and it is modified.</td>
<td>The event postcondition/method is incorrect and it is modified.</td>
<td></td>
</tr>
<tr>
<td>The basic type is relevant and it is added to the CSUD</td>
<td>The derived type is relevant and it is added to the CSUD</td>
<td>The event type is relevant and it is added to the CSUD</td>
<td></td>
</tr>
<tr>
<td>The postcondition of an event is not satisfied.</td>
<td>The event postcondition/method is incorrect and it is modified.</td>
<td>The method is not correct and it is modified.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Semantic error in an expression</th>
<th>Assert non-occurrence fails</th>
<th>A static constraint needs to be changed</th>
</tr>
</thead>
<tbody>
<tr>
<td>The effect of an event type is incorrect</td>
<td>A derivation rule is incorrect</td>
<td>A precondition is added/updated</td>
</tr>
<tr>
<td>Some constraint is invalid and it is modified.</td>
<td>The expression is corrected</td>
<td>The CSUD is changed</td>
</tr>
</tbody>
</table>

**Iteration 53**

**Iteration objective**

S38: ReleaseTicket

**Current test case**

```java
test S38{
    load testConfiguration3;
    load created_tickets;
    li := new LogIn(username:='mary', password:='yyy');
    assert occurrence li;
    rt:=new ReleaseTicket(staffMember:=generalConsultant, ticket:=ticket1);
    assert occurrence rt;
    assert true ticket1.assignedStaff.isUndefined();

    assert true [ticket1.internalNote->one(i|i.datetime=sys.currentDateTime and i.text='Released ticket' and i.subject='Ticket unassigned' and i.author='Mary')];
}
```
TDCM application: Summary of changes performed in the schema

- [TicketsManagementAndTracking.cstl] <line 1578> ReleaseTicket is not defined in the CSUT as a class or an association

```
event ReleaseTicket
operations
effect()
end

association releaseTicket_staffMember between
  ReleaseTicket[*]
  StaffMember[*]
end

association releaseTicket_ticket between
  ReleaseTicket[*]
  Ticket[*]
end

calendar Date releaseTicket::effect()
post:
  self.ticket.assignedStaff->isEmpty()
  and (InternalNote.allInstances- InternalNote.allInstances@pre)
  ->one(i | i.oclIsNew()
  and i.datetime=System.allInstances()->any(true).currentDateTime
  and i.subject='Ticket unassigned'
  and i.text='Released ticket'
  and i.author=self.staffMember.firstName
  and i.ticket=self.ticket)
```

Time spent

<table>
<thead>
<tr>
<th>Time to write test cases (in minutes)</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time to complete the iteration (in minutes)</td>
<td>13</td>
</tr>
</tbody>
</table>

Errors and failures that drive the conceptual modeling

<table>
<thead>
<tr>
<th>A basic type involved in a test case does not exist in the CSUD</th>
<th>A derived type involved in a test case does not exist in the CSUD</th>
<th>An event type involved in a test case does not exist in the CSUD</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Inconsistent state before the occurrence of an event</th>
<th>Inconsistent state after the occurrence of an event</th>
<th>The postcondition of an event is not satisfied.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Some static constraint is invalid and it is modified.</td>
<td>Some initial integrity constraint is invalid and it is modified.</td>
<td>The event postcondition/method is incorrect and it is modified.</td>
</tr>
<tr>
<td>The event postcondition/method is incorrect and it is modified.</td>
<td>Some constraint is invalid and it is modified.</td>
<td>The method is not correct and it is modified.</td>
</tr>
<tr>
<td>Some assertion about the IB state fails or contains an error</td>
<td>Assert non-occurrence fails</td>
<td>Semantic error in an expression</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Assert consistency fails</th>
<th>A static constraint needs to be changed</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Iteration 54

Iteration objective

S39: ReleaseTicket_ticketIsNotVisible

Current test case

test S39{
  load testConfiguration3;
  load created_tickets;
  li := new LogIn(username:='martin', password:='ttt');
  assert occurrence li;
  rt:=new ReleaseTicket(staffMember:=technicalActive, ticket:=ticket2);
  assert non-occurrence rt;
}

TDCM application: Summary of changes performed in the schema

- Added
- Updated

- Preconditions of the domain event rt:ReleaseTicket are satisfied and consequently, the event can occur

  context ReleaseTicket ini inv theTicketsVisible:
  self.staffMember.isAdministrator or
  (self.staffMember.staffGroup.departmentsAccess->including(self.staffMember.department))
  ->includes(self.ticket.assignedDepartment)

Time spent

| TIME TO WRITE TEST CASES (IN MINUTES) | 1.5 |
| TIME TO COMPLETE THE ITERATION (IN MINUTES) | 2 |

Errors and failures that drive the conceptual modeling

<table>
<thead>
<tr>
<th>A basic type involved in a test case does not exist in the CSUD</th>
<th>A derived type involved in a test case does not exist in the CSUD</th>
<th>An event type involved in a test case does not exist in the CSUD</th>
</tr>
</thead>
<tbody>
<tr>
<td>The basic type is relevant and it is added to the CSUD</td>
<td>The derived type is relevant and it is added to the CSUD</td>
<td>The event type is relevant and it is added to the CSUD</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Inconsistent state before the occurrence of an event</th>
<th>Inconsistent state after the occurrence of an event</th>
<th>The postcondition of an event is not satisfied</th>
</tr>
</thead>
<tbody>
<tr>
<td>Some static constraint is invalid and it is modified.</td>
<td>The event postcondition/method is incorrect and it is modified.</td>
<td>Some constraint is invalid and it is modified.</td>
</tr>
<tr>
<td>Some initial integrity constraint is invalid and it is modified.</td>
<td>The method is not correct and it is modified.</td>
<td>The postcondition is not correct and it is modified.</td>
</tr>
</tbody>
</table>
An assertion about the IB state fails or contains an error | Assert non-occurrence fails | Semantic error in an expression
---|---|---
The effect of an event type is not correct | A derivation rule is incorrect | A precondition is added/updated

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Iteration 55**

**Iteration objective**

S40: ReleaseTicket_notAssigned

**Current test case**

```plaintext
test S40{
    load testConfiguration3;
    load created_tickets;
    li := new LogIn(username:='mary', password:='yyy');
    assert occurrence li;
    rt:=new ReleaseTicket(staffMember:=generalConsultant, ticket:=ticket4);
    assert non-occurrence rt;
}
```

**TDCM application: Summary of changes performed in the schema**

- Added
- Updated

- Preconditions of the domain event rt:ReleaseTicket are satisfied and consequently, the event can occur

  ```plaintext
  context ReleaseTicket ini inv theTicketIsAssigned:
  self.ticket.assignedStaff.isDefined()
  ```

**Time spent**

| TIME TO WRITE TEST CASES (IN MINUTES) | 2 |
| TIME TO COMPLETE THE ITERATION (IN MINUTES) | 4 |
Errors and failures that drive the conceptual modeling

<table>
<thead>
<tr>
<th>A basic type involved in a test case does not exist in the CSUD</th>
<th>A derived type involved in a test case does not exist in the CSUD</th>
<th>An event type involved in a test case does not exist in the CSUD</th>
</tr>
</thead>
<tbody>
<tr>
<td>The basic type is relevant and it is added to the CSUD</td>
<td>The derived type is relevant and it is added to the CSUD</td>
<td>The event type is relevant and it is added to the CSUD</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Inconsistent state before the occurrence of an event</th>
<th>Inconsistent state after the occurrence of an event</th>
<th>The postcondition of an event is not satisfied.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Some static constraint is invalid and it is modified.</td>
<td>Some initial integrity constraint is invalid and it is modified.</td>
<td>The event postcondition/method is incorrect and it is modified.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Some constraint is invalid and it is modified.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The method is not correct and it is modified.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The postcondition is not correct and it is modified.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>An assertion about the IB state fails or contains an error</th>
<th>Assert non-occurrence fails</th>
<th>Semantic error in an expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>The effect of an event type is incorrect</td>
<td>A derivation rule is incorrect</td>
<td>The expression is corrected</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The CSUD is changed</td>
</tr>
</tbody>
</table>

Iteration 56

Iteration objective

S41: ReleaseTicket_NotLoggedIn

Current test case

test S41{
    load testConfiguration3;
    load created_tickets;
    rt:=new ReleaseTicket(staffMember:=generalConsultant, ticket:=ticket2);
    assert non-occurrence rt;
}

TDCM application: Summary of changes performed in the schema

- Preconditions of the domain event rt:ReleaseTicket are satisfied and consequently, the event can occur

    context ReleaseTicket ini inv staffMemberIsLoggedIn:
    self.staffMember.isLogIn

Time spent

<table>
<thead>
<tr>
<th>TIME TO WRITE TEST CASES (IN MINUTES)</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>TIME TO COMPLETE THE ITERATION (IN MINUTES)</td>
<td>1,5</td>
</tr>
</tbody>
</table>
Errors and failures that drive the conceptual modeling

<table>
<thead>
<tr>
<th>A basic type involved in a test case does not exist in the CSUD</th>
<th>A derived type involved in a test case does not exist in the CSUD</th>
<th>An event type involved in a test case does not exist in the CSUD</th>
</tr>
</thead>
<tbody>
<tr>
<td>The basic type is relevant and it is added to the CSUD</td>
<td>The derived type is relevant and it is added to the CSUD</td>
<td>The event type is relevant and it is added to the CSUD</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Inconsistent state before the occurrence of an event</th>
<th>Inconsistent state after the occurrence of an event</th>
<th>The postcondition of an event is not satisfied.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Some static constraint is invalid and it is modified.</td>
<td>Some initial integrity constraint is invalid and it is modified.</td>
<td>Some constraint is invalid and it is modified.</td>
</tr>
<tr>
<td></td>
<td>The event postcondition/method is incorrect and it is modified.</td>
<td>The method is not correct and it is modified.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The postcondition is not correct and it is modified.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>An assertion about the IB state fails or contains an error</th>
<th>Assert non-occurrence fails</th>
<th>Semantic error in an expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>The effect of an event type is not correct</td>
<td>A derivation rule is incorrect</td>
<td>The expression is corrected</td>
</tr>
<tr>
<td></td>
<td>A precondition is added/updated</td>
<td>The CSUD is changed</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Assert consistency fails</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>A static constraint needs to be changed</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Iteration 57

Iteration objective

S42: EditTicket

Current test case

test S42{
    load testConfiguration3;
    load created_tickets;
    li := new LogIn(username:='mary', password:='yyy');
    assert occurrence li;
    dt3:=new Datetime(value:=[(sys.currentDateTime.value+3)]);
    et:=new EditTicket(staffMember:=generalConsultant, ticket:=ticket1,
        emailAddress:='mary2@marnes.mar',
        fullName:='Mary Marnes2',
        subject:='Error operating system2',
        telephone:='xxx2',
        ext:='xx2',
        dueDateTime:=dt3,
        priority:=#Low,
        helpTopic:=helpTopicUse,
        editionInternalNote:='The customer asks for this changes');
    assert occurrence et;
    assert equals ticket1.email 'mary2@marnes.mar';
    assert equals ticket1.fullName 'Mary Marnes2';
    assert equals ticket1.subject 'Error operating system2';
    assert equals ticket1.telephone 'xxx2';
    assert equals ticket1.ext 'xx2';
    assert equals ticket1.priority #Low;
    assert equals ticket1.helpTopic helpTopicUse;
    assert true [ticket1.internalNote->one(i|i.datetime=sys.currentDateTime and
        i.text='The customer asks for this changes' and]
TDCM application: Summary of changes performed in the schema

- [TicketsManagementAndTracking.cstl] <line 1643> EditTicket is not defined in the CSUT as a class or an association

```
event EditTicket
attributes
emailAddress: String
fullName: String
subject: String
telephone: String[0..1]
ext: String[0..1]
priority: Priority
dueDatetime: Datetime
editionInternalNote: String
operations
effect()
end

association editTicket_staffMember between
   EditTicket[*]
   StaffMember[1]
end

association editTicket_ticket between
   EditTicket[*]
   Ticket[1]
end

association editTicket_helpTopic between
   EditTicket[*]
   HelpTopic[1]
end

context EditTicket::effect()
post:
self.ticket.email=self.emailAddress and
self.ticket.fullName=self.fullName and
self.ticket.subject=self.subject and
self.ticket.telephone=self.telephone and
self.ticket.ext=self.ext and
self.ticket.priority=self.priority and
self.ticket.helpTopic=self.helpTopic and
self.ticket.dueDatetime=self.dueDatetime and
(InternalNote.allInstances- InternalNote.allInstances@pre)
   ->one(i | i.oclsNew() and i.datetime=System.allInstances()->first(any(true),currentDateTime
   and i.subject='Ticket updated'
   and i.text=self.editionInternalNote
   and i.author=self.staffMember.firstName
   and i.ticket=self.ticket)
```

Time spent

| Time to Complete Test Cases (in minutes) | 6 |
| Time to Complete the Iteration (in minutes) | 32 |
Errors and failures that drive the conceptual modeling

<table>
<thead>
<tr>
<th>Errors or failures</th>
<th>A basic type involved in a test case does not exist in the CSUD</th>
<th>A derived type involved in a test case does not exist in the CSUD</th>
<th>An event type involved in a test case does not exist in the CSUD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inconsistent state before the occurrence of an event</td>
<td>The basic type is relevant and it is added to the CSUD</td>
<td>The derived type is relevant and it is added to the CSUD</td>
<td>The event type is relevant and it is added to the CSUD</td>
</tr>
<tr>
<td>An assertion about the IB state fails or contains an error</td>
<td>Some static constraint is invalid and it is modified.</td>
<td>Some initial integrity constraint is invalid and it is modified.</td>
<td>The event postcondition/method is incorrect and it is modified.</td>
</tr>
<tr>
<td>Assert consistency fails</td>
<td>The postcondition of an event is not satisfied.</td>
<td>The postcondition of an event is not satisfied.</td>
<td>The postcondition of an event is not satisfied.</td>
</tr>
<tr>
<td>Assert non-occurrence fails</td>
<td>The postcondition is not correct and it is modified.</td>
<td>The method is not correct and it is modified.</td>
<td>The postcondition is not correct and it is modified.</td>
</tr>
<tr>
<td>Semantic error in an expression</td>
<td>The expression is corrected</td>
<td>The expression is corrected</td>
<td>The expression is corrected</td>
</tr>
<tr>
<td>The effect of an event type is not correct</td>
<td>A derivation rule is incorrect</td>
<td>A precondition is added/updated</td>
<td>The CSUD is changed</td>
</tr>
<tr>
<td>A static constraint needs to be changed</td>
<td>The postcondition is not correct and it is modified.</td>
<td>The postcondition is not correct and it is modified.</td>
<td>The postcondition is not correct and it is modified.</td>
</tr>
</tbody>
</table>

Iteration 58

Iteration objective

S43: EditTicket_ticketIsNotVisible

Current test case

test S43{
    load testConfiguration3;
    load created_tickets;
    li := new LogIn(username:='martin', password:='ttt');
    assert occurrence li;
    dt3:=new Datetime(value:=[(sys.currentDateTime.value+2)];
    et:=new EditTicket(staffMember:=technicalActive, ticket:=ticket2,
        emailAddress:='john2@johnes.nes',
        fullName:='John Johnes2',
        subject:='Can I reply a ticket? Yes or no?,'
        telephone:='yyy2',
        ext:='yy2',
        dueDateTime:=dt3,
        priority:=#Normal,
        helpTopic:=helpTopicUse,
        editionInternalNote:='The customer asks for this changes');
    assert non-occurrence et;
}

April 2011

Research report
TDCM application: Summary of changes performed in the schema

- Added
- Updated

- Preconditions of the domain event et:EditTicket are satisfied and consequently, the event can occur

```plaintext
context EditTicket ini inv theTicketIsVisible:
  self.staffMember.isAdministrator or
  (self.staffMember.staffGroup.departmentsAccess->including(self.staffMember.department))
  ->includes(self.ticket.assignedDepartment)
```

**Time spent**

<table>
<thead>
<tr>
<th>Time to Write Test Cases (in minutes)</th>
<th>2.5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time to Complete the Iteration (in minutes)</td>
<td>5.5</td>
</tr>
</tbody>
</table>

**Errors and failures that drive the conceptual modeling**

<table>
<thead>
<tr>
<th>A basic type involved in a test case does not exist in the CSUD</th>
<th>A derived type involved in a test case does not exist in the CSUD</th>
<th>An event type involved in a test case does not exist in the CSUD</th>
</tr>
</thead>
<tbody>
<tr>
<td>The basic type is relevant and it is added to the CSUD</td>
<td>The derived type is relevant and it is added to the CSUD</td>
<td>The event type is relevant and it is added to the CSUD</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Inconsistent state before the occurrence of an event</th>
<th>Inconsistent state after the occurrence of an event</th>
<th>The postcondition of an event is not satisfied</th>
</tr>
</thead>
<tbody>
<tr>
<td>Some static constraint is invalid and it is modified.</td>
<td>Some initial integrity constraint is invalid and it is modified.</td>
<td>The event postcondition/method is incorrect and it is modified.</td>
</tr>
<tr>
<td>The event postcondition/method is incorrect and it is modified.</td>
<td>Some constraint is invalid and it is modified.</td>
<td>The method is not correct and it is modified.</td>
</tr>
<tr>
<td>The postcondition of an event is not satisfied.</td>
<td>Some constraint is invalid and it is modified.</td>
<td>The postcondition is not correct and it is modified.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>An assertion about the IB state fails or contains an error</th>
<th>Assert non-occurrence fails</th>
<th>Semantic error in an expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>The effect of an event type is not correct</td>
<td>A derivation rule is incorrect</td>
<td>The expression is corrected</td>
</tr>
<tr>
<td>A precondition is added/updated</td>
<td></td>
<td>The CSUD is changed</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Assert consistency fails</th>
</tr>
</thead>
<tbody>
<tr>
<td>A static constraint needs to be changed</td>
</tr>
</tbody>
</table>

**Iteration 59**

**Iteration objective**

S44: EditTicket_NotAllowed

**Current test case**

```plaintext
test S44{
  load testConfiguration3;
```
load created_tickets;

li := new LogIn(username:='martin', password:='ttt');
assert occurrence li;
dt3:=new Datetime(value:=(sys.currentDateTime.value+2));
et:=new EditTicket(staffMember:=technicalActive, ticket:=ticket1,
    emailAddress:='mary2@marnes.mar',
    fullName:='John Johnes2',
    subject:='Can I reply a ticket? Yes or no?',
    telephone:='yyy2',
    ext:='yy2',
    dueDatetime:=dt3,
    priority:=#Normal,
    helpTopic:=helpTopicUse,
    editionInternalNote:='The customer asks for this changes');
assert non-occurrence et;
}

TDCM application: Summary of changes performed in the schema

- Preconditions of the domain event et:EditTicket are satisfied and consequently, the event can occur

context EditTicket ini inv staffMemberIsNotAllowedToEditTickets:
    self.staffMember.staffGroup.canEditTickets

Time spent

| TIME TO WRITE TEST CASES (IN MINUTES) | 2 |
| TIME TO COMPLETE THE ITERATION (IN MINUTES) | 4 |

Errors and failures that drive the conceptual modeling

<table>
<thead>
<tr>
<th>A basic type involved in a test case does not exist in the CSUD</th>
<th>A derived type involved in a test case does not exist in the CSUD</th>
<th>An event type involved in a test case does not exist in the CSUD</th>
</tr>
</thead>
<tbody>
<tr>
<td>The basic type is relevant and it is added to the CSUD</td>
<td>The derived type is relevant and it is added to the CSUD</td>
<td>The event type is relevant and it is added to the CSUD</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Inconsistent state before the occurrence of an event</th>
<th>Inconsistent state after the occurrence of an event</th>
<th>The postcondition of an event is not satisfied.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Some static constraint is invalid and it is modified.</td>
<td>Some initial integrity constraint is invalid and it is modified.</td>
<td>Some constraint is invalid and it is modified.</td>
</tr>
<tr>
<td>The event postcondition/method is incorrect and it is modified.</td>
<td>The method is not correct and it is modified.</td>
<td>The postcondition is not correct and it is modified.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>An assertion about the IB state fails or contains an error</th>
<th>Assert non-occurrence fails</th>
<th>Semantic error in an expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>The effect of an event type is not correct</td>
<td>A derivation rule is incorrect</td>
<td>A precondition is added/updated</td>
</tr>
<tr>
<td>The expression is corrected</td>
<td>The CSUD is changed</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Assert consistency fails</th>
<th>A static constraint needs to be changed</th>
</tr>
</thead>
</table>
Iteration 60

Iteration objective

S45: EditTicket_NotAllowedButAdministrator

Current test case

test S45{
  load testConfiguration3;
  load created_tickets;
  technicalActive.isAdministrator:=true;
  li := new LogIn(username:='martin', password:='ttt');
  assert occurrence li;
  dt3:=new Datetime(value:=[sys.currentDateTime.value+2]);
  et:=new EditTicket(staffMember:=technicalActive, ticket:=ticket1, emailAddress:='mary2@marm.es.mar',
                     fullName:='John Johnes2',
                     subject:='Can I reply a ticket? Yes or no?',
                     telephone:='yyy2',
                     ext:='yy2',
                     dueDateTime:=dt3,
                     priority:='#Normal',
                     helpTopic:=helpTopicUse,
                     editionInternalNote:='The customer asks for this changes');
  assert occurrence et;
}

TDCM application: Summary of changes performed in the schema

- Added
- Updated

- [TicketsManagementAndTracking.cstl] <line 1735> Initial IC 'staffMemberIsNotAllowedToEditTickets ' of et:EditTicket event fails

context EditTicket ini inv staffMemberIsNotAllowedToEditTickets:
  self.staffMember.staffGroup.canEditTickets or self.staffMember.isAdministrator

Time spent

| TIME TO WRITE TEST CASES (IN MINUTES) | 2 |
| TIME TO COMPLETE THE ITERATION (IN MINUTES) | 1 |

Errors and failures that drive the conceptual modeling

<table>
<thead>
<tr>
<th>A basic type involved in a test case does not exist in the CSUD</th>
<th>A derived type involved in a test case does not exist in the CSUD</th>
<th>An event type involved in a test case does not exist in the CSUD</th>
</tr>
</thead>
<tbody>
<tr>
<td>The basic type is relevant and it is added to the CSUD</td>
<td>The derived type is relevant and it is added to the CSUD</td>
<td>The event type is relevant and it is added to the CSUD</td>
</tr>
</tbody>
</table>
Iteration 61

Iteration objective

S46: EditTicket_NotLoggedIn

Current test case

test S46{
    load testConfiguration3;
    load created_tickets;
    dt3:=new Datetime(value:=[(sys.currentDateTime.value+3)]);
    et:=new EditTicket(staffMember:=generalConsultant, ticket:=ticket1,
        emailAddress:='mary2@marnes.mar',
        fullName:='Mary Marnes2',
        subject:='Error operating system2',
        telephone:='xxx2',
        ext:='xx2',
        dueDatetime:=dt3,
        priority:=#Low,
        helpTopic:=helpTopicUse,
        editionInternalNote:='The customer asks for this changes');
    assert non-occurrence et;
}

TDCM application: Summary of changes performed in the schema

- Preconditions of the domain event et:EditTicket are satisfied and consequently, the event can occur

context EditTicket ini inv staffMemberIsLoggedIn:
    self.staffMember.isLoggedin

---

Inconsistent state before the occurrence of an event | Inconsistent state after the occurrence of an event | The postcondition of an event is not satisfied

- Some static constraint is invalid and it is modified.
- Some initial integrity constraint is invalid and it is modified.
- The event postcondition/method is incorrect and it is modified.
- Some constraint is invalid and it is modified.
- The postcondition is not correct and it is modified.

- The event postcondition/method is incorrect and it is modified.
- Some constraint is invalid and it is modified.
- The postcondition is not correct and it is modified.

An assertion about the IB state fails or contains an error | Assert non-occurrence fails | Semantic error in an expression

- The effect of an event type is not correct
- A derivation rule is incorrect
- A precondition is added/updated
- The expression is corrected
- The CSUD is changed

Assert consistency fails | A static constraint needs to be changed

- Preconditions of the domain event et:EditTicket are satisfied and consequently, the event can occur
Time spent

<table>
<thead>
<tr>
<th>Time to write test cases (in minutes)</th>
<th>1.5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time to complete the iteration (in minutes)</td>
<td>2</td>
</tr>
</tbody>
</table>

Errors and failures that drive the conceptual modeling

<table>
<thead>
<tr>
<th>A basic type involved in a test case does not exist in the CSUD</th>
<th>A derived type involved in a test case does not exist in the CSUD</th>
<th>An event type involved in a test case does not exist in the CSUD</th>
</tr>
</thead>
<tbody>
<tr>
<td>The basic type is relevant and it is added to the CSUD</td>
<td>The derived type is relevant and it is added to the CSUD</td>
<td>The event type is relevant and it is added to the CSUD</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Inconsistent state before the occurrence of an event</th>
<th>Inconsistent state after the occurrence of an event</th>
<th>The postcondition of an event is not satisfied.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Some static constraint is invalid and it is modified.</td>
<td>Some initial integrity constraint is invalid and it is modified.</td>
<td>The event postcondition/method is incorrect and it is modified.</td>
</tr>
<tr>
<td>Some static constraint is invalid and it is modified.</td>
<td>Some initial integrity constraint is invalid and it is modified.</td>
<td>Some constraint is invalid and it is modified.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>An assertion about the IB state fails or contains an error</th>
<th>Assert non-occurrence fails</th>
<th>Semantic error in an expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>The effect of an event type is not correct</td>
<td>A derivation rule is incorrect</td>
<td>The expression is corrected</td>
</tr>
<tr>
<td>Assert consistency fails</td>
<td>A precondition is added/updated</td>
<td>The CSUD is changed</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Assert consistency fails</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>A static constraint needs to be changed</td>
<td></td>
</tr>
</tbody>
</table>

**Iteration 62**

**Iteration objective**

S47: PostTicketReply_alertsAndAutoresponsesActive

**Current test case**

test S47()
load test:Configuration1;
helpTopicInstallation.autoresponse:=#Enabled;
load created_tickets;
li := new LogIn(username:='mary', password:='yyy');
assert occurrence li;
rt:=new PostTicketReply(staffMember:=generalConsultant, ticket:=ticket1,
response:='You should choose the installation executable...');
assert occurrence rt;
assert true [ticket1.ticketThreadMessage->one(m|m.datetime=sys.currentDateTime and
m.text='You should choose the installation executable...' and
m.author='Mary')];
assert equals ticket1.lastMessageDatetime sys.currentDateTime;
assert equals ticket1.lastRespondent generalConsultant;
assert equals ticket1.lastRespondent generalConsultant;
TDCM application: Summary of changes performed in the schema

- [TicketsManagementAndTracking.cstl] <line 1768> PostTicketReply is not defined in the CSUT as a class or an association

```cstl
context PostTicketReply::effect()
post:

let sendNewMessageAlertToLastRespondent:Boolean=
    StaffNoticesAlertsSettings.allInstances()->any(true).alertWhenNewMessage and
    StaffNoticesAlertsSettings.allInstances()->any(true).alertWhenNewMessageStaff->includes(#LastRespondent)
in
let sendNewMessageAlertToAssignedStaff:Boolean=
    StaffNoticesAlertsSettings.allInstances()->any(true).alertWhenNewMessage and
    StaffNoticesAlertsSettings.allInstances()->any(true).alertWhenNewMessageStaff->includes(#AssignedStaff)
in
let sendNewMessageAlertToDepartmentManager:Boolean=
    StaffNoticesAlertsSettings.allInstances()->any(true).alertWhenNewMessage and
    StaffNoticesAlertsSettings.allInstances()->any(true).alertWhenNewMessageStaff->includes(#DepartmentManager)
in
let staffAlertsFromEMailAddress: String=
    EmailSettings.allInstances()->any(true).defaultStaffAlertsEmail.address
in
let sendAutoresponse:Boolean=
    if CustomerAutoresponsesSettings.allInstances()->any(true).autorespondWhenNewMessageAppendedToTicket then
        if self.ticket.helpTopic->notEmpty then
            if self.ticket.helpTopic.autoresponse=#Enabled then true else false
        else false
    else false
```
end if
else
    self.ticket.assignedDepartment.newAddedMessageIsNotified
end if
else false
endif

in
    self.ticket.ticketThreadMessage->one(tdm | tdm.oclIsNew() and tdm.datetime=System.allInstances()->any(true).currentDateTime and tdm.text=self.response and tdm.author=self.staffMember.firstName and self.ticket.lastMessageDatetime=System.allInstances()->any(true).currentDateTime)

and self.ticket.lastRespondent=self.staffMember

and (sendAutoresponse implies EMail.allInstances() exists(e|e.fromAddress=self.ticket.assignedDepartment.autoresponseEmail.address and e.toAddress=self.ticket.email and e.ticketNumber=self.ticket.number))

-- staff notices and (sendNewMessageAlertToLastRespondent and self.ticket.lastRespondent@pre->notEmpty() and not(self.ticket.lastRespondent@pre.isInVacationMode or self.ticket.lastRespondent@pre.status=#Disabled) implies EMail.allInstances() exists(e|e.fromAddress=staffAlertsFromEMailAddress and e.toAddress=self.ticket.lastRespondent@pre and e.ticketNumber=self.ticket.number))

and (sendNewMessageAlertToAssignedStaff and self.ticket.assignedStaff->notEmpty() and not(self.ticket.assignedStaff.isInVacationMode or self.ticket.assignedStaff.status=#Disabled) implies EMail.allInstances() exists(e|e.fromAddress=staffAlertsFromEMailAddress and e.toAddress=self.ticket.assignedStaff.emailAddress and e.ticketNumber=self.ticket.number))

---

### Time spent

| Time to write test cases (in minutes) | 9.5 |
| Time to complete the iteration (in minutes) | 27 |

### Errors and failures that drive the conceptual modeling

<table>
<thead>
<tr>
<th>A basic type involved in a test case does not exist in the CSUD</th>
<th>A derived type involved in a test case does not exist in the CSUD</th>
<th>An event type involved in a test case does not exist in the CSUD</th>
</tr>
</thead>
<tbody>
<tr>
<td>The basic type is relevant and it is added to the CSUD</td>
<td>The derived type is relevant and it is added to the CSUD</td>
<td>The event type is relevant and it is added to the CSUD</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Inconsistent state before the occurrence of an event</th>
<th>Inconsistent state after the occurrence of an event</th>
<th>The postcondition of an event is not satisfied</th>
</tr>
</thead>
<tbody>
<tr>
<td>Some static constraint is invalid and it is modified.</td>
<td>Some initial integrity constraint is invalid and it is modified.</td>
<td>The event postcondition/method is incorrect and it is modified.</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>An assertion about the IB state fails or contains an error</th>
<th>Assert non-occurrence fails</th>
<th>Semantic error in an expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>The effect of an event type is not correct</td>
<td>A derivation rule is incorrect</td>
<td>The expression is added/updated</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>The CSUD is changed</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Assert consistency fails</th>
<th>A static constraint needs to be changed</th>
</tr>
</thead>
</table>
**Iteration 63**

**Iteration objective**

S48: PostTicketReply_alertsAndAutoresponsesDisabled

**Current test case**

```java
test S48{
load testConfiguration4;
helpTopicInstallation.autoresponse:=#Enabled;
load created_tickets;

li := new Login(username:='mary', password:='yyy');
assert occurrence li;

rt:=new PostTicketReply(staffMember:=generalConsultant, ticket:=ticket1,
    response:='You should choose the installation executable...');
assert occurrence rt;

assert true [ticket1.ticketThreadMessage->one(m|m.datetime=sys.currentDateTime and
    m.text='You should choose the installation executable...' and
    m.author='Mary')];

assert equals ticket1.lastMessageDatetime sys.currentDateTime;
assert equals ticket1.lastRespondent generalConsultant;

//autoresponse
assert false [EMail.allInstances())->exists(e|e.emailKind.oclIsTypeOf(TicketResponseNotice) and
    e.fromAddress='technical_at_support.com' and
    e.toAddress='mary_at_marnes.mar' and
    e.ticketNumber=1)];

//notice to assigned staff
assert false [EMail.allInstances())->exists(e|e.emailKind.oclIsTypeOf(NewMessageAlertToStaff) and
    e.fromAddress='general_at_support.com' and
    e.toAddress='mary_at_support.com' and
    e.ticketNumber=1)];

//notice to department manager
assert false [EMail.allInstances())->exists(e|e.emailKind.oclIsTypeOf(NewMessageAlertToStaff) and
    e.fromAddress='general_at_support.com' and
    e.toAddress='john_at_support.com' and
    e.ticketNumber=1)];
}
```

**TDCM application: Summary of changes performed in the schema**

The CSUD has not been changed.

**Time spent**

<table>
<thead>
<tr>
<th>TIME TO WRITE TEST CASES (IN MINUTES)</th>
<th>1,5</th>
</tr>
</thead>
<tbody>
<tr>
<td>TIME TO COMPLETE THE ITERATION (IN MINUTES)</td>
<td>0,5</td>
</tr>
</tbody>
</table>
Iteration 64

Iteration objective

S49: PostTicketReply_ticketIsNotVisible

Current test case

```java
test S49{
    load testConfiguration4;
    load created_tickets;
    li := new LogIn(username:='martin', password:='ttt');
    assert occurrence li;
    rt:=new PostTicketReply(staffMember:=technicalActive, ticket:=ticket2,
        response:='You should choose the installation executable...');
    assert non-occurrence rt;
}
```

TDCM application: Summary of changes performed in the schema

- Added
  - Preconditions of the domain event rt:PostTicketReply are satisfied and consequently, the event can occur

    ```java
    context PostTicketReply ini inv theTicketsVisible:
    self.staffMember.isAdministrator or
    (self.staffMember.staffGroup.departmentsAccess->including(self.staffMember.department))
    -injects(self.ticket.assignedDepartment)
    ```

Time spent

| TIME TO WRITE TEST CASES (IN MINUTES) | 1 |
| TIME TO COMPLETE THE ITERATION (IN MINUTES) | 3 |

Errors and failures that drive the conceptual modeling

| A basic type involved in a test case does not exist in the CSUD | A derived type involved in a test case does not exist in the CSUD | An event type involved in a test case does not exist in the CSUD |
| Inconsistent state before the occurrence of an event | Inconsistent state after the occurrence of an event | The postcondition of an event is not satisfied. |
| Some static constraint is invalid and it is modified. | Some initial integrity constraint is invalid and it is modified. | The event postcondition/method is incorrect and it is modified. |
| Some constraint is invalid and it is modified. | Some constraint is invalid and it is modified. | Some constraint is incorrect and it is modified. |
| The method is not correct and it is modified. | The method is not correct and it is modified. | The postcondition is not correct and it is modified. |
### Iteration 65

#### Iteration objective

S50: PostTicketReply_NotLoggedIn

#### Current test case

```plaintext
test S50{
  load testConfiguration1;
  load created_tickets;
  rt:=new PostTicketReply(staffMember:=generalConsultant, ticket:=ticket1,
                 response:='You should choose the installation executable...');
  assert non-occurrence rt;
}
```

#### TDCM application: Summary of changes performed in the schema

- **Added**
- **Updated**

- Preconditions of the domain event rt:PostTicketReply are satisfied and consequently, the event can occur

  ```plaintext
  context PostTicketReply ini inv staffMemberIsLoggedIn:
  self.staffMember.isLoggedIn
  ```

#### Time spent

<table>
<thead>
<tr>
<th>Time to Write Test Cases (in minutes)</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time to Complete the Iteration (in minutes)</td>
<td>1.5</td>
</tr>
</tbody>
</table>

#### Errors and failures that drive the conceptual modeling

<table>
<thead>
<tr>
<th>A basic type involved in a test case does not exist in the CSUD</th>
<th>A derived type involved in a test case does not exist in the CSUD</th>
<th>An event type involved in a test case does not exist in the CSUD</th>
</tr>
</thead>
<tbody>
<tr>
<td>The basic type is relevant and it is added to the CSUD</td>
<td>The derived type is relevant and it is added to the CSUD</td>
<td>The event type is relevant and it is added to the CSUD</td>
</tr>
</tbody>
</table>
Inconsistent state before the occurrence of an event | Inconsistent state after the occurrence of an event | The postcondition of an event is not satisfied
---|---|---
Some static constraint is invalid and it is modified. | Some initial integrity constraint is invalid and it is modified. | The method is not correct and it is modified.

An assertion about the IB state fails or contains an error | Assert non-occurrence fails | Semantic error in an expression
---|---|---
The effect of an event type is not correct | A derivation rule is incorrect | The expression is corrected | The CSUD is changed

Iteration 66

Iteration objective

S51: PostTicketInternalNote_staffAlertsEnabled

Current test case

```java
    test S51{
        load testConfiguration1;
        load created_tickets;

        li := new LogIn(username:='martin', password:='ttt');
        assert occurrence li;

        pin:=new PostTicketInternalNote(staffMember:=technicalActive, ticket:=ticket1,
            title:='No tickets?', note:='It seems that she does not have tickets');
        assert occurrence pin;

        assert true [ticket1.internalNote->one(m|m.datetime=sys.currentDateTime and
            m.subject='No tickets?' and
            m.text='It seems that she does not have tickets' and
            m.author='Martin')];

        //notice to assigned staff
        assert true [EMail.allInstances()->exists(e|e.emailKind.oclIsTypeOf(NewInternalNoteAlertToStaff) and
            e.fromAddress='general_at_support.com' and
            e.toAddress='mary_at_support.com' and
            e.ticketNumber=1)];

        //notice to department manager
        assert true [EMail.allInstances()->exists(e|e.emailKind.oclIsTypeOf(NewInternalNoteAlertToStaff) and
            e.fromAddress='general_at_support.com' and
            e.toAddress='john_at_support.com' and
            e.ticketNumber=1)];
    }
```
TDCM application: Summary of changes performed in the schema

- Added

[TicketsManagementAndTracking.cstl] <line 1875> PostTicketInternalNote is not defined in the CSUT as a class or an association

```plaintext
event PostTicketInternalNote
  attributes
title: String
note: String
operations
effect()
end

association postTicketInternalNote_ticket between
  PostTicketInternalNote[*]
  Ticket[1]
end

association postTicketInternalNote_staffMember between
  PostTicketInternalNote[*]
  StaffMember[1]
end
```

context PostTicketInternalNote::effect()
post:
```
let sendNewMessageAlertToLastRespondent:Boolean=
  StaffNoticesAlertsSettings.allInstances()->
  any(true).alertWhenNewMessage and
  StaffNoticesAlertsSettings.allInstances()->
  any(true).alertWhenNewMessageStaff->includes(#LastRespondent)
in
let sendNewMessageAlertToAssignedStaff:Boolean=
  StaffNoticesAlertsSettings.allInstances()->
  any(true).alertWhenNewMessage and
  StaffNoticesAlertsSettings.allInstances()->
  any(true).alertWhenNewMessageStaff->includes(#AssignedStaff)
in
let sendNewMessageAlertToDepartmentManager:Boolean=
  StaffNoticesAlertsSettings.allInstances()->
  any(true).alertWhenNewMessage and
  StaffNoticesAlertsSettings.allInstances()->
  any(true).alertWhenNewMessageStaff->includes(#DepartmentManager)
in
let staffAlertsFromEMailAddress:String=
  EmailSettings.allInstances()->
  any(true).defaultStaffAlertsEmail.address
in
self.ticket.internalNote->one(int | int.oclIsNew() and
  int.datetime=System.allInstances()->
  any(true).currentDateTime
  and int.subject=self.title
  and int.text=self.note
  and int.author=self.staffMember.firstName
  and self.ticket.lastMessageDatetime=System.allInstances()->
  any(true).currentDateTime)
```

-- staff notices
and (sendNewMessageAlertToLastRespondent and
  self.ticket.lastRespondent@pre->notEmpty() and
  not(self.ticket.lastRespondent@pre.isInVacationMode or self.ticket.lastRespondent@pre.status=#Disabled)
  implies
  EMail.allInstances()->
  exists(e|e.fromAddress=staffAlertsFromEMailAddress and
  e.toAddress=self.ticket.lastRespondent@pre and
  e.ticketNumber=self.ticket.number))
```
and (sendNewMessageAlertToAssignedStaff
  and self.ticket.assignedStaff->notEmpty() and
  not(self.ticket.assignedStaff.isInVacationMode or self.ticket.assignedStaff.status=#Disabled)
  implies
  EMail.allInstances()->
  exists(e|e.fromAddress=staffAlertsFromEMailAddress and
  e.toAddress=self.ticket.assignedStaff.emailAddress and
  e.ticketNumber=self.ticket.number))
```
DEVELOPMENT OF THE CONCEPTUAL SCHEMA OF THE osTICKET SYSTEM BY APPLYING TDCM

Time spent

<table>
<thead>
<tr>
<th>Event Description</th>
<th>Time (in minutes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time to write test cases</td>
<td>11</td>
</tr>
<tr>
<td>Time to complete the iteration</td>
<td>19</td>
</tr>
</tbody>
</table>

Errors and failures that drive the conceptual modeling

<table>
<thead>
<tr>
<th>Event Description</th>
<th>Time (in minutes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A basic type involved in a test case does not exist in the CSUD</td>
<td>4</td>
</tr>
<tr>
<td>A derived type involved in a test case does not exist in the CSUD</td>
<td>1</td>
</tr>
<tr>
<td>An event type involved in a test case does not exist in the CSUD</td>
<td>1</td>
</tr>
<tr>
<td>Inconsistent state before the occurrence of an event</td>
<td></td>
</tr>
<tr>
<td>Inconsistent state after the occurrence of an event</td>
<td></td>
</tr>
<tr>
<td>Some static constraint is invalid and it is modified.</td>
<td></td>
</tr>
<tr>
<td>Some initial integrity constraint is invalid and it is modified.</td>
<td></td>
</tr>
<tr>
<td>Some constraint is invalid and it is modified.</td>
<td></td>
</tr>
<tr>
<td>Some constraint is not correct and it is modified.</td>
<td></td>
</tr>
<tr>
<td>The postcondition of an event is not satisfied.</td>
<td></td>
</tr>
</tbody>
</table>

Iteration 67

Iteration objective

S52: PostTicketInternalNote_staffAlertsDisabled

Current test case

test S52/
load testConfiguration4;
load created_tickets;
li := new LogIn(username:='martin', password:='ttt');
assert occurrence li;

pin:=new PostTicketInternalNote(staffMember:=technicalActive, ticket:=ticket5,
title:='No tickets?', note:='It seems that she does not have tickets');
assert occurrence pin;

assert true [ticket5.internalNote->one(m|m.datetime=sys.currentDateTime and 
m.subject='No tickets?'and 
m.text='It seems that she does not have tickets' and 
m.author='Martin')];

//notice to assigned staff
assert false [EMail.allInstances()->exists(e|e.emailKind.oclIsTypeOf(NewInternalNoteAlertToStaff) and 
e.fromAddress='general_at_support.com' and 
e.subject='New Ticket Internal Note Staff Alerts Disabled')]


The conceptual schema has not been changed

Time spent

| Time to write test cases (in minutes) | 2 |
| Time to complete the iteration (in minutes) | 0.5 |

Iteration 68

Iteration objective

S53: PostTicketInternalNote.TicketNotVisible

Current test case

test S53{
    load testConfiguration4;
    load created_tickets;
    li := new LogIn(username:='martin', password:='ttt');
    assert occurrence li;
    pin:=new PostTicketInternalNote(staffMember:=technicalActive, ticket:=ticket4,
        title:='Checked button', note:='Checked that the button appears');
    assert non-occurrence pin;
}

TDCM application: Summary of changes performed in the schema

- Added
- Updated

Preconditions of the domain event pin:PostTicketInternalNote are satisfied and consequently, the event can occur

context PostTicketInternalNote ini inv theTicketIsVisible:
    self.staffMember.isAdministrator or
    (self.staffMember.staffGroup.departmentsAccess->including(self.staffMember.department))
Time spent

<table>
<thead>
<tr>
<th>Errors and failures that drive the conceptual modeling</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>A basic type involved in a test case does not exist in the CSUD</strong></td>
</tr>
<tr>
<td>The basic type is relevant and it is added to the CSUD</td>
</tr>
<tr>
<td><strong>Inconsistent state before the occurrence of an event</strong></td>
</tr>
<tr>
<td>Some static constraint is invalid and it is modified.</td>
</tr>
<tr>
<td><strong>An assertion about the IB state fails or contains an error</strong></td>
</tr>
<tr>
<td>The effect of an event type is not correct</td>
</tr>
<tr>
<td><strong>Assert consistency fails</strong></td>
</tr>
<tr>
<td><strong>1</strong></td>
</tr>
</tbody>
</table>

Iteration 69

Iteration objective

S54: PostTicketInternalNote_NotLoggedIn

Current test case

test S54{
    load testConfiguration4;
    load created_tickets;

    pin:=new PostTicketInternalNote(staffMember:=technicalActive, ticket:=ticket1, title:='No tickets?', note:='It seems that she does not have tickets');
    assert non-occurrence pin;
}
TDCM application: Summary of changes performed in the schema

- Added
  - Preconditions of the domain event pin:PostTicketInternalNote are satisfied and consequently, the event can occur

```plaintext
context PostTicketInternalNote ini inv staffMemberIsLoggedIn:
  self.staffMember.isLoggedIn
```

### Time spent

<table>
<thead>
<tr>
<th>Description</th>
<th>Time (in minutes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time to write test cases</td>
<td>0.5</td>
</tr>
<tr>
<td>Time to complete the iteration</td>
<td>1</td>
</tr>
</tbody>
</table>

### Refactoring

We realize that we could represent all the events that require logged in staff and a visible ticket as subclasses of an abstract class named `StaffTicketEvent`.

```plaintext
event StaffTicketEvent
end

association staffTicketEvent_ticket between
  StaffTicketEvent[*]
  Ticket[1]
end

association staffTicketEvent_staffMember between
  StaffTicketEvent[*]
  StaffMember[1]
end

context StaffTicketEvent ini inv theTicketIsVisible:
  self.staffMember.isAdministrator or
  (self.staffMember.staffGroup.departmentsAccess->including(self.staffMember.department))
  ->includes(self.ticket.assignedDepartment)
context StaffTicketEvent ini inv staffMemberIsLoggedIn:
  self.staffMember.isLoggedin
```

The events `PostTicketInternalNote, PostTicketReply, ReleaseTicket, EditTicket, AssignTicket, MarkTicketOverdue, ChangeTicketPriority` are specified as a subclass of `StaffTicketEvent`. The associations with the ticket and the staff member, and the initial integrity constraints that check that the ticket is visible and the staff member is logged in are deleted from each particular event.

After running all test cases, we are sure that after refactoring, the verdict of all test cases remains `Pass`. 
Errors and failures that drive the conceptual modeling

<table>
<thead>
<tr>
<th>A basic type involved in a test case does not exist in the CSUD</th>
<th>A derived type involved in a test case does not exist in the CSUD</th>
<th>An event type involved in a test case does not exist in the CSUD</th>
</tr>
</thead>
<tbody>
<tr>
<td>The basic type is relevant and it is added to the CSUD</td>
<td>The derived type is relevant and it is added to the CSUD</td>
<td>The event type is relevant and it is added to the CSUD</td>
</tr>
</tbody>
</table>

Inconsistent state before the occurrence of an event

<table>
<thead>
<tr>
<th>Inconsistent state after the occurrence of an event</th>
<th>The postcondition of an event is not satisfied.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Some static constraint is invalid and it is modified.</td>
<td>The event postcondition/method is incorrect and it is modified.</td>
</tr>
<tr>
<td>Some initial integrity constraint is invalid and it is modified.</td>
<td>Some constraint is invalid and it is modified.</td>
</tr>
<tr>
<td>The event occurrence is not correct.</td>
<td>The method is not correct and it is modified.</td>
</tr>
<tr>
<td>The postcondition is not correct.</td>
<td>The postcondition is not correct and it is modified.</td>
</tr>
</tbody>
</table>

An assertion about the IB state fails or contains an error

<table>
<thead>
<tr>
<th>Assert non-occurrence fails</th>
<th>Semantic error in an expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>The effect of an event type is not correct</td>
<td>The expression is correct.</td>
</tr>
<tr>
<td>A derivation rule is incorrect</td>
<td>The CSUD is changed.</td>
</tr>
<tr>
<td>A precondition is added/updated</td>
<td></td>
</tr>
<tr>
<td>The expression is corrected</td>
<td></td>
</tr>
</tbody>
</table>

Assert consistency fails

<table>
<thead>
<tr>
<th>A static constraint needs to be changed</th>
</tr>
</thead>
</table>

Iteration 70

Iteration objective

S55: TransferTicket_staffAlertsEnabled

Current test case

test S55{
  load testConfiguration4;
  load created_tickets;

  li := new LogIn(username:='mary', password:='yyy');
  assert occurrence li;

  tt:=new TransferDepartment(staffMember:=generalConsultant, ticket:=ticket1,
   department:=deptTechnical,
   note:= 'This is a technical question');
  assert occurrence tt;

  assert equals ticket1.assignedDepartment deptTechnical;
}

TDCM application: Summary of changes performed in the schema

- [TicketsManagementAndTracking.cstl] <line 1963> TransferDepartment is not defined in the CSUT as a class or an association
DEVELOPMENT OF THE CONCEPTUAL SCHEMA OF THE osTICKET SYSTEM BY APPLYING TDCM

event TransferDepartment<StaffTicketEvent
attributes
    note:String
operations
effect()
end

association transferDepartment_department between
    TransferDepartment[*]
    Department[1]
end

context TransferDepartment::effect()
post:
    self.ticket.assignedDepartment=self.department and
    self.ticket.internalNote->one(int | int.oclIsNew() and
    int.datetime=System.allInstances()->any(true).currentDateTime
    and int.subject='Department transfer'
    and int.text=self.note
    and self.ticket.lastMessageDatetime=System.allInstances()->any(true).currentDateTime)

Time spent

<table>
<thead>
<tr>
<th>TIME TO WRITE TEST CASES (IN MINUTES)</th>
<th>4,5</th>
</tr>
</thead>
<tbody>
<tr>
<td>TIME TO COMPLETE THE ITERATION (IN MINUTES)</td>
<td>6,5</td>
</tr>
</tbody>
</table>

Errors and failures that drive the conceptual modeling

<table>
<thead>
<tr>
<th>A basic type involved in a test case does not exist in the CSUD</th>
<th>A derived type involved in a test case does not exist in the CSUD</th>
<th>An event type involved in a test case does not exist in the CSUD</th>
</tr>
</thead>
<tbody>
<tr>
<td>The basic type is relevant and it is added to the CSUD</td>
<td>The derived type is relevant and it is added to the CSUD</td>
<td>The event type is relevant and it is added to the CSUD</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

Inconsistent state before the occurrence of an event

<table>
<thead>
<tr>
<th>Some static constraint is invalid and it is modified.</th>
<th>Some initial integrity constraint is invalid and it is modified.</th>
<th>The event postcondition/method is incorrect and it is modified.</th>
</tr>
</thead>
</table>

The postcondition of an event is not satisfied.

<table>
<thead>
<tr>
<th>An assertion about the IB state fails or contains an error</th>
<th>Assert non-occurrence fails</th>
<th>Semantic error in an expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>The effect of an event type is not correct</td>
<td>A derivation rule is incorrect</td>
<td>The expression is corrected</td>
</tr>
<tr>
<td>A precondition is added/updated</td>
<td>The CSUD is changed</td>
<td></td>
</tr>
</tbody>
</table>

Assert consistency fails

A static constraint needs to be changed
DEVELOPMENT OF THE CONCEPTUAL SCHEMA OF THE osTICKET SYSTEM BY APPLYING TDCM

Iteration 71

Iteration objective

S56: TransferTicket_ticketIsNotVisible

Current test case

test S56{
    load testConfiguration4;
    load created_tickets;
    technicalActive.staffGroup:=maximumPrivilegesGroup;
    maximumPrivilegesGroup.departmentsAccess:=Set{dptTechnical};
    li := new LogIn(username:='martin', password:='ttt');
    assert occurrence li;

    tt:=new TransferDepartment(staffMember:=technicalActive, ticket:=ticket2, department:=dptTechnical, note:='This is a technical question');
    assert non-occurrence tt;
}

TDCM application: Summary of changes performed in the schema

The CSUT has not been changed.

Time spent

| TIME TO WRITE TEST CASES (IN MINUTES) | 1,5 |
| TIME TO COMPLETE THE ITERATION (IN MINUTES) | 0,5 |

Iteration 72

Iteration objective

S57: TransferTicket_SameDepartment

Current test case

test S57{
    load testConfiguration4;
    load created_tickets;

    li := new LogIn(username:='mary', password:='yyy');
    assert occurrence li;

tt:=new TransferDepartment(staffMember:=generalConsultant, ticket:=ticket2, department:=deptGeneral, note:="This is a technical question");
assert non-occurrence tt;
}

TDCM application: Summary of changes performed in the schema

- Added
  - Preconditions of the domain event pin:PostTicketInternalNote are satisfied and consequently, the event can occur

context TransferDepartment ini inv departmentsDifferent:
  self.department <> self.ticket.assignedDepartment

<table>
<thead>
<tr>
<th>Errors and failures that drive the conceptual modeling</th>
</tr>
</thead>
<tbody>
<tr>
<td>A basic type involved in a test case does not exist in the CSUD</td>
</tr>
<tr>
<td>Inconsistent state before the occurrence of an event</td>
</tr>
<tr>
<td>Some static constraint is invalid and it is modified.</td>
</tr>
<tr>
<td>An assertion about the IB state fails or contains an error</td>
</tr>
<tr>
<td>The effect of an event type is not correct</td>
</tr>
<tr>
<td>Assert consistency fails</td>
</tr>
<tr>
<td>A static constraint needs to be changed</td>
</tr>
</tbody>
</table>

Iteration 73

Iteration objective

S58: TransferTicket_NotAllowedToTransfer
Current test case

test S58{
load testConfiguration3;
load created_tickets;

li := new LogIn(username:='martin', password:='ttt');
assert occurrence li;

tt:=new TransferDepartment(staffMember:=generalConsultant, ticket:=ticket2,
department:=deptGeneral,
note:='This is a technical question');
assert non-occurrence tt;
}

TDCM application: Summary of changes performed in the schema

- Preconditions of the domain event tt:TransferDepartment are satisfied and consequently, the event can occur

  context TransferDepartment ini inv staffMembersAllowedToTransfer:
  self.staffMember.staffGroup.canTransferTickets

Time spent

| TIME TO WRITE TEST CASES (IN MINUTES) | 1.5 |
| TIME TO COMPLETE THE ITERATION (IN MINUTES) | 3 |

Errors and failures that drive the conceptual modeling

<table>
<thead>
<tr>
<th>A basic type involved in a test case does not exist in the CSUD</th>
<th>A derived type involved in a test case does not exist in the CSUD</th>
<th>An event type involved in a test case does not exist in the CSUD</th>
</tr>
</thead>
<tbody>
<tr>
<td>The basic type is relevant and it is added to the CSUD</td>
<td>The derived type is relevant and it is added to the CSUD</td>
<td>The event type is relevant and it is added to the CSUD</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Inconsistent state before the occurrence of an event</th>
<th>Inconsistent state after the occurrence of an event</th>
<th>The postcondition of an event is not satisfied.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Some static constraint is invalid and it is modified.</td>
<td>Some initial integrity constraint is invalid and it is modified.</td>
<td>The event postcondition/method is incorrect and it is modified.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>An assertion about the IB state fails or contains an error</th>
<th>Assert non-occurrence fails</th>
<th>Semantic error in an expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>The effect of an event type is not correct</td>
<td>A derivation rule is incorrect</td>
<td>A precondition is added/updated</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Assert consistency fails</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>A static constraint needs to be changed</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Iteration 74

Iteration objective

S59: TransferTicket_NotLoggedIn

Current test case

test S59{
    load testConfiguration3;
    load created_tickets;

    tt:=new TransferDepartment(staffMember:=generalConsultant, ticket:=ticket1,
                                department:=dptGeneral,
                                note:='This is a technical question');
    assert non-occurrence tt;
}

TDCM application: Summary of changes performed in the schema

The CSUD has not been changed

Time spent

<table>
<thead>
<tr>
<th>TIME TO WRITE TEST CASES (IN MINUTES)</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>TIME TO COMPLETE THE ITERATION (IN MINUTES)</td>
<td>0,5</td>
</tr>
</tbody>
</table>

Iteration 75

Iteration objective

S60: Close ticket

Current test case

test S60{
    load testConfiguration3;
    load created_tickets;

    li := new LogIn(username:='mary', password:='yyy');
    assert occurrence li;

    ct:=new CloseTicket(staffMember:=generalConsultant, ticket:=ticket1);
    assert occurrence ct;
    assert equals ticket1.ticketStatus #Closed;

    assert true [ticket1.internalNote->one(m|m.datetime=sys.currentDateTime and
     m.subject='Ticket closed'and m.text='Ticket closed without response' and}
m.author='Mary');

c1:=new CloseTicket(staffMember:=generalConsultant, ticket:=ticket1);
assert non-occurrence ct;
}

**TDCM application: Summary of changes performed in the schema**

- **Added**
  - [TicketsManagementAndTracking.cstl] <line 2044> CloseTicket is not defined in the CSUT as a class or an association
    
    ```
    event CloseTicket<StaffTicketEvent
    operations
    effect()
    end
    
    context CloseTicket::effect()
    post:
    self.ticket.ticketStatus=#Closed and
    self.ticket.internalNote->one(int | int.oclIsNew() and int.datetime=System.allInstances()->any(true).currentDateTime and int.subject='Ticket closed' and int.text='Ticket closed without response' and int.author=self.staffMember.firstName
    and self.ticket.lastMessageDateTime=System.allInstances()->any(true).currentDateTime)
    ```

- Preconditions of the domain event ct:CloseTicket are satisfied and consequently, the event can occur
  
  ```
  context CloseTicket ini inv ticketIsNotClosed:
  not (self.ticket.ticketStatus=#Closed)
  ```

**Time spent**

<table>
<thead>
<tr>
<th>TIME TO WRITE TEST CASES (IN MINUTES)</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>TIME TO COMPLETE THE ITERATION (IN MINUTES)</td>
<td>14</td>
</tr>
</tbody>
</table>

**Errors and failures that drive the conceptual modeling**

<table>
<thead>
<tr>
<th>A basic type involved in a test case does not exist in the CSUD</th>
<th>A derived type involved in a test case does not exist in the CSUD</th>
<th>An event type involved in a test case does not exist in the CSUD</th>
</tr>
</thead>
<tbody>
<tr>
<td>The basic type is relevant and it is added to the CSUD</td>
<td>The derived type is relevant and it is added to the CSUD</td>
<td>The event type is relevant and it is added to the CSUD</td>
</tr>
<tr>
<td>Inconsistent state before the occurrence of an event</td>
<td>Inconsistent state after the occurrence of an event</td>
<td>The postcondition of an event is not satisfied</td>
</tr>
<tr>
<td>Some static constraint is invalid and it is modified</td>
<td>Some initial integrity constraint is invalid and it is modified</td>
<td>The event postcondition/method is incorrect and it is modified</td>
</tr>
<tr>
<td>Some constraint is invalid and it is modified</td>
<td>The event postcondition/method is incorrect and it is modified</td>
<td>Some constraint is invalid and it is modified</td>
</tr>
<tr>
<td>The method is not correct and it is modified</td>
<td>The postcondition is not satisfied and it is modified</td>
<td>The postcondition is not correct and it is modified</td>
</tr>
<tr>
<td>An assertion about the IB state fails or contains an error</td>
<td>Assert non-occurrence fails</td>
<td>Semantic error in an expression</td>
</tr>
<tr>
<td>The effect of an event type is not correct</td>
<td>A derivation rule is incorrect</td>
<td>The preconditions added/updated</td>
</tr>
<tr>
<td>The expression is corrected</td>
<td>The CSUD is changed</td>
<td></td>
</tr>
</tbody>
</table>
Iteration 76

Iteration objective

S61: CloseTicket_ticketIsNotVisibleOrNotAllowed

Current test case

```plaintext
test S61{
load testConfiguration3;
load created_tickets;
li := new LogIn(username := 'mary', password := 'yyy');
assert occurrence li;
generalConsultant.staffGroup.departmentsAccess := Set{dptGeneral};
ct := new CloseTicket(staffMember := generalConsultant, ticket := ticket3);
assert non-occurrence ct;
generalConsultant.staffGroup.departmentsAccess := Set{dptGeneral, dptTechnical};
generalConsultant.staffGroup.canCloseTickets := false;
ct := new CloseTicket(staffMember := generalConsultant, ticket := ticket3);
assert non-occurrence ct;
}
```

TDCM application: Summary of changes performed in the schema

- Added
  - Preconditions of the domain event ct:CloseTicket are satisfied and consequently, the event can occur

    ```plaintext
    context CloseTicket ini inv staffMembersAllowedToClose:
    self.staffMember.staffGroup.canCloseTickets
    ```

Time spent

| TIME TO WRITE TEST CASES (IN MINUTES) | 8 |
| TIME TO COMPLETE THE ITERATION (IN MINUTES) | 5 |

Errors and failures that drive the conceptual modeling

<table>
<thead>
<tr>
<th>A basic type involved in a test case does not exist in the CSUD</th>
<th>A derived type involved in a test case does not exist in the CSUD</th>
<th>An event type involved in a test case does not exist in the CSUD</th>
</tr>
</thead>
<tbody>
<tr>
<td>The basic type is relevant and it is added to the CSUD</td>
<td>The derived type is relevant and it is added to the CSUD</td>
<td>The event type is relevant and it is added to the CSUD</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Inconsistent state before the occurrence of an event</th>
<th>Inconsistent state after the occurrence of an event</th>
<th>The postcondition of an event is not satisfied</th>
</tr>
</thead>
<tbody>
<tr>
<td>Some static constraint is invalid and it is modified.</td>
<td>Some initial integrity constraint is invalid and it is modified.</td>
<td>The event precondition/method is incorrect and it is modified.</td>
</tr>
<tr>
<td>Some constraint is invalid and it is modified.</td>
<td>The method is not correct and it is modified.</td>
<td>The postcondition is not correct and it is modified.</td>
</tr>
</tbody>
</table>
### Iteration 77

#### Iteration objective

S62: CloseTicket_NotLoggedIn

#### Current test case

```plaintext
test S62{
    load testConfiguration3;
    load created_tickets;
    ct:=new CloseTicket(staffMember:=generalConsultant, ticket:=ticket3);
    assert non-occurrence ct;
}
```

#### TDCM application: Summary of changes performed in the schema

The CSUD has not been changed

#### Time spent

| TIME TO WRITE TEST CASES (IN MINUTES) | 1,5 |
| TIME TO COMPLETE THE ITERATION (IN MINUTES) | 0,5 |

### Iteration 78

#### Iteration objective

S63: CloseTicketWithResponse_alertsAndAutoresponsesEnabled
Current test case

test S63{
  load testConfiguration1;
  load created_tickets;
  helpTopicInstallation.autoresponse:=#Enabled;

  li := new LogIn(username:='mary', password:='yyy');
  assert occurrence li;

  ct := new CloseTicketWithResponse(staffMember:=generalConsultant, ticket:=ticket1,
    response:='Ticket solved');
  assert occurrence ct;
  assert equals ticket1.ticketStatus #Closed;

  assert true [ticket1.internalNote->one(m|m.datetime=sys.currentDateTime and
    m.subject='Ticket closed' and
    m.text='Ticket closed on reply' and
    m.author='Mary')];

  assert true [EMail.allInstances()->exists(e|e.emailKind.oclIsTypeOf(TicketResponseNotice) and
    e.fromAddress='technical_at_support.com' and
    e.toAddress='mary_at_marnes.mar' and
    e.ticketNumber=1)];

  assert true [ticket1.ticketThreadMessage->one(m|m.datetime=sys.currentDateTime and
    m.text='Ticket solved' and
    m.author='Mary')];
}

TDCM application: Summary of changes performed in the schema

- [TicketsManagementAndTracking.cstl] <line 2099> CloseTicketWithResponse is not defined in the CSUT as a class or an association

  event CloseTicketWithResponse<StaffTicketEvent
  attributes
    response:String
  operations
    effect()
  end

  context CloseTicketWithResponse::effect()
  post:
    let sendAutoresponse:Boolean:=
      if CustomerAutoresponsesSettings.allInstances()->any(true).autorespondWhenNewMessageAppendedToTicket
        if self.ticket.helpTopic->notEmpty then
          if self.ticket.helpTopic.autoresponse=#Enabled then true
          else false
        endif
      else false
      endif
    else
      self.ticket.assignedDepartment.newAddedMessageIsNotified
    endif
  else false
  endif
  in
  self.ticket.ticketStatus=#Closed
  self.ticket.internalNote->one(int|int.oclIsNew() and
    int.datetime=System.allInstances()->any(true).currentDateTime
    and int.subject='Ticket closed' and
    int.author='Mary'
  )
and int.text='Ticket closed on reply'
    and int.author=self.staffMember.firstName
    and self.ticket.lastMessageDateatime=System.allInstances()->any(true).currentDateTime
    and self.ticket.ticketThreadMessage->one(tdm | tdm.oclIsNew() and tdm.datetime=System.allInstances()->any(true).currentDateTime
    and tdm.text=self.response
    and tdm.author=self.staffMember.firstName
    and self.ticket.lastMessageDateatime=System.allInstances()->any(true).currentDateTime
    and self.ticket.lastRespondent=self.staffMember

and (sendAutoresponse implies
    EMail.allInstances()->exists(e|e.fromAddress=self.ticket.assignedDepartment.autoresponseEmail.address and e.toAddress=self.ticket.email and e.ticketNumber=self.ticket.number))

- Preconditions of the domain event ct:CloseTicket are satisfied and consequently, the event can occur

context CloseTicketWithResponse ini inv ticketIsNotClosed:
    not (self.ticket.ticketStatus=#Closed)

Time spent

| TIME TO WRITE TEST CASES (IN MINUTES) | 9 |
| TIME TO COMPLETE THE ITERATION (IN MINUTES) | 34 |

Errors and failures that drive the conceptual modeling

<table>
<thead>
<tr>
<th>A basic type involved in a test case does not exist in the CSUD</th>
<th>A derived type involved in a test case does not exist in the CSUD</th>
<th>An event type involved in a test case does not exist in the CSUD</th>
</tr>
</thead>
<tbody>
<tr>
<td>The basic type is relevant and it is added to the CSUD</td>
<td>The derived type is relevant and it is added to the CSUD</td>
<td>The event type is relevant and it is added to the CSUD</td>
</tr>
<tr>
<td>Inconsistent state before the occurrence of an event</td>
<td>Inconsistent state after the occurrence of an event</td>
<td>The postcondition of an event is not satisfied.</td>
</tr>
<tr>
<td>Some static constraint is invalid and it is modified.</td>
<td>Some initial integrity constraint is invalid and it is modified.</td>
<td>Some constraint is invalid and it is modified.</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>An assertion about the IB state fails or contains an error</td>
<td>Assert non-occurrence fails</td>
<td>Semantic error in an expression</td>
</tr>
<tr>
<td>The effect of an event type is not correct</td>
<td>A derivation rule is incorrect</td>
<td>The expression is corrected</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Assert consistency fails</td>
<td></td>
<td></td>
</tr>
<tr>
<td>A static constraint needs to be changed</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Iteration 79

Iteration objective

S64: CloseTicketWithResponse_alertsAndAutoresponsesDisabled

Current test case

test S64{
load testConfiguration4;
load created_tickets;

li := new LogIn(username:='mary', password:='yyy');
assert occurrence li;

tc:=new CloseTicketWithResponse(staffMember:=generalConsultant, ticket:=ticket1,
response:='Ticket solved');
assert occurrence ct;

assert equals ticket1.ticketStatus #Closed;
assert true [ticket1.internalNote->one(m|m.datetime=sys.currentDateTime and
m.subject='Ticket closed'and
m.text='Ticket closed on reply' and
m.author='Mary')];

assert false [EMail.allInstances()->exists(e|e.emailKind.oclIsTypeOf(TicketResponseNotice) and
 e.fromAddress='technical_at_support.com' and
 e.toAddress='mary_at_marnes.mar' and
 e.ticketNumber=1)];

assert true [ticket1.ticketThreadMessage->one(m|m.datetime=sys.currentDateTime and
m.text='Ticket solved' and
m.author='Mary')];
}

TDCM application: Summary of changes performed in the schema

The CSUT has not been changed.

Time spent

| TIME TO WRITE TEST CASES (IN MINUTES) | 2 |
| TIME TO COMPLETE THE ITERATION (IN MINUTES) | 0.5 |
Iteration 80

Iteration objective

S65: CloseTicketWithResponse_ticketIsNotVisibleOrNotAllowed

Current test case

test S65{
  load testConfiguration3;
  load created_tickets;
  li := new LogIn(username:='mary', password:='yyy');
  assert occurrence li;
  generalConsultant.staffGroup.departmentsAccess:=Set{dptGeneral};
  ct:=new CloseTicketWithResponse(staffMember:=generalConsultant, ticket:=ticket3,
    response:='Ticket solved');
  assert non-occurrence ct;
  generalConsultant.staffGroup.departmentsAccess:=Set{dptGeneral,dptTechnical};
  generalConsultant.staffGroup.canCloseTickets:=false;
  ct:=new CloseTicketWithResponse(staffMember:=generalConsultant, ticket:=ticket3,
    response:='Ticket solved');
  assert non-occurrence ct;
}

TDCM application: Summary of changes performed in the schema

- Added
- Updated

- Preconditions of the domain event ct:CloseTicketWithResponse are satisfied and consequently, the event can occur

  context CloseTicketWithResponse
    init inv staffMemberIsAllowedToClose:
      self.staffMember.staffGroup.canCloseTickets

Time spent

<table>
<thead>
<tr>
<th>TIME TO WRITE TEST CASES (IN MINUTES)</th>
<th>2,5</th>
</tr>
</thead>
<tbody>
<tr>
<td>TIME TO COMPLETE THE ITERATION (IN MINUTES)</td>
<td>3</td>
</tr>
</tbody>
</table>

Errors and failures that drive the conceptual modeling

<table>
<thead>
<tr>
<th>A basic type involved in a test case does not exist in the CSUD</th>
<th>A derived type involved in a test case does not exist in the CSUD</th>
<th>An event type involved in a test case does not exist in the CSUD</th>
</tr>
</thead>
<tbody>
<tr>
<td>The basic type is relevant and it is added to the CSUD</td>
<td>The derived type is relevant and it is added to the CSUD</td>
<td>The event type is relevant and it is added to the CSUD</td>
</tr>
</tbody>
</table>
Inconsistent state before the occurrence of an event | Inconsistent state after the occurrence of an event | The postcondition of an event is not satisfied.
--- | --- | ---
Some static constraint is invalid and it is modified. | Some initial integrity constraint is invalid and it is modified. | Some constraint is invalid and it is modified.

The event postcondition/method is incorrect and it is modified. | The method is not correct and it is modified. | The postcondition is not correct and it is modified.

---

An assertion about the IB state fails or contains an error | Assert non-occurrence fails | Semantic error in an expression
--- | --- | ---
The effect of an event type is not correct | A derivation rule is incorrect | A precondition is added/updated
A precondition is added/updated | The expression is corrected | The CSUD is changed

| Assert consistency fails | 1 |
A static constraint needs to be changed

---

**Iteration 81**

**Iteration objective**

S66: CloseTicketWithReply_notLoggedIn

**Current test case**

```java
test S66{
load testConfiguration3;
load created_tickets;

ct:=new CloseTicketWithResponse(staffMember:=generalConsultant, ticket:=ticket1, response:='Ticket solved');
assert non-occurrence ct;
}
```

**TDCM application: Summary of changes performed in the schema**

The CSUD has not been changed.

**Time spent**

| TIME TO WRITE TEST CASES (IN MINUTES) | 0,5 |
| TIME TO COMPLETE THE ITERATION (IN MINUTES) | 0,5 |
Iteration 82

Iteration objective

S67: ReopenTicket

Current test case

test S67 {
  load testConfiguration3;
  load created_tickets;
  ticket1.ticketStatus := #Closed;
  li := new Login(username:='mary', password:='yyy');
  assert occurrence li;
  rot := new ReopenTicket(staffMember:=generalConsultant, ticket:=ticket1);
  assert occurrence rot;
  assert equals ticket1.ticketStatus #Open;
  assert true (ticket1.internalNote->one(m|m.datetime=sys.currentDateTime and
      m.subject='Ticket reopened' and
      m.text='Ticket reopened without comments' and
      m.author='Mary'));
  rot := new ReopenTicket(staffMember:=generalConsultant, ticket:=ticket1);
  assert non-occurrence rot;
}

TDCM application: Summary of changes performed in the schema

- Added
- Updated

- [TicketsManagementAndTracking.cstl] <line 2201> ReopenTicket is not defined in the CSUT as a class or an association

  event ReopenTicket<StaffTicketEvent
  operations
effect()
end

class ReopenTicket::effect()
post:
  self.ticket.ticketStatus := #Open and
  self.ticket.internalNote->one(int | int.oclIsNew() and
      int.datetime=System.allInstances()->any(true).currentDateTime and
      int.subject='Ticket reopened' and
      int.text='Ticket reopened without comments' and
      int.author=self.staffMember.firstName)
and self.ticket.lastMessageDateTime=System.allInstances()->any(true).currentDateTime

- Preconditions of the domain event rot:ReopenTicket are satisfied and consequently, the event can occur

  context ReopenTicket ini inv ticketIsClosed:
  self.ticket.ticketStatus := #Closed
Time spent

| TIME TO WRITE TEST CASES (IN MINUTES) | 5.5 |
| TIME TO COMPLETE THE ITERATION (IN MINUTES) | 13 |

Errors and failures that drive the conceptual modeling

<table>
<thead>
<tr>
<th>A basic type involved in a test case does not exist in the CSUD</th>
<th>A derived type involved in a test case does not exist in the CSUD</th>
<th>An event type involved in a test case does not exist in the CSUD</th>
</tr>
</thead>
<tbody>
<tr>
<td>The basic type is relevant and it is added to the CSUD</td>
<td>The derived type is relevant and it is added to the CSUD</td>
<td>The event type is relevant and it is added to the CSUD</td>
</tr>
<tr>
<td>Inconsistent state before the occurrence of an event</td>
<td>Inconsistent state after the occurrence of an event</td>
<td>The postcondition of an event is not satisfied.</td>
</tr>
<tr>
<td>Some static constraint is invalid and it is modified</td>
<td>Some initial integrity constraint is invalid and it is modified</td>
<td>The method is not correct and it is modified.</td>
</tr>
<tr>
<td>The event postcondition/method is incorrect and it is modified</td>
<td>Some constraint is invalid and it is modified.</td>
<td>The postcondition is not correct and it is modified.</td>
</tr>
<tr>
<td>The method is not correct and it is modified.</td>
<td>Some constraint is invalid and it is modified.</td>
<td>The postcondition is not correct and it is modified.</td>
</tr>
<tr>
<td>The effect of an event type is not correct</td>
<td>A precondition is added/updated</td>
<td>The expression is corrected</td>
</tr>
<tr>
<td>Assert non-occurrence fails</td>
<td>A derivation rule is incorrect</td>
<td>The CSUD is changed</td>
</tr>
<tr>
<td>Semantic error in an expression</td>
<td>The expression is corrected</td>
<td></td>
</tr>
<tr>
<td>Assert consistency fails</td>
<td></td>
<td></td>
</tr>
<tr>
<td>A static constraint needs to be changed</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Iteration 83

Iteration objective

S68: ReopenTicket_ticketIsNotVisible

Current test case

```plaintext
test S68{
  load testConfiguration3;
  load created_tickets;
  ticket2.ticketStatus:=#Closed;
  li := new LogIn(username:='martin', password:='ttt');
  assert occurrence li;
  rot:=new ReopenTicket(staffMember:=technicalActive, ticket:=ticket2);
  assert non-occurrence rot;
}
```
TDCM application: Summary of changes performed in the schema

The conceptual schema has not been changed

Time spent

<table>
<thead>
<tr>
<th>Time to write test cases (in minutes)</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time to complete the iteration (in minutes)</td>
<td>0.5</td>
</tr>
</tbody>
</table>

Iteration 84

Iteration objective

S69: ReopenTicket_NotLoggedIn

Current test case

```
test S69{
    load testConfiguration3;
    load created_tickets;
    ticket1.ticketStatus:=#Closed;
    rot:=new ReopenTicket(staffMember:=generalConsultant, ticket:=ticket1);
    assert non-occurrence rot;
}
```

TDCM application: Summary of changes performed in the schema

The conceptual schema has not been changed

Time spent

<table>
<thead>
<tr>
<th>Time to write test cases (in minutes)</th>
<th>0.5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time to complete the iteration (in minutes)</td>
<td>0.5</td>
</tr>
</tbody>
</table>

Iteration 85

Iteration objective

S70: ReopenTicketWithReply_alertsAndAutoresponsesEnabled
Current test case

test S70{
   load testConfiguration1;
   helpTopicInstallation.autoresponse:=#Enabled;
   load created_tickets;
   ticket1.ticketStatus:=#Closed;
   li := new LogIn(username:='mary', password:='yyy');
   assert occurrence li;
   rot:=new ReopenTicketWithResponse(staffMember:=generalConsultant, ticket:=ticket1,
                                    response:='The customer is not satisfied');
   assert occurrence rot;
   assert equals ticket1.ticketStatus #Open;
   assert true [ticket1.internalNote->one(m|m.datetime=sys.currentDateTime and
                                                m.subject='Ticket status changed to open and
                                                m.text='A staff member reopened the ticket on reply' and
                                                m.author='Mary')];
   assert true [EMail.allInstances()->exists(e|e.emailKind.oclIsTypeOf(TicketResponseNotice) and
                                                e.fromAddress='technical_at_support.com' and
                                                e.toAddress='mary_at_marnes.mar' and
                                                e.ticketNumber=1)];
   assert true [ticket1.ticketThreadMessage->one(m|m.datetime=sys.currentDateTime and
                                                m.text='The customer is not satisfied' and
                                                m.author='Mary')];
   rot:=new ReopenTicketWithResponse(staffMember:=generalConsultant, ticket:=ticket1,
                                    response:='The customer is not satisfied');
   assert non-occurrence rot;
}

TDCM application: Summary of changes performed in the schema

- [TicketsManagementAndTracking.cstl] <line 2249> ReopenTicketWithResponse is not defined in the CSUT as a class or an association

    event ReopenTicketWithResponse<StaffTicketEvent
    attributes
    response:String
    operations
    effect() end

context ReopenTicketWithResponse::effect()
post:
let sendAutoresponse:Boolean= if CustomerAutoresponsesSettings.allInstances()->any(true).autorespondWhenNewMessageAppendedToTicket then
   if self.ticket.helpTopic->notEmpty then
      if self.ticket.helpTopic.autoresponse=#Enabled then true
      else false
   endif
   else
      self.ticket.assignedDepartment.newAddedMessageIsNotified
   endif
   else false
endif
in
self.ticket.ticketStatus=#Open and
self.ticket.internalNote->one(int | int.oclIsNew() and int.datetime=System.allInstances()->any(true).currentDateTime and int.subject='Ticket status changed to open' and int.text='A staff member reopened the ticket on reply' and int.author=self.staffMember.firstName and self.ticket.lastMessageDateTime=System.allInstances()->any(true).currentDateTime) and self.ticket.ticketThreadMessage->one(tdm | tdm.oclIsNew() and tdm.datetime=System.allInstances()->any(true).currentDateTime and tdm.text=self.response and tdm.author=self.staffMember.firstName and self.ticket.lastMessageDateTime=System.allInstances()->any(true).currentDateTime) and self.ticket.lastRespondent=self.staffMember and (sendAutoresponse implies EMail.allInstances()->exists(e|e.fromAddress=self.ticket.assignedDepartment.autoresponseEmail.address and e.toAddress=self.ticket.email and e.ticketNumber=self.ticket.number))

context ReopenTicketWithResponse ini inv ticketIsClosed:
  self.ticket.ticketStatus=#Closed

Preconditions of the domain event rot:ReopenTicketWithResponse are satisfied and consequently, the event can occur

context ReopenTicketWithResponse ini inv ticketsIsClosed:
  self.ticket.ticketStatus=#Closed

Time spent

<table>
<thead>
<tr>
<th>TIME TO WRITE TEST CASES (IN MINUTES)</th>
<th>8,5</th>
</tr>
</thead>
<tbody>
<tr>
<td>TIME TO COMPLETE THE ITERATION (IN MINUTES)</td>
<td>12</td>
</tr>
</tbody>
</table>

Errors and failures that drive the conceptual modeling

<table>
<thead>
<tr>
<th>A basic type involved in a test case does not exist in the CSUD</th>
<th>A derived type involved in a test case does not exist in the CSUD</th>
<th>An event type involved in a test case does not exist in the CSUD</th>
</tr>
</thead>
<tbody>
<tr>
<td>The basic type is relevant and it is added to the CSUD</td>
<td>The derived type is relevant and it is added to the CSUD</td>
<td>The event type is relevant and it is added to the CSUD</td>
</tr>
<tr>
<td>□ Inconsistent state before the occurrence of an event</td>
<td>□ Inconsistent state after the occurrence of an event</td>
<td>□ The postcondition of an event is not satisfied.</td>
</tr>
<tr>
<td>Some static constraint is invalid and it is modified.</td>
<td>Some initial integrity constraint is invalid and it is modified.</td>
<td>The event postcondition/method is incorrect and it is modified.</td>
</tr>
<tr>
<td>□ An assertion about the IB state fails or contains an error</td>
<td>□ Assert non-occurrence fails</td>
<td>□ Semantic error in an expression</td>
</tr>
<tr>
<td>The effect of an event type is not correct</td>
<td>A derivation rule is incorrect</td>
<td>The expression is corrected</td>
</tr>
<tr>
<td>□ Assert consistency fails</td>
<td>A precondition is added/updated</td>
<td>The CSUD is changed</td>
</tr>
<tr>
<td>A static constraint needs to be changed</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Iteration 86

Iteration objective

S71: ReopenTicketWithReply_alertsAndAutoresponsesDisabled

Current test case

test S71{
    load testConfiguration4;
    load created_tickets;
    ticket1.ticketStatus := #Closed;
    li := new Login(username:='mary', password:='yyy');
    assert occurrence li;
    rot := new ReopenTicketWithResponse(staffMember:=generalConsultant, ticket:=ticket1,
                                            response:='The customer is not satisfied');
    assert occurrence rot;
    assert equals ticket1.ticketStatus #Open;
    assert true [ticket1.internalNote->one(m|m.datetime=sys.currentDateTime and
                                           m.subject='Ticket status changed to open' and
                                           m.text='A staff member reopened the ticket on reply' and
                                           m.author='Mary')];

    assert false [EMail.allInstances()->exists(e|e.emailKind.oclIsTypeOf(TicketResponseNotice) and
                                           e.fromAddress='technical_at_support.com' and
                                           e.toAddress='mary_at_marnes.mar' and
                                           e.ticketNumber=1)];

    assert true [ticket1.ticketThreadMessage->one(m|m.datetime=sys.currentDateTime and
                                                   m.text='The customer is not satisfied' and
                                                   m.author='Mary')];
}

TDCM application: Summary of changes performed in the schema

The CSUD has not been changed

Time spent

| Time to write test cases (in minutes) | 2 |
| Time to complete the iteration (in minutes) | 0,5 |
Iteration 87

Iteration objective

S72: ReopenTicketWithReply_ticketIsNotVisible

Current test case

test S72{
    load testConfiguration3;
    load created_tickets;
    ticket1.ticketStatus:=#Closed;
    li := new LogIn(username:='martin', password:='ttt');
    assert occurrence li;
    rot:=new ReopenTicketWithResponse(staffMember:=technicalActive, ticket:=ticket2,
        response:='The customer is not satisfied');
    assert non-occurrence rot;
}

TDCM application: Summary of changes performed in the schema

The CSUD has not been changed

Time spent

<table>
<thead>
<tr>
<th>TIME TO WRITE TEST CASES (IN MINUTES)</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>TIME TO COMPLETE THE ITERATION (IN MINUTES)</td>
<td>0,5</td>
</tr>
</tbody>
</table>

Iteration 88

Iteration objective

S73: ReopenTicketWithReply_NotLoggedIn

Current test case

test S73{
    load testConfiguration1;
    helpTopicInstallation.autoresponse:=#Enabled;
    load created_tickets;
    ticket1.ticketStatus:=#Closed;
    rot:=new ReopenTicketWithResponse(staffMember:=generalConsultant, ticket:=ticket1,
        response:='The customer is not satisfied');
    assert non-occurrence rot;
}
TDCM application: Summary of changes performed in the schema

The CSUD has not been changed

Time spent

| TIME TO WRITE TEST CASES (IN MINUTES) | 1,5 |
| TIME TO COMPLETE THE ITERATION (IN MINUTES) | 0,5 |

Iteration 89

Iteration objective

S74: BAnTicketCloseEmail

Current test case

```
test S74{
    load testConfiguration3;
    load created_tickets;
    li := new Login(username:=’mary’, password:=’yyy’);
    assert occurrence li;
    cbt:=new BanEmailAndCloseTicket(staffMember:=generalConsultant, ticket:=ticket1);
    assert occurrence cbt;  
    assert equals ticket1.ticketStatus #Closed;
    assert true EmailSettings.allInstances->any(true).banList->includes(ticket1.email);
    cbt:=new BanEmailAndCloseTicket(staffMember:=generalConsultant, ticket:=ticket1);
    assert non-occurrence cbt;
}
```

TDCM application: Summary of changes performed in the schema

- Added
  
  `[TicketsManagementAndTracking.cstl] <line 2347> BanEmailAndCloseTicket is not defined in the CSUT as a class or an association

    event BanEmailAndCloseTicket<StaffTicketEvent
    operations
    effect() 
    end

    context BanEmailAndCloseTicket::effect()
    post:
    self.ticket.ticketStatus=#Closed and 
    EmailSettings.allInstances[](true).banList->includes(self.ticket.email) and...`
self.ticket.internalNote->one(int | int.oclIsNew() 
 and int.datetime=System.allInstances()->any(true).currentDateTime 
 and int.subject='Ticket closed' 
 and int.text='Email added to banlist and ticket status set to closed' 
 and int.author=self.staffMember.firstName 
 and self.ticket.lastMessageDatetime=System.allInstances()->any(true).currentDateTime)

• `osTicketCSUT.use:1414:40: Undefined operation named `banList' in expression `EmailSettings.banList()'.

```plaintext
class EmailSettings
attributes
administrationEmail:String
banList:Set{String}
end
```

• Preconditions of the domain event `cbt:BanEmailAndCloseTicket` are satisfied and consequently, the event can occur

```plaintext
context BanEmailAndCloseTicket ini inv ticketIsNotClosed:
not (self.ticket.ticketStatus=#Closed)
```

### Time spent

<table>
<thead>
<tr>
<th>TIME TO WRITE TEST CASES (IN MINUTES)</th>
<th>5.5</th>
</tr>
</thead>
<tbody>
<tr>
<td>TIME TO COMPLETE THE ITERATION (IN MINUTES)</td>
<td>9.5</td>
</tr>
</tbody>
</table>

### Errors and failures that drive the conceptual modeling

<table>
<thead>
<tr>
<th>A basic type involved in a test case does not exist in the CSUD</th>
<th>A derived type involved in a test case does not exist in the CSUD</th>
<th>An event type involved in a test case does not exist in the CSUD</th>
</tr>
</thead>
<tbody>
<tr>
<td>The basic type is relevant and it is added to the CSUD</td>
<td>The derived type is relevant and it is added to the CSUD</td>
<td>The event type is relevant and it is added to the CSUD</td>
</tr>
<tr>
<td>Inconsistent state before the occurrence of an event</td>
<td>Inconsistent state after the occurrence of an event</td>
<td>The postcondition of an event is not satisfied</td>
</tr>
<tr>
<td>Some static constraint is invalid and it is modified</td>
<td>Some initial integrity constraint is invalid and it is modified</td>
<td>The event postcondition/method is incorrect and it is modified</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>An assertion about the IB state fails or contains an error</td>
<td>Assert non-occurrence fails</td>
<td>Semantic error in an expression</td>
</tr>
<tr>
<td>The effect of an event type is not correct</td>
<td>A derivation rule is incorrect</td>
<td>The expression is corrected</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Assert consistency fails</td>
<td>A static constraint needs to be changed</td>
<td>The CSUD is changed</td>
</tr>
</tbody>
</table>
Iteration 90

Iteration objective

S75: BanTicketCloseEmail_ticketIsNotVisible

Current test case

```
test S75{
  load testConfiguration3;
  technicalActive.staffGroup.canBanEmails:=true;
  load created_tickets;
  li := new LogIn(username:='martin', password:='ttt');
  assert occurrence li;
  cbt:=new BanEmailAndCloseTicket(staffMember:=technicalActive, ticket:=ticket1);
  assert non-occurrence cbt;
}
```

TDCM application: Summary of changes performed in the schema

The CSUD has not been changed.

Time spent

```
| TIME TO WRITE TEST CASES (IN MINUTES) | 1 |
| TIME TO COMPLETE THE ITERATION (IN MINUTES) | 0.5 |
```

Iteration 91

Iteration objective

S76: BanTicketCloseEmail_notAllowed

Current test case

```
test S76{
  load testConfiguration3;
  load created_tickets;
  generalConsultant.staffGroup.canBanEmails:=false;
  li := new LogIn(username:='mary', password:='yyy');
  assert occurrence li;
  cbt:=new BanEmailAndCloseTicket(staffMember:=generalConsultant, ticket:=ticket2);
  assert non-occurrence cbt;
}
```
TDCM application: Summary of changes performed in the schema

- Added
- Updated

- Preconditions of the domain event `cbt:BanEmailAndCloseTicket` are satisfied and consequently, the event can occur

```plaintext
context BanEmailAndCloseTicket in inv staffMemberIsAllowedToBanEmails:
    self.staffMember.staffGroup.canBanEmails
```

**Time spent**

<table>
<thead>
<tr>
<th>Time to write test cases (in minutes)</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time to complete the iteration (in minutes)</td>
<td>2.5</td>
</tr>
</tbody>
</table>

**Errors and failures that drive the conceptual modeling**

<table>
<thead>
<tr>
<th>A basic type involved in a test case does not exist in the CSUD</th>
<th>A derived type involved in a test case does not exist in the CSUD</th>
<th>An event type involved in a test case does not exist in the CSUD</th>
</tr>
</thead>
<tbody>
<tr>
<td>The basic type is relevant and it is added to the CSUD</td>
<td>The derived type is relevant and it is added to the CSUD</td>
<td>The event type is relevant and it is added to the CSUD</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Inconsistent state before the occurrence of an event</th>
<th>Inconsistent state after the occurrence of an event</th>
<th>The postcondition of an event is not satisfied.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Some static constraint is invalid and it is modified.</td>
<td>Some initial integrity constraint is invalid and it is modified.</td>
<td>The event postcondition/method is invalid and it is modified.</td>
</tr>
<tr>
<td></td>
<td>The event postcondition/method is incorrect and it is modified.</td>
<td>The method is not correct and it is modified.</td>
</tr>
<tr>
<td></td>
<td>Some constraint is invalid and it is modified.</td>
<td>The postcondition is not correct and it is modified.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>An assertion about the IB state fails or contains an error</th>
<th>Assert non-occurrence fails</th>
<th>Semantic error in an expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>The effect of an event type is not correct</td>
<td>A derivation rule is incorrect</td>
<td>The expression is corrected</td>
</tr>
<tr>
<td>A preconidion is added/updated</td>
<td>The postcondition is not correct and it is modified.</td>
<td>The CSUD is changed</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Assert consistency fails</th>
</tr>
</thead>
<tbody>
<tr>
<td>A static constraint needs to be changed</td>
</tr>
</tbody>
</table>

**Iteration 92**

**Iteration objective**

S77: BanTicketCloseEmail_NotLoggedln

**Current test case**

```plaintext
test S77{
    load testConfiguration3;
    load created_tickets;

cbt:=new BanEmailAndCloseTicket(staffMember:=generalConsultant, ticket:=ticket2);
assert non-occurrence cbt;
}
```
TDCM application: Summary of changes performed in the schema

The CSUD has not been changed.

Time spent

<table>
<thead>
<tr>
<th>TIME TO WRITE TEST CASES (IN MINUTES)</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>TIME TO COMPLETE THE ITERATION (IN MINUTES)</td>
<td>2.5</td>
</tr>
</tbody>
</table>

Iteration 93

Iteration objective

S78: DeleteTicket

Current test case

```plaintext
test S78{
    load testConfiguration3;
    load created_tickets;
    li := new Login(username:='mary', password:='yyy');
    assert occurrence li;
    dt:=new DeleteTicket(staffMember:=generalConsultant, ticket:=ticket1);
    assert occurrence dt;
    assert true [not(Ticket.allInstances()->exists(tt:number=1))];
}
```

TDCM application: Summary of changes performed in the schema

- Added
  - [TicketsManagementAndTracking.cstl] <line 2401> DeleteTicket is not defined in the CSUT as a class or an association

```plaintext
  event DeleteTicket
  operations
effect() end

  association deleteTicket_ticket between
  DeleteTicket[*]
  Ticket[0..1]
  end

  association deleteTicket_staffMember between
  DeleteTicket[*]
  StaffMember[1]
  end

  context DeleteTicket::effect()
```
DEVELOPMENT OF THE CONCEPTUAL
SCHEMA OF THE osTICKET
SYSTEM BY APPLYING TDCM

Time spent

| TIME TO WRITE TEST CASES (IN MINUTES) | 4 |
| TIME TO COMPLETE THE ITERATION (IN MINUTES) | 22 |

Errors and failures that drive the conceptual modeling

<table>
<thead>
<tr>
<th>Problem Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>A basic type involved in a test case does not exist in the CSUD</td>
<td>The basic type is relevant and it is added to the CSUD</td>
</tr>
<tr>
<td>A derived type involved in a test case does not exist in the CSUD</td>
<td>The derived type is relevant and it is added to the CSUD</td>
</tr>
<tr>
<td>An event type involved in a test case does not exist in the CSUD</td>
<td>The event type is relevant and it is added to the CSUD</td>
</tr>
<tr>
<td>Inconsistent state before the occurrence of an event</td>
<td>The postcondition of an event is not satisfied.</td>
</tr>
<tr>
<td>Inconsistent state after the occurrence of an event</td>
<td>Some static constraint is invalid and it is modified.</td>
</tr>
<tr>
<td>Some initial integrity constraint is invalid and it is modified.</td>
<td>The event postcondition/method is incorrect and it is modified.</td>
</tr>
<tr>
<td>The event postcondition/method is incorrect and it is modified.</td>
<td>Some constraint is invalid and it is modified.</td>
</tr>
<tr>
<td>The method is not correct and it is modified.</td>
<td>The postcondition is not correct and it is modified.</td>
</tr>
<tr>
<td>An assertion about the IB state fails or contains an error</td>
<td>Assert non-occurrence fails</td>
</tr>
<tr>
<td>Assert occurrence fails</td>
<td>Semantic error in an expression</td>
</tr>
<tr>
<td>The effect of an event type is not correct</td>
<td>A derivation rule is incorrect</td>
</tr>
<tr>
<td>A derivation rule is incorrect</td>
<td>The expression is corrected</td>
</tr>
<tr>
<td>A precondition is added/updated</td>
<td>The CSUD is changed</td>
</tr>
<tr>
<td>The expression is corrected</td>
<td></td>
</tr>
<tr>
<td>Some constraint is invalid and it is modified.</td>
<td></td>
</tr>
<tr>
<td>The method is not correct and it is modified.</td>
<td></td>
</tr>
<tr>
<td>The postcondition is not correct and it is modified.</td>
<td></td>
</tr>
<tr>
<td>Assert consistency fails</td>
<td></td>
</tr>
<tr>
<td>A static constraint needs to be changed</td>
<td></td>
</tr>
</tbody>
</table>

Iteration 94

Iteration objective

S79: DeleteTicket_ticketIsNotVisible

Current test case

test S79{
  load testConfiguration3;
  load created_tickets;
  technicalActive.staffGroup.canDeleteTickets:=true;
  li := new LogIn(username:='mary', password:='yyy');
  assert occurrence li;
  dt:=new DeleteTicket(staffMember:=technicalActive, ticket:=ticket2);
  assert non-occurrence dt;
}
TDCM application: Summary of changes performed in the schema

- Added
  - Preconditions of the domain event dt:DeleteTicket are satisfied and consequently, the event can occur

    context DeleteTicket ini inv theTicketsVisible:
    self.staffMember.isAdministrator or
    (self.staffMember.staffGroup.departmentsAccess->including(self.staffMember.department))
    ->includes(self.ticket.assignedDepartment)

Time spent

<table>
<thead>
<tr>
<th>Description</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time to write test cases (in minutes)</td>
<td>1.5</td>
</tr>
<tr>
<td>Time to complete the iteration (in minutes)</td>
<td>3</td>
</tr>
</tbody>
</table>

Errors and failures that drive the conceptual modeling

<table>
<thead>
<tr>
<th>Error/Failure</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>A basic type involved in a test case does not exist in the CSUD</td>
<td>The basic type is relevant and it is added to the CSUD</td>
</tr>
<tr>
<td>A derived type involved in a test case does not exist in the CSUD</td>
<td>The derived type is relevant and it is added to the CSUD</td>
</tr>
<tr>
<td>An event type involved in a test case does not exist in the CSUD</td>
<td>The event type is relevant and it is added to the CSUD</td>
</tr>
<tr>
<td>Inconsistent state before the occurrence of an event</td>
<td>Some static constraint is invalid and it is modified.</td>
</tr>
<tr>
<td>Inconsistent state after the occurrence of an event</td>
<td>Some initial integrity constraint is invalid and it is modified.</td>
</tr>
<tr>
<td>The postcondition of an event is not satisfied</td>
<td>The event postcondition/method is incorrect and it is modified.</td>
</tr>
<tr>
<td>The assertion about the IB state fails or contains an error</td>
<td>The postcondition is not correct and it is modified.</td>
</tr>
<tr>
<td>Assert non-occurrence fails</td>
<td>The method is not correct and it is modified.</td>
</tr>
<tr>
<td>Semantic error in an expression</td>
<td>The postcondition is not satisfied.</td>
</tr>
<tr>
<td>The effect of an event type is not correct</td>
<td>The expression is corrected.</td>
</tr>
<tr>
<td>A derivation rule is incorrect</td>
<td>The CSUD is changed.</td>
</tr>
<tr>
<td>A precondition is added/updated</td>
<td>The expression is corrected.</td>
</tr>
<tr>
<td>A constraint is invalid and it is modified</td>
<td>The method is not correct and it is modified.</td>
</tr>
<tr>
<td>Assert consistency fails</td>
<td>The postcondition is not satisfied.</td>
</tr>
<tr>
<td>A static constraint needs to be changed</td>
<td>The expression is corrected.</td>
</tr>
</tbody>
</table>

Iteration 95

Iteration objective

S80: DeleteTicket_notAllowed

Current test case

```java
// Load test configuration
load testConfiguration3;
load created_tickets;
technicalActive.staffGroup.canDeleteTickets := false;

l := new Login(username:='mary', password:='yyy');
```
assert occurrence li;

dt:=new DeleteTicket(staffMember:=technicalActive, ticket:=ticket1);
assert non-occurrence dt;

TDCM application: Summary of changes performed in the schema

- Added
  - Preconditions of the domain event dt:DeleteTicket are satisfied and consequently, the event can occur

```plaintext
context DeleteTicket ini inv staffMembersAllowedToDeleteTickets:
    self.staffMember.isAdministrator or
    self.staffMember.staffGroup.canDeleteTickets
```

Time spent

<table>
<thead>
<tr>
<th>Time to Write Test Cases (in minutes)</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time to Complete the Iteration (in minutes)</td>
<td>1.5</td>
</tr>
</tbody>
</table>

Errors and failures that drive the conceptual modeling

<table>
<thead>
<tr>
<th>A basic type involved in a test case does not exist in the CSUD</th>
<th>A derived type involved in a test case does not exist in the CSUD</th>
<th>An event type involved in a test case does not exist in the CSUD</th>
</tr>
</thead>
<tbody>
<tr>
<td>The basic type is relevant and it is added to the CSUD</td>
<td>The derived type is relevant and it is added to the CSUD</td>
<td>The event type is relevant and it is added to the CSUD</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Inconsistent state before the occurrence of an event</th>
<th>Inconsistent state after the occurrence of an event</th>
<th>The postcondition of an event is not satisfied.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Some static constraint is invalid and it is modified.</td>
<td>Some initial integrity constraint is invalid and it is modified.</td>
<td>The event postcondition/method is incorrect and it is modified.</td>
</tr>
<tr>
<td></td>
<td>The event postcondition/method is incorrect and it is modified.</td>
<td>Some constraint is invalid and it is modified.</td>
</tr>
<tr>
<td></td>
<td>Some constraint is invalid and it is modified.</td>
<td>The method is not correct and it is modified.</td>
</tr>
<tr>
<td></td>
<td>The postcondition is not correct.</td>
<td>The postcondition is not correct.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>An assertion about the IB state fails or contains an error</th>
<th>Assert non-occurrence fails</th>
<th>Semantic error in an expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>The effect of an event type is not correct</td>
<td>A derivation rule is incorrect</td>
<td>The expression is corrected</td>
</tr>
<tr>
<td>A precondition is added/updated</td>
<td>The postcondition is not correct</td>
<td>The CSUD is changed</td>
</tr>
<tr>
<td>Assert consistency fails</td>
<td></td>
<td>1</td>
</tr>
</tbody>
</table>

Iteration 96

Iteration objective

S81: DeleteTicket_notLoggedIn
Current test case

```plaintext
test S81{
    load testConfiguration3;
    load created_tickets;
    dt:=new DeleteTicket(staffMember:=generalConsultant, ticket:=ticket1);
    assert non-occurrence dt;
}
```

TDCM application: Summary of changes performed in the schema

- **Added**
  - Preconditions of the domain event `dt:DeleteTicket` are satisfied and consequently, the event can occur

```
context DeleteTicket ini inv staffMemberIsLoggedIn:
    self.staffMember.isLoggedin
```

Time spent

| TIME TO WRITE TEST CASES (IN MINUTES) | 1 |
| TIME TO COMPLETE THE ITERATION (IN MINUTES) | 1 |

Errors and failures that drive the conceptual modeling

<table>
<thead>
<tr>
<th>A basic type involved in a test case does not exist in the CSUD</th>
<th>A derived type involved in a test case does not exist in the CSUD</th>
<th>An event type involved in a test case does not exist in the CSUD</th>
</tr>
</thead>
<tbody>
<tr>
<td>The basic type is relevant and it is added to the CSUD</td>
<td>The derived type is relevant and it is added to the CSUD</td>
<td>The event type is relevant and it is added to the CSUD</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Inconsistent state before the occurrence of an event</th>
<th>Inconsistent state after the occurrence of an event</th>
<th>The postcondition of an event is not satisfied.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Some static constraint is invalid and it is modified.</td>
<td>Some initial integrity constraint is invalid and it is modified.</td>
<td>The event postcondition/method is incorrect and it is modified.</td>
</tr>
<tr>
<td>The event postcondition/method is incorrect and it is modified.</td>
<td>Some constraint is invalid and it is modified.</td>
<td>Some constraint is not correct and it is modified.</td>
</tr>
<tr>
<td>The method is not correct and it is modified.</td>
<td>The postcondition is not correct and it is modified.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>An assertion about the IB state fails or contains an error</th>
<th>Assert non-occurrence fails</th>
<th>Semantic error in an expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>The effect of an event type is not correct</td>
<td>A derivation rule is incorrect</td>
<td>A precondition is added/updated</td>
</tr>
<tr>
<td>The expression is corrected</td>
<td>The CSUD is changed</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Assert consistency fails</th>
<th>A static constraint needs to be changed</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>
Iteration 97

Iteration objective

S82: BannedEmailsCannotCreateTickets

Current test case

```java
Iteration objective

S82: BannedEmailsCannotCreateTickets

Current test case

test S82{
  load testConfiguration4;
  EmailSettings.allInstances() -> any(true).banList := Set{'hello_at_helloworld.hel'};
  li := new LogIn(username:='mary', password:='yyy');
  assert occurrence li;
  nt1:=new NewTicketOffline;
  nt1.fullName:='Mary Marnes';
  nt1.email:='hello_at_helloworld.hel';
  nt1.telephone:='xxxxxxxx';
  nt1.ext:='xxxxxxxx';
  nt1.source:=#Phone;
  nt1.assignedDepartment:=dptTechnical;
  nt1.helpTopic:=helpTopicInstallation;
  nt1.subject:='Error operating system';
  nt1.message:='The installation process does not finish...';
  nt1.internalNote:='It seems that the correct installer is being used';
  dt2:=new Datetime(value:=[(sys.currentDateTime.value+2)]);
  nt1.dueDatetime:=dt2;
  nt1.priority:=#Normal;
  nt1.assignedStaff:=generalConsultant;
  nt1.creator:=generalConsultant;
  assert non-occurrence nt1;

  nt4:=new NewTicketOnline;
  nt4.fullName:='James Jordan';
  nt4.email:='hello_at_helloworld.hel';
  nt4.helpTopic:=helpTopicUse;
  nt4.subject:='Reopening ticket';
  nt4.message:='I do not know how to reopen one of my closed tickets';
  assert non-occurrence nt4;

  nt5:=new NewTicketByEmail;
  nt5.toAddress:='technical_at_support.com';
  nt5.fromName:='Marta Johnes';
  nt5.fromAddress:='hello_at_helloworld.hel';
  nt5.subject:='See my tickets';
  nt5.message:='Can I see my tickets?';
  assert non-occurrence nt5;
}

TDCM application: Summary of changes performed in the schema

- Added
  - Preconditions of the domain event nt1:NewTicketOffline are satisfied and consequently, the event can occur

  context NewTicketOffline ini inv emailsNotInBanList:
    EmailSettings.allInstances() -> any(true).banList -> notEmpty() implies
    EmailSettings.allInstances() -> any(true).banList -> excludes(self.email)
DEVELOPMENT OF THE CONCEPTUAL
SCHEMA OF THE osTICKET
SYSTEM BY APPLYING TDCM

- Preconditions of the domain event nt4: NewTicketOnline are satisfied and consequently, the event can occur
  
  context NewTicketOnline ini inv emailIsNotInBanList:
  EmailSettings.allInstances()->any(true).banList->notEmpty() implies
  EmailSettings.allInstances()->any(true).banList->excludes(self.fromEmail)

- Preconditions of the domain event nt5: NewTicketByEmail are satisfied and consequently, the event can occur
  
  context NewTicketEmail ini inv emailIsNotInBanList:
  EmailSettings.allInstances()->any(true).banList->notEmpty() implies
  EmailSettings.allInstances()->any(true).banList->excludes(self.email)

Time spent

<table>
<thead>
<tr>
<th>Time to write test cases (in minutes)</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time to complete the iteration (in minutes)</td>
<td>8.5</td>
</tr>
</tbody>
</table>

Errors and failures that drive the conceptual modeling

<table>
<thead>
<tr>
<th>A basic type involved in a test case does not exist in the CSUD</th>
<th>A derived type involved in a test case does not exist in the CSUD</th>
<th>An event type involved in a test case does not exist in the CSUD</th>
</tr>
</thead>
<tbody>
<tr>
<td>The basic type is relevant and it is added to the CSUD</td>
<td>The derived type is relevant and it is added to the CSUD</td>
<td>The event type is relevant and it is added to the CSUD</td>
</tr>
<tr>
<td>Inconsistent state before the occurrence of an event</td>
<td>Inconsistent state after the occurrence of an event</td>
<td>The postcondition of an event is not satisfied.</td>
</tr>
<tr>
<td>Some static constraint is invalid and it is modified.</td>
<td>Some initial integrity constraint is invalid and it is modified.</td>
<td>Some constraint is invalid and it is modified.</td>
</tr>
<tr>
<td>Some constraint is invalid and it is modified.</td>
<td>The event postcondition/method is incorrect and it is modified.</td>
<td>The method is not correct and it is modified.</td>
</tr>
<tr>
<td>The effect of an event type is not correct</td>
<td>A derivation rule is incorrect</td>
<td>The postcondition is not correct and it is modified.</td>
</tr>
<tr>
<td>Assert non-occurrence fails</td>
<td>A precondition is added/updated</td>
<td>The expression is corrected</td>
</tr>
<tr>
<td>Semantic error in an expression</td>
<td>3</td>
<td>The CSUD is changed</td>
</tr>
</tbody>
</table>

Iteration 98

Iteration objective

S83: CheckOverdueTickets_staffAlertsDisabled

Current test case

```java
test S83{
  load testConfiguration5;
  load created_tickets;
}```
DEVELOPMENT OF THE CONCEPTUAL
SCHEMA OF THE osTICKET
SYSTEM BY APPLYING TDCM

April 2011

DEPARTMENT OF SERVICE AND
INFORMATION SYSTEM ENGINEERING

TDCM application: Summary of changes performed in the schema

---

**Added**

- [TicketsManagementAndTracking.cstl] <line 2499> CheckOverdueTickets is not defined in the CSUT as a class or an association

```plaintext
dt3:=new Datetime(value:=(sys.currentDateTime.value+2));
sys.currentDateTime:=dt3;

cot:=new CheckOverdueTickets;
assert occurrence cot;

assert equals ticket1.isOverdue true;

// notice to assigned staff
assert false [EMail.allInstances()->exists(e|e.emailKind.oclIsTypeOf(NewInternalNoteAlertToStaff) and
  e.fromAddress='general_at_support.com' and
  e.toAddress='mary_at_support.com' and
  e.ticketNumber=1)];

// notice to department member
assert false [EMail.allInstances()->exists(e|e.emailKind.oclIsTypeOf(NewInternalNoteAlertToStaff) and
  e.fromAddress='general_at_support.com' and
  e.toAddress='john_at_support.com' and
  e.ticketNumber=1)];
```

---

Department of Service and
Information System Engineering
DEVELOPMENT OF THE CONCEPTUAL SCHEMA OF THE osTICKET SYSTEM BY APPLYING TDCM

EMail.allInstances() ->exists(e|e.fromAddress=staffAlertsFromEMailAddress and
e.toAddress=t.assignedDepartment.departmentManager.emailAddress and
e.ticketNumber=t.ticketNumber)

and (sendOverdueTicketAlertToDepartmentMembers
implies
  t.assignedDepartment.staffMember->forAll(m|
  (m.status=#Enabled and not(m.isInVacationMode))
implies
    EMail.allInstances() ->exists(e|e.fromAddress=staffAlertsFromEMailAddress and
    e.toAddress=m.emailAddress and
    e.ticketNumber=t.ticketNumber)))

Time spent

| TIME TO WRITE TEST CASES (IN MINUTES) | 7.5 |
| TIME TO COMPLETE THE ITERATION (IN MINUTES) | 24 |

Errors and failures that drive the conceptual modeling

<table>
<thead>
<tr>
<th>A basic type involved in a test case does not exist in the CSUD</th>
<th>A derived type involved in a test case does not exist in the CSUD</th>
<th>An event type involved in a test case does not exist in the CSUD</th>
</tr>
</thead>
<tbody>
<tr>
<td>The basic type is relevant and it is added to the CSUD</td>
<td>The derived type is relevant and it is added to the CSUD</td>
<td>The event type is relevant and it is added to the CSUD</td>
</tr>
<tr>
<td>Inconsistent state before the occurrence of an event</td>
<td>Inconsistent state after the occurrence of an event</td>
<td>The postcondition of an event is not satisfied.</td>
</tr>
<tr>
<td>Some static constraint is invalid and it is modified.</td>
<td>Some initial integrity constraint is invalid and it is modified.</td>
<td>The event postcondition/method is incorrect and it is modified.</td>
</tr>
<tr>
<td>An assertion about the IB state fails or contains an error</td>
<td>Assert non-occurrence fails</td>
<td>Semantic error in an expression</td>
</tr>
<tr>
<td>The effect of an event type is not correct</td>
<td>A derivation rule is incorrect</td>
<td>A precondition is added/updated</td>
</tr>
<tr>
<td>Assert consistency fails</td>
<td>The expression is corrected</td>
<td>The CSUD is changed</td>
</tr>
<tr>
<td>A static constraint needs to be changed</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Iteration 99

Iteration objective

S84: CheckOverdueTickets_staffAlertsEnabled

Current test case

test S84{
  load testConfiguration1;
  load created_tickets;
dt3:=new Datetime(value:=[sys.currentDateTime.value+2]);
sys.currentDateTime:=dt3;

cot:=new CheckOverdueTickets;
assert occurrence cot;

assert equals ticket1.isOverdue true;

//notice to assigned staff
assert true [EMail.allInstances()->exists(e|e.emailKind.oclIsTypeOf(OverdueTicketAlertToStaff) and
e.fromAddress='general_at_support.com' and
e.toAddress='mary_at_support.com' and
e.ticketNumber=1)];

//notice to department member
assert true [EMail.allInstances()->exists(e|e.emailKind.oclIsTypeOf(OverdueTicketAlertToStaff) and
e.fromAddress='general_at_support.com' and
e.toAddress='john_at_support.com' and
e.ticketNumber=1)];

TDCM application: Summary of changes performed in the schema

The CSUD has not been changed

Time spent

<table>
<thead>
<tr>
<th>TIME TO WRITE TEST CASES (IN MINUTES)</th>
<th>3,5</th>
</tr>
</thead>
<tbody>
<tr>
<td>TIME TO COMPLETE THE ITERATION (IN MINUTES)</td>
<td>0,5</td>
</tr>
</tbody>
</table>

Iteration 100

Iteration objective

S85: ViewTickets_open_myTickets_overdue_closed

Current test case

test S85{ 
load testConfiguration3;
load created_tickets;

li := new LogIn(username:='john', password:='xxx');
assert occurrence li;

dts:=new DisplayTicketsByStatus(consultant:=generalAdministrator, status:=#OpenTickets);
assert occurrence dts;
assert equals dts.answer() [Sequence{
Tuple(createDate=1,department='Technical support',email='mary_at_marnes.mar',number=1,priority=#Normal,subject='Error operating system'),
Tuple(createDate=1,department='General support',email='mary_at_marnes.mar',number=2,priority=#High,subject='Can I reply a ticket?');
}
TDCM application: Summary of changes performed in the schema

- **Added**
  - [TicketsManagementAndTracking.cstl] <Line 2557>:1:1: Undefined enumeration literal 'Assigned'.
    ```
    enum StatusFilter{OpenTickets,ClosedTickets,AssignedToMe,OverdueTickets}
    ```
event DisplayTicketsByStatus
attributes
status:StatusFilter
operations
answer():Sequence(Tuple(number:Integer,createDate:Integer,subject:String,department:String,
priority:Priority,email:String))=
   let visibleDepartments:Set(Department)=
      self.consultant.staffGroup.departmentsAccess->including(self.consultant.department)
   in
      if self.status=#OpenTickets then
         Ticket.allInstances
         ->select(t|t.ticketStatus=#Open and visibleDepartments->includes(t.assignedDepartment))
         -> collect (t | Tuple  {number : t.number, createDate:t.creationDatetime.value,
          subject:t.subject,department:t.assignedDepartment.name, priority:t.priority, email:t.email})->sortedBy(number)
      else
         if self.status=#AssignedToMe then
            Ticket.allInstances
            ->select(t|t.assignedStaff=self.consultant and t.ticketStatus<>#Closed and visibleDepartments->
            includes(t.assignedDepartment))
            -> collect (t | Tuple  {number : t.number, createDate:t.creationDatetime.value,
            subject:t.subject,department:t.assignedDepartment.name, priority:t.priority, email:t.email})->sortedBy(number)
         else
            if self.status=#OverdueTickets then
               Ticket.allInstances
               ->select(t|t.isOverdue and visibleDepartments->includes(t.assignedDepartment))
               -> collect (t | Tuple  {number : t.number, createDate:t.creationDatetime.value,
               subject:t.subject,department:t.assignedDepartment.name, priority:t.priority, email:t.email})->sortedBy(number)
            else
               Ticket.allInstances
               ->select(t|t.ticketStatus=#Closed and visibleDepartments->includes(t.assignedDepartment))
               -> collect (t | Tuple  {number : t.number, createDate:t.creationDatetime.value,
               subject:t.subject,department:t.assignedDepartment.name, priority:t.priority, email:t.email})->sortedBy(number)
            endif
         endif
      endif
   endif
   effect()
end

The result is Sequence[{createDate=1,department="Technical support",email="mary_at_marnes.mar",number=1,priority=#Normal,subject="Error operating system"},Tuple {createDate=2,department="General support",email="james_at_jordan.jam",number=4,priority=#Normal,subject="Reopening ticket"}] but it is expected to be Sequence[{createDate=1,department="Technical support",email="mary_at_marnes.mar",number=1,priority=#Normal,subject="Error operating system"}]

We realize that when asking for overdue, closed tickets are also shown

event DisplayTicketsByStatus
attributes
status:StatusFilter
operations
answer():Sequence(Tuple(number:Integer,createDate:Integer,subject:String,department:String,
priority:Priority,email:String))=
   let visibleDepartments:Set(Department)=
      self.consultant.staffGroup.departmentsAccess->including(self.consultant.department)
   in
      if self.status=#OpenTickets then
         Ticket.allInstances
         ->select(t|t.ticketStatus=#Open and visibleDepartments->includes(t.assignedDepartment))
         -> collect (t | Tuple  {number : t.number, createDate:t.creationDatetime.value,
         subject:t.subject,department:t.assignedDepartment.name, priority:t.priority, email:t.email})->sortedBy(number)
      else
         if self.status=#AssignedToMe then
            Ticket.allInstances
            ->select(t|t.assignedStaff=self.consultant and t.ticketStatus<>#Closed and visibleDepartments->
            includes(t.assignedDepartment))
            -> collect (t | Tuple  {number : t.number, createDate:t.creationDatetime.value,
            subject:t.subject,department:t.assignedDepartment.name, priority:t.priority, email:t.email})->sortedBy(number)
         else
            if self.status=#OverdueTickets then
               Ticket.allInstances
               ->select(t|t.isOverdue and visibleDepartments->includes(t.assignedDepartment))
               -> collect (t | Tuple  {number : t.number, createDate:t.creationDatetime.value,
               subject:t.subject,department:t.assignedDepartment.name, priority:t.priority, email:t.email})->sortedBy(number)
            else
               Ticket.allInstances
               ->select(t|t.ticketStatus=#Closed and visibleDepartments->includes(t.assignedDepartment))
               -> collect (t | Tuple  {number : t.number, createDate:t.creationDatetime.value,
               subject:t.subject,department:t.assignedDepartment.name, priority:t.priority, email:t.email})->sortedBy(number)
            endif
         endif
      endif
   endif
   effect()
DEVELOPMENT OF THE CONCEPTUAL
SCHEMA OF THE osTICKET
SYSTEM BY APPLYING TDCM

Research report
April 2011

Department of Service and
Information System Engineering

-> collect (t | Tuple {number : t.number, creationDateTime.value, subject:t.subject, assignedDepartment.name, priority:t.priority, email:t.email})->sortedBy(number)
else if self.status=#OverdueTickets then
    Ticket.allInstances->select(t|t.isOverdue and t.ticketStatus<>#Closed and visibleDepartments->includes(t.assignedDepartment))
-> collect (t | Tuple {number : t.number, creationDateTime.value, subject:t.subject,assignedDepartment.name, priority:t.priority, email:t.email})->sortedBy(number)
else
    Ticket.allInstances->select(t|t.ticketStatus=#Closed and visibleDepartments->includes(t.assignedDepartment))
-> collect (t | Tuple {number : t.number, creationDateTime.value, subject:t.subject,assignedDepartment.name, priority:t.priority, email:t.email})->sortedBy(number)
effect()
end

Time spent

| TIME TO WRITE TEST CASES (IN MINUTES) | 13 |
| TIME TO COMPLETE THE ITERATION (IN MINUTES) | 14,5 |

Errors and failures that drive the conceptual modeling

<table>
<thead>
<tr>
<th>A basic type involved in a test case does not exist in the CSUD</th>
<th>A derived type involved in a test case does not exist in the CSUD</th>
<th>An event type involved in a test case does not exist in the CSUD</th>
</tr>
</thead>
<tbody>
<tr>
<td>The basic type is relevant and it is added to the CSUD</td>
<td>The derived type is relevant and it is added to the CSUD</td>
<td>The event type is relevant and it is added to the CSUD</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Inconsistent state before the occurrence of an event</th>
<th>Inconsistent state after the occurrence of an event</th>
<th>The postcondition of an event is not satisfied.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Some static constraint is invalid and it is modified.</td>
<td>Some initial integrity constraint is invalid and it is modified.</td>
<td>The event precondition/method is incorrect and it is modified.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Semantic error in an expression</th>
</tr>
</thead>
</table>

1

An assertion about the IB state fails or contains an error

<table>
<thead>
<tr>
<th>Assert non-occurrence fails</th>
<th>Semantic error in an expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>The effect of an event type is not correct</td>
<td>The expression is incorrect</td>
</tr>
<tr>
<td>A derivation rule is incorrect</td>
<td>The CSUD is changed</td>
</tr>
<tr>
<td>A precondition is added/updated</td>
<td></td>
</tr>
</tbody>
</table>

1

Assert consistency fails

<table>
<thead>
<tr>
<th>A static constraint needs to be changed</th>
</tr>
</thead>
</table>
In the following, we present the graphical UML form of the resultant conceptual schema of the osTicket system, obtained by applying TDCM.

**Structural schema**
context TicketSettings inv hasOnlyOneInstance:
    TicketSettings.allInstances()->size()=1

context TicketSettings inv specifiesTheMaximumNumberOfOpenTicketPerMailIfNotUnlimited:
    self.openTicketsPerMailAreLimited implies self.maximumOpenTicketsPerMail.isDefined()

context CustomerAutoresponsesSettings inv hasOnlyOneInstance:
    CustomerAutoresponsesSettings.allInstances()->size()=1

context StaffNoticesAlertsSettings inv hasOnlyOneInstance:
    StaffNoticesAlertsSettings.allInstances()->size()=1

**Behavioral schema**

The resultant conceptual schema contains 24 event types. We present the UML graphical form of one of them (the assignment of tickets) and its event specification and initial integrity constraints (preconditions) in OCL:

```oclm
context AssignTicket::effect()
post:
let staffAlertsFromEMailAddress:String=
    EmailSettings.allInstances()->any(true).defaultStaffAlertsEmail.address
in
self.ticket.assignedStaff=self.assignee
and
(InternaNote.allInstances().InternalNote.allInstances@pre)
    ->one(i | i.oclIsNew() and i.datetime=System.allInstances()->any(true).currentDateTime
        and i.subject='Ticket Reassigned'
        and i.text=self.assignmentText
        and i.author=self.staffMember.firstName
        and i.ticket=self.ticket)
and self.ticket.lastMessageDateTime=System.allInstances()->any(true).currentDateTime
and (EMail.allInstances()->exists(e|e.fromAddress=staffAlertsFromEMailAddress and e.toAddress=self.assignee.emailAddress and e.ticketNumber=self.ticket.number))

context AssignTicket ini inv assigneeIsNotInVacationMode:
not(self.assignee.isInVacationMode)
```

We also reproduce the other event types in its USEx format:
effect()
end

context NewTicketOnline ini inv priorityMayBeSetWhenAllowed:
  if TicketSettings.allInstances()->any(true).customersCanChangePriority then
    self.priority.isDefined()
  else
    self.priority.isUndefined()
  endif

context NewTicketOnline ini inv emailIsNotInBanList:
  EmailSettings.allInstances()->any(true).banList->notEmpty() implies
  EmailSettings.allInstances()->any(true).banList->excludes(self.email)

association newTicketOnline_helpTopic between
  NewTicketOnline[*]
  HelpTopic[0..1]
end

context NewTicketOnline ini inv helpTopicSpecifiedIfAvailable:
  if HelpTopic.allInstances()->select(hp|hp.status=#Enabled)->size()>0 then
    self.helpTopic->size()=1
  else
    self.helpTopic->size()==0
  endif

context NewTicketOnline ini inv helpTopicIsEnabled:
  self.helpTopic->notEmpty() implies self.helpTopic.status=#Enabled

context NewTicketOnline ini inv helpDeskStatusIsOnline:
  GeneralSettings.allInstances()->any(true).status=#Online

context NewTicketOnline ini inv maximumOpenTicketsLimitIsNotViolated:
  if TicketSettings.allInstances()->any(true).openTicketsPerMailAreLimited then
    Ticket.allInstances->select(t|t.email=self.email)->size()<TicketSettings.allInstances()->any(true).maximumOpenTicketsPerMail
  else true
  endif

context NewTicketOnline::effect()
post:
  let defaultPriority:Priority=
    if self.helpTopic->notEmpty() then
      self.helpTopic.newTicketPriority
    else
      TicketSettings.allInstances()->any(true).priority
    endif
  in
  let assignedPriority:Priority=
    if TicketSettings.allInstances()->any(true).customersCanChangePriority then
      self.priority
    else
      defaultPriority
    endif
  in
  let defaultDepartment:Department=
    if self.helpTopic->notEmpty() then
      self.helpTopic.newTicketDepartment
    else
      Department.allInstances()->any(did.isDefault)
    endif
  in
  let sendNewTicketAlertToAdministrator:Boolean=
    StaffNoticesAlertsSettings.allInstances()->any(true).alertWhenNewTicketCreated and
    StaffNoticesAlertsSettings.allInstances()->any(true).alertWhenNewTicketCreatedStaff->includes(#Administrator)
  in
  let sendNewTicketAlertToDepartmentManager:Boolean=
    StaffNoticesAlertsSettings.allInstances()->any(true).alertWhenNewTicketCreated and
    StaffNoticesAlertsSettings.allInstances()->any(true).alertWhenNewTicketCreatedStaff->includes(#DepartmentManager)
  in
  let sendNewTicketAlertToDepartmentMembers:Boolean=
    StaffNoticesAlertsSettings.allInstances()->any(true).alertWhenNewTicketCreated and
    StaffNoticesAlertsSettings.allInstances()->any(true).alertWhenNewTicketCreatedStaff->includes(True)
StaffNoticesAlertsSettings.allInstances()->any(true).alertWhenNewTicketCreated and StaffNoticesAlertsSettings.allInstances()->any(true).alertWhenNewTicketCreatedStaff->includes(#DepartmentMembers)
in
let staffAlertsFromEMailAddress:String=
  EmailSettings.allInstances()->any(true).defaultStaffAlertsEmail.address
in
let sendAutoresponse:Boolean=
  if CustomerAutoresponsesSettings.allInstances()->any(true).autorespondWhenNewTicketCreatedByCustomer then
    if self.helpTopic->notEmpty then
      if self.helpTopic.autoresponse=#Enabled then true
      else false
    endif
  else
    defaultDepartment.newTicketAutoresponseIsSent
  endif
  else false
endif

in
(Ticket.allInstances- Ticket.allInstances@pre)
->one(t | t.oclIsNew() and self.createdTicket=t and t.fullName=self.fullName and t.email=self.email and t.telephone=self.telephone and t.subject=self.subject and t.message=self.message and t.ticketStatus=#Open and t.priority=assignedPriority and t.source=#Web and self.helpTopic=self.helpTopic and t.assignedDepartment=defaultDepartment and (TicketThreadMessage.allInstances- TicketThreadMessage.allInstances@pre)
  ->one(tdm | tdm.oclIsNew() and tdm.datetime=System.allInstances()->any(true).currentDateTime and tdm.text=self.message and tdm.author=self.fullName and tdm.ticket=t and t.lastMessageDatetime=System.allInstances()->any(true).currentDateTime)
  -- autoresponses
  and (sendAutoresponse implies
    EMail.allInstances()-exists(e|e.fromAddress=t.assignedDepartment.autoresponseEmail.address and e.toAddress=t.email and e.ticketNumber=t.number))

  -- staff notices
  and (sendNewTicketAlertToAdministrator implies
    EMail.allInstances()-exists(e|e.fromAddress=staffAlertsFromEMailAddress and e.toAddress=EmailSettings.allInstances()->any(true).administrationEmail and e.ticketNumber=t.number))
and (sendNewTicketAlertToDepartmentManager and t.assignedDepartment.departmentManager->notEmpty() and t.assignedDepartment.departmentManager.status=#Enabled and not(t.assignedDepartment.departmentManager.isInVacationMode) implies
  EMail.allInstances()-exists(e|e.fromAddress=staffAlertsFromEMailAddress and e.toAddress=t.assignedDepartment.departmentManager.emailAddress and e.ticketNumber=t.number))
and (sendNewTicketAlertToDepartmentMembers implies
  t.assignedDepartment.staffMember->forAll(m| (m.status=#Enabled and not(m.isInVacationMode)) implies
    EMail.allInstances()-exists(e|e.fromAddress=staffAlertsFromEMailAddress and e.toAddress=m.emailAddress and e.ticketNumber=t.number)))
)
event NewTicketByEmail
attributes
toAddress:String
fromName:String
fromAddress:String
subject:String
message:String
createdTicket:Ticket[0..1]
operations
effect()
end

context NewTicketByEmail in
post:
let incomingEmailAccount:EmailAccount=>
  EmailAccount.allInstances()->any(e|e.address=self.toAddress)
in
let assignedPriority:Priority=>
  if TicketSettings.allInstances()->any(true).useEmailPriorityWhenAvailable then
    incomingEmailAccount.defaultNewPriority
  else
    TicketSettings.allInstances()->any(true).priority
  endif
in
let defaultDepartment:Department=>
  incomingEmailAccount.defaultNewTicketDepartment
in
let sendNewTicketAlertToAdministrator:Boolean=>
  StaffNoticesAlertsSettings.allInstances()->any(true).alertWhenNewTicketCreated and
  StaffNoticesAlertsSettings.allInstances()->any(true).alertWhenNewTicketCreatedStaff->includes(#Administrator)
in
let sendNewTicketAlertToDepartmentManager:Boolean=>
  StaffNoticesAlertsSettings.allInstances()->any(true).alertWhenNewTicketCreated and
  StaffNoticesAlertsSettings.allInstances()->any(true).alertWhenNewTicketCreatedStaff->includes(#DepartmentManager)
in
let sendNewTicketAlertToDepartmentMembers:Boolean=>
  StaffNoticesAlertsSettings.allInstances()->any(true).alertWhenNewTicketCreated and
  StaffNoticesAlertsSettings.allInstances()->any(true).alertWhenNewTicketCreatedStaff->includes(#DepartmentMembers)
in
let staffAlertsFromEMailAddress:String=>
  EmailSettings.allInstances()->any(true).defaultStaffAlertsEmail.address
in
let sendAutoresponse:Boolean=>
  if (incomingEmailAccount.autoresponsesStatus=#Enabled) then true
  else false
  endif
in
(Ticket.allInstances- Ticket.allInstances@pre)
->one(t | t.oclIsNew() and self.createdTicket=t)
and t.assignedDepartment=incomingEmailAccount.defaultNewTicketDepartment
and (TicketThreadMessage.allInstances- TicketThreadMessage.allInstances@pre)
  ->one(tdm | tdm.oclIsNew())
and tdm.datetime=System.allInstances()->any(true).currentDateTime
and tdm.text=self.message
and tdm.author=self.fromName
and tdm.ticket=t
and t.lastMessageDatetime=System.allInstances()->any(true).currentDateTime)

context NewTicketByEmail in
post:
let maximumOpenTicketsLimitIsNotViolated:
  if TicketSettings.allInstances()->any(true).openTicketsPerMailAreLimited then
    if incomingEmailAccount.openTicketsPerMailAreLimited then
      true
    else
      incomingEmailAccount.openTicketsPerMailAreLimited
    endif
    endif
  else
    true
  endif
in
DEVELOPMENT OF THE CONCEPTUAL
SCHEMA OF THE osTICKET
SYSTEM BY APPLYING TDCM

Ticket.allInstances->select(tt.email=self.fromAddress)->size()<TicketSettings.allInstances()->any(true).maximumOpenTicketsPerMail
else true
endif

class NewTicketByEmail
def __init__(self):
    context NewTicketByEmail ini inv theIncomingEmailIsValid:
        EmailAccount.allInstances()->one(address=self.toAddress)

class NewTicketByEmail
def __init__(self):
    context NewTicketByEmail ini inv helpDeskStatusIsOnline:
        GeneralSettings.allInstances()->any(true).status=#Online

event DisplayTicketsAssociatedToEmail
attributes
    email: String
    ticketNumber: Integer
operations
    answer(): Set(Tuple(number: Integer, createDate: Integer, status: TicketStatus, department: String, email: String))=
        Ticket.allInstances->sortedBy(number)->collect(t | Tuple {number : t.number, createDate: t.creationDatetime.value, status: t.ticketStatus, subject: t.subject, department: t.assignedDepartment.name, email: t.email})->asSet()
effect()
end

class DisplayTicketsAssociatedToEmail
def __init__(self):
    context DisplayTicketsAssociatedToEmail ini inv thereAreTicketsAssociatedToTheEmail:
        Ticket.allInstances()->size()>0

event ReplyTicketByCustomer
attributes
    replyText: String
operations
effect()
end

association replyTicketByCustomer_ticket between
    ReplyTicketByCustomer[*]
    Ticket[1]
end

context ReplyTicketByCustomer::effect()
post:
    let sendNewMessageAlertToLastRespondent:Boolean=
        StaffNoticesAlertsSettings.allInstances()->any(true).alertWhenNewMessage and
        StaffNoticesAlertsSettings.allInstances()->any(true).alertWhenNewMessageStaff->includes(#LastRespondent)
in
    let sendNewMessageAlertToAssignedStaff:Boolean=
        StaffNoticesAlertsSettings.allInstances()->any(true).alertWhenNewMessage and
        StaffNoticesAlertsSettings.allInstances()->any(true).alertWhenNewMessageStaff->includes(#AssignedStaff)
in
    let sendNewMessageAlertToDepartmentManager:Boolean=
        StaffNoticesAlertsSettings.allInstances()->any(true).alertWhenNewMessage and
        StaffNoticesAlertsSettings.allInstances()->any(true).alertWhenNewMessageStaff->includes(#DepartmentManager)
in
    let staffAlertsFromEMailAddress:String=
        EmailSettings.allInstances()->any(true).defaultStaffAlertsEmail.address
in
    let sendAutoresponse:Boolean=
        if CustomerAutoresponsesSettings.allInstances()->any(true).autorespondWhenNewTicketCreatedByCustomer then
            if self.ticket.helpTopic->notEmpty then
                if self.ticket.helpTopic.autoresponse=#Enabled then true
                else false
            else false
        else false
    endif
endif
else false
endif
DEVELOPMENT OF THE CONCEPTUAL
SCHEMA OF THE osTICKET
SYSTEM BY APPLYING TDCM

-- autoresponses
and (sendAutoresponse implies
  EMail.allInstances()->|exists(e|e.fromAddress=osTicket.assignedDepartment.autoresponseEmail.address and
e.toAddress=osTicket.ticket.email and
e.ticketNumber=osTicket.ticket.number))

-- staff notices
and (sendNewMessageAlertToLastRespondent and
  osTicket.ticket.lastRespondent->|notEmpty() and
  not(osTicket.ticket.lastRespondent.isInVacationMode or osTicket.ticket.lastRespondent.status=#Disabled)
  implies
  EMail.allInstances()->|exists(e|e.fromAddress=staffAlertsFromEMailAddress and
e.toAddress=osTicket.ticket.lastRespondent and
e.ticketNumber=osTicket.ticket.number))

and (sendNewMessageAlertToAssignedStaff
  and osTicket.assignedStaff->|notEmpty() and
  not(osTicket.assignedStaff.isInVacationMode or osTicket.assignedStaff.status=#Disabled)
  implies
  EMail.allInstances()->|exists(e|e.fromAddress=staffAlertsFromEMailAddress and
e.toAddress=osTicket.assignedStaff.emailAddress and
e.ticketNumber=osTicket.ticket.number))

event LogIn
attributes
  username: String
  password: String
operations
effect()
end

context LogIn::effect()
post:
  StaffMember.allInstances()->|any(sm|sm.username=self.username and sm.password=self.password).isLoggedIn=true

context StaffMember inv usernameIsUnique:
  StaffMember.allInstances()->|isUnique(username)

context LogIn ini inv isNotLoggedIn:
  StaffMember.allInstances()->|select(sm|sm.username=self.username and sm.password=self.password)->|size()=0
  implies
  StaffMember.allInstances()->|any(sm|sm.username=self.username and sm.password=self.password).isLoggedIn=false

context LogIn ini inv accessDataIsValid:
  StaffMember.allInstances()->|select(sm|sm.username=self.username and sm.password=self.password)->|size()=0

context LogIn ini inv staffMemberIsEnabled:
  StaffMember.allInstances()->|select(sm|sm.username=self.username and sm.password=self.password)->|size()=0
  implies
  StaffMember.allInstances()->|any(sm|sm.username=self.username and sm.password=self.password).status=#Enabled

event LogOut
operations
effect()
end
association logOut_staffMember between
LogOut[*]
StaffMember[1]
end

context LogOut::effect()
post:
self.staffMember.isLoggedIn=false

context LogOut ini inv isNotLoggedIn:
self.staffMember.isLoggedIn=true

event NewTicketOffline
attributes
fullName: String
email: String
telephone: String[0..1]
ext: String[0..1]
source: TicketSource
subject: String
message: String
priority: Priority[0..1]
createdTicket: Ticket[0..1]
internalNote: String[0..1]
dueDatetime: Datetime[0..1]
operations
effect()
end

context NewTicketOffline ini inv emailIsNotInBanList:
EmailSettings.allInstances()->any(true).banList->notEmpty() implies
EmailSettings.allInstances()->any(true).banList->excludes(self.email)

association newTicketOffline_department between
NewTicketOffline[*]
Department[1] role assignedDepartment
end

association newTicketOffline_helpTopic between
NewTicketOffline[*]
HelpTopic[0..1]
end

association newTicketOffline_assignedStaff between
NewTicketOffline[*]
StaffMember[0..1] role assignedStaff
end

association newTicketOffline_creator between
NewTicketOffline[*] role newTicketOfflineOfCreator
StaffMember[1] role creator
end

context NewTicketOffline ini inv creatorIsLoggedIn:
self.creator.isLoggedIn

context NewTicketOffline ini inv creatorIsAllowedToCreateTickets:
self.creator.staffGroup.canCreateTickets

context NewTicketOffline ini inv helpTopicSpecifiedIfAvailable:
if HelpTopic.allInstances()->select(ht|ht.status=##Enabled)->size()>0 then
self.helpTopic->size()=1
else
self.helpTopic->size()=0
endif

context NewTicketOffline ini inv helpTopicsEnabled:
self.helpTopic->notEmpty() implies self.helpTopic.status=##Enabled
context NewTicketOffline ini inv maximumOpenTicketsLimitIsNotViolated:
if TicketSettings.allInstances()->any(true).openTicketsPerMailAreLimited then
    Ticket.allInstances()->select(t|t.email=self.email)->size()<TicketSettings.allInstances()->any(true).maximumOpenTicketsPerMail
else true
endif

context NewTicketOffline::effect() post:
let sendNewTicketAlertToAdministrator:Boolean=
    StaffNoticesAlertsSettings.allInstances()->any(true).alertWhenNewTicketCreated and
    StaffNoticesAlertsSettings.allInstances()->any(true).alertWhenNewTicketCreatedStaff->includes(#Administrator)
in
let sendNewTicketAlertToDepartmentManager:Boolean=
    StaffNoticesAlertsSettings.allInstances()->any(true).alertWhenNewTicketCreated and
    StaffNoticesAlertsSettings.allInstances()->any(true).alertWhenNewTicketCreatedStaff->includes(#DepartmentManager)
in
let sendNewTicketAlertToDepartmentMembers:Boolean=
    StaffNoticesAlertsSettings.allInstances()->any(true).alertWhenNewTicketCreated and
    StaffNoticesAlertsSettings.allInstances()->any(true).alertWhenNewTicketCreatedStaff->includes(#DepartmentMembers)
in
let staffAlertsFromEMailAddress:String=
    EmailSettings.allInstances()->any(true).defaultStaffAlertsEmail.address
in
let sendAutoresponse:Boolean=
    if CustomerAutoresponsesSettings.allInstances()->any(true).autorespondWhenNewTicketCreatedByCustomer then
        if self.helpTopic->notEmpty then
            if self.helpTopic.autoresponse=#Enabled then true
            else false
        endif
        else false
    endif
    else false
endif
in
(Ticket.allInstances- Ticket.allInstances@pre)
->one(t | t.oclIsNew() and self.createdTicket=t and t.fullName=self.fullName and t.telephone=self.telephone and t.ext=self.ext and t.subject=self.subject and t.message=self.message and t.ticketStatus=#Open and t.priority=self.priority and t.source=self.source and t.dueDatetime=self.dueDatetime and t.helpTopic=self.helpTopic and t.assignedDepartment=self.assignedDepartment and t.assignedStaff=self.assignedStaff)
and (TicketThreadMessage.allInstances- TicketThreadMessage.allInstances@pre)
->one(tdm | tdm.oclIsNew() and tdm.datetime=System.allInstances()->any(true).currentDateTime and tdm.text=self.message and tdm.author=self.fullName and tdm.ticket=t and t.lastMessageDatetime=System.allInstances()->any(true).currentDateTime)

-- autoresponses
and (sendAutoresponse implies
    EMail.allInstances()->exists(e|e.fromAddress=t.assignedDepartment.autoresponseEmail.address and e.toAddress=t.email and e.ticketNumber=t.number))

-- staff notices
and (sendNewTicketAlertToAdministrator implies
    EMail.allInstances()->exists(e|e.fromAddress=staffAlertsFromEMailAddress and e.toAddress=EmailSettings.allInstances()->any(true).administrationEmail and e.ticketNumber=t.number)
and (sendNewTicketAlertToDepartmentManager
    and t.assignedDepartment.departmentManager->notEmpty()
    and t.assignedDepartment.departmentManager.status=#Enabled
    and not(t.assignedDepartment.departmentManager.isInVacationMode)
    implies
    EMail.allInstances() ->exists(e|e.fromAddress=staffAlertsFromEMailAddress and
    e.toAddress=t.assignedDepartment.departmentManager.emailAddress and
    e.ticketNumber=t.number())
)

and (sendNewTicketAlertToDepartmentMembers
    implies
    t.assignedDepartment.staffMember->forAll(m|
    (m.status=#Enabled and not(m.isInVacationMode))
    implies
    EMail.allInstances() ->exists(e|e.fromAddress=staffAlertsFromEMailAddress and
    e.toAddress=m.emailAddress and
    e.ticketNumber=t.number()))
)

event DisplayTicketsByStatus
attributes
status:StatusFilter
operations
answer():Sequence(Tuple(number:Integer,createDate:Integer,subject:String,department:String, priority:Priority,email:String))= let visibleDepartments:Set(Department)=
    self.consultant.staffGroup.departmentsAccess->including(self.consultant.department) in
    if self.status=#OpenTickets then
        Ticket.allInstances
        ->select(t|t.ticketStatus=#Open and visibleDepartments->includes(t.assignedDepartment))
        -> collect (t | Tuple {number : t.number, createDate:t.creationDatetime.value,
        subject:t.subject,department:t.assignedDepartment.name, priority:t.priority, email:t.email})->sortBy(number)
    else
        if self.status=#AssignedToMe then
            Ticket.allInstances
            ->select(t|t.assignedStaff=self.consultant and t.ticketStatus<>#Closed and visibleDepartments->includes(t.assignedDepartment))
            -> collect (t | Tuple {number : t.number, createDate:t.creationDatetime.value,
            subject:t.subject,department:t.assignedDepartment.name, priority:t.priority, email:t.email})->sortBy(number)
        else
            if self.status=#OverdueTickets then
                Ticket.allInstances
                ->select(t|t.isOverdue and t.ticketStatus<>#Closed and visibleDepartments->includes(t.assignedDepartment))
                -> collect (t | Tuple {number : t.number, createDate:t.creationDatetime.value,
                subject:t.subject,department:t.assignedDepartment.name, priority:t.priority, email:t.email})->sortBy(number)
            else
                Ticket.allInstances
                ->select(t|t.ticketStatus=#Closed and visibleDepartments->includes(t.assignedDepartment))
                -> collect (t | Tuple {number : t.number, createDate:t.creationDatetime.value,
                subject:t.subject,department:t.assignedDepartment.name, priority:t.priority, email:t.email})->sortBy(number)
            endif
        endif
    endif
effect()
end

association displayTicketsByStatus_consultant between
DisplayTicketsByStatus[*] role newTicketOfflineOfConsultant
StaffMember[1] role consultant
end

context DisplayTicketsByStatus ini inv consultantIsLoggedIn:
    self.consultant.isLoggedIn

event ChangeTicketPriority<StaffTicketEvent
attributes
DEVELOPMENT OF THE CONCEPTUAL SCHEMA OF THE osTICKET SYSTEM BY APPLYING TDCM

newPriority::Priority
operations
effect()
end

context ChangeTicketPriority::effect()
post:
self.ticket.priority=self.newPriority
and
(InternalNote.allInstances- InternalNote.allInstances@pre)
->one(i | i.oclIsNew()
 and i.datetime[System.allInstances()-any(true).currentDateTime
 and i.subject='Ticket priority changed'
 and i.text='The ticket priority has been changed'
 and i.author=self.staffMember.firstName
 and i.ticket=self.ticket)
and self.ticket.lastMessageDateTime[System.allInstances()-any(true).currentDateTime

event MarkTicketOverdue<StaffTicketEvent
operations
effect()
end

context MarkTicketOverdue::effect()
post:
selticket.isOverdue
and
(InternalNote.allInstances- InternalNote.allInstances@pre)
->one(i | i.oclIsNew()
 and i.datetime[System.allInstances()-any(true).currentDateTime
 and i.subject='Ticket Marked Overdue'
 and i.text='Ticket flagged as overdue'
 and i.author=self.staffMember.firstName
 and i.ticket=self.ticket)
and self.ticket.lastMessageDateTime[System.allInstances()-any(true).currentDateTime

context MarkTicketOverdue ini inv staffMembersAnAdministrator:
sel.staffMember.isAdministrator

event AssignTicket<StaffTicketEvent
attributes
assignmentText::String
operations
effect()
end

association assignTicket_assignee between
AssignTicket[*] role assignTicketOfAssignee
StaffMember[1] role assignee
end

context AssignTicket::effect()
post:
let staffAlertsFromEMailAddress:String=
EmailSettings.allInstances()-any(true).defaultStaffAlertsEmail.address
in
self.ticket.assignedStaff=self.assignee
and
(InternalNote.allInstances- InternalNote.allInstances@pre)
->one(i | i.oclIsNew()
 and i.datetime[System.allInstances()-any(true).currentDateTime
 and i.subject='Ticket Reassigned'
 and i.text=self.assignmentText
 and i.author=self.staffMember.firstName
 and i.ticket=self.ticket)
and self.ticket.lastMessageDateTime=System.allInstances().any(true).currentDateTime
and (EMail.allInstances()->:exists(e|e.fromAddress=staffAlertsFromEMailAddress and e.toAddress=self.assignee.emailAddress and e.ticketNumber=self.ticket.number))

context AssignTicket ini inv assigneeIsNotInVacationMode:
    not(self.assignee.isInVacationMode)

event ReleaseTicket<StaffTicketEvent
operations
effect()
end

context ReleaseTicket::effect()
post:
    self.ticket.assignedStaff->isEmpty()
and (InternalNote.allInstances- InternalNote.allInstances@pre)
    ->one(i | i.oclIsNew() and i.datetime=System.allInstances().any(true).currentDateTime
    and i.subject='Ticket unassigned' and i.text='Released ticket' and i.author=self.staffMember.firstName and i.ticket=self.ticket)
context ReleaseTicket ini inv theTicketIsAssigned:
    self.ticket.assignedStaff.isDefined()

event EditTicket<StaffTicketEvent
attributes
    emailAddress: String
    fullName: String
    subject: String
    telephone: String[0..1]
    ext: String[0..1]
    priority: Priority
    dueDatetime: Datetime
editionInternalNote: String
operations
effect()
end

association editTicket_helpTopic between
    EditTicket[*]
    HelpTopic[1]
end

context EditTicket::effect()
post:
    self.ticket.email=self.emailAddress and
    self.ticket.fullName=self.fullName and
    self.ticket.subject=self.subject and
    self.ticket.telephone=self.telephone and
    self.ticket.ext=self.ext and
    self.ticket.priority=self.priority and
    self.ticket.helpTopic=self.helpTopic and
    self.ticket.dueDatetime=self.dueDatetime and
    (InternalNote.allInstances- InternalNote.allInstances@pre)
    ->one(i | i.oclIsNew() and i.datetime=System.allInstances().any(true).currentDateTime
and i.subject="Ticket updated"
and i.text=self.editionInternalNote
and i.author=self.staffMember.firstName
and i.ticket=self.ticket)

context EditTicket ini inv staffMemberIsNotAllowedToEditTickets:
  self.staffMember.staffGroup.canEditTickets or self.staffMember.isAdministrator

event PostTicketReply<StaffTicketEvent
attributes
  response:String
 operations
effect()
end

context PostTicketReply::effect()
post:
let sendNewMessageAlertToLastRespondent:Boolean=
  StaffNoticesAlertsSettings.allInstances()->any(true).alertWhenNewMessage and
  StaffNoticesAlertsSettings.allInstances()->any(true).alertWhenNewMessageStaff->includes(#LastRespondent)
in
let sendNewMessageAlertToAssignedStaff:Boolean=
  StaffNoticesAlertsSettings.allInstances()->any(true).alertWhenNewMessage and
  StaffNoticesAlertsSettings.allInstances()->any(true).alertWhenNewMessageStaff->includes(#AssignedStaff)
in
let sendNewMessageAlertToDepartmentManager:Boolean=
  StaffNoticesAlertsSettings.allInstances()->any(true).alertWhenNewMessage and
  StaffNoticesAlertsSettings.allInstances()->any(true).alertWhenNewMessageStaff->includes(#DepartmentManager)
in
let staffAlertsFromEMailAddress:String=
  EmailSettings.allInstances()->any(true).defaultStaffAlertsEmail.address
in
let sendAutoresponse:Boolean=
  if CustomerAutoresponsesSettings.allInstances()->any(true).autorespondWhenNewMessageAppendedToTicket then
    if self.ticket.helpTopic->notEmpty then
      if self.ticket.helpTopic.autoresponse=#Enabled then true
      else false
    else false
  endif
else false
endif
in
self.ticket.ticketThreadMessage->one(tdm | tdm.oclIsNew()
  and tdm.datetime=System.allInstances()->any(true).currentDateTime
  and tdm.text=self.response
  and tdm.author=self.staffMember.firstName
  and self.ticket.lastMessageDatetime=System.allInstances()->any(true).currentDateTime)
and self.ticket.lastRespondent=self.staffMember

and (sendAutoresponse implies
  EMail.allInstances()->exists(e|e.fromAddress=self.ticket.assignedDepartment.autoresponseEmail.address and
e.toAddress=self.ticket.email and
e.ticketNumber=self.ticket.number))

-- staff notices
and (sendNewMessageAlertToLastRespondent and
  self.ticket.lastRespondent@pre->notEmpty() and
  not(self.ticket.lastRespondent@pre.isInVacationMode or self.ticket.lastRespondent@pre.status=#Disabled)
implies
  EMail.allInstances()->exists(e|e.fromAddress=staffAlertsFromEMailAddress and
DEVELOPMENT OF THE CONCEPTUAL SCHEMA OF THE osTICKET SYSTEM BY APPLYING TDCM

Research report
April 2011

Department of Service and Information System Engineering

---

e.toAddress=self.ticket.lastRespondent@pre and
e.ticketNumber=self.ticket.number)

and (sendNewMessageAlertToAssignedStaff
and self.ticket.assignedStaff->notEmpty())
and (not(self.ticket.assignedStaff.isInVacationMode or self.ticket.assignedStaff.status=#Disabled)
implies
EMail.allInstances()->exists(e|e.fromAddress=staffAlertsFromEMailAddress and
e.toAddress=self.ticket.assignedStaff.emailAddress and
e.ticketNumber=self.ticket.number))

---

event PostTicketInternalNote<StaffTicketEvent
attributes
title: String
note: String
operations
effect()
end

custom PostTicketInternalNote::effect()
post:

let sendNewMessageAlertToLastRespondent:Boolean=
StaffNoticesAlertsSettings.allInstances()->any(true).alertWhenNewMessage and
StaffNoticesAlertsSettings.allInstances()->any(true).alertWhenNewMessageStaff->includes(#LastRespondent)
in

let sendNewMessageAlertToAssignedStaff:Boolean=
StaffNoticesAlertsSettings.allInstances()->any(true).alertWhenNewMessage and
StaffNoticesAlertsSettings.allInstances()->any(true).alertWhenNewMessageStaff->includes(#AssignedStaff)
in

let sendNewMessageAlertToDepartmentManager:Boolean=
StaffNoticesAlertsSettings.allInstances()->any(true).alertWhenNewMessage and
StaffNoticesAlertsSettings.allInstances()->any(true).alertWhenNewMessageStaff->includes(#DepartmentManager)
in

let staffAlertsFromEMailAddress:String=
EmailSettings.allInstances()->any(true).defaultStaffAlertsEmail.address
in

self.ticket.internalNote->one(int | int.oclIsNew() and int.datetime=System.allInstances()->any(true).currentDateTime and int.subject=self.title and int.text=self.note and int.author=self.staffMember.firstName and self.ticket.lastMessageDatetime=System.allInstances()->any(true).currentDateTime)

-- staff notices
and (sendNewMessageAlertToLastRespondent and
self.ticket.lastRespondent@pre->notEmpty() and
not(self.ticket.lastRespondent@pre.isInVacationMode or self.ticket.lastRespondent@pre.status=#Disabled)
implies
EMail.allInstances()->exists(e|e.fromAddress=staffAlertsFromEMailAddress and
e.toAddress=self.ticket.lastRespondent@pre and
e.ticketNumber=self.ticket.number))

and (sendNewMessageAlertToAssignedStaff
and self.ticket.assignedStaff->notEmpty() and
not(self.ticket.assignedStaff.isInVacationMode or self.ticket.assignedStaff.status=#Disabled)
implies
EMail.allInstances()->exists(e|e.fromAddress=staffAlertsFromEMailAddress and
e.toAddress=self.ticket.assignedStaff.emailAddress and
e.ticketNumber=self.ticket.number))

---
abstract event StaffTicketEvent
end

association staffTicketEvent_ticket between
  StaffTicketEvent[*]
  Ticket[1]
end

association staffTicketEvent_staffMember between
  StaffTicketEvent[*]
  StaffMember[1]
end

context StaffTicketEvent ini inv theTicketIsVisible:
  self.staffMember.isAdministrator or
  (self.staffMember.staffGroup.departmentsAccess->including(self.staffMember.department)->
   ->includes(self.ticket.assignedDepartment))

context StaffTicketEvent ini inv staffMemberIsLoggedIn:
  self.staffMember.isLoggedin

event TransferDepartment<StaffTicketEvent
attributes
  note: String
operations
effect()
end

association transferDepartment_department between
  TransferDepartment[*]
  Department[1]
end

context TransferDepartment::effect()
post:
  self.ticket.assignedDepartment=self.department and
  self.ticket.internalNote->one(int | int.oclIsNew()
    and int.datetime=System.allInstances()->any(true).currentDateTime
    and int.subject='Department transfer'
    and int.text=self.note
    and int.author=self.staffMember.firstName
    and self.ticket.lastMessageDatetime=System.allInstances()->any(true).currentDateTime)

context TransferDepartment ini inv departmentIsDifferent:
  self.department <> self.ticket.assignedDepartment

context TransferDepartment ini inv staffMemberIsAllowedToTransfer:
  self.staffMember.staffGroup.canTransferTickets

event CloseTicket<StaffTicketEvent
operations
effect()
end

context CloseTicket::effect()
post:
  self.ticket.ticketStatus=#Closed and
  self.ticket.internalNote->one(int | int.oclIsNew()
    and int.datetime=System.allInstances()->any(true).currentDateTime
    and int.subject='Ticket closed without response'
    and int.text='Ticket closed without response'
    and int.author=self.staffMember.firstName
    and self.ticket.lastMessageDatetime=System.allInstances()->any(true).currentDateTime)

context CloseTicket ini inv staffMemberIsAllowedToClose:
  self.staffMember.staffGroup.canCloseTickets
context CloseTicket ini inv ticketIsNotClosed:
    not (self.ticket.ticketStatus=#Closed)

event CloseTicketWithResponse<StaffTicketEvent
attributes
response:String
operations
effect()
end

closecontext CloseTicketWithResponse::effect()

post:
let sendAutoresponse:Boolean=
    if CustomerAutoresponsesSettings.allInstances()->any(true).autorespondWhenNewMessageAppendedToTicket then
        if self.ticket.helpTopic->notEmpty then
            if self.ticket.helpTopic.autoresponse=#Enabled then true
            else false
        endif
    else
        self.ticket.assignedDepartment.newAddedMessagesNotified
    endif
else false
endif
end

in
self.ticket.ticketStatus=#Closed and
self.ticket.internalNote->one(int | int.oclIsNew() and int.datetime=System.allInstances()->any(true).currentDateTime and int.subject='Ticket closed' and int.text='Ticket closed on reply' and int.author=self.staffMember.firstName and self.ticket.lastMessageDatetime=System.allInstances()->any(true).currentDateTime)
and
self.ticket.ticketThreadMessage->one(tdm | tdm.oclIsNew() and tdm.datetime=System.allInstances()->any(true).currentDateTime and tdm.text=self.response and tdm.author=self.staffMember.firstName and self.ticket.lastMessageDatetime=System.allInstances()->any(true).currentDateTime)
and
self.ticket.lastRespondent=self.staffMember

and (sendAutoresponse implies
    EMail.allInstances()->exists(e|e.fromAddress=self.ticket.assignedDepartment.autoresponseEmail.address and e.toAddress=self.ticket.email and e.ticketNumber=self.ticket.number))

context CloseTicketWithResponse ini inv staffMemberIsAllowedToClose:
    self.staffMember.staffGroup.canCloseTickets

context CloseTicketWithResponse ini inv ticketIsNotClosed:
not (self.ticket.ticketStatus=#Closed)

event ReopenTicket<StaffTicketEvent
operations
effect()
end

context ReopenTicket::effect()

post:
self.ticket.ticketStatus=#Open and
self.ticket.internalNote->one(int | int.oclIsNew() and int.datetime=System.allInstances()->any(true).currentDateTime and int.subject='Ticket reopened' and int.text='Ticket reopened without comments' and int.author=self.staffMember.firstName and self.ticket.lastMessageDatetime=System.allInstances()->any(true).currentDateTime)

context ReopenTicket ini inv ticketIsClosed:
    self.ticket.ticketStatus=#Closed
event ReopenTicketWithResponse<StaffTicketEvent
attributes
response: String
operations
effect()
end

class ReopenTicketWithResponse::effect():
post:
let sendAutoreponse:Boolean =
  if CustomerAutoresponsesSettings.allInstances()[->any(true)].autorespondWhenNewMessageAppendedToTicket then
    if self.ticket.helpTopic->notEmpty then
      if self.ticket.helpTopic.autoresponse=#Enabled then true
      else false
    endif
  else false
  endif
  self.ticket.assignedDepartment.newAddedMessageIsNotified
end
else false
endif
in
self.ticket.ticketStatus=#Open and
self.ticket.internalNote->one(int | int.oclIsNew())
  and int.datetime=System.allInstances()->any(true).currentDateTime
  and int.subject='Ticket status changed to open'
  and int.text='A staff member reopened the ticket on reply'
  and int.author=self.staffMember.firstName
and self.ticket.lastMessageDateTime=System.allInstances()->any(true).currentDateTime
and
self.ticket.ticketThreadMessage->one(tdm | tdm.oclIsNew())
  and tdm.datetime=System.allInstances()->any(true).currentDateTime
  and tdm.text=self.response
  and tdm.author=self.staffMember.firstName
and self.ticket.lastMessageDateTime=System.allInstances()->any(true).currentDateTime
and self.ticket.lastRespondent=self.staffMember

and (sendAutoreponse implies
  EMail.allInstances()[->exists(ele | ele.fromAddress=self.ticket.assignedDepartment.autoresponseEmail.address and
e.toAddress=self.ticket.email and
e.ticketNumber=self.ticket.number)])
context ReopenTicketWithResponse ini inv ticketsClosed:
self.ticket.ticketStatus=#Closed
context ReopenTicketWithResponse ini inv ticketsNotClosed:
not (self.ticket.ticketStatus=#Closed)

event BanEmailAndCloseTicket<StaffTicketEvent
operations
effect()
end

class BanEmailAndCloseTicket::effect():
post:
self.ticket.ticketStatus=#Closed and
EmailSettings.allInstances()[->any(true)].banList->includes(self.ticket.email) and
self.ticket.internalNote->one(int | int.oclIsNew())
  and int.datetime=System.allInstances()->any(true).currentDateTime
  and int.subject='Ticket closed'
  and int.text='Email added to banlist and ticket status set to closed'
  and int.author=self.staffMember.firstName
and self.ticket.lastMessageDateTime=System.allInstances()->any(true).currentDateTime
context BanEmailAndCloseTicket ini inv staffMemberIsAllowedToBanEmails:
self.staffMember.staffGroup.canBanEmails
context BanEmailAndCloseTicket ini inv ticketsNotClosed:
not (self.ticket.ticketStatus=#Closed)
event DeleteTicket
operations
effect()
end

association deleteTicket_ticket between
DeleteTicket[*]
Ticket[0..1]
end

association deleteTicket_staffMember between
DeleteTicket[*]
StaffMember[1]
end

context DeleteTicket::effect()
post:
Ticket.allInstances()->excludes(self.ticket@pre) and
self.ticket@pre.internalNote@pre->forAll(int|InternalNote.allInstances()->excludes(int))
end

context DeleteTicket ini inv theTicketIsVisible:
self.staffMember.isAdministrator or
(self.staffMember.staffGroup.departmentsAccess->including(self.staffMember.department)->>

context DeleteTicket ini inv staffMemberIsLoggedIn:
self.staffMember.isLoggedIn

context DeleteTicket ini inv staffMemberIsAllowedToDeleteTickets:
self.staffMember.isAdministrator or
self.staffMember.staffGroup.canDeleteTickets

event CheckOverdueTickets
operations
effect()
end

context CheckOverdueTickets::effect()
post:
let sendOverdueTicketAlertToAdministrator:Boolean=
StaffNoticesAlertsSettings.allInstances()->any(true).alertWhenTicketOverdue and
StaffNoticesAlertsSettings.allInstances()->any(true).alertWhenTicketOverdueStaff->includes(#Administrator)
in
let sendOverdueTicketAlertToDepartmentManager:Boolean=
StaffNoticesAlertsSettings.allInstances()->any(true).alertWhenTicketOverdue and
StaffNoticesAlertsSettings.allInstances()->any(true).alertWhenTicketOverdueStaff->includes(#DepartmentManager)
in
let sendOverdueTicketAlertToDepartmentMembers:Boolean=
StaffNoticesAlertsSettings.allInstances()->any(true).alertWhenTicketOverdue and
 StaffNoticesAlertsSettings.allInstances()->any(true).alertWhenTicketOverdueStaff->includes(#DepartmentMembers)
in
let staffAlertsFromEmailAddress:String=
EmailSettings.allInstances()->any(true).defaultStaffAlertsEmail.address
in
Ticket.allInstances()->any(true).currentDateTime.value>t.dueDateTime.value+TicketSettings.allInstances()->any(true).ticketGracePeriod)
   and not(t.isOverdue)
  -> forAll(t.isOverdue)

-- staff notices
and (sendOverdueTicketAlertToAdministrator implies
EMAIL.allInstances()->exists(e|exists(e.addressStaffAlertsFromEmailAddress and
   e.toAddress>EmailSettings.allInstances()->any(true).administrationEmail and
   e.ticketNumber=t.number))
and (sendOverdueTicketAlertToDepartmentManager
   and t.assignedDepartment.departmentManager->notEmpty()
   and t.assignedDepartment.departmentManager.status=#Enabled)
and not(t.assignedDepartment.departmentManager.isInVacationMode)
implies
EMail.allInstances()->exists(e|e.fromAddress=staffAlertsFromEMailAddress and
  e.toAddress=t.assignedDepartment.departmentManager.emailAddress and
  e.ticketNumber=t.number)

and (sendOverdueTicketAlertToDepartmentMembers
implies
t.assignedDepartment.staffMember->forAll(m|
  m.status=#Enabled and not(m.isInVacationMode))
implies
EMail.allInstances()->exists(e|e.fromAddress=staffAlertsFromEMailAddress and
  e.toAddress=m.emailAddress and
  e.ticketNumber=t.number)))

}
6. Case study analysis

In this section, we analyze the experimentation reported in this document. We analyze the properties of the resultant conceptual schema, the testing effort, the kinds of errors and failures, and the characteristics of the TDCM iterations performed in order to develop the conceptual schema of the osTicket system.

6.1. The resultant conceptual schema

The resultant conceptual schema is the schema obtained in the last iteration as a result of the evolution of the schema by applying TDMC. You can download a zipped file with the CSTLProcessor and the case study files (the resultant conceptual schema in the USEx executable format, the methods file, and the CSTL test programs).

Table 1 summarizes the number of schema elements that constitute the resultant conceptual schema obtained by applying TDCM in the osTicket case study:

<table>
<thead>
<tr>
<th>osTicket Conceptual Schema</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Classes</td>
<td>28</td>
</tr>
<tr>
<td>Attributes</td>
<td>92</td>
</tr>
<tr>
<td>Associations</td>
<td>44</td>
</tr>
<tr>
<td>Event types</td>
<td>24</td>
</tr>
<tr>
<td>Integrity constraints</td>
<td>51</td>
</tr>
</tbody>
</table>

Table 1. Schema elements of the osTicket Conceptual schema
Quality properties of the resultant conceptual schema

The resultant conceptual schema is correct according to the expectations formalized in the processed test cases (the knowledge included in the conceptual schema fulfills the expectations formalized as test case assertions).

The resultant schema is also complete according to the test set, because the knowledge it contains makes possible the test set execution.

However, more user stories could be designed and, consequently, more test cases could be specified in order to increase our confidence about the correctness and the completeness, by testing the schema in more representative cases. This is a drawback inherent to all the testing processes, because the number of possible test cases is infinite. In this case study, we learned that it is very important to specify the test cases based on a representative set of user stories according to a planned testing strategy.

All knowledge defined in the resultant conceptual schema is relevant. The passing test set and its associated conceptual schema are not enough to assert the relevance of the schema (the defined knowledge is correct and necessary but the schema could contain irrelevant knowledge that does not alter the verdict of the test cases). However, the CSTL processor allows to automatically check whether the basic types, derived types, valid type configurations or domain event types are participants of any test case or not. If all the elements of the schema are needed in at least one correct test case, it is because the defined knowledge is relevant for the system.

---

**Fig. 3.** Test case execution report provided by the CSTL processor
Figure 4 shows the coverage analysis report provided by the CSTL Processor at the end of the last iteration. It allows us to ensure the relevance of the knowledge defined in the resultant conceptual schema.

6.2. The test set

The conceptual schema was obtained from an empty schema which was evolved in 100 iterations in which the stories defined in Section 2 were processed as test cases. Table 2 summarizes some data about the test set processed in this case study.

<table>
<thead>
<tr>
<th>osTicket Test Set</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of test cases</td>
</tr>
<tr>
<td>Lines of test cases</td>
</tr>
<tr>
<td>Fixture components</td>
</tr>
</tbody>
</table>

Table 2. Test set summary

We finished the TDCM application when two conditions hold: 1) we formalized as test cases all the representative stories according to our testing strategy, 2) the verdict of all the test cases became Pass.
6.3. TDCM iterations

The conceptual schema developed in this case study is the result of 100 TDCM iterations. This figure has no particular significance in this case study. There was not any intention for a particular number of iterations. In the following, we categorize the errors and failures fixed by applying TDCM and the changes they drove to evolve the schema.

Errors/failures categorization

In this case study we have categorized the errors and failures which may be obtained during the execution of test cases by applying TDCM. Neither syntactical errors nor incorrectly formalized expectations in test cases are considered in this table.

Figure 5 summarizes the categorization of errors/failures which has been used and refined. We also suggest the applicable changes to fix each error/failure type.

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
<th>Suggested changes to the schema to fix the error/failure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rel_BT</td>
<td>An expected relevant base type (entity type or relationship type) is not specified in the conceptual schema</td>
<td>Add_Rel_BT</td>
</tr>
<tr>
<td>Rel_DT</td>
<td>An expected relevant derived type is not specified in the conceptual schema</td>
<td>Add_Rel_DT</td>
</tr>
<tr>
<td>Rel_ET</td>
<td>An expected relevant event type (domain event or query) is not specified in the conceptual schema</td>
<td>Add_Rel_ET</td>
</tr>
<tr>
<td>EvOc_bef</td>
<td>The IB state before an expected event occurrence is inconsistent (the event specification is invalid)</td>
<td>Some (too restrictive) static constraints or preconditions are updated Chg_constraint Chg_event_specification</td>
</tr>
<tr>
<td>EvOc_after</td>
<td>The IB state after an expected occurrence of an event is inconsistent (the event specification is invalid)</td>
<td>The event postcondition, the event method or an static constraint are updated. Chg_event_specification</td>
</tr>
<tr>
<td>EvOc_post</td>
<td>The postcondition is not satisfied after an expected event occurrence</td>
<td>Either the method or the postcondition are updated.      Chg_event_specification</td>
</tr>
<tr>
<td>Sem_exp</td>
<td>An OCL expression in a test case or in the conceptual schema are not valid or inconsistent (e.g. invalid operations for specific types)</td>
<td>Either the expression in the test case is corrected or an element of the schema needs to be changed according to the semantic error revealed. Chg_element_type Chg_exp</td>
</tr>
</tbody>
</table>
Table 3. Errors and failures categorization

This categorization and their associated actions may be useful guidelines to help making progress in TDCM more efficiently. In TDCM, errors and failures to be fixed are the essence for progress. When errors/failures are revealed, then the modeler may use this table to find out applicable actions to change the schema in order to fix each error/failure.

In the case study, the errors/failures which drive the changes in each iteration are reported and classified using this categorization.

Errors and failures that drive conceptual modeling in the case study

Chart 1. Errors and failures revealed (percentages)
In charts 1 and 2, we analyze the errors and failures revealed by applying TDCM to the osTicket case study, according to the categorization described in the previous section. Table 4 summarizes the errors reported by the CSTLProcessor during the TDCM iterations.
We observe that TDCM drives the development of the conceptual schema by promoting to fix three main kinds of errors/failures:

- 58.1% of the errors/failures correspond to relevant types (basic types, derived types or event types) which have not been defined yet in the schema (Rel_BT+Rel_DT+Rel_ET). Rel_BT, Rel_DDT and Rel_ET are proportional to the relevant knowledge defined in the schema.

- 17.5% of the errors/failures correspond to erroneous definitions of domain event types (EvOc_bef+EvOc_after+EvOc_post), either because the state before the occurrence is inconsistent, or because the state after the occurrence is inconsistent, or because the postcondition is not satisfied.

- The rest of the errors correspond to unexpected results (assertions that fail). Most of them are assertions about the non-occurrence of events (11%), and about the IB state (9.7%). Others are assertions that check the consistency of an IB state (2%) –this assertions are only applied in the basics&configuration are of knowledge, where the implicit structural events are only considered–.

In this case study, some iterations have also been driven by other semantic errors in OCL expressions, such as incompatible types, invalid operations for some types, etc.

![Chart 3. Changes to the conceptual schema while applying TDCM (percentages)](chart)

The errors and failures which are reported by the CSTLProcessor in each iteration need to be fixed according to the TDCM cycle. Changing the schema to fix the errors/failures makes progress in the incremental development of the schema. By analyzing the kinds of actions applied to fix the previously analyzed errors/failures, we observe:

- Fixing the errors about missing relevant knowledge is almost trivial: they need to be added. Note that the percentage of the changes Add_Rel_BT (52.4%), Add_Rel_DT (0.5%) and Add_Rel_ET (5.2%) are exactly the same as the errors revealed due to missing relevant knowledge in the conceptual schema.
- 36.9% of changes correspond to the refinement of event specifications (precondition, postcondition, method) in order to be correctly defined according to the general definition of domain events.

- 3.2% of the changes correspond to the addition/refinement of static constraints of the schema. These changes are usually induced by invalid IB states when an event occurrence is asserted or when the assertion about the consistency of the IB state fails.

- Semantic errors in expressions reveal inconsistencies in the schema. They need to be corrected either by changing the expression or changing the type of an element of the schema in order to make possible the evaluation of the expression. In this case study, all of these errors have been corrected by changing the type of a schema element (1.7%).

![Chart 4. Number of changes to the conceptual schema while applying TDCM](image)

It is important to note that these results point out a tendency about the most common errors/failures and their induced changes by analyzing the application of TDCM in a concrete case study. However, the kind of errors/failures revealed and the changes driven by TDCM also depend on the knowledge of the universe of discourse of the system for which we develop the conceptual schema.

### 6.4. Iterations analysis

In this section, we analyze and compare the 100 iterations (we name them as It1, It2, ..., It100) that have been performed by applying TDCM to this case study. A complete report about each iteration is described in Section 3.
Test cases specification

Chart 5. Time invest on testing specification in each iteration

Chart 6. Lines of test cases added/updated in each iteration
As we can observe in first bar chart 6, the numbers of lines of test cases added to the test set vary in each iteration depending on the processed story.

If we compare bar chart 6 (lines of test cases added) with the time spent by specifying the test cases in each iteration (bar chart 5), we can observe that the testing specification time is not directly proportional to the lines of test cases added in all cases.

Chart 7 represents the testing specification productivity (lines of test added/minute). We observe that, in general, the productivity tends to vary cyclically (the productivity increases periodically). If we analyze the iterations, we may observe that test cases may be grouped into similar stories (e.g. stories which are tested with variations or using different initial states or conditions). The first time we specify a story with very different testing objectives, the testing specification productivity decreases, but when we specify story variations as test cases, then the productivity increases.

We realize that there are peaks of productivity in the iterations when previously used testing structures are reused. In contrast, the testing specification consumes more time when we specify stories with new (and probably unknown) structures.
TDCM iterations productivity

Bar chart 8 analyzes the development time used to complete each TDCM iteration and the changes applied to the schema due to the fixing actions induced by the revealed errors/failures.

We can observe that most of the iterations are productive because they drive changes in the schema. Moreover, in many iterations, more development time implies more changes to the schema induced by errors and failures revealed by TDCM. This is also important to note, that some iterations have an insignificant development time and no changes are done to the schema. It means that the verdict of the test set is Pass from the first execution and, consequently, the iteration does not make progress in the TDCM cycle. Nevertheless, these iterations increase our confidence about the validity of the schema.

The time spent in a TDCM iteration is the time to fix the errors/failures (that is to evolve the schema). At the end of each iteration, we obtain an executable conceptual schema with a test set that validates its correctness and completeness.

We have also automatically analyzed, at the end of each iteration, the basic types, the derived types and the domain event types which have been tested in at least one of the passing test cases of the current test set. This analysis provides a measure of the basic coverage of the test set and allows us to identify elements in the schema that have not been tested.
When applying TDCM in this case study, we realized that, at the end of each iteration, the coverage was 100% (all the elements of the schema are participants of a valid and passing test case). The coverage report shown in Fig. 4 was obtained once each iteration was finished. Therefore, for all the elements of the schema, at the end of each iteration, its relevance was justified by the test set.

The previous accumulated bar chart represents the total time spent in each iteration. In most iterations, the time spent in the TDCM development (fixing errors and failures) is greater than the time spent in fixing errors/failures and changing the schema. It means that in most of the iterations, the testing specification time worth the while because the test case encourages and drives the evolution of the conceptual schema.

Again, the exceptions are those iterations which does not evolve the schema (although they increase confidence about the correctness of the schema). In these iterations, the TDCM iteration time is insignificant, because no changes are done in the schema (these are iterations that pass in the first execution). In these iterations, the time spent by designing and specifying the test case is higher in comparison with the TDCM iteration time spent.
Errors and failures evolution

Chart 10. Errors/failures revealed during TDCM application

Chart 11. Changes to the conceptual schema during TDCM application
Finally, the two previous charts allows us to analyze the distribution of the errors/failures and the changes which have been performed to fix them while TDCM is applied.

We can observe that, in first iterations, the main errors found correspond to relevant types which are not in the schema. These iterations correspond to the processing of test cases about the basics and configuration of the system. The action to solve them is trivial for this kind of errors: It is required to add them to the conceptual schema. Once the main static schema elements are specified in the schema, the type of failures and errors that drive the schema evolution change significantly. As we add the first domain events, we detect inconsistent states that require refining static constraints and correctly specifying the effect of these events, but only some static knowledge is required to be added.

Charts 10 and 11 also support that not all kinds of knowledge require the same effort to be evolved or corrected according to the processed test cases. In the first iterations, the number of errors/failures is greater because we basically add relevant knowledge (although the time spent in fixing these errors is lower than the time invest on fixing the errors and failures of the next iterations). After the basic static knowledge is added (which is necessary to support the execution of the following test cases), we process stories which are sequences of event occurrences (like a common user story). When we specify the effect of the events and we make assertions about the IB states reached by the events, the required effort is greater because it is less evident how to change the schema in order to reach the verdict Pass.

**Iterations summary**

Table 5 summarizes some aggregated data related to the TDCM iterations applied in the development of the osTicket conceptual schema.

<table>
<thead>
<tr>
<th>TDCM iterations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of iterations</td>
</tr>
<tr>
<td>Total development time of the iterations</td>
</tr>
<tr>
<td>Total time to specify test cases</td>
</tr>
<tr>
<td>Total time to evolve the conceptual schema under development</td>
</tr>
<tr>
<td>Average of lines per test case</td>
</tr>
<tr>
<td>Average of testing specification time per iteration</td>
</tr>
<tr>
<td>Average of conceptual schema development time per iteration</td>
</tr>
<tr>
<td>Average of changes to the conceptual schema per iteration</td>
</tr>
</tbody>
</table>

*Table 5. Iterations summary*
7. Conclusions

- We have applied Test-Driven Conceptual Modeling for the reverse engineering development of the conceptual schema of a widely-used and real-world ticket support system.

- After the application of TDCM in this case study, we obtained an executable conceptual schema with a test set that checks the correctness and the completeness of the schema according to the expectations formalized as tests. In other words, TDCM iterations drive the evolution of the conceptual schema and continuously perform its validation.

- The time spent to specify the test cases in each iteration varies depending on the formalized stories. However, we observe that testing structures are reused and, therefore, the testing specification productivity tends to increase as we make progress in TDCM.

- The time spent in the conceptual schema evolution (by fixing errors/failures) is greater than the time used to specify the test cases. Therefore, most of the test cases are productive because they lead to make progress in the evolution of the schema. The exception are those iterations that pass in the first execution (they increase our confidence about the validity of the schema, but they do not drive changes).

- At the end of each TDCM iteration, the basic coverage of the elements of the schema is 100%. It means that, at the end of each iteration, for all the elements of the schema, its relevance is justified and, in at least one case, its correctness has been tested.

- The most common errors/failures revealed correspond to missing relevant types, to invalid definitions of domain event types and to failing assertions (either due to incorrect domain event effects, invalid static constraints or incorrect derivation rules).
In first iterations, the most common errors are about missing relevant types (which are necessary to build IB states). After that, the most common errors/failures are about the correct definition of domain event types and the correctness (according to the assertions) of the reached IB states.

The time spent on fixing errors/failures is not proportional in all cases to the number of errors fixed in each iteration. It means that not all errors/failures require the same effort to be fixed. The analysis suggests that missing relevant types are trivial to be fixed (they need to be added). However, the changes to fix failing assertions about the state of the domain or incorrect domain event specifications may require different actions such as changing derivation rules, integrity constraints or the precondition and postcondition of the effect of the event.

We have identified and categorized the errors and failures that may be revealed and the associated changes in the schema that may be applied to fix them. This categorization may help in the application of TDCM.
8. References


Appendix A. USE specification of the resultant schema

```plaintext
model osTicket
  enum DepartmentType {Public, Private} --it2
  enum Priority {Low, Normal, High}
  enum Status {Enabled, Disabled}
  enum HelpDeskStatus {Online, Offline} --it7
  enum TicketsMode {Sequential, Random}
  enum StaffRole {Administrator, DepartmentManager, DepartmentMembers, LastRespondent, AssignedStaff}
  enum TicketStatus {Open, Closed}
  enum TicketSource {Web, EMail, Phone, Other}
  enum StatusFilter {OpenTickets, ClosedTickets, AssignedToMe, OverdueTickets}

  class Datetime
    attributes
    value:Integer
  end

  class System
    attributes
    currentDateTime: Datetime
    aleat: Integer
  end

  class EMail
    attributes
    /timeStamp: Datetime = System.allInstances()->any(true).currentDateTime constant
    fromAddress: String
    toAddress: String
    ticketNumber: Integer
  end

  association eMail_emailKind between EMail[*] EMailKind[1]
end

class EmailTemplate
  attributes
  name: String
  internalNotes: String
```

DEVELOPMENT OF THE CONCEPTUAL SCHEMA OF THE osTICKET SYSTEM BY APPLYING TDCM

class EmailKind
attributes
subject: String
message: String
end

class NewTicketAutoresponse<EmailKind
end

association newTicketAutoresponse_emailTemplate between
NewTicketAutoresponse[1]
   EmailTemplate[1]
end

class NewMessageAutoresponse<EmailKind
end

association newMessageAutoresponse_emailTemplate between
NewMessageAutoresponse[1]
   EmailTemplate[1]
end

class NewTicketNotice<EmailKind
end

association newTicketNotice_emailTemplate between
NewTicketNotice[1]
   EmailTemplate[1]
end

class OverTicketLimitNotice<EmailKind
end

association overTicketLimitNotice_emailTemplate between
OverTicketLimitNotice[1]
   EmailTemplate[1]
end

class TicketResponseNotice<EmailKind
end

association ticketResponseNotice_emailTemplate between
TicketResponseNotice[1]
   EmailTemplate[1]
end

class NewTicketAlertToStaff<EmailKind
end

association newTicketAlertToStaff_emailTemplate between
NewTicketAlertToStaff[1]
   EmailTemplate[1]
end

class NewMessageAlertToStaff<EmailKind
end

association newMessageAlertToStaff_emailTemplate between
NewMessageAlertToStaff[1]
   EmailTemplate[1]
end

class NewInternalNoteAlertToStaff<EmailKind
end

association newInternalNoteAlertToStaff_emailTemplate between
NewInternalNoteAlertToStaff[1]
   EmailTemplate[1]
end
class TicketAssignedAlertToStaff<EmailKind
end

association ticketAssignedAlertToStaff_emailTemplate between
    TicketAssignedAlertToStaff[1]
    EmailTemplate[1]
end

class OverdueTicketAlertToStaff<EmailKind
end

association overdueTicketAlertToStaff_emailTemplate between
    OverdueTicketAlertToStaff[1]
    EmailTemplate[1]
end

---it2

class Department
attributes
    name: String
    type: DepartmentType
    newTicketAutoresponseIsSent: Boolean
    newAddedMessagesIsNotified: Boolean
    isDefault: Boolean=false
end

class Department
context Department inv hasAlwaysOneDefault:
    Department.allInstances() -> select(d|d.isDefault) -> size()==1

association department_emailTemplate between
    Department[*] role departmentOfEmailTemplate
    EmailTemplate[1]
end

association department_departmentManager between
    Department[*] role departmentOfManager
    StaffMember[0..1] role departmentManager
end

association departmentAutoresponseEmail between
    Department[*] role departmentOfAutoresponseEmail
    EmailAccount[1] role autoresponseEmail
end

association departmentOutgoingEmail between
    Department[*] role departmentOutgoingEmail
    EmailAccount[1] role outgoingEmail
end

class EmailAccount
attributes
    address: String
    fromName: String
    defaultNewPriority: Priority
    autoresponsesStatus: Status
end

association EmailAccount_defaultNewTicketDepartment between
    EmailAccount[*]
    Department[1] role defaultNewTicketDepartment
end

class StaffMember
attributes
    username: String
    firstName: String
    lastName: String
    emailAddress: String
DEVELOPMENT OF THE CONCEPTUAL
SCHEMA OF THE osTiCKET
SYSTEM BY APPLYING TDCM

Research report
April 2011

officPhone:String[0..1]
phoneExtension: String [0..1]
mobilePhone: String[0..1]
signature: String[0..1]
password: String
status: Status
isAdministrator: Boolean
isInVacationMode: Boolean
isLoggedIn: Boolean=false
end

association staffMember_department between
StaffMember[*]
   Department[1]
end

association staffMember_staffGroup between
StaffMember[*]
   StaffGroup[1]
end

class StaffGroup
attributes
   name: String
   status: Status
   canCreateTickets: Boolean
   canEditTickets: Boolean
   canCloseTickets: Boolean
   canTransferTickets: Boolean
   canDeleteTickets: Boolean
   canBanEmails: Boolean
end

association staffGroup_departmentsAccess between
   StaffGroup[*]
   Department[*] role departmentsAccess
end

---it4
class EmailSettings
attributes
   administrationEmail: String
   banList: String[*]
end

association emailSettings_emailAccount between
   EmailSettings[*] role emailSettingsOfDefaultSystemEmail
   EmailAccount[1] role defaultSystemEmail
end

association emailSettings_defaultStaffAlertsEmail between
   EmailSettings[*] role emailSettingsOfDefaultStaffAlertsEmail
   EmailAccount[1] role defaultStaffAlertsEmail
end

context EmailSettings inv hasOnlyOneInstance:
   EmailSettings.allInstances().size()=1

---it5
class HelpTopic
attributes
   name: String
   status: Status
   autoresponse: Status
   newTicketPriority: Priority
end

association helpTopic_newTicketDepartment between
   HelpTopic[*]
   Department[1] role newTicketDepartment
DEVELOPMENT OF THE CONCEPTUAL
SCHEMA OF THE osTICKET
SYSTEM BY APPLYING TDCM

April 2011

Research report

Department of Service and Information System Engineering
DEVELOPMENT OF THE CONCEPTUAL SCHEMA OF THE osTICKET SYSTEM BY APPLYING TDCM

telephone: String [0..1]
ext: String [0..1]
subject: String
text: String [0..1]
message: String
priority: Priority
source: TicketSource
creationDateTime: Datetime -> any(true).currentDateTime constant
dueDateTime: Datetime [0..1]
lastResponseDateTime: Datetime [0..1]
lastMessageDateTime: Datetime [0..1]
number: Integer = if TicketSettings.allInstances()->any(true).mode = #Sequential then
  Ticket.allInstances()->size()
else System.allInstances()->aleat
endif constant
isOverdue: Boolean = false

operations

end

association ticket_helpTopic between
  Ticket[*]
  HelpTopic[0..1]
end

association ticket_assignedStaff between
  Ticket[*]
  StaffMember[0..1] role assignedStaff
end

class TicketThreadMessage

attributes

datetime: Datetime
text: String
author: String

end
class InternalNote

attributes

datetime: Datetime
subject: String
text: String
author: String

end

association ticket_internalNote between
  Ticket[1]
  InternalNote[*]
end

association ticket_ticketThreadMessage between
  Ticket[1]
  TicketThreadMessage[*]
end

association ticket_assignedDepartment between
  Ticket[*]
  Department[1] role assignedDepartment
end

association ticket_lastRespondent between
  Ticket[*]
  role ticketOfLastRespondent
  StaffMember[0..1] role lastRespondent
end

event NewTicketOnline

attributes

fullName: String
eMail: String
telephone: String [0..1]

ext: String[0..1]
subject: String
message: String
priority: Priority[0..1]
createdTicket: Ticket[0..1]

operations
effect()
end

context NewTicketOnline ini inv priorityMayBeSetWhenAllowed:
  if TicketSettings.allInstances()->select(hp|hp.status=#Enabled)->size()>0 then
    self.priority.isDefined()
  else
    self.priority.isUndefined()
  endif
end

context NewTicketOnline ini inv emailIsNotInBanList:
  EmailSettings.allInstances()->any(true).banList->notEmpty() implies
  EmailSettings.allInstances()->any(true).banList->excludes(self.email)
end

association newTicketOnline_helpTopic between
  NewTicketOnline[*]
  HelpTopic[0..1]
end

context NewTicketOnline ini inv helpTopicSpecifiedIfAvailable:
  if HelpTopic.allInstances()->select(hp|hp.status=#Enabled)->size()>0 then
    self.helpTopic->size()=1
  else
    self.helpTopic->size()=0
  endif
end

context NewTicketOnline ini inv helpDeskStatusIsOnline:
  GeneralSettings.allInstances()->any(true).status=#Online
end

context NewTicketOnline ini inv maximumOpenTicketsLimitIsNotViolated:
  if TicketSettings.allInstances()->any(true).openTicketsPerMailAreLimited then
    Ticket.allInstances()->select(tt|tt.email=self.email)->size()<TicketSettings.allInstances()->any(true).maximumOpenTicketsPerMail
  else
    true
  endif
end

context NewTicketOnline::effect()
post:
  let defaultPriority: Priority=
    if self.helpTopic->notEmpty() then
      self.helpTopic.newTicketPriority
    else
      TicketSettings.allInstances()->any(true).priority
    endif
  in
  let assignedPriority: Priority=
    if TicketSettings.allInstances()->any(true).customersCanChangePriority then
      self.priority
    else
      defaultPriority
    endif
  in
  let defaultDepartment: Department=
    if self.helpTopic->notEmpty() then
      self.helpTopic.newTicketDepartment
    else
      Department.allInstances()->any(dcl|dcl.isDefault)
    endif
  in
  let sendNewTicketAlertToAdministrator: Boolean=
    StaffNoticesAlertsSettings.allInstances()->any(true).alertWhenNewTicketCreated and
    StaffNoticesAlertsSettings.allInstances()->any(true).alertWhenNewTicketCreatedStaff->includes(#Administrator)
DEVELOPMENT OF THE CONCEPTUAL SCHEMA OF THE osTICKET SYSTEM BY APPLYING TDCM

April 2011

Research report

in

let sendNewTicketAlertToDepartmentManager:Boolean=
StaffNoticesAlertsSettings.allInstances()->any(true).alertWhenNewTicketCreatedStaff->includes(#DepartmentManager)

in

let sendNewTicketAlertToDepartmentMembers:Boolean=
StaffNoticesAlertsSettings.allInstances()->any(true).alertWhenNewTicketCreatedStaff->includes(#DepartmentMembers)

in

let staffAlertsFromEMailAddress:String=
EmailSettings.allInstances()->any(true).defaultStaffAlertsEmail.address

in

let sendAutoresponse:Boolean=
if CustomerAutoresponsesSettings.allInstances()->any(true).autorespondWhenNewTicketCreatedByCustomer then
if self.helpTopic->notEmpty then
if self.helpTopic.autoresponse=#Enabled then true
else false
endif
else
defaultDepartment.newTicketAutoresponseIsSent
endif
else false
endif

in

(Ticket.allInstances- Ticket.allInstances@pre)
->one(t | t.oclIsNew() and self.createdTicket=t and t.fullName=self.fullName and t.email=self.email and t.telephone=self.telephone and t.subject=self.subject and t.message=self.message and t.ticketStatus=#Open and t.priority=assignedPriority and t.source=#Web and t.helpTopic=self.helpTopic and t.assignedDepartment=defaultDepartment and (TicketThreadMessage.allInstances- TicketThreadMessage.allInstances@pre)
->one(tdm | tdm.oclIsNew() and tdm.datetime=System.allInstances()->any(true).currentDateTime and tdm.text=self.message and tdm.author=self.fullName and tdm.ticket=t and t.lastMessageDatetime=System.allInstances()->any(true).currentDateTime)
-- autoresponses
and (sendAutoresponse implies
EMail.allInstances()-exists(ele | ele.fromAddress=t.assignedDepartment.autoresponseEmail.address and e.toAddress=t.email and e.ticketNumber=t.number))

-- staff notices
and (sendNewTicketAlertToAdministrator implies
EMail.allInstances()-exists(ele | ele.fromAddress=staffAlertsFromEMailAddress and e.toAddress=EmailSettings.allInstances()->any(true).administrationEmail and e.ticketNumber=t.number))

and (sendNewTicketAlertToDepartmentManager
and t.assignedDepartment.departmentManager->notEmpty() and t.assignedDepartment.departmentManager.status=#Enabled and not(t.assignedDepartment.departmentManager.isInVacationMode)
implies
EMail.allInstances()-exists(ele | ele.fromAddress=staffAlertsFromEMailAddress and e.toAddress=t.assignedDepartment.departmentManager.emailAddress and e.ticketNumber=t.number))

and (sendNewTicketAlertToDepartmentMembers
implies
(t.assignedDepartment.staffMember->forAll(m | (m.status=#Enabled and not(m.isInVacationMode))))
implies
  EMail.allInstances()->exists(e|e.fromAddress=staffAlertsFromEMailAddress and
e.toAddress=m.emailAddress and
e.ticketNumber=t.number))
)

event NewTicketByEmail
attributes
toAddress:String
fromName:String
fromAddress:String
subject:String
message:String
createdTicket:Ticket[0..1]
operations
effect()
end

class NewTicketByEmail
ini
inv
 null

context NewTicketByEmail
ini
inv
emailIsNotInBanList:
  EmailSettings.allInstances()->any(true).banList->notEmpty() implies
    EmailSettings.allInstances()->any(true).banList->excludes(self.fromAddress)

context NewTicketByEmail::effect()
post:
  let incomingEmailAccount:EmailAccount=
    EmailAccount.allInstances()->any(e|e.address=self.toAddress)
in
let assignedPriority:Priority=
  if TicketSettings.allInstances()->any(true).useEmailPriorityWhenAvailable then
    incomingEmailAccount.defaultNewPriority
  else
    TicketSettings.allInstances()->any(true).priority
  endif
in
let defaultDepartment:Department=
  incomingEmailAccount.defaultNewTicketDepartment
in
let sendNewTicketAlertToAdministrator:Boolean=
  StaffNoticesAlertsSettings.allInstances()->any(true).alertWhenNewTicketCreated and
  StaffNoticesAlertsSettings.allInstances()->any(true).alertWhenNewTicketCreatedStaff->includes(#Administrator)
in
let sendNewTicketAlertToDepartmentManager:Boolean=
  StaffNoticesAlertsSettings.allInstances()->any(true).alertWhenNewTicketCreated and
  StaffNoticesAlertsSettings.allInstances()->any(true).alertWhenNewTicketCreatedStaff->includes(#DepartmentManager)
in
let sendNewTicketAlertToDepartmentMembers:Boolean=
  StaffNoticesAlertsSettings.allInstances()->any(true).alertWhenNewTicketCreated and
  StaffNoticesAlertsSettings.allInstances()->any(true).alertWhenNewTicketCreatedStaff->includes(#DepartmentMembers)
in
let staffAlertsFromEMailAddress:String=
  EmailSettings.allInstances()->any(true).defaultStaffAlertsEmail.address
in
let sendAutoresponse:Boolean=
  if (incomingEmailAccount.autoresponsesStatus=#Enabled) then true
  else false
  endif
in
  (Ticket.allInstances- Ticket.allInstances@pre)
  ->one(t | t.oclIsNew() and self.createdTicket=t
and self.assignedDepartment=incomingEmailAccount.defaultNewTicketDepartment
and (TicketThreadMessage.allInstances- TicketThreadMessage.allInstances@pre)
  ->one(tdm | tdm.oclIsNew() and tdm.datetime=System.allInstances()->any(true).currentDateTime
and tdm.text=self.message
and tdm.author=self.fromName
and tdm.ticket=t
and t.lastMessageDateTime=System.allInstances()->any(true).currentDateTime)
context NewTicketByEmail
  ini
  maxmimumOpenTicketsLimitsNotViolated:
  if TicketSettings.allInstances()->any(true).openTicketsPerMailAreLimited then
    if Ticket.allInstances->select(tt.email=self.fromAddress)->size()<TicketSettings.allInstances()->any(true).maximumOpenTicketsPerMail
      else true
    endif
  end

context NewTicketByEmail
  ini
  theIncomingEmailIsValid:
  EmailAccount.allInstances()->one(address=self.toAddress)

context NewTicketByEmail
  ini
  helpDeskStatusIsOnline:
  GeneralSettings.allInstances()->any(true).status=#Online

context NewTicketsAssociatedToEmail
  ini
  thereAreTicketsAssociatedToTheEmail:
  Ticket.allInstances()->select(t|t.email=self.email)->size()>0

context NewTicketsAssociatedToEmail
  ini
  accessDataIsValid:
  Ticket.allInstances()->select(t|t.email=self.email).number->includes(self.ticketNumber)

event DisplayTicketsAssociatedToEmail
  attributes
  email: String
  ticketNumber: Integer
  operations
  answer(): Set(Tuple(number: Integer, createDate: Integer, status: String, department: String, email: String)) =
    Ticket.allInstances
    -> sortedBy(number)
    -> collect (t | Tuple {number : t.number, createDate: t.creationDatetime.value, status: t.ticketStatus, subject: t.subject, department: t.assignedDepartment.name, email: t.email})
  effect()
end

class DisplayTicketsAssociatedToEmail
end

class ReplyTicketByCustomer
end

context ReplyTicketByCustomer::effect()
  post:
  let sendNewMessageAlertToLastRespondent:Boolean=
    StaffNoticesAlertsSettings.allInstances()->any(true).alertWhenNewMessage and
    StaffNoticesAlertsSettings.allInstances()->any(true).alertWhenNewMessageStaff->includes(#LastRespondent)
  in
  let sendNewMessageAlertToAssignedStaff:Boolean=
    StaffNoticesAlertsSettings.allInstances()->any(true).alertWhenNewMessage and
    StaffNoticesAlertsSettings.allInstances()->any(true).alertWhenNewMessageStaff->includes(#AssignedStaff)
  in
  let sendNewMessageAlertToDepartmentManager:Boolean=
    StaffNoticesAlertsSettings.allInstances()->any(true).alertWhenNewMessage and
    StaffNoticesAlertsSettings.allInstances()->any(true).alertWhenNewMessageStaff->includes(#DepartmentManager)
  in
  let staffAlertsFromEMailAddress:String=
    EmailSettings.allInstances()->any(true).defaultStaffAlertsEmail.address
  in
  let sendAutoresponse:Boolean=
    if CustomerAutoresponsesSettings.allInstances()->any(true).autorespondWhenNewTicketCreatedByCustomer then
      if self.ticket.helpTopic->notEmpty then
        if self.ticket.helpTopic.autoresponse=#Enabled then true
        else false
      endif
    else false
  end
DEVELOPMENT OF THE CONCEPTUAL SCHEMA OF THE osTICKET SYSTEM BY APPLYING TDCM

April 2011

Research report

else
  self.ticket.assignedDepartment.newTicketAutoresponsesSent
endif
else false
endif
in
  self.ticket.ticketThreadMessage->one(tdm | tdm.oclsIsNew())
  and tdm.datetime=System.allInstances()->any(true).currentDateTime
  and tdm.text=self.replyText
  and tdm.author=self.ticket.fullName
  and self.ticket.lastMessageDatetime=System.allInstances()->any(true).currentDateTime

-- autoresponses
and (sendAutoresponse implies
  EMail.allInstances()->exists(e|e.fromAddress=self.ticket.assignedDepartment.autoresponseEmail.address and
  e.toAddress=self.ticket.email and
  e.ticketNumber=self.ticket.number))

-- staff notices
and (sendNewMessageAlertToLastRespondent and
  self.ticket.lastRespondent->notEmpty() and
  not(self.ticket.lastRespondent.isInVacationMode or self.ticket.lastRespondent.status=#Disabled)
  implies
  EMail.allInstances()->exists(e|e.fromAddress=staffAlertsFromEMailAddress and
  e.toAddress=self.ticket.lastRespondent and
  e.ticketNumber=self.ticket.number ))

and (sendNewMessageAlertToAssignedStaff
  and self.ticket.assignedStaff->notEmpty()
  and not(self.ticket.assignedStaff.isInVacationMode or self.ticket.assignedStaff.status=#Disabled)
  implies
  EMail.allInstances()->exists(e|e.fromAddress=staffAlertsFromEMailAddress and
  e.toAddress=self.ticket.assignedStaff.emailAddress and
  e.ticketNumber=self.ticket.number))

event LogIn
attributes
  username:String
  password:String
operations
  effect()
end

context LogIn::effect()
post:
  StaffMember.allInstances()->any(sm|sm.username=self.username and sm.password=self.password).isLoggedIn=true

context StaffMember inv usernameIsUnique:
  StaffMember.allInstances()->isUnique(username)

context LogIn inv isNotLoggedIn:
  StaffMember.allInstances()->select(sm|sm.username=self.username and sm.password=self.password)->size()>0
  implies
  StaffMember.allInstances()->any(sm|sm.username=self.username and sm.password=self.password).isLoggedIn=false

context LogIn inv accessDataisValid:
  StaffMember.allInstances()->select(sm|sm.username=self.username and sm.password=self.password)->size()>0

context LogIn inv staffMemberIsEnabled:
  StaffMember.allInstances()->select(sm|sm.username=self.username and sm.password=self.password)->size()>0
  implies
  StaffMember.allInstances()->any(sm|sm.username=self.username and sm.password=self.password).status=#Enabled

and StaffMember.allInstances()->any(sm|sm.username=self.username and sm.password=self.password).staffGroup.status=#Enabled

event LogOut
operations
DEVELOPMENT OF THE CONCEPTUAL SCHEMA OF THE osTICKET SYSTEM BY APPLYING TDCM

April 2011

Research report

Effect:

context LogOut::effect()
post:
self.staffMember.isLoggedIn=false

class LogOut::effect()
context LogOut::ini
inv isNotLoggedIn:
self.staffMember.isLoggedIn=true

event NewTicketOffline
attributes
fullName:String
email: String
telephone: String[0..1]
ext: String[0..1]
source: TicketSource
subject: String
message: String
priority: Priority[0..1]
createdTicket: Ticket[0..1]
internalNote: String[0..1]
dueDatetime: Datetime[0..1]
operations
effect()
end

class NewTicketOffline::ini
inv emailIsNotInBanList:
EmailSettings.allInstances()->any(true).banList->notEmpty() implies
EmailSettings.allInstances()->any(true).banList->excludes(self.email)

association newTicketOffline_department between
NewTicketOffline[*]
Department[1] role assignedDepartment
end

association newTicketOffline_helpTopic between
NewTicketOffline[*]
HelpTopic[0..1]
end

association newTicketOffline_assignedStaff between
NewTicketOffline[*]
StaffMember[0..1] role assignedStaff
end

association newTicketOffline_creator between
NewTicketOffline[*] role newTicketOfflineOfCreator
StaffMember[1] role creator
end

class NewTicketOffline::ini
inv creatorIsLoggedIn:
self.creator.isLoggedIn

class NewTicketOffline::ini
inv creatorIsAllowedToCreateTickets:
self.creator.staffGroup.canCreateTickets

class NewTicketOffline::ini
inv helpTopicSpecifiedIfAvailable:
if HelpTopic.allInstances()->select(hp|hp.status=#Enabled)->size()=0 then
self.helpTopic->size()=1
else
self.helpTopic->size()=0
endif

context NewTicketOffline ini inv helpTopicIsEnabled:
  self.helpTopic->notEmpty() implies self.helpTopic.status=#Enabled

context NewTicketOffline ini inv maximumOpenTicketsLimitIsNotViolated:
  if TicketSettings.allInstances()->any(true).openTicketsPerMailAreLimited then
    Ticket.allInstances()->select(tt.email=self.email)->size()<TicketSettings.allInstances()->any(true).maximumOpenTicketsPerMail
  else true
endif

context NewTicketOffline::effect()
post:
  let sendNewTicketAlertToAdministrator:Boolean=
    StaffNoticesAlertsSettings.allInstances()->any(true).alertWhenNewTicketCreated and
    StaffNoticesAlertsSettings.allInstances()->any(true).alertWhenNewTicketCreatedStaff->includes(#Administrator)
in
  let sendNewTicketAlertToDepartmentManager:Boolean=
    StaffNoticesAlertsSettings.allInstances()->any(true).alertWhenNewTicketCreated and
    StaffNoticesAlertsSettings.allInstances()->any(true).alertWhenNewTicketCreatedStaff->includes(#DepartmentManager)
in
  let sendNewTicketAlertToDepartmentMembers:Boolean=
    StaffNoticesAlertsSettings.allInstances()->any(true).alertWhenNewTicketCreated and
    StaffNoticesAlertsSettings.allInstances()->any(true).alertWhenNewTicketCreatedStaff->includes(#DepartmentMembers)
in
  let staffAlertsFromEMailAddress:String=
    EmailSettings.allInstances()->any(true).defaultStaffAlertsEmail.address
in
  let sendAutoresponse:Boolean=
    if CustomerAutoresponsesSettings.allInstances()->any(true).autorespondWhenNewTicketCreatedByCustomer then
      if self.helpTopic->notEmpty then
        if self.helpTopic.autoresponse=#Enabled then true
        else false
      endif
    else false
    endif
in
(Ticket.allInstances- Ticket.allInstances@pre)
->one(t | t.oclIsNew() and self.createdTicket=t and self.fullName=self.fullName and self.email=self.email and self.telephone=self.telephone and self.subject=self.subject and self.message=self.message and self.ticketStatus=#Open and self.priority=self.priority and self.source=self.source and self.dueDatetime=self.dueDatetime and self.helpTopic=self.helpTopic and self.assignedDepartment=self.assignedDepartment and self.assignedStaff=self.assignedStaff and (TicketThreadMessage.allInstances- TicketThreadMessage.allInstances@pre)
  ->one(tdm | tdm.oclIsNew() and tdm.datetime=System.allInstances()->any(true).currentDateTime and tdm.text=self.message and tdm.author=self.fullName and tdm.ticket=t and t.lastMessageDatetime=System.allInstances()->any(true).currentDateTime)
  -- autoresponses
  and (sendAutoresponse implies
    EMail.allInstances()->exists(e|e.fromAddress=t.assignedDepartment.autoresponseEmail.address and e.toAddress=t.email and e.ticketNumber=t.number))
  -- staff notices
and (sendNewTicketAlertToAdministrator implies
    EMail.allInstances[]->exists(e|fromAddress=staffAlertsFromEMailAddress and
e.toAddress=EmailSettings.allInstances[]->any(true).administrationEmail and
e.ticketNumber=t.number))

and (sendNewTicketAlertToDepartmentManager
    and t.assignedDepartment.departmentManager->notEmpty()
    and t.assignedDepartment.departmentManager.status=#Enabled
    and not(t.assignedDepartment.departmentManager.isInVacationMode)
    implies
    EMail.allInstances[]->exists(e|fromAddress=staffAlertsFromEMailAddress and
e.toAddress=t.assignedDepartment.departmentManager.emailAddress and
e.ticketNumber=t.number))

and (sendNewTicketAlertToDepartmentMembers
    implies
    t.assignedDepartment.staffMember->forAll(m|
        (m.status=#Enabled and not(m.isInVacationMode))
    implies
    EMail.allInstances[]->exists(e|fromAddress=staffAlertsFromEMailAddress and
e.toAddress=m.emailAddress and
e.ticketNumber=t.number))
)

event DisplayTicketsByStatus
attributes
  status:StatusFilter
operations
  answer():Sequence(Tuple(number:Integer,createDate:Integer,subject:String,department:String, priority:Priority,email:String))=
    let visibleDepartments:Set(Department)=
      self.consultant.staffGroup.departmentsAccess->including(self.consultant.department)
    in
    if self.status=#OpenTickets then
      Ticket.allInstances
      ->select(t|t.ticketStatus=#Open and visibleDepartments->includes(t.assignedDepartment))
      -> collect (t | Tuple  {number : t.number, createDate:t.creationDatetime.value,
      subject:t.subject,department:t.assignedDepartment.name, priority:t.priority, email:t.email})->sortedBy(number)
    else
      if self.status=#AssignedToMe then
        Ticket.allInstances
        ->select(t|t.assignedStaff=self.consultant and t.ticketStatus<>#Closed and visibleDepartments-
        >includes(t.assignedDepartment))
        -> collect (t | Tuple  {number : t.number, createDate:t.creationDatetime.value,
      subject:t.subject,department:t.assignedDepartment.name, priority:t.priority, email:t.email})->sortedBy(number)
      else
        if self.status=#OverdueTickets then
          Ticket.allInstances
          ->select(t|t.isOverdue and t.ticketStatus<>#Closed and visibleDepartments->includes(t.assignedDepartment))
          -> collect (t | Tuple  {number : t.number, createDate:t.creationDatetime.value,
      subject:t.subject,department:t.assignedDepartment.name, priority:t.priority, email:t.email})->sortedBy(number)
        else
          Ticket.allInstances
          ->select(t|t.ticketStatus=#Closed and visibleDepartments->includes(t.assignedDepartment))
          -> collect (t | Tuple  {number : t.number, createDate:t.creationDatetime.value,
      subject:t.subject,department:t.assignedDepartment.name, priority:t.priority, email:t.email})->sortedBy(number)
        endif
      endif
    endif

association displayTicketsByStatus_consultant between
  DisplayTicketsByStatus[*] role newTicketOfflineOfConsultant
  StaffMember[1] role consultant

end
context DisplayTicketsByStatus ini inv consultantisLoggedIn:
    self.consultant.isLoggedIn

event ChangeTicketPriority<StaffTicketEvent
attributes
    newPriority:Priority
operations
effect()
end

context ChangeTicketPriority::effect()
post:
    self.ticket.priority=self.newPriority
and
    (InternalNote.allInstances- InternalNote.allInstances@pre)
        ->one(i | i.oclIsNew() and i.datetime=System.allInstances()->any(true).currentDateTime
and i.subject='Ticket priority changed'
and i.text='The ticket priority has been changed'
and i.author=self.staffMember.firstName
and i.ticket=self.ticket)
and self.ticket.lastMessageDatetime=System.allInstances()->any(true).currentDateTime

event MarkTicketOverdue<StaffTicketEvent
operations
effect()
end

context MarkTicketOverdue::effect()
post:
    self.ticket.isOverdue
and
    (InternalNote.allInstances- InternalNote.allInstances@pre)
        ->one(i | i.oclIsNew() and i.datetime=System.allInstances()->any(true).currentDateTime
and i.subject='Ticket Marked Overdue'
and i.text='Ticket flagged as overdue'
and i.author=self.staffMember.firstName
and i.ticket=self.ticket)
and self.ticket.lastMessageDatetime=System.allInstances()->any(true).currentDateTime

context MarkTicketOverdue ini inv staffMemberIsAnAdministrator:
    self.staffMember.isAdministrator

event AssignTicket<StaffTicketEvent
attributes
    assignmentText:String
operations
effect()
end

association assignTicket_assignee between
    AssignTicket[*] role assignTicketOfAssignee
    StaffMember[1] role assignee
end
context AssignTicket::effect()
post:
let staffAlertsFromEMailAddress:String=
  EmailSettings.allInstances()->any(true).defaultStaffAlertsEmail.address
in
  self.ticket.assignedStaff=self.assignee
and
  (InternalNote.allInstances- InternalNote.allInstances@pre)
    ->one( i | i.oclIsNew() and i.datetime=System.allInstances()->any(true).currentDateTime
    and i.subject='Ticket Reassigned'
    and i.text=self.assignmentText
    and i.author=self.staffMember.firstName
    and i.ticket=self.ticket)
and
  self.ticket.lastMessageDatetime=System.allInstances()->any(true).currentDateTime
and
  (EMail.allInstances()->exists(e|e.fromAddress=staffAlertsFromEMailAddress and
    e.toAddress=self.assignee.emailAddress and
    e.ticketNumber=self.ticket.number))
context AssignTicket ini inv assigneeIsNotInVacationMode:
  not(self.assignee.isInVacationMode)

event ReleaseTicket<StaffTicketEvent
operations
  effect()
end

context ReleaseTicket::effect()
post:
  self.ticket.assignedStaff->isEmpty()
and
  (InternalNote.allInstances- InternalNote.allInstances@pre)
    ->one( i | i.oclIsNew() and i.datetime=System.allInstances()->any(true).currentDateTime
    and i.subject='Ticket unassigned'
    and i.text='Released ticket'
    and i.author=self.staffMember.firstName
    and i.ticket=self.ticket)
context ReleaseTicket ini inv theTicketIsAssigned:
  self.ticket.assignedStaff.isDefined()
context EditTicket::effect()
post:
self.ticket.email=self.emailAddress and
self.ticket.fullName=self.fullName and
self.ticket.statusText=self.statusText and
self.ticket.telephone=self.telephone and
self.ticket.ext=self.ext and
self.ticket.priority=self.priority and
self.ticket.helpTopic=self.helpTopic and
self.ticket.dueDatetime=self.dueDatetime and

(InternalNote.allInstances- InternalNote.allInstances@pre)
->one(i | i.oclIsNew() and i.datetime=System.allInstances()->any(true).currentDateTime
and i.subject='Ticket updated' and i.text=self.editionInternalNote
and i.author=self.staffMember.firstName
and i.ticket=self.ticket)

context EditTicket ini inv staffMembersIsNotAllowedToEditTickets:
self.staffMember.staffGroup.canEditTickets or self.staffMember.isUserAdmin

event PostTicketReply<StaffTicketEvent
attributes
response:String
operations
effect()
end

context PostTicketReply::effect()
post:
let sendNewMessageAlertToLastRespondent:Boolean=
StaffNoticesAlertsSettings.allInstances()->any(true).alertWhenNewMessage and
StaffNoticesAlertsSettings.allInstances()->any(true).alertWhenNewMessageStaff->includes(#LastRespondent)
in
let sendNewMessageAlertToAssignedStaff:Boolean=
StaffNoticesAlertsSettings.allInstances()->any(true).alertWhenNewMessage and
StaffNoticesAlertsSettings.allInstances()->any(true).alertWhenNewMessageStaff->includes(#AssignedStaff)
in
let sendNewMessageAlertToDepartmentManager:Boolean=
StaffNoticesAlertsSettings.allInstances()->any(true).alertWhenNewMessage and
StaffNoticesAlertsSettings.allInstances()->any(true).alertWhenNewMessageStaff->includes(#DepartmentManager)
in
let staffAlertsFromEMailAddress:String=
EmailSettings.allInstances()->any(true).defaultStaffAlertsEmail.address
in
let sendAutoresponse:Boolean=
if CustomerAutoresponsesSettings.allInstances()->any(true).autorespondWhenNewMessageAppendedToTicket then
if self.ticket.helpTopic->notEmpty then
if self.ticket.helpTopic.autoresponse=#Enabled then true
else false
endif
else
self.ticket.assignedDepartment.newAddedMessageIsNotified
endif
else false
endif
in
self.ticket.ticketThreadMessage->one(tdm | tdm.oclIsNew() and tdm.datetime=System.allInstances()->any(true).currentDateTime and tdm.text=self.response
and tdm.author=self.staffMember.firstName
and self.ticket.lastMessageDatetime=System.allInstances()->any(true).currentDateTime)
and self.ticket.lastRespondent=self.staffMember

and (sendAutoresponse implies
    EMail.allInstances()->exists(e|e.fromAddress=self.ticket.assignedDepartment.autorresponseEmail.address and e.toAddress=self.ticket.email and e.ticketNumber=self.ticket.number))

-- staff notices
and (sendNewMessageAlertToLastRespondent and
    self.ticket.lastRespondent@pre->notEmpty() and
    not(self.ticket.lastRespondent@pre.oclIsNew() or self.ticket.lastRespondent@pre.isInVacationMode or self.ticket.lastRespondent@pre.status=#Disabled)
    implies
    EMail.allInstances()->exists(e|e.fromAddress=staffAlertsFromEMailAddress and e.toAddress=self.ticket.lastRespondent@pre and e.ticketNumber=self.ticket.number))

and (sendNewMessageAlertToAssignedStaff
    and self.ticket.assignedStaff->notEmpty()
    and not(self.ticket.assignedStaff.isInVacationMode or self.ticket.assignedStaff.status=#Disabled)
    implies
    EMail.allInstances()->exists(e|e.fromAddress=staffAlertsFromEMailAddress and e.toAddress=self.ticket.assignedStaff.emailAddress and e.ticketNumber=self.ticket.number))

event PostTicketInternalNote<StaffTicketEvent
attributes
    title:String
    note:String
operations
effect() end

context PostTicketInternalNote::effect() post:
    let sendNewMessageAlertToLastRespondent:Boolean=
        StaffNoticesAlertsSettings.allInstances()->any(true).alertWhenNewMessage and
        StaffNoticesAlertsSettings.allInstances()->any(true).alertWhenNewMessageStaff->includes(#LastRespondent)
    in
    let sendNewMessageAlertToAssignedStaff:Boolean=
        StaffNoticesAlertsSettings.allInstances()->any(true).alertWhenNewMessage and
        StaffNoticesAlertsSettings.allInstances()->any(true).alertWhenNewMessageStaff->includes(#AssignedStaff)
    in
    let sendNewMessageAlertToDepartmentManager:Boolean=
        StaffNoticesAlertsSettings.allInstances()->any(true).alertWhenNewMessage and
        StaffNoticesAlertsSettings.allInstances()->any(true).alertWhenNewMessageStaff->includes(#DepartmentManager)
    in
    let staffAlertsFromEMailAddress:String=
        EmailSettings.allInstances()->any(true).defaultStaffAlertsEmail.address
    in
    self.ticket.internalNote->one(int | int.oclIsNew()
    and int.datetime=System.allInstances()->any(true).currentDateTime
    and int.subject=self.title
    and int.text=self.note
    and int.author=self.staffMember.firstName
    and self.ticket.lastMessageDatetime=System.allInstances()->any(true).currentDateTime)

-- staff notices
and (sendNewMessageAlertToLastRespondent and
    self.ticket.lastRespondent@pre->notEmpty() and
    not(self.ticket.lastRespondent@pre.oclIsNew() or self.ticket.lastRespondent@pre.isInVacationMode or self.ticket.lastRespondent@pre.status=#Disabled)
    implies
    EMail.allInstances()->exists(e|e.fromAddress=staffAlertsFromEMailAddress and e.toAddress=self.ticket.lastRespondent@pre and e.ticketNumber=self.ticket.number))
event StaffTicketEvent
end

association staffTicketEvent_ticket between
    StaffTicketEvent[*]
    Ticket[1]
end

association staffTicketEvent_staffMember between
    StaffTicketEvent[*]
    StaffMember[1]
end

context StaffTicketEvent ini inv theTicketIsVisible:
    self.staffMember.isAdministrator or
    (self.staffMember.staffGroup.departmentsAccess->including(self.staffMember.department))-
    >includes(self.ticket.assignedDepartment)

context StaffTicketEvent ini inv staffMemberIsLoggedIn:
    self.staffMember.isLoggedIn

event TransferDepartment<StaffTicketEvent
attributes
    note:String
operations
    effect()
end

association transferDepartment_department between
    TransferDepartment[*]
    Department[1]
end

context TransferDepartment::effect()
    post:
        self.ticket.assignedDepartment=self.department and
        self.ticket.internalNote->one(int | int.oclIsNew() and
        int.datetime=System.allInstances()->any(true).currentDateTime
        and int.subject='Department transfer'
        and int.text=self.note
        and int.author=self.staffMember.firstName
        and self.ticket.lastMessageDatetime=System.allInstances()->any(true).currentDateTime)

context TransferDepartment ini inv departmentIsDifferent:
    self.department <> self.ticket.assignedDepartment

context TransferDepartment ini inv staffMemberIsAllowedToTransfer:
    self.staffMember.staffGroup.canTransferTickets

event CloseTicket<StaffTicketEvent
operations
    effect()
end
context CloseTicket::effect()
post:
self.ticket.ticketStatus=#Closed and
self.ticket.internalNote->one(int | int.oclIsNew() and int.datetime=System.allInstances()->any(true).currentDateTime and int.subject='Ticket closed' and int.text='Ticket closed without response' and int.author=self.staffMember.firstName and self.ticket.lastMessageDatetime=System.allInstances()->any(true).currentDateTime)

context CloseTicket ini inv staffMembersAllowedToClose:
self.staffMember.staffGroup.canCloseTickets

context CloseTicket ini inv ticketsNotClosed:
not (self.ticket.ticketStatus=#Closed)

event CloseTicketWithResponse<StaffTicketEvent
attributes
response:String
operations
effect()
end

context CloseTicketWithResponse::effect()
post:
let sendAutoresponse:Boolean=
if CustomerAutoresponsesSettings.allInstances()->any(true).autorepondWhenNewMessageAppendedToTicket then
if self.ticket.helpTopic->notEmpty then
if self.ticket.helpTopic.autoresponse=#Enabled then true else false endif
else
self.ticket.assignedDepartment.newAddedMessageIsNotified else false endif
endif
in
self.ticket.ticketStatus=#Closed and
self.ticket.internalNote->one(int | int.oclIsNew() and int.datetime=System.allInstances()->any(true).currentDateTime and int.subject='Ticket closed' and int.text='Ticket closed on reply' and int.author=self.staffMember.firstName and self.ticket.lastMessageDatetime=System.allInstances()->any(true).currentDateTime) and
self.ticket.ticketThreadMessage->one(tdm | tdm.oclIsNew() and tdm.datetime=System.allInstances()->any(true).currentDateTime and tdm.text=self.response and tdm.author=self.staffMember.firstName and self.ticket.lastMessageDatetime=System.allInstances()->any(true).currentDateTime)
and self.ticket.lastRespondent=self.staffMember

and (sendAutoresponse implies EMail.allInstances()->exists(e|e.fromAddress=self.ticket.assignedDepartment.autoresponseEmail.address and e.toAddress=self.ticket.email and e.ticketNumber=self.ticket.number))

context CloseTicketWithResponse ini inv staffMembersAllowedToClose:
self.staffMember.staffGroup.canCloseTickets

context CloseTicketWithResponse ini inv ticketsNotClosed:
not (self.ticket.ticketStatus=#Closed)

event ReopenTicket<StaffTicketEvent
operations
effect()
end

context ReopenTicket::effect()
post:
self.ticket.ticketStatus=#Open and
self.ticket.internalNote->one(int | int.oclIsNew() and int.datetime=System.allInstances()->any(true).currentDateTime
and int.subject='Ticket reopened' and int.author=self.staffMember.firstName
and self.ticket.lastMessageDatetime=System.allInstances()->any(true).currentDateTime)
context ReopenTicket ini inv ticketIsClosed:
self.ticket.ticketStatus=#Closed

event ReopenTicketWithResponse<StaffTicketEvent
attributes
response:String
operations
effect() end
context ReopenTicketWithResponse::effect() post:
let sendAutoresponse:Boolean=
if CustomerAutoresponsesSettings.allInstances()->any(true).autorespondWhenNewMessageAppendedToTicket then
if self.ticket.helpTopic->notEmpty then
if self.ticket.helpTopic.autoresponse=#Enabled then true
else false
endif
else
self.ticket.assignedDepartment.newAddedMessageIsNotified
endif
else false
endif
in
self.ticket.ticketStatus=#Open and
self.ticket.internalNote->one(int | int.oclIsNew() and int.datetime=System.allInstances()->any(true).currentDateTime
and int.subject='Ticket status changed to open' and int.author=self.staffMember.firstName
and self.ticket.lastMessageDatetime=System.allInstances()->any(true).currentDateTime)
and
self.ticket.ticketThreadMessage->one(tdm | tdm.oclIsNew() and tdm.datetime=System.allInstances()->any(true).currentDateTime
and tdm.text=self.response and tdm.author=self.staffMember.firstName
and self.ticket.lastMessageDatetime=System.allInstances()->any(true).currentDateTime)
and self.ticket.lastRespondent=self.staffMember and
(sendAutoresponse implies 
Email.allInstances()->exists(e|e.fromAddress=self.ticket.assignedDepartment.autoresponseEmail.address and
  e.toAddress=self.ticket.email and
  e.ticketNumber=self.ticket.number))
context ReopenTicketWithResponse ini inv ticketIsClosed:
self.ticket.ticketStatus=#Closed

event BanEmailAndCloseTicket<StaffTicketEvent
operations
effect() end
context BanEmailAndCloseTicket::effect() post:
self.ticket.ticketStatus=#Closed and
EmailSettings.allInstances()->any(true).banList->includes(self.ticket.email)

and int.subject="Ticket closed"
and int.text="Email added to banlist and ticket status set to closed"
and int.author->self.staffMember.firstName
and self.ticket.lastMessageDateTime=System.allInstances()->any(true).currentDateTime)

context BanEmailAndCloseTicket ini inv ticketsNotClosed:
not (self.ticket.ticketStatus=#Closed)

context BanEmailAndCloseTicket ini inv staffMembersAllowedToBanEmails:
self.staffMember.staffGroup.canBanEmails

event DeleteTicket
operations
effect()
end

association deleteTicket_ticket between
DeleteTicket[*]
Ticket[0..1]
end

association deleteTicket_staffMember between
DeleteTicket[*]
StaffMember[1]
end

class DeleteTicket::effect()
post:
Ticket.allInstances()->excludes(self.ticket@pre) and
self.ticket@pre.internalNote@pre->forAll(int|InternalNote.allInstances()->excludes(int))
-- and self.ticket@pre.ticketThreadMessage@pre->forAll(ttm|TicketThreadMessage.allInstances()->excludes(ttm))
class DeleteTicket ini inv theTicketIsVisible:
self.staffMember.isAdministrator or
(self.staffMember.staffGroup.departmentsAccess->including(self.staffMember.department))-
>includes(self.ticket.assignedDepartment)
class DeleteTicket ini inv staffMemberIsLoggedIn:
self.staffMember.isLoggedIn
class DeleteTicket ini inv staffMemberIsAllowedToDeleteTickets:
self.staffMember.isAdministrator or
self.staffMember.staffGroup.canDeleteTickets

event CheckOverdueTickets
operations
effect()
end

class CheckOverdueTickets::effect()
post:
let sendOverdueTicketAlertToAdministrator:Boolean=
StaffNoticesAlertsSettings.allInstances()->any(true).alertWhenTicketOverdue and
StaffNoticesAlertsSettings.allInstances()->any(true).alertWhenTicketOverdueStaff->includes(#Administrator)
in
let sendOverdueTicketAlertToDepartmentManager:Boolean=
StaffNoticesAlertsSettings.allInstances()->any(true).alertWhenTicketOverdue and
StaffNoticesAlertsSettings.allInstances()->any(true).alertWhenTicketOverdueStaff->includes(#DepartmentManager)
in
let sendOverdueTicketAlertToDepartmentMembers:Boolean=
StaffNoticesAlertsSettings.allInstances()->any(true).alertWhenTicketOverdue and
StaffNoticesAlertsSettings.allInstances()->any(true).alertWhenTicketOverdueStaff->includes(#DepartmentMembers)
in
let staffAlertsFromEMailAddress:String=
EmailSettings.allInstances()->any(true).defaultStaffAlertsEmail.address
in
Ticket.allInstances()->select(t| (System.allInstances()>
-any(true).currentDateTime.value>t.dueDatetime.value+TicketSettings.allInstances()>
-any(true).ticketGracePeriod))
DEVELOPMENT OF THE CONCEPTUAL
SCHEMA OF THE osTICKET
SYSTEM BY APPLYING TDCM

and not(t.isOverdue))
- > forAll(t, t.isOverdue

-- staff notices
and (sendOverdueTicketAlertToAdministrator implies
  EMail.allInstances() ->exists(e|e.fromAddress=staffAlertsFromEMailAddress and
e.toAddress=EmailSettings.allInstances()->any(true).administrationEmail and
e.ticketNumber=t.number))

and (sendOverdueTicketAlertToDepartmentManager
  and t.assignedDepartment.departmentManager ->notEmpty()
  and t.assignedDepartment.departmentManager.status=#Enabled
  and not(t.assignedDepartment.departmentManager.isInVacationMode)
  implies
  EMail.allInstances() ->exists(e|e.fromAddress=staffAlertsFromEMailAddress and
e.toAddress=t.assignedDepartment.departmentManager.emailAddress and
e.ticketNumber=t.number))

and (sendOverdueTicketAlertToDepartmentMembers
  implies
  t.assignedDepartment.staffMember ->forAll(m|
  (m.status=#Enabled and not(m.isInVacationMode))
  implies
  EMail.allInstances() ->exists(e|e.fromAddress=staffAlertsFromEMailAddress and
e.toAddress=m.emailAddress and
e.ticketNumber=t.number)))

}
Appendix B. Methods specification

method SendMail{
  m1:=new EMail;
  m1.fromAddress:=self.fromAddress;
  m1.toAddress:=self.toAddress;
  m1.emailKind:=self.emailKind;
  m1.ticketNumber:=self.ticketNumber;
  m1.timeStamp:=System.allInstances()-any(true).currentDateTime;
  self.createdEmail:=m1;
}

method NewTicketOnline{
  t1:=new Ticket;
  self.createdTicket:=t1;
  t1.fullName:=self.fullName;
  t1.email:=self.email;
  t1.telephone:=self.telephone;
  t1.ext:=self.ext;
  t1.subject:=self.subject;
  t1.message:=self.message;
  t1.ticketStatus:=#Open;
  t1.helpTopic:=self.helpTopic;
  dp1:=if self.helpTopic->notEmpty() then
    self.helpTopic.newTicketPriority
  else
    TicketSettings.allInstances()-any(true).priority
  endif;
  ap1:=dp1;
  if TicketSettings.allInstances()-any(true).customersCanChangePriority then
    ap1:=self.priority;
  endif
  t1.priority:=ap1;
  t1.source:=#Web;
  ddpt1:=if self.helpTopic->notEmpty() then
    self.helpTopic.newTicketDepartment
  else
    Department.allInstances()-any(d|d.isDefault)
  endif;
  t1.assignedDepartment:=ddpt1;
  tdm1:=new TicketThreadMessage;
  tdm1.datetime:=System.allInstances()-any(true).currentDateTime;
  tdm1.text:=self.message;
  tdm1.author:=self.fullName;
  tdm1.ticket:=t1;
  t1.lastMessageDatetime:=System.allInstances()-any(true).currentDateTime;
  //Autoreponses
  sendAutoresponse:=false;
if CustomerAutoresponsesSettings.allInstances()->any(true).autorespondWhenNewTicketCreatedByCustomer then
  if self.helpTopic->notEmpty then
    if self.helpTopic.autoresponse=#Enabled then sendAutoresponse:=true; endif
  else
    sendAutoresponse:=ddpt1.newTicketAutoresponsesSent;
  endif
endif

if sendAutoresponse then
  m1:=new EMail(fromAddress:=t1.assignedDepartment.autoresponseEmail.address,
    toAddress:=t1.email,
    ticketNumber:=t1.number,emailKind:=NewTicketAutoresponse.allInstances()->any(true));
endif

//Staff alerts
sendNewTicketAlertToAdministrator:=
  StaffNotices Alerts Settings.allInstances()->any(true).alertWhenNewTicketCreated and
  StaffNotices Alerts Settings.allInstances()->any(true).alertWhenNewTicketCreatedStaff->includes(#Administrator)];

sendNewTicketAlertToDepartmentManager:=
  [StaffNotices Alerts Settings.allInstances()->any(true).alertWhenNewTicketCreated and
  StaffNotices Alerts Settings.allInstances()->any(true).alertWhenNewTicketCreatedStaff->includes(#DepartmentManager)];

sendNewTicketAlertToDepartmentMembers:=
  [StaffNotices Alerts Settings.allInstances()->any(true).alertWhenNewTicketCreated and
  StaffNotices Alerts Settings.allInstances()->any(true).alertWhenNewTicketCreatedStaff->includes(#DepartmentMembers)];

staffAlertsFromEMailAddress:=
  [EmailSettings.allInstances()->any(true).defaultStaffAlertsEmail.address];

if sendNewTicketAlertToAdministrator then
  m1:=new EMail(fromAddress:=staffAlertsFromEMailAddress,
    toAddress:=EmailSettings.allInstances()->any(true).administrationEmail,
    ticketNumber:=t1.number,emailKind:=NewTicketAlertToStaff.allInstances()->any(true));
endif

if sendNewTicketAlertToDepartmentManager then
  if t1.assignedDepartment.departmentManager->notEmpty() then
    if t1.assignedDepartment.departmentManager.status=#Enabled then
      if t1.assignedDepartment.departmentManager.isInVacationMode=false then
        m1:=new EMail(fromAddress:=staffAlertsFromEMailAddress,
          toAddress:=t1.assignedDepartment.departmentManager.emailAddress,
          ticketNumber:=t1.number,emailKind:=NewTicketAlertToStaff.allInstances()->any(true));
      endif
    endif
  endif
endif
endif

if sendNewTicketAlertToDepartmentMembers then
  i:=1;
  staff:=t1.assignedDepartment.staffMember->asSequence();
  while i<t1.assignedDepartment.staffMember->size() do
    currentStaffMember:=staff->at(i);
    if currentStaffMember.isInVacationMode=false then
      if currentStaffMember.status=#Enabled then
        m1:=new EMail(fromAddress:=staffAlertsFromEMailAddress,
          toAddress:=currentStaffMember.emailAddress,
          ticketNumber:=t1.number, emailKind:=NewTicketAlertToStaff.allInstances()->any(true));
      endif
    endif
  endwhile
endif

method NewTicketByEmail{
  t1:=new Ticket;
  self.createdTicket:=t1;
  t1.fullName:=self.fromName;
  t1.email:=self.fromAddress;
}
t1.subject:=self.subject;
t1.message:=self.message;
t1.ticketStatus:=#Open;

incomingEmailAccount:=EmailAccount.allInstances()->any(true).address=self.toAddress;
dp1:=TicketSettings.allInstances()->any(true).priority;
if TicketSettings.allInstances()->any(true).useEmailPriorityWhenAvailable then
dp1:=incomingEmailAccount.defaultNewPriority;
endif
ap1:=dp1;

t1.priority:=ap1;
t1.source:=#EMail;

dc1:=TicketDepartment.allInstances()->any(true).defaultNewTicketDepartment;
t1.assignedDepartment:=dc1;

tdm1:=new TicketThreadMessage;
tdm1.datetime:=System.allInstances()->any(true).currentDateTime;
tdm1.text:=self.message;
tdm1.author:=self.fromName;
tdm1.ticket:=t1;
t1.lastMessageDateTime:=System.allInstances()->any(true).currentDateTime;

sendAutoresponse:=true;
//Autoresponses
if incomingEmailAccount.autoresponsesStatus=#Enabled then
    sendAutoresponse:=true;
else
    sendAutoresponse:=false;
endif

if sendAutoresponse then
    m1:=new EMail(fromAddress:=t1.assignedDepartment.autoresponseEmail.address,
        toAddress:=t1.email,
        ticketNumber:=t1.number,emailKind:=NewTicketAutoresponse.allInstances()->any(true));
endif

//Staff alerts
sendNewTicketAlertToAdministrator:=
    StaffNoticesAlertsSettings.allInstances()->any(true).alertWhenNewTicketCreated and
    StaffNoticesAlertsSettings.allInstances()->any(true).alertWhenNewTicketCreatedStaff->includes(#Administrator);

sendNewTicketAlertToDepartmentManager:=
    StaffNoticesAlertsSettings.allInstances()->any(true).alertWhenNewTicketCreated and
    StaffNoticesAlertsSettings.allInstances()->any(true).alertWhenNewTicketCreatedStaff->includes(#DepartmentManager);

sendNewTicketAlertToDepartmentMembers:=
    StaffNoticesAlertsSettings.allInstances()->any(true).alertWhenNewTicketCreated and
    StaffNoticesAlertsSettings.allInstances()->any(true).alertWhenNewTicketCreatedStaff->includes(#DepartmentMembers);

staffAlertsFromEMailAddress:=
    EmailSettings.allInstances()->any(true).defaultStaffAlertsEmail.address;

if sendNewTicketAlertToAdministrator then
    m1:=new EMail(fromAddress:=staffAlertsFromEMailAddress,
        toAddress:=EmailSettings.allInstances()->any(true).administrationEmail,
        ticketNumber:=t1.number,emailKind:=NewTicketAlertToStaff.allInstances()->any(true));
endif

if sendNewTicketAlertToDepartmentManager then
    if t1.assignedDepartment.departmentManager->notEmpty() then
        if t1.assignedDepartment.departmentManager.status=#Enabled then
            m1:=new EMail(fromAddress:=staffAlertsFromEMailAddress,
                toAddress:=t1.assignedDepartment.departmentManager.emailAddress,
                ticketNumber:=t1.number,emailKind:=NewTicketAlertToStaff.allInstances()->any(true));
        endif
    endif
endif
DEVELOPMENT OF THE CONCEPTUAL SCHEMA OF THE osTICKET SYSTEM BY APPLYING TDCM

April 2011

Department of Service and Information System Engineering

method DisplayTicketsAssociatedToEmail{
}

method ReplyTicketByCustomer{

//Staff alerts
sendNewMessageAlertToLastRespondent:=[
StaffNoticesAlertsSettings.allInstances()->any(true).alertWhenNewMessage and
StaffNoticesAlertsSettings.allInstances()->any(true).alertWhenNewMessageStaff->includes(#LastRespondent);
]

sendNewMessageAlertToAssignedStaff:=[
StaffNoticesAlertsSettings.allInstances()->any(true).alertWhenNewMessage and
StaffNoticesAlertsSettings.allInstances()->any(true).alertWhenNewMessageStaff->includes(#AssignedStaff);
]

sendNewMessageAlertToDepartmentManager:=[
StaffNoticesAlertsSettings.allInstances()->any(true).alertWhenNewMessage and
StaffNoticesAlertsSettings.allInstances()->any(true).alertWhenNewMessageStaff->includes(#DepartmentManager);
]

staffAlertsFromEMailAddress:=
[EmailSettings.allInstances()->any(true).defaultStaffAlertsEmail.address]

if sendNewMessageAlertToLastRespondent then
if self.ticket.lastRespondent->notEmpty() then
if self.ticket.lastRespondent.status=#Enabled then
m1:=new EMail(fromAddress:=staffAlertsFromEMailAddress,
toAddress:=[self.ticket.lastRespondent.emailAddress],
ticketNumber:=self.ticket.number,emailKind:=NewMessageAlertToStaff.allInstances()->any(true));
endif
endif
endif
endif
if sendNewMessageAlertToAssignedStaff then
if self.ticket.assignedStaff->notEmpty() then
if self.ticket.assignedStaff.status=#Enabled then
if self.ticket.assignedStaff.isInVacationMode=false then
m1:=new EMail(fromAddress:=staffAlertsFromEMailAddress,
toAddress:=[self.ticket.assignedStaff.emailAddress],
ticketNumber:=self.ticket.number,emailKind:=NewMessageAlertToStaff.allInstances()->any(true));
endif
endif
endif
endif
if sendNewMessageAlertToDepartmentManager then
if self.ticket.assignedDepartment.departmentManager->notEmpty() then
if self.ticket.assignedDepartment.departmentManager.status=#Enabled then
if self.ticket.assignedDepartment.departmentManager.isInVacationMode=false then
endif
endif
endif
endif
endif
m1:=new EMail(fromAddress:=staffAlertsFrom EM ailAddress, toAddress:=self.ticket.assignedDepartment.departmentManager.emailAddress, ticketNumber:=self.ticket. number, emailKind:=NewMessageAlertToStaff.allInstances()->any(true));
endif
endif
endif
endif

tdm1:=new TicketThreadMessage;
tdm1.datetime:=System.allInstances()->any(true).currentDateTime;
tdm1.text:=self.replyText;
tdm1.author:=self.ticket.fullName;
tdm1.ticket:=self.ticket;
self.ticket.lastMessageDatetime:=System.allInstances()->any(true).currentDateTime;

sendAutoresponse:=false;
if CustomerAutoresponsesSettings.allInstances()->any(true).autorespondWhenNewMessageAppendedToTicket then
  if self.ticket.helpTopic->notEmpty then
    if self.ticket.helpTopic.autoresponse=#Enabled then sendAutoresponse:=true;
    endif
  else
    sendAutoresponse:=self.ticket.assignedDepartment.newAddedMessagesNotified;
  endif
endif

if sendAutoresponse then
  m1:=new EMail(fromAddress:=self.ticket.assignedDepartment.autoresponseEmail.address,
    toAddress:=self.ticket.email, ticketNumber:=self.ticket.number, emailKind:=NewMessageAutoresponse.allInstances()->any(true));
endif

}

method LogIn{
  ls:=[StaffMember.allInstances()->any(sm|sm.username=self.username and sm.password=self.password)];
  ls.isLoggedIn:=true;
}

method LogOut{
  self.staffMember.isLoggedIn:=false;
}

method NewTicketOffline{
  t1:=new Ticket;
  self.createdTicket:=t1;
  t1.fullName:=self.fullName;
  t1.email:=self.email;
  t1.telephone:=self.telephone;
  t1.ext:=self.ext;
  t1.subject:=self.subject;
  t1.message:=self.message;
  t1.ticketStatus:=#Open;
  t1.helpTopic:=self.helpTopic;
  t1.priority:=self.priority;
  t1.source:=self.source;
  t1.assignedDepartment:=self.assignedDepartment;
  t1.assignedStaff:=self.assignedStaff;
  t1.dueDatetime:=self.dueDatetime;

tdm1:=new TicketThreadMessage;
tdm1.datetime:=System.allInstances()->any(true).currentDateTime;
tdm1.text:=self.message;
tdm1.author:=self.fullName;
tdm1.ticket:=t1;
t1.lastMessageDateTime:=System.allInstances()->any(true).currentDateTime;

// Autoresponses
sendAutoresponse:=false;
DEVELOPMENT OF THE CONCEPTUAL SCHEMA OF THE osTICKET SYSTEM BY APPLYING TDCM

Research report
April 2011

Department of Service and Information System Engineering

if CustomerAutoresponsesSettings.allInstances()[i].autorespondWhenNewTicketCreatedByCustomer then
    if self.helpTopic[0].autoresponse && self.assignedDepartment.newTicketAutoresponsesSent then
        sendAutoresponse := true;
        endif
        else
            sendAutoresponse := self.assignedDepartment.newTicketAutoresponsesSent;
        endif
        endif

if sendAutoresponse then
    m1 := new EMail(fromAddress := self.assignedDepartment.autoresponseEmail.address,
                    toAddress := t1.email,
                    ticketNumber := t1.number,
                    emailKind := NewTicketNotice.allInstances()[i].autorespondWhenNewTicketCreatedByCustomer);
endif

if CustomerAutoresponsesSettings.allInstances()[j].autorespondWhenNewTicketCreatedByCustomer then
    if self.helpTopic[0].autoresponse && self.assignedDepartment.newTicketAutoresponsesSent then
        sendAutoresponse := true;
        endif
        else
            sendAutoresponse := self.assignedDepartment.newTicketAutoresponsesSent;
        endif
        endif

if sendAutoresponse then
    m1 := new EMail(fromAddress := self.assignedDepartment.autoresponseEmail.address,
                    toAddress := self.assignedDepartment.autoresponseEmail.address,
                    ticketNumber := t1.number,
                    emailKind := NewTicketNotice.allInstances()[j].autorespondWhenNewTicketCreatedByCustomer);
endif

if CustomerAutoresponsesSettings.allInstances()[k].autorespondWhenNewTicketCreatedByCustomer then
    if self.helpTopic[0].autoresponse && self.assignedDepartment.newTicketAutoresponsesSent then
        sendAutoresponse := true;
        endif
        else
            sendAutoresponse := self.assignedDepartment.newTicketAutoresponsesSent;
        endif
        endif

if sendAutoresponse then
    m1 := new EMail(fromAddress := self.assignedDepartment.autoresponseEmail.address,
                    toAddress := self.assignedDepartment.autoresponseEmail.address,
                    ticketNumber := t1.number,
                    emailKind := NewTicketNotice.allInstances()[k].autorespondWhenNewTicketCreatedByCustomer);
endif

// Staff alerts
sendNewTicketAlertToAdministrator :=
    [StaffNoticesAlertsSettings.allInstances()[i].alertWhenNewTicketCreated and
    StaffNoticesAlertsSettings.allInstances()[j].alertWhenNewTicketCreatedStaff.includes(#Administrator)];

sendNewTicketAlertToDepartmentManager :=
    [StaffNoticesAlertsSettings.allInstances()[i].alertWhenNewTicketCreated and
    StaffNoticesAlertsSettings.allInstances()[j].alertWhenNewTicketCreatedStaff.includes(#DepartmentManager)];

sendNewTicketAlertToDepartmentMembers :=
    [StaffNoticesAlertsSettings.allInstances()[i].alertWhenNewTicketCreated and
    StaffNoticesAlertsSettings.allInstances()[j].alertWhenNewTicketCreatedStaff.includes(#DepartmentMembers)];

staffAlertsFromEMailAddress :=
    [EmailSettings.allInstances()[i].defaultStaffAlertsEmail.address];

if sendNewTicketAlertToAdministrator then
    m1 := new EMail(fromAddress := staffAlertsFromEMailAddress,
                    toAddress := EmailSettings.allInstances()[i].administrationEmail,
                    ticketNumber := t1.number,
                    emailKind := NewTicketAlertToStaff.allInstances()[i].alertWhenNewTicketCreated);
endif

if sendNewTicketAlertToDepartmentManager then
    if t1.assignedDepartment.departmentManager->notEmpty() then
        if t1.assignedDepartment.departmentManager.status && t1.assignedDepartment.departmentManager.isInVacationMode = false then
            m1 := new EMail(fromAddress := staffAlertsFromEMailAddress,
                            toAddress := t1.assignedDepartment.departmentManager.emailAddress,
                            ticketNumber := t1.number,
                            emailKind := NewTicketAlertToStaff.allInstances()[j].alertWhenNewTicketCreated);
        endif
        endif
endif

if sendNewTicketAlertToDepartmentMembers then
    i := 1;
    staff := t1.assignedDepartment.staffMember->asSequence();
    while i < t1.assignedDepartment.staffMember->size() do
        currentStaffMember := staff->at(i);
        if currentStaffMember.isInVacationMode = false then
            if currentStaffMember.status && currentStaffMember.isInVacationMode = false then
                m1 := new EMail(fromAddress := staffAlertsFromEMailAddress,
                                toAddress := currentStaffMember.emailAddress,
                                ticketNumber := t1.number,
                                emailKind := NewTicketAlertToStaff.allInstances()[k].alertWhenNewTicketCreated);
            endif
            endif
        i := i + 1;
        endwhile
    endif

method DisplayTicketsByStatus{
}

method ChangeTicketPriority{
    self.ticket.priority := self.newPriority;
DEVELOPMENT OF THE CONCEPTUAL
SCHEMA OF THE osTICKET
SYSTEM BY APPLYING TDCM

Research report
April 2011

Department of Service and
Information System Engineering

```java
i:=new InternalNote;
i.datetime:=System.allInstances()->any(true).currentDateTime;
i.subject:='Ticket priority changed';
i.text:='The ticket priority has been changed';
i.author:=self.staffMember.firstName;
i.ticket:=self.ticket;
self.ticket.lastMessageDatetime:=System.allInstances()->any(true).currentDateTime;
}

method MarkTicketOverdue{
    self.ticket.isOverdue:=true;
    i:=new InternalNote;
    i.datetime:=System.allInstances()->any(true).currentDateTime;
    i.subject:='Ticket Marked Overdue';
i.text:='Ticket flagged as overdue';
i.author:=self.staffMember.firstName;
i.ticket:=self.ticket;
self.ticket.lastMessageDatetime:=System.allInstances()->any(true).currentDateTime;
}

method AssignTicket{
    self.ticket.assignedStaff:=self.assignee;
    staffAlertsFromEMailAddress:=
    [EmailSettings.allInstances()->any(true).defaultStaffAlertsEmail.address];
    i:=new InternalNote;
    i.datetime:=System.allInstances()->any(true).currentDateTime;
    i.subject:='Ticket Reassigned';
i.text:=self.assignmentText;
i.author:=self.staffMember.firstName;
i.ticket:=self.ticket;
self.ticket.lastMessageDatetime:=System.allInstances()->any(true).currentDateTime;
    m1:=new EMail(fromAddress:=staffAlertsFromEMailAddress,
toAddress:=self.assignee.emailAddress,
ticketNumber:=self.ticket.number,emailKind:=TicketAssignedAlertToStaff.allInstances()->any(true));
}

method ReleaseTicket{
    self.ticket.assignedStaff:=Set{};
    i:=new InternalNote;
    i.datetime:=System.allInstances()->any(true).currentDateTime;
    i.subject:='Ticket unassigned';
i.text:='Released ticket';
i.author:=self.staffMember.firstName;
i.ticket:=self.ticket;
self.ticket.lastMessageDatetime:=System.allInstances()->any(true).currentDateTime;
}

method EditTicket{
    self.ticket.email:=self.emailAddress;
    self.ticket.fullName:=self.fullName;
    self.ticket.subject:=self.subject;
    self.ticket.telephone:=self.telephone;
    self.ticket.ext:=self.ext;
    self.ticket.priority:=self.priority;
    self.ticket.helpTopic:=self.helpTopic;
    self.ticket.dueDatetime:=self.dueDatetime;
    i:=new InternalNote;
    i.datetime:=System.allInstances()->any(true).currentDateTime;
    i.subject:='Ticket updated';
i.text:=self.editionInternalNote;
i.author:=self.staffMember.firstName;
i.ticket:=self.ticket;
```
method PostTicketReply{
    // Staff alerts
    sendNewMessageAlertToLastRespondent:=
        StaffNoticesAlertsSettings.allInstances()->any(true).alertWhenNewMessage and
        StaffNoticesAlertsSettings.allInstances()->any(true).alertWhenNewMessageStaff->includes(#LastRespondent);

    sendNewMessageAlertToAssignedStaff:=
        StaffNoticesAlertsSettings.allInstances()->any(true).alertWhenNewMessage and
        StaffNoticesAlertsSettings.allInstances()->any(true).alertWhenNewMessageStaff->includes(#AssignedStaff);

    sendNewMessageAlertToDepartmentManager:=
        StaffNoticesAlertsSettings.allInstances()->any(true).alertWhenNewMessage and
        StaffNoticesAlertsSettings.allInstances()->any(true).alertWhenNewMessageStaff->includes(#DepartmentManager);

    staffAlertsFromEMailAddress:=
        [EmailSettings.allInstances()->any(true).defaultStaffAlertsEmail.address];

    sendAutoresponse:=false;
    if CustomerAutoresponsesSettings.allInstances()->any(true).autorespondWhenNewMessageAppendedToTicket then
        if self.ticket.helpTopic->notEmpty then
            if self.ticket.helpTopic.autoresponse=#Enabled then sendAutoresponse:=true;
            else
                sendAutoresponse:=self.ticket.assignedDepartment.newAddedMessagesNotified;
            endif
        endif
    else
        sendAutoresponse:=false;
    endif

    if sendAutoresponse then
        m1:=new EMail(fromAddress:=self.ticket.assignedDepartment.autoresponseEmail.address,
                    toAddress:=self.ticket.email,
                    ticketNumber:=self.ticket.number,emailKind:=TicketResponseNotice.allInstances()->any(true));
    endif

    if sendNewMessageAlertToLastRespondent then
        if self.ticket.lastRespondent->notEmpty() then
            if self.ticket.lastRespondent.status=#Enabled then
                if self.ticket.lastRespondent.isInVacationMode=false then
                    m1:=new EMail(fromAddress:=staffAlertsFromEMailAddress,
                                toAddress:=self.ticket.lastRespondent.emailAddress,
                                ticketNumber:=self.ticket.number,emailKind:=NewMessageAlertToStaff.allInstances()->any(true));
                endif
            endif
        endif
    endif

    if sendNewMessageAlertToAssignedStaff then
        if self.ticket.assignedStaff->notEmpty() then
            if self.ticket.assignedStaff.status=#Enabled then
                if self.ticket.assignedStaff.isInVacationMode=false then
                    m1:=new EMail(fromAddress:=staffAlertsFromEMailAddress,
                                toAddress:=self.ticket.assignedStaff.emailAddress,
                                ticketNumber:=self.ticket.number,emailKind:=NewMessageAlertToStaff.allInstances()->any(true));
                endif
            endif
        endif
    endif

    if sendNewMessageAlertToDepartmentManager then
        if self.ticket.assignedDepartment.departmentManager->notEmpty() then
            if self.ticket.assignedDepartment.departmentManager.status=#Enabled then
                if self.ticket.assignedDepartment.departmentManager.isInVacationMode=false then
                    m1:=new EMail(fromAddress:=staffAlertsFromEMailAddress,
                                toAddress:=self.ticket.assignedDepartment.departmentManager.emailAddress,
                                ticketNumber:=self.ticket.number,emailKind:=NewMessageAlertToStaff.allInstances()->any(true));
                endif
            endif
        endif
    endif
}
self.ticket.lastRespondent:=self.staffMember;

tdm1:=new TicketThreadMessage;
tdm1.datetime:=System.allInstances()->any(true).currentDateTime;
tdm1.text:=self.response;
tdm1.author:=self.staffMember.firstName;
tdm1.ticket:=self.ticket;
self.ticket.lastMessageDatetime:=System.allInstances()->any(true).currentDateTime;

}

method PostTicketInternalNote{
    //Staff alerts
    sendNewMessageAlertToLastRespondent:=[
        StaffNoticesAlertsSettings.allInstances()->any(true).alertWhenNewMessage and
        StaffNoticesAlertsSettings.allInstances()->any(true).alertWhenNewMessageStaff->includes(#LastRespondent)];
    sendNewMessageAlertToAssignedStaff:=[
        StaffNoticesAlertsSettings.allInstances()->any(true).alertWhenNewMessage and
        StaffNoticesAlertsSettings.allInstances()->any(true).alertWhenNewMessageStaff->includes(#AssignedStaff)];
    sendNewMessageAlertToDepartmentManager:=[
        StaffNoticesAlertsSettings.allInstances()->any(true).alertWhenNewMessage and
        StaffNoticesAlertsSettings.allInstances()->any(true).alertWhenNewMessageStaff->includes(#DepartmentManager)];
    staffAlertsFromEMailAddress:=
        [EmailSettings.allInstances()->any(true).defaultStaffAlertsEmail.address]
        if sendNewMessageAlertToLastRespondent then
        if  self.ticket.lastRespondent->notEmpty() then
            if  self.ticket.lastRespondent.status=#Enabled then
                if self.ticket.lastRespondent.isInVacationMode=false then
                    m1:=new EMail(fromAddress:=staffAlertsFromEMailAddress,
                        toAddress:=[self.ticket.lastRespondent.emailAddress],
                        ticketNumber:=self.ticket.number,emailKind:=NewInternalNoteAlertToStaff.allInstances()->any(true));
                endif
            endif
        endif
    endif
    endif
    endif
    if sendNewMessageAlertToAssignedStaff then
    if  self.ticket.assignedStaff->notEmpty() then
        if  self.ticket.assignedStaff.status=#Enabled then
            if self.ticket.assignedStaff.isInVacationMode=false then
                m1:=new EMail(fromAddress:=staffAlertsFromEMailAddress,
                    toAddress:=[self.ticket.assignedStaff.emailAddress],
                    ticketNumber:=self.ticket.number,emailKind:=NewInternalNoteAlertToStaff.allInstances()->any(true));
            endif
        endif
    endif
    endif
    endif
    if sendNewMessageAlertToDepartmentManager then
    if  self.ticket.assignedDepartment.departmentManager->notEmpty() then
        if  self.ticket.assignedDepartment.departmentManager.status=#Enabled then
            if self.ticket.assignedDepartment.departmentManager.isInVacationMode=false then
                m1:=new EMail(fromAddress:=staffAlertsFromEMailAddress,
                    toAddress:=[self.ticket.assignedDepartment.departmentManager.emailAddress],
                    ticketNumber:=self.ticket.number,emailKind:=NewInternalNoteAlertToStaff.allInstances()->any(true));
            endif
        endif
    endif
    endif
    endif

    i:=new InternalNote;
    i.datetime:=System.allInstances()->any(true).currentDateTime;
    i.subject:=self.title;
DEVELOPMENT OF THE CONCEPTUAL
SCHEMA OF THE osTICKET
SYSTEM BY APPLYING TDCM

Research report
April 201

i.text:=self.note;
i.author:=self.staffMember.firstName;
i.ticket:=self.ticket;
sel.s.ticket.lastMessageDatet.:=System.all..Instances().-any(true).currentDateTime;

}

method TransferDepartment{

self.ticket.assignedDepartment:=self.department;

i:=new InternalNote;
i.datetime:=System.allInstances()-any(true).currentDateTime;
i.subject:="Department transfer";
i.text:=self.note;
i.author:=self.staffMember.firstName;
i.ticket:=self.ticket;
sel.s.ticket.lastMessageDatet.:=System.all..Instances().-any(true).currentDateTime;

}

method CloseTicket{

self.ticket.ticketStatus:="Closed";

i:=new InternalNote;
i.datetime:=System.allInstances()-any(true).currentDateTime;
i.subject:="Ticket closed";
i.text:="Ticket closed without response";
i.author:=self.staffMember.firstName;
i.ticket:=self.ticket;
sel.s.ticket.lastMessageDatet.:=System.all..Instances().-any(true).currentDateTime;

}

method CloseTicketWithResponse{

self.ticket.ticketStatus:="Closed";

tdm1:=new TicketThreadMessage;
tdm1.datetime:=System.allInstances()-any(true).currentDateTime;
tdm1.text:=self.response;
tdm1.author:=self.staffMember.firstName;
tdm1.ticket:=self.ticket;
self.ticket.lastMessageDatet.:=System.all..Instances().-any(true).currentDateTime;

self.ticket.lastRespondent:=self.staffMember;

i:=new InternalNote;
i.datetime:=System.allInstances()-any(true).currentDateTime;
i.subject:="Ticket closed";
i.text:="Ticket closed on reply";
i.author:=self.staffMember.firstName;
i.ticket:=self.ticket;
sel.s.ticket.lastMessageDatet.:=System.all..Instances().-any(true).currentDateTime;

sendAutoresponse:=false;
if CustomerAutonresponseesSettings.allInstances()-any(true).autorespondWhenNewMessageAppendedToTicket then
  if self.ticket.helpTopic->notEmpty then
    if self.ticket.helpTopic.autoresponse=#Enabled then sendAutoresponse:=true;
    else
      sendAutoresponse:=self.ticket.assignedDepartment.newAddedMessagesNotified;
    endif
  endif
endif
if sendAutoresponse then


m1:=new EMail(fromAddress:=self.ticket.assignedDepartment.autoresponseEmail.address, 
toAddress:=self.ticket.email, 
ticketNumber:=self.ticket.number,emailKind:=TicketResponseNotice.allInstances()->any(true));
endif
}

method ReopenTicket{

    self.ticket.ticketStatus:=#Open;

    i:=new InternalNote;
    i.datetime:=System.allInstances()->any(true).currentDateTime;
    i.subject:='Ticket reopened';
    i.text:='Ticket reopened without comments';
    i.author:=self.staffMember.firstName;
    i.ticket:=self.ticket;
    self.ticket.lastMessageDatetime:=System.allInstances()->any(true).currentDateTime;
}

method ReopenTicketWithResponse{

    self.ticket.ticketStatus:=#Open;

    tdm1:=new TicketThreadMessage;
    tdm1.datetime:=System.allInstances()->any(true).currentDateTime;
    tdm1.text:=self.response;
    tdm1.author:=self.staffMember.firstName;
    tdm1.ticket:=self.ticket;
    self.ticket.lastMessageDatetime:=System.allInstances()->any(true).currentDateTime;

    self.ticket.lastRespondent:=self.staffMember;

    i:=new InternalNote;
    i.datetime:=System.allInstances()->any(true).currentDateTime;
    i.subject:='Ticket status changed to open';
    i.text:='A staff member reopened the ticket on reply';
    i.author:=self.staffMember.firstName;
    i.ticket:=self.ticket;
    self.ticket.lastMessageDatetime:=System.allInstances()->any(true).currentDateTime;

    sendAutoresponse:=false;
    if CustomerAutoresponsesSettings.allInstances()->any(true).autorespondWhenNewMessageAppendedToTicket then
        if self.ticket.helpTopic->notEmpty then
            if self.ticket.helpTopic.autoresponse=#Enabled then sendAutoresponse:=true;
        endif
    else
        sendAutoresponse:=self.ticket.assignedDepartment.newAddedMessageIsNotified;
    endif
endif

if sendAutoresponse then
    m1:=new EMail(fromAddress:=self.ticket.assignedDepartment.autoresponseEmail.address, 
toAddress:=self.ticket.email, 
ticketNumber:=self.ticket.number,emailKind:=TicketResponseNotice.allInstances()->any(true));
endif
}

method BanEmailAndCloseTicket{

    self.ticket.ticketStatus:=#Closed;

    i:=new InternalNote;
    i.datetime:=System.allInstances()->any(true).currentDateTime;
    i.subject:='Ticket closed';
    i.text:='Email added to banlist and ticket status set to closed';
    i.author:=self.staffMember.firstName;
    i.ticket:=self.ticket;
self.ticket.lastMessageDatetime:=System.allInstances()>>any(true).currentDateTime;
bl:=Set{self.ticket.email};
if EmailSettings.allInstances()>>any(true).banList->notEmpty then
    bl:=EmailSettings.allInstances()>>any(true).banList->including(self.ticket.email);
endif
EmailSettings.allInstances()>>any(true).banList:=bl;
}

method DeleteTicket{
    internalNotes:=self.ticket.internalNote;
ticketThreadMessage:=self.ticket.ticketThreadMessage;
delete self.ticket;
while internalNotes->size()>0 do
    delete internalNotes->any(true);
endwhile
while ticketThreadMessage->size()>0 do
    delete ticketThreadMessage->any(true);
endwhile
}

method CheckOverdueTickets{
    //Staff alerts
    sendOverdueAlertToAssignedStaff:=[
        StaffNoticesAlertsSettings.allInstances()>>any(true).alertWhenTicketOverdue and
        StaffNoticesAlertsSettings.allInstances()>>any(true).alertWhenTicketOverdueStaff->includes(#AssignedStaff)];
    sendOverdueAlertToDepartmentManager:=[
        StaffNoticesAlertsSettings.allInstances()>>any(true).alertWhenTicketOverdue and
        StaffNoticesAlertsSettings.allInstances()>>any(true).alertWhenTicketOverdueStaff->includes(#DepartmentManager)];
    sendOverdueAlertToDepartmentMembers:=[
        StaffNoticesAlertsSettings.allInstances()>>any(true).alertWhenTicketOverdue and
        StaffNoticesAlertsSettings.allInstances()>>any(true).alertWhenTicketOverdueStaff->includes(#DepartmentMembers)];
    staffAlertsFromEMailAddress:=[
        EmailSettings.allInstances()>>any(true).defaultStaffAlertsEmail.address];
    newOverdueTickets:=
        Ticket.allInstances()>>select(t|(System.allInstances()>>any(true).currentDateTime.value>(t.dueDatetime.value+TicketSettings.allInstances()>>any(true).ticketGracePeriod)) and
        not(t.isOverdue))->asSequence();
sizeOverdue:=newOverdueTickets->size();
ii:=1;
while ii<=sizeOverdue do
    overdueTicket:=newOverdueTickets->at(ii);
    overdueTicket.isOverdue:=true;
    if sendOverdueAlertToAssignedStaff then
        if overdueTicket.assignedStaff->notEmpty() then
            if overdueTicket.assignedStaff.status=#Enabled then
                if overdueTicket.assignedStaff.isInVacationMode=false then
                    m1:=new EMail(fromAddress:=staffAlertsFromEMailAddress,
                    toAddress:=[overdueTicket.assignedStaff.emailAddress],
                    ticketNumber:=overdueTicket.number,emailKind:="OverdueTicketAlertToStaff.allInstances()>>any(true)"));
                endif
            endif
        endif
    endif
    endif
}

if sendOverdueAlertToDepartmentManager then
    if overdueTicket.assignedDepartment.departmentManager->notEmpty() then
        //Department Manager alerts
        // Send email alerts to department manager
        // ... (similar logic as for staff alerts)
    endif
endif
if overdueTicket.assignedDepartment.departmentManager.status=#Enabled then
if overdueTicket.assignedDepartment.departmentManager.isInVacationMode=false then
m1:=new EMail(fromAddress:=staffAlertsFromEMailAddress,
toAddress:=[overdueTicket.assignedDepartment.departmentManager.emailAddress],
ticketNumber:=overdueTicket.number,emailKind:=OverdueTicketAlertToStaff.allInstances()->any(true));
endif
endif
endif
endif

if sendOverdueAlertToDepartmentMembers then
z:=1;
staff:=overdueTicket.assignedDepartment.staffMember->asSequence();
while z<staff->size() do
   currentStaffMember:=staff->at(z);
   if currentStaffMember.isInVacationMode=false then
      if currentStaffMember.status=#Enabled then
         m1:=new EMail(fromAddress:=staffAlertsFromEMailAddress,
toAddress:=currentStaffMember.emailAddress,
ticketNumber:=overdueTicket.number, emailKind:=OverdueTicketAlertToStaff.allInstances()->any(true));
      endif
   endif
   z:=z+1;
endwhile
endif
Appendix C. Test set

testprogram ConfigurationAndBasics{

  fixturecomponent CompatibleConfigurationAndBasics{

    template_default:=new EmailTemplate(name:='Default');
    template_default.internalNotes:='Email templates by default';

    ek1.emailTemplate:=template_default;
    template_default.newTicketAutoresponse.subject:='X';
    template_default.newTicketAutoresponse.message:='Y';

    ek2:=new NewMessageAutoresponse(subject:='X',message:='Y');
    ek2.emailTemplate:=template_default;
    template_default.newMessageAutoresponse.subject:='X';
    template_default.newMessageAutoresponse.message:='Y';

    ek3.emailTemplate:=template_default;
    template_default.newTicketNotice.subject:='X';
    template_default.newTicketNotice.message:='Y';

    ek4:=new OverTicketLimitNotice(subject:='X',message:='Y');
    ek4.emailTemplate:=template_default;
    template_default.overTicketLimitNotice.subject:='X';
    template_default.overTicketLimitNotice.message:='Y';

    ek5:=new TicketResponseNotice(subject:='X',message:='Y');
    ek5.emailTemplate:=template_default;
    template_default.ticketResponseNotice.subject:='X';
    template_default.ticketResponseNotice.message:='Y';

    ek6:=new NewTicketAlertToStaff(subject:='X',message:='Y');
    ek6.emailTemplate:=template_default;
    template_default.newTicketAlertToStaff.subject:='X';
    template_default.newTicketAlertToStaff.message:='Y';

    ek7:=new NewMessageAlertToStaff(subject:='X',message:='Y');
    ek7.emailTemplate:=template_default;
    template_default.newMessageAlertToStaff.subject:='X';
    template_default.newMessageAlertToStaff.message:='Y';

    ek8:=new NewInternalNoteAlertToStaff(subject:='X',message:='Y');
    ek8.emailTemplate:=template_default;
    template_default.newInternalNoteAlertToStaff.subject:='X';
    template_default.newInternalNoteAlertToStaff.message:='Y';

    ek9:=new TicketAssignedAlertToStaff(subject:='X',message:='Y');
    ek9.emailTemplate:=template_default;
    template_default.ticketAssignedAlertToStaff.subject:='X';
    template_default.ticketAssignedAlertToStaff.message:='Y';

    ek10:=new OverdueTicketAlertToStaff(subject:='X',message:='Y');
    ek10.emailTemplate:=template_default;
    template_default.overdueTicketAlertToStaff.subject:='X';
  }
template_default.overdueTicketAlertToStaff.message := 'Y';

dptGeneral := new Department(name:='General support');
dptGeneral.type := #Public;
dptGeneral.emailTemplate := template_default;
dptGeneral.newTicketAutoresponsesSent := true;
dptGeneral.newAddedMessagesNotified := true;
dptGeneral.isDefault := true;

dptTechnical := new Department(name:='Technical support');
dptTechnical.type := #Private;
dptTechnical.emailTemplate := template_default;
dptTechnical.newTicketAutoresponsesSent := true;
dptTechnical.newAddedMessagesNotified := true;

generalSupportEmailAccount := new EmailAccount(address:='general_at_support.com');
generalSupportEmailAccount.fromName := 'General questions';
generalSupportEmailAccount.defaultNewPriority := #Low;
generalSupportEmailAccount.defaultNewTicketDepartment := dptGeneral;
generalSupportEmailAccount.autoresponsesStatus := #Disabled;

technicalSupportEmailAccount := new EmailAccount(address:='technical_at_support.com');
technicalSupportEmailAccount.fromName := 'B Support';
technicalSupportEmailAccount.defaultNewPriority := #High;
technicalSupportEmailAccount.defaultNewTicketDepartment := dptTechnical;
technicalSupportEmailAccount.autoresponsesStatus := #Disabled;

dptGeneral.outgoingEmail := generalSupportEmailAccount;
dptGeneral.autorresponseEmail := generalSupportEmailAccount;

dptTechnical.outgoingEmail := technicalSupportEmailAccount;
dptTechnical.autorresponseEmail := technicalSupportEmailAccount;

generalAdministrator := new StaffMember(username:='john');
generalAdministrator.department := dptGeneral;
generalAdministrator.firstName := 'John';
generalAdministrator.lastName := 'Johny';
generalAdministrator.emailAddress := 'john_at_support.com';
generalAdministrator.officePhone := '11111';
generalAdministrator.phoneExtension := '11';
generalAdministrator.mobilePhone := '111111';
generalAdministrator.signature := 'John Johny';
generalAdministrator.password := 'xxx';
generalAdministrator.status := #Enabled;
generalAdministrator.isAdmin := true;
generalAdministrator.isInVacationMode := false;

dptTechnical.departmentManager := generalAdministrator;

maximumPrivilegesGroup := new StaffGroup(name:='Maximum Privileges Group');
maximumPrivilegesGroup.status := #Enabled;
maximumPrivilegesGroup.departmentsAccess := Set{dptGeneral, dptTechnical};
maximumPrivilegesGroup.canCreateTickets := true;
maximumPrivilegesGroup.canEditTickets := true;
maximumPrivilegesGroup.canCloseTickets := true;
maximumPrivilegesGroup.canTransferTickets := true;
maximumPrivilegesGroup.canDeleteTickets := true;
maximumPrivilegesGroup.canBanEmails := true;

generalAdministrator.staffGroup := maximumPrivilegesGroup;

//Iteration 3
minimumPrivilegesGroup := new StaffGroup(name:='Minimum Privileges Group');
minimumPrivilegesGroup.status := #Enabled;
minimumPrivilegesGroup.departmentsAccess := Set{};
minimumPrivilegesGroup.canCreateTickets := false;
minimumPrivilegesGroup.canEditTickets := false;
minimumPrivilegesGroup.canCloseTickets := false;
minimumPrivilegesGroup.canTransferTickets := false;
minimumPrivilegesGroup.canDeleteTickets:=false;
minimumPrivilegesGroup.canBanEmails:=false;

inactiveGroup:=new StaffGroup(name:='Inactive Group');
inactiveGroup.status:=#Disabled;
inactiveGroup.departmentsAccess:=Set{};
inactiveGroup.canCreateTickets:=true;
inactiveGroup.canEditTickets:=false;
inactiveGroup.canCloseTickets:=true;
inactiveGroup.canTransferTickets:=false;
inactiveGroup.canDeleteTickets:=true;
inactiveGroup.canBanEmails:=false;

generalConsultant:=new StaffMember(username:='mary');
generalConsultant.department:=dptGeneral;
generalConsultant.firstName:='Mary';
generalConsultant.lastName:='Mayer';
generalConsultant.emailAddress:='mary_at_support.com';
generalConsultant.officePhone:='22222';
generalConsultant.phoneExtension:='22';
generalConsultant.mobilePhone:='22222';
generalConsultant.signature:='Mary Mayer';
generalConsultant.password:='yyy';
generalConsultant.status:=#Enabled;
generalConsultant.isAdministrator:=false;
generalConsultant.isInVacationMode:=false;
generalConsultant.staffGroup:=maximumPrivilegesGroup;

generalConsultantVacation:=new StaffMember(username:='david');
generalConsultantVacation.department:=dptGeneral;
generalConsultantVacation.firstName:='David';
generalConsultantVacation.lastName:='Dassel';
generalConsultantVacation.emailAddress:='david_at_support.com';
generalConsultantVacation.officePhone:='33333';
generalConsultantVacation.phoneExtension:='33';
generalConsultantVacation.mobilePhone:='33333';
generalConsultantVacation.signature:='David Dassel';
generalConsultantVacation.password:='zzz';
generalConsultantVacation.status:=#Enabled;
generalConsultantVacation.isAdministrator:=false;
generalConsultantVacation.isInVacationMode:=false;
generalConsultantVacation.staffGroup:=maximumPrivilegesGroup;

technicalActive:=new StaffMember(username:='martin');
technicalActive.department:=dptTechnical;
technicalActive.firstName:='Martin';
technicalActive.lastName:='Martech';
technicalActive.emailAddress:='martin_at_support.com';
technicalActive.password:='ttt';
technicalActive.status:=#Enabled;
technicalActive.isAdministrator:=false;
technicalActive.isInVacationMode:=false;
technicalActive.staffGroup:=minimumPrivilegesGroup;

technicalInactive:=new StaffMember(username:='patricia');
technicalInactive.department:=dptTechnical;
technicalInactive.firstName:='Patricia';
technicalInactive.lastName:='Pauls';
technicalInactive.emailAddress:='patricia_at_support.com';
technicalInactive.password:='ttt';
technicalInactive.status:=#Disabled;
technicalInactive.isAdministrator:=false;
technicalInactive.isInVacationMode:=false;
technicalInactive.staffGroup:=minimumPrivilegesGroup;

emailSettings:=new EmailSettings;
emailSettings.defaultSystemEmail:=generalSupportEmailAccount;
emailSettings.defaultStaffAlertsEmail:=generalSupportEmailAccount;
emailSettings.administrationEmail:='system_at_support.com';

helpTopicUse:=new HelpTopic(name:='Use');
DEVELOPMENT OF THE CONCEPTUAL SCHEMA OF THE osTICKET SYSTEM BY APPLYING TDCM

Research report
April 2011

helpTopicUse.status:=#Enabled;
helpTopicUse.autoresponse:=#Enabled;
helpTopicUse.newTicketPriority:=#Normal;
helpTopicUse.newTicketDepartment:=dptGeneral;

helpTopicInstallation:=new HelpTopic(name:='Installation');
helpTopicInstallation.status:=#Enabled;
helpTopicInstallation.autoresponse:=#Disabled;
helpTopicInstallation.newTicketPriority:=#High;
helpTopicInstallation.newTicketDepartment:=dptTechnical;

helpTopicDisabled:=new HelpTopic(name:='Offers');
helpTopicDisabled.status:=#Disabled;
helpTopicDisabled.autoresponse:=#Disabled;
helpTopicDisabled.newTicketPriority:=#Low;
helpTopicDisabled.newTicketDepartment:=dptGeneral;

fixturecomponent GeneralSettingsOnline{
generalSettings:=new GeneralSettings;
generalSettings.status:=#Online;
generalSettings.helpdeskURL:='http://onlinesupport.com';
generalSettings.helpdeskName:='Online customer support';
generalSettings.defaultEmailTemplate:=template_default;
}

fixturecomponent GeneralSettingsOffline{
generalSettings:=new GeneralSettings;
generalSettings.status:=#Offline;
generalSettings.defaultEmailTemplate:=template_default;
}

fixturecomponent TicketSettingsSequential{
ticketSettings:=new TicketSettings;
ticketSettings.mode:=#Sequential;
ticketSettings.priority:=#Normal;
ticketSettings.customersCanChangePriority:=false;
ticketSettings.useEmailPriorityWhenAvailable:=true;
ticketSettings.openTicketsPerMailAreLimited:=true;
ticketSettings.maximumOpenTicketsPerMail:=2;
ticketSettings.ticketGracePeriod:=0;
ticketSettings.reopenedTicketsAreAssignedToLastRespondent:=true;
}

fixturecomponent TicketSettingsRandom{
ticketSettings:=new TicketSettings;
ticketSettings.mode:=#Random;
ticketSettings.priority:=#High;
ticketSettings.customersCanChangePriority:=true;
ticketSettings.useEmailPriorityWhenAvailable:=false;
ticketSettings.ticketGracePeriod:=2;
ticketSettings.reopenedTicketsAreAssignedToLastRespondent:=false;
}

fixturecomponent CustomerAutoresponsesActive{
customerAutoresponsesSettings:=new CustomerAutoresponsesSettings;
customerAutoresponsesSettings.autorespondWhenNewTicketCreatedByCustomer:=true;
customerAutoresponsesSettings.autorespondWhenNewTicketCreatedByStaff:=true;
customerAutoresponsesSettings.autorespondWhenNewMessageAppendedToTicket:=true;
customerAutoresponsesSettings.autorespondWhenMaximumOpenTicketsOfCustomer:=true;
}

fixturecomponent CustomerAutoresponsesInactive{
customerAutoresponsesSettings:=new CustomerAutoresponsesSettings;
customerAutoresponsesSettings.autorespondWhenNewTicketCreatedByCustomer:=false;
customerAutoresponsesSettings.autorespondWhenNewTicketCreatedByStaff:=false;
customerAutoresponsesSettings.autorespondWhenNewMessageAppendedToTicket:=false;
customerAutoresponsesSettings.autorespondWhenMaximumOpenTicketsOfCustomer:=false;
}
fixturecomponent StaffNoticesAlertsInactive{  
staffNoticesAlertsSettings:=new StaffNoticesAlertsSettings;  
staffNoticesAlertsSettings.alertWhenNewTicketCreated:=false;  
staffNoticesAlertsSettings.alertWhenNewMessage:=false;  
staffNoticesAlertsSettings.alertWhenInternalNote:=false;  
staffNoticesAlertsSettings.alertWhenTicketOverdue:=false;  
}

fixturecomponent StaffNoticesAlertsActive{  
staffNoticesAlertsSettings:=new StaffNoticesAlertsSettings;  
staffNoticesAlertsSettings.alertWhenNewTicketCreated:=true;  
staffNoticesAlertsSettings.alertWhenNewTicketCreatedStaff:=Set{#Administrator,#DepartmentManager,#DepartmentMembers};  
staffNoticesAlertsSettings.alertWhenNewMessage:=true;  
staffNoticesAlertsSettings.alertWhenNewMessageStaff:=Set#{LastRespondent,#AssignedStaff,#DepartmentManager};  
staffNoticesAlertsSettings.alertWhenInternalNote:=true;  
staffNoticesAlertsSettings.alertWhenInternalNoteStaff:=Set#{LastRespondent,#AssignedStaff,#DepartmentManager};  
staffNoticesAlertsSettings.alertWhenTicketOverdue:=true;  
staffNoticesAlertsSettings.alertWhenTicketOverdueStaff:=Set#{AssignedStaff,#DepartmentManager,#DepartmentMembers};  
}

test testConfiguration1{  
load CompatibleConfigurationAndBasics;  
load GeneralSettingsOnline;  
load TicketSettingsSequential;  
load CustomerAutoresponsesActive;  
load StaffNoticesAlertsActive;  
assert consistency;  
}

test testConfiguration2{  
load CompatibleConfigurationAndBasics;  
load GeneralSettingsOnline;  
load TicketSettingsSequential;  
load CustomerAutoresponsesActive;  
load StaffNoticesAlertsInactive;  
assert consistency;  
}

test testConfiguration3{  
load CompatibleConfigurationAndBasics;  
load GeneralSettingsOnline;  
load TicketSettingsSequential;  
load CustomerAutoresponsesInactive;  
load StaffNoticesAlertsActive;  
assert consistency;  
}

test testConfiguration4{  
load CompatibleConfigurationAndBasics;  
load GeneralSettingsOnline;  
load TicketSettingsSequential;  
load CustomerAutoresponsesInactive;  
load StaffNoticesAlertsInactive;  
assert consistency;  
}

test testConfiguration5{  
load CompatibleConfigurationAndBasics;  
load GeneralSettingsOnline;  
load TicketSettingsRandom;  
load CustomerAutoresponsesActive;  
load StaffNoticesAlertsActive;  
assert consistency;  
}

test testConfiguration6{  
load CompatibleConfigurationAndBasics;  
load GeneralSettingsOnline;  
load GeneralSettingsOnline;  
load GeneralSettingsOnline;
DEVELOPMENT OF THE CONCEPTUAL
SCHEMA OF THE osTICKET
SYSTEM BY APPLYING TDCM

April 2011

Research report

load TicketSettingsRandom;
load CustomerAutoresponsesActive;
load StaffNoticesAlertsInactive;
assert consistency;
}

test testConfiguration7{
load CompatibleConfigurationAndBasics;
load GeneralSettingsOnline;
load TicketSettingsRandom;
load CustomerAutoresponsesActive;
load StaffNoticesAlertsInactive;
assert consistency;
}

test testConfiguration8{
load CompatibleConfigurationAndBasics;
load GeneralSettingsOnline;
load TicketSettingsRandom;
load CustomerAutoresponsesInactive;
load StaffNoticesAlertsActive;
assert consistency;
}

test testConfiguration9{
load CompatibleConfigurationAndBasics;
load GeneralSettingsOffline;
load TicketSettingsSequential;
load CustomerAutoresponsesActive;
load StaffNoticesAlertsActive;
assert consistency;
}

test testConfiguration10{
load CompatibleConfigurationAndBasics;
load GeneralSettingsOffline;
load TicketSettingsSequential;
load CustomerAutoresponsesActive;
load StaffNoticesAlertsInactive;
assert consistency;
}

test testConfiguration11{
load CompatibleConfigurationAndBasics;
load GeneralSettingsOffline;
load TicketSettingsSequential;
load CustomerAutoresponsesInactive;
load StaffNoticesAlertsActive;
assert consistency;
}

test testConfiguration12{
load CompatibleConfigurationAndBasics;
load GeneralSettingsOffline;
load TicketSettingsSequential;
load CustomerAutoresponsesInactive;
load StaffNoticesAlertsInactive;
assert consistency;
}

test testConfiguration13{
load CompatibleConfigurationAndBasics;
load GeneralSettingsOffline;
load TicketSettingsRandom;
load CustomerAutoresponsesActive;
load StaffNoticesAlertsActive;
assert consistency;
}

test testConfiguration14{
load CompatibleConfigurationAndBasics;
DEVELOPMENT OF THE CONCEPTUAL
SCHEMA OF THE osTICKET
SYSTEM BY APPLYING TDCM

April 2011

Department of Service and Information System Engineering

load GeneralSettingsOffline;
load TicketSettingsRandom;
load CustomerAutoresponsesActive;
load StaffNoticesAlertsInactive;
assert consistency;
}

test testConfiguration15{
load CompatibleConfigurationAndBasics;
load GeneralSettingsOffline;
load TicketSettingsRandom;
load CustomerAutoresponsesInactive;
load StaffNoticesAlertsActive;
assert consistency;
}

test testConfiguration16{
load CompatibleConfigurationAndBasics;
load GeneralSettingsOffline;
load TicketSettingsRandom;
load CustomerAutoresponsesInactive;
load StaffNoticesAlertsInactive;
assert consistency;
}

testprogram TicketsManagementAndTracking{

now := new Datetime(value:=1);
after:=new Datetime(value:=2);
sys := new System(currentDateTime:=now);
sys.aleat:=5;

fixturecomponent CompatibleConfigurationAndBasics[

    template_default:=new EmailTemplate(name:='Default');
    template_default.internalNotes:='Email templates by default';

    ek1.emailTemplate:=template_default;
    template_default.newTicketAutoresponse.subject:='X';
    template_default.newTicketAutoresponse.message:='Y';

    ek2:=new NewMessageAutoresponse(subject:='X',message:='Y');
    ek2.emailTemplate:=template_default;
    template_default.newMessageAutoresponse.subject:='X';
    template_default.newMessageAutoresponse.message:='Y';

    ek3.emailTemplate:=template_default;
    template_default.newTicketNotice.subject:='X';
    template_default.newTicketNotice.message:='Y';

    ek4:=new OverTicketLimitNotice(subject:='X',message:='Y');
    ek4.emailTemplate:=template_default;
    template_default.overTicketLimitNotice.subject:='X';
    template_default.overTicketLimitNotice.message:='Y';

    ek5:=new TicketResponseNotice(subject:='X',message:='Y');
    ek5.emailTemplate:=template_default;
    template_default.ticketResponseNotice.subject:='X';
    template_default.ticketResponseNotice.message:='Y';

    ek6:=new NewTicketAlertToStaff(subject:='X',message:='Y');
    ek6.emailTemplate:=template_default;
    template_default.newTicketAlertToStaff.subject:='X';
    template_default.newTicketAlertToStaff.message:='Y';

    ek7:=new NewMessageAlertToStaff(subject:='X',message:='Y');
    ek7.emailTemplate:=template_default;
}
template_default.newMessageAlertToStaff.subject = 'X';
template_default.newMessageAlertToStaff.message = 'Y';

ek8 := new NewInternalNoteAlertToStaff(subject = 'X', message = 'Y');
ek8.emailTemplate := template_default;
template_default.newInternalNoteAlertToStaff.subject = 'X';
template_default.newInternalNoteAlertToStaff.message = 'Y';

ek9 := new TicketAssignedAlertToStaff(subject = 'X', message = 'Y');
ek9.emailTemplate := template_default;
template_default.ticketAssignedAlertToStaff.subject = 'X';
template_default.ticketAssignedAlertToStaff.message = 'Y';

ek10 := new OverdueTicketAlertToStaff(subject = 'X', message = 'Y');
ek10.emailTemplate := template_default;
template_default.overdueTicketAlertToStaff.subject = 'X';
template_default.overdueTicketAlertToStaff.message = 'Y';

dptGeneral := new Department(name = 'General support');
dptGeneral.type := #Public;
dptGeneral.emailTemplate := template_default;
dptGeneral.newTicketAutoresponsesIsSent := true;
dptGeneral.newAddedMessagesIsNotified := true;
dptGeneral.isDefault := true;

dptTechnical := new Department(name = 'Technical support');
dptTechnical.type := #Private;
dptTechnical.emailTemplate := template_default;
dptTechnical.newTicketAutoresponsesIsSent := true;
dptTechnical.newAddedMessagesIsNotified := true;

generalSupportEmailAccount := new EmailAccount(address = 'general_at_support.com');
generalSupportEmailAccount.fromName = 'General questions';
generalSupportEmailAccount.defaultNewPriority := #Low;
generalSupportEmailAccount.defaultNewTicketDepartment := dptGeneral;
generalSupportEmailAccount.autoresponsesStatus := #Enabled;

technicalSupportEmailAccount := new EmailAccount(address = 'technical_at_support.com');
technicalSupportEmailAccount.fromName = 'B Support';
technicalSupportEmailAccount.defaultNewPriority := #High;
technicalSupportEmailAccount.defaultNewTicketDepartment := dptTechnical;
technicalSupportEmailAccount.autoresponsesStatus := #Disabled;

dptGeneral.outgoingEmail := generalSupportEmailAccount;
dptGeneral.autoresponseEmail := generalSupportEmailAccount;

dptTechnical.outgoingEmail := technicalSupportEmailAccount;
dptTechnical.autoresponseEmail := technicalSupportEmailAccount;

generalAdministrator := new StaffMember(username = 'john');
generalAdministrator.department := dptGeneral;
generalAdministrator.firstName := 'John';
generalAdministrator.lastName := 'Johny';
generalAdministrator.emailAddress := 'john_at_support.com';
generalAdministrator.officePhone := '111111';
generalAdministrator.phoneExtension := '11';
generalAdministrator.mobilePhone := '111111';
generalAdministrator.signature := 'John Johny';
generalAdministrator.password := 'xxx';
generalAdministrator.status := #Enabled;
generalAdministrator.isAdministrator := true;
generalAdministrator.isInVacationMode := false;

dptTechnical.departmentManager := generalAdministrator;

maximumPrivilegesGroup := new StaffGroup(name := 'Maximum Privileges Group');
maximumPrivilegesGroup.status := #Enabled;
maximumPrivilegesGroup.departmentsAccess := Set{dptGeneral, dptTechnical};
maximumPrivilegesGroup.canCreateTickets := true;
maximumPrivilegesGroup.canEditTickets := true;
maximumPrivilegesGroup.canCloseTickets := true;
maximumPrivilegesGroup.canTransferTickets := true;
maximumPrivilegesGroup.canDeleteTickets := true;
maximumPrivilegesGroup.canBanEmails := true;

generalAdministrator.staffGroup := maximumPrivilegesGroup;

// Iteration 3
minimumPrivilegesGroup := new StaffGroup(name := 'Minimum Privileges Group');
minimumPrivilegesGroup.status := #Enabled;
minimumPrivilegesGroup.departmentsAccess := Set{};
minimumPrivilegesGroup.canCreateTickets := false;
minimumPrivilegesGroup.canEditTickets := false;
minimumPrivilegesGroup.canCloseTickets := false;
minimumPrivilegesGroup.canTransferTickets := false;
minimumPrivilegesGroup.canDeleteTickets := false;
minimumPrivilegesGroup.canBanEmails := false;

inactiveGroup := new StaffGroup(name := 'Inactive Group');
inactiveGroup.status := #Disabled;
inactiveGroup.departmentsAccess := Set{};
inactiveGroup.canCreateTickets := true;
inactiveGroup.canEditTickets := false;
inactiveGroup.canCloseTickets := true;
inactiveGroup.canTransferTickets := false;
inactiveGroup.canDeleteTickets := true;
inactiveGroup.canBanEmails := false;

generalConsultant := new StaffMember(username := 'mary');
generalConsultant.department := dptGeneral;
generalConsultant.firstName := 'Mary';
generalConsultant.lastName := 'Mayer';
generalConsultant.emailAddress := 'mary_at_support.com';
generalConsultant.officePhone := '22222';
generalConsultant.phoneExtension := '22';
generalConsultant.mobilePhone := '22222';
generalConsultant.signature := 'Mary Mayer';
generalConsultant.password := 'yyy';
generalConsultant.status := #Enabled;
generalConsultant.isAdministrator := false;
generalConsultant.isInVacationMode := false;
generalConsultant.staffGroup := maximumPrivilegesGroup;

generalConsultantVacation := new StaffMember(username := 'david');
generalConsultantVacation.department := dptGeneral;
generalConsultantVacation.firstName := 'David';
generalConsultantVacation.lastName := 'Dassel';
generalConsultantVacation.emailAddress := 'david_at_support.com';
generalConsultantVacation.officePhone := '33333';
generalConsultantVacation.phoneExtension := '33';
generalConsultantVacation.mobilePhone := '33333';
generalConsultantVacation.signature := 'David Dassel';
generalConsultantVacation.password := 'zzz';
generalConsultantVacation.status := #Enabled;
generalConsultantVacation.isAdministrator := false;
generalConsultantVacation.isInVacationMode := true;
generalConsultantVacation.staffGroup := maximumPrivilegesGroup;

technicalActive := new StaffMember(username := 'martin');
technicalActive.department := dptTechnical;
technicalActive.firstName := 'Martin';
technicalActive.lastName := 'Martech';
technicalActive.emailAddress := 'martin_at_support.com';
technicalActive.password := 'ttt';
technicalActive.status := #Enabled;
technicalActive.isAdministrator := false;
technicalActive.isInVacationMode := false;
technicalActive.staffGroup := minimumPrivilegesGroup;
technicalInactive := new StaffMember(username := 'patricia');
technicalInactive.department:=dptTechnical;
technicalInactive.firstName:='Patricia';
technicalInactive.lastName:=‘Pauls’;
technicalInactive.emailAddress:='patricia_at_support.com';
technicalInactive.password:='uuu';
technicalInactive.status:=#Disabled;
technicalInactive.isAdministrator:=false;
technicalInactive.isInVacationMode:=false;
technicalInactive.staffGroup:=minimumPrivilegesGroup;

emailSettings:=new EmailSettings;
emailSettings.defaultSystemEmail:=generalSupportEmailAccount;
emailSettings.defaultStaffAlertsEmail:=generalSupportEmailAccount;
emailSettings.administrationEmail:=’system_at_support.com’;

helpTopicUse:=new HelpTopic(name:='Use');
helpTopicUse.status:=#Enabled;
helpTopicUse.autoresponse:=#Enabled;
helpTopicUse.newTicketPriority:=#Normal;
helpTopicUse.newTicketDepartment:=dptGeneral;

helpTopicInstallation:=new HelpTopic(name:='Installation');
helpTopicInstallation.status:=#Enabled;
helpTopicInstallation.autoresponse:=#Disabled;
helpTopicInstallation.newTicketPriority:=#High;
helpTopicInstallation.newTicketDepartment:=dptTechnical;

helpTopicDisabled:=new HelpTopic(name:='Offers');
helpTopicDisabled.status:=#Disabled;
helpTopicDisabled.autoresponse:=#Disabled;
helpTopicDisabled.newTicketPriority:=#Low;
helpTopicDisabled.newTicketDepartment:=dptGeneral;

}

fixturecomponent GeneralSettingsOnline{
    generalSettings:=new GeneralSettings;
    generalSettings.status:=#Online;
    generalSettings.helpdeskURL:='http://onlinesupport.com';
    generalSettings.helpdeskName:='Online customer support';
    generalSettings.defaultEmailTemplate:=template_default;
}

fixturecomponent GeneralSettingsOffline{
    generalSettings:=new GeneralSettings;
    generalSettings.helpdeskURL:='http://offlinesupport.com';
    generalSettings.status:=#Offline;
    generalSettings.defaultEmailTemplate:=template_default;
}

fixturecomponent TicketSettingsSequential{
    ticketSettings:=new TicketSettings;
    ticketSettings.mode:=#Sequential;
    ticketSettings.priority:=#Normal;
    ticketSettings.customersCanChangePriority:=false;
    ticketSettings.useEmailPriorityWhenAvailable:=true;
    ticketSettings.openTicketsPerMailAreLimited:=true;
    ticketSettings.maximumOpenTicketsPerMail:=2;
    ticketSettings.ticketGracePeriod:=0;
    ticketSettings.reopenedTicketsAreAssignedToLastResponder:=true;
}

fixturecomponent TicketSettingsRandom{
    ticketSettings:=new TicketSettings;
    ticketSettings.mode:=#Random;
    ticketSettings.priority:=#High;
    ticketSettings.customersCanChangePriority:=true;
    ticketSettings.useEmailPriorityWhenAvailable:=false;
    //ticketSettings.maximumOpenTicketsPerMail:=#Unlimited;
    ticketSettings.ticketGracePeriod:=2;
    ticketSettings.reopenedTicketsAreAssignedToLastResponder:=false;
}
fixturecomponent CustomerAutoresponsesActive{
    customerAutoresponsesSettings:=new CustomerAutoresponsesSettings;
    customerAutoresponsesSettings.autorespondWhenNewTicketCreatedByCustomer:=true;
    customerAutoresponsesSettings.autorespondWhenNewTicketCreatedByStaff:=true;
    customerAutoresponsesSettings.autorespondWhenNewMessageAppendedToTicket:=true;
    customerAutoresponsesSettings.autorespondWhenMaximumOpenTicketsOfCustomer:=true;
}

fixturecomponent CustomerAutoresponsesInactive{
    customerAutoresponsesSettings:=new CustomerAutoresponsesSettings;
    customerAutoresponsesSettings.autorespondWhenNewTicketCreatedByCustomer:=false;
    customerAutoresponsesSettings.autorespondWhenNewTicketCreatedByStaff:=false;
    customerAutoresponsesSettings.autorespondWhenNewMessageAppendedToTicket:=false;
    customerAutoresponsesSettings.autorespondWhenMaximumOpenTicketsOfCustomer:=false;
}

fixturecomponent StaffNoticesAlertsInactive{
    staffNoticesAlertsSettings:=new StaffNoticesAlertsSettings;
    staffNoticesAlertsSettings.alertWhenNewTicketCreated:=false;
    staffNoticesAlertsSettings.alertWhenNewMessage:=false;
    staffNoticesAlertsSettings.alertWhenInternalNote:=false;
    staffNoticesAlertsSettings.alertWhenTicketOverdue:=false;
}

fixturecomponent StaffNoticesAlertsActive{
    staffNoticesAlertsSettings:=new StaffNoticesAlertsSettings;
    staffNoticesAlertsSettings.alertWhenNewTicketCreatedStaff:=Set{#Administrator,#DepartmentManager,#DepartmentMembers};
    staffNoticesAlertsSettings.alertWhenNewMessage:=true;
    staffNoticesAlertsSettings.alertWhenInternalNoteStaff:=Set{#LastRespondent,#AssignedStaff,#DepartmentManager};
    staffNoticesAlertsSettings.alertWhenTicketOverdueStaff:=Set{#AssignedStaff,#DepartmentManager,#DepartmentMembers};
}

fixturecomponent testConfiguration1{
    load CompatibleConfigurationAndBasics;
    load GeneralSettingsOnline;
    load TicketSettingsSequential;
    load CustomerAutoresponsesActive;
    load StaffNoticesAlertsActive;
}

fixturecomponent testConfiguration2{
    load CompatibleConfigurationAndBasics;
    load GeneralSettingsOnline;
    load TicketSettingsSequential;
    load CustomerAutoresponsesActive;
    load StaffNoticesAlertsInactive;
}

fixturecomponent testConfiguration3{
    load CompatibleConfigurationAndBasics;
    load GeneralSettingsOnline;
    load TicketSettingsSequential;
    load CustomerAutoresponsesInactive;
    load StaffNoticesAlertsActive;
}

fixturecomponent testConfiguration4{
    load CompatibleConfigurationAndBasics;
    load GeneralSettingsOnline;
    load TicketSettingsSequential;
    load CustomerAutoresponsesInactive;
    load StaffNoticesAlertsInactive;
}
DEVELOPMENT OF THE CONCEPTUAL SCHEMA OF THE osTICKET SYSTEM BY APPLYING TDCM

fixturecomponent testConfiguration5{
  load CompatibleConfigurationAndBasics;
  load GeneralSettingsOnline;
  load TicketSettingsRandom;
  load CustomerAutoresponsesActive;
  load StaffNoticesAlertsActive;
}

fixturecomponent testConfiguration6{
  load CompatibleConfigurationAndBasics;
  load GeneralSettingsOnline;
  load TicketSettingsRandom;
  load CustomerAutoresponsesActive;
  load StaffNoticesAlertsInactive;
}

fixturecomponent testConfiguration7{
  load CompatibleConfigurationAndBasics;
  load GeneralSettingsOnline;
  load TicketSettingsRandom;
  load CustomerAutoresponsesInactive;
  load StaffNoticesAlertsActive;
}

fixturecomponent testConfiguration8{
  load CompatibleConfigurationAndBasics;
  load GeneralSettingsOnline;
  load TicketSettingsRandom;
  load CustomerAutoresponsesInactive;
  load StaffNoticesAlertsInactive;
}

fixturecomponent testConfiguration9{
  load CompatibleConfigurationAndBasics;
  load GeneralSettingsOffline;
  load TicketSettingsSequential;
  load CustomerAutoresponsesActive;
  load StaffNoticesAlertsActive;
}

fixturecomponent testConfiguration10{
  load CompatibleConfigurationAndBasics;
  load GeneralSettingsOffline;
  load TicketSettingsSequential;
  load CustomerAutoresponsesActive;
  load StaffNoticesAlertsInactive;
}

fixturecomponent testConfiguration11{
  load CompatibleConfigurationAndBasics;
  load GeneralSettingsOffline;
  load TicketSettingsSequential;
  load CustomerAutoresponsesInactive;
  load StaffNoticesAlertsActive;
}

fixturecomponent testConfiguration12{
  load CompatibleConfigurationAndBasics;
  load GeneralSettingsOffline;
  load TicketSettingsSequential;
  load CustomerAutoresponsesInactive;
  load StaffNoticesAlertsInactive;
}

fixturecomponent testConfiguration13{
  load CompatibleConfigurationAndBasics;
  load GeneralSettingsOffline;
  load TicketSettingsRandom;
  load CustomerAutoresponsesActive;
  load StaffNoticesAlertsActive;
}
fixturecomponent testConfiguration14{
  load CompatibleConfigurationAndBasics;
  load GeneralSettingsOffline;
  load TicketSettingsRandom;
  load CustomerAutoresponsesActive;
  load StaffNoticesAlertsInactive;
}

fixturecomponent testConfiguration15{
  load CompatibleConfigurationAndBasics;
  load GeneralSettingsOffline;
  load TicketSettingsRandom;
  load CustomerAutoresponsesInactive;
  load StaffNoticesAlertsActive;
}

fixturecomponent testConfiguration16{
  load CompatibleConfigurationAndBasics;
  load GeneralSettingsOffline;
  load TicketSettingsRandom;
  load CustomerAutoresponsesInactive;
  load StaffNoticesAlertsInactive;
}

fixturecomponent created_tickets{
  li := new LogIn(username:='mary', password:='yyy');
  assert occurrence li;

  nt1:=new NewTicketOffline;
  nt1.fullName:='Mary Marnes';
  nt1.email:='mary_at_marnes.mar';
  nt1.telephone:='xxxxxxxx';
  nt1.ext:='xxxxxxxx';
  nt1.source:=#Phone;
  nt1.assignedDepartment:=dptTechnical;
  nt1.helpTopic:=helpTopicInstallation;
  nt1.subject:='Error operating system';
  nt1.message:='The installation process does not finish....';
  nt1.internalNote:='It seems that the correct installer is being used';
  dt2:=new Datetime(value:=[(sys.currentDateTime.value+2)]);
  nt1.dueDatetime:=dt2;
  nt1.priority:=#Normal;
  nt1.assignedStaff:=generalConsultant;
  nt1.creator:=generalConsultant;
  assert occurrence nt1;
  ticket1 := nt1.createdTicket;

  nt2:=new NewTicketOffline;
  nt2.fullName:='John Johnes';
  nt2.email:='mary_at_marnes.mar';
  nt2.source:=#Other;
  nt2.assignedDepartment:=dptGeneral;
  nt2.helpTopic:=helpTopicUse;
  nt2.subject:='Can I reply a ticket?';
  nt2.message:='I do not know how to reply a ticket';
  nt2.priority:=#High;
  nt2.assignedStaff:=generalConsultant;
  nt2.creator:=generalConsultant;
  assert occurrence nt2;
  ticket2 := nt2.createdTicket;

  lo := new LogOut(staffMember:=generalConsultant);
  assert occurrence lo;

  li := new LogIn(username:='john', password:='xxx');
  assert occurrence li;

  nt3:=new NewTicketOffline;
  nt3.fullName:='Martin Pope';
  nt3.email:='martin_at_pope.mar';

nt3.source:=#Phone;
nt3.assignedDepartment:=dptTechnical;
nt3.helpTopic:=helpTopicUse;
nt3.subject:=‘Error while login’;
nt3.message:="I get an error when I try to login’;
nt3.priority:=#Low;
dt3:=new Datetime(value:=[(sys.currentDateTime.value+5)]);
nt3.dueDatetime:=dt3;
nt3.assignedStaff:=technicalActive;
assert occurrence nt3;
ticket3 := nt3.createdTicket;

dt4:=new Datetime(value:=[(sys.currentDateTime.value+1)]);
sys.currentDateTime:=dt4;

lo := new LogOut(staffMember:=generalAdministrator);
assert occurrence lo;

nt4:=new NewTicketOnline;
nt4.fullName:='James Jordan';
nt4.email:='james_at_jordan.jam';
nt4.helpTopic:=helpTopicUse;
nt4.subject:="Reopening ticket’;
nt4.message:="I do not know how to reopen one of my closed tickets’;
assert occurrence nt4;
ticket4 := nt4.createdTicket;

nt5:=new NewTicketByEmail;
t5.toAddress:="technical_at_support.com’;
t5.fromName:="Marta Johnes’;
t5.fromAddress:="marta_at_johnes.mar’;
t5.subject:="See my tickets’;
nt5.message:="Can I see my tickets?’;
assert occurrence nt5;
ticket5 := nt5.createdTicket;
}

test S1{
load testConfiguration1;

nt:=new NewTicketOnline;
nt.fullName:='Mary Marnes';
nt.email:='mary_at_marnes.mar';
nt.telephone:='xxxxxxxx';
nt.ext:='xxxxxxxx';
nt.helpTopic:=helpTopicInstallation;
nt.subject:="Error operating system’;
nt.message:="The installation process does not finish....’;
assert occurrence nt;
ticket1:=nt.createdTicket;

assert equals ticket1.number 1;
assert equals ticket1.ticketStatus #Open;
assert equals ticket1.subject ‘Error operating system’;
assert equals ticket1.priority #High;
assert true ticket1.assignedStaff->isEmpty();
assert equals ticket1.source #Web;
assert equals ticket1.creationDatetime sys.currentDateTime;
assert true ticket1.dueDatetime.isUndefined();
assert true ticket1.lastResponseDatetime.isUndefined();
assert equals ticket1.assignedDepartment dptTechnical;

assert true [ticket1.ticketThreadMessage->one(m|m.datetime=sys.currentDateTime and m.text='The installation process does not finish....’ and m.author='Mary Marnes’)];

assert equals ticket1.lastMessageDatetime sys.currentDateTime;
no autoreponses

assert true [|not(EMail.allInstances()->exists(e|e.emailKind.oclIsTypeOf(NewTicketAutoresponse)))];

// notice to administrator
assert true [EMail.allInstances()->exists(e|e.emailKind.oclIsTypeOf(NewTicketAlertToStaff) and
            e.fromAddress='general_at_support.com' and
            e.toAddress='system_at_support.com' and
            e.ticketNumber=1)];

// notice to department manager
assert true [EMail.allInstances()->exists(e|e.emailKind.oclIsTypeOf(NewTicketAlertToStaff) and
            e.fromAddress='general_at_support.com' and
            e.toAddress='john_at_support.com' and
            e.ticketNumber=1)];

// notice to department members
assert true [EMail.allInstances()->exists(e|e.emailKind.oclIsTypeOf(NewTicketAlertToStaff) and
            e.fromAddress='general_at_support.com' and
            e.toAddress='martin_at_support.com' and
            e.ticketNumber=1)];

// TICKET 2
ticketSettings.customersCanChangePriority:=true;

nt2:=new NewTicketOnline;
nt2.fullName:='James Jordan';
nt2.email:='james_at_jordan.jam';
nt2.telephone:='xxxxxxxx';
nt2.ext:='xxxxxxxx';
nt2.priority:=#Low;
nt2.helpTopic:=helpTopicUse;
nt2.subject:='Reopening ticket';
nt2.message:='I do not know how to reopen one of my closed tickets';
assert occurrence nt2;

ticket2:=nt2.createdTicket;
assert equals ticket2.number 2;
assert equals ticket2.priority #Low;
assert equals ticket2.assignedDepartment dptGeneral;

// autoreponses

assert true [EMail.allInstances()->exists(e|e.emailKind.oclIsTypeOf(NewTicketAutoresponse) and
            e.fromAddress='general_at_support.com' and
            e.toAddress='james_at_jordan.jam' and
            e.ticketNumber=2)];

// notice to administrator
assert true [EMail.allInstances()->exists(e|e.emailKind.oclIsTypeOf(NewTicketAlertToStaff) and
            e.fromAddress='general_at_support.com' and
            e.toAddress='system_at_support.com' and
            e.ticketNumber=2)];

// notice to department members
assert true [EMail.allInstances()->exists(e|e.emailKind.oclIsTypeOf(NewTicketAlertToStaff) and
            e.fromAddress='general_at_support.com' and
            e.toAddress='john_at_support.com' and
            e.ticketNumber=2)];
assert true [EMail.allInstances()->exists(e|e.emailKind.oclIsTypeOf(NewTicketAlertToStaff) and
            e.fromAddress='general_at_support.com' and
            e.toAddress='mary_at_support.com' and
            e.ticketNumber=2)];
assert false [EMail.allInstances()->exists(e|e.emailKind.oclIsTypeOf(NewTicketAlertToStaff) and
            e.fromAddress='general_at_support.com' and
            e.toAddress='david_at_support.com' and
            e.ticketNumber=2)];
DEVELOPMENT OF THE CONCEPTUAL SCHEMA OF THE osTICKET SYSTEM BY APPLYING TDCM

Research report
April 2011

Department of Service and Information System Engineering

{test S2[
  load testConfiguration1;
  nt1:=new NewTicketOnline;
  nt1.fullName='Mary Marnes';
  nt1.email='mary_at_marnes.mar';
  nt1.telephone='xxxxxxxx';
  nt1.ext='xxxxxxxx';
  nt1.helpTopic=helpTopicInstallation;
  nt1.subject='Error operating system';
  nt1.message='The installation process does not finish....';
  assert occurrence nt1;
  ticket1:=nt1.createdTicket;
  nt2:=new NewTicketOnline;
  nt2.fullName='Mary Marnes';
  nt2.email='mary_at_marnes.mar';
  nt2.telephone='xxxxxxxx';
  nt2.ext='xxxxxxxx';
  nt2.helpTopic=helpTopicInstallation;
  nt2.subject='Reopening ticket';
  nt2.message='I do not know how to reopen one of my closed tickets';
  assert occurrence nt2;
  ticket2:=nt2.createdTicket;
  nt3:=new NewTicketOnline;
  nt3.fullName='Mary Marnes';
  nt3.email='mary_at_marnes.mar';
  nt3.telephone='xxxxxxxx';
  nt3.ext='xxxxxxxx';
  nt3.helpTopic=helpTopicInstallation;
  nt3.subject='Customize graphical interface';
  nt3.message='May I change the background color?';
  assert non-occurrence nt3;
]}

test S3[
  load testConfiguration8;
  nt1:=new NewTicketOnline;
  nt1.fullName='Mary Marnes';
  nt1.email='mary_at_marnes.mar';
  nt1.telephone='xxxxxxxx';
  nt1.ext='xxxxxxxx';
  nt1.priority=#Normal;
  nt1.helpTopic=helpTopicInstallation;
  nt1.subject='Error operating system';
  nt1.message='The installation process does not finish....';
  assert occurrence nt1;
  ticket1:=nt1.createdTicket;
  assert equals ticket1.number 5;
  //5 is the aleatory number specified for testing purposes
  assert equals ticket1.assignedDepartment dptTechnical;
  //no autoresponses
  assert true [not(EMail.allInstances()->exists(e|e.emailKind.oclIsTypeOf(NewTicketAutoresponse)))];
  //notice to administrator
  assert false [EMail.allInstances()->exists(e|e.emailKind.oclIsTypeOf(NewTicketAlertToStaff) and e.fromAddress='general_at_support.com') and

e.toAddress='system_at_support.com' and
e.ticketNumber=5];

// notice to department manager
assert false [EMail.allInstances()->exists(e|e.emailKind.oclIsTypeOf(NewTicketAlertToStaff) and
e.fromAddress='general_at_support.com' and
e.toAddress='john_at_support.com' and
e.ticketNumber=5)];

// notice to department members
assert false [EMail.allInstances()->exists(e|e.emailKind.oclIsTypeOf(NewTicketAlertToStaff) and
e.fromAddress='general_at_support.com' and
e.toAddress='martin_at_support.com' and
e.ticketNumber=5)];
}

test S4{
load testConfiguration3;

nt1::new NewTicketOnline;
nt1.fullName:='James Jordan';
nt1.email:='james_at_jordan.jam';
nt1.telephone:='xxxxxxxx';
nt1.ext:='xxxxxxxx';
nt1.helpTopic:=helpTopicUse;
nt1.subject:='Reopening ticket';
nt1.message:='I do not know how to reopen one of my closed tickets';
assert occurrence nt1;

ticket1:=nt1.createdTicket;

assert equals ticket1.assignedDepartment dptGeneral;

// no autoresponses
assert true [not(EMail.allInstances()->exists(e|e.emailKind.oclIsTypeOf(NewTicketAutoresponse)))];

// notice to administrator
assert true [E-Mail.allInstances()->exists(e|e.emailKind.oclIsTypeOf(NewTicketAlertToStaff) and
e.fromAddress='general_at_support.com' and
e.toAddress='system_at_support.com' and
e.ticketNumber=1)];

// notice to department members
assert true [E-Mail.allInstances()->exists(e|e.emailKind.oclIsTypeOf(NewTicketAlertToStaff) and
e.fromAddress='general_at_support.com' and
e.toAddress='john_at_support.com' and
e.ticketNumber=1)];

assert true [E-Mail.allInstances()->exists(e|e.emailKind.oclIsTypeOf(NewTicketAlertToStaff) and
e.fromAddress='general_at_support.com' and
e.toAddress='mary_at_support.com' and
e.ticketNumber=1)];

assert false [E-Mail.allInstances()->exists(e|e.emailKind.oclIsTypeOf(NewTicketAlertToStaff) and
e.fromAddress='general_at_support.com' and
e.toAddress='david_at_support.com' and
e.ticketNumber=1)];
}

test S5{
load testConfiguration4;
helpTopicUse.status:=#Disabled;
helpTopicInstallation.status:=#Disabled;

nt1::new NewTicketOnline;
nt1.fullName:='James Jordan';
nt1.email:='james_at_jordan.jam';
nt1.subject:='Reopening ticket';
nt1.message:='I do not know how to reopen one of my closed tickets';
assert occurrence nt1;

ticket1:=nt1.createdTicket;

assert equals ticket1.assignedDepartment dptGeneral;
assert true ticket1.helpTopic->isEmpty();
assert equals ticket1.priority #Normal;

//no autoreponses
assert false [EMail.allInstances()->exists(e|e.emailKind.oclIsTypeOf(NewTicketAutoresponse))];

//notice to administrator
assert false [EMail.allInstances()->exists(e|e.emailKind.oclIsTypeOf(NewTicketAlertToStaff) and
e.fromAddress='general_at_support.com' and
e.toAddress='system_at_support.com' and
e.ticketNumber=1)];

//notice to department members
assert false [EMail.allInstances()->exists(e|e.emailKind.oclIsTypeOf(NewTicketAlertToStaff) and
e.fromAddress='general_at_support.com' and
e.toAddress='john_at_support.com' and
e.ticketNumber=1)];

assert false [EMail.allInstances()->exists(e|e.emailKind.oclIsTypeOf(NewTicketAlertToStaff) and
e.fromAddress='general_at_support.com' and
e.toAddress='mary_at_support.com' and
e.ticketNumber=1)];

assert false [EMail.allInstances()->exists(e|e.emailKind.oclIsTypeOf(NewTicketAlertToStaff) and
e.fromAddress='general_at_support.com' and
e.toAddress='david_at_support.com' and
e.ticketNumber=1)];

}

test S6{
load testConfiguration5;
nt1:=new NewTicketOnline;
nt1.fullName:='James Jordan';
nt1.email:='james_at_jordan.jam';
nt1.subject:='Reopening ticket';
nt1.message:='I do not know how to reopen one of my closed tickets';
nt1.helpTopic:=helpTopicDisabled;
nt1.priority:=#Low;
assert non-occurrence nt1;
}

test S7{
load testConfiguration9;
nt1:=new NewTicketOnline;
nt1.fullName:='James Jordan';
nt1.email:='james_at_jordan.jam';
nt1.subject:='Reopening ticket';
nt1.message:='I do not know how to reopen one of my closed tickets';
nt1.helpTopic:=helpTopicUse;
nt1.priority:=#Low;
assert non-occurrence nt1;
}

test S8{
load testConfiguration3;
nt1:=new NewTicketByEmail;
nt1.toAddress:='general_at_support.com';
nt1.fromName:='James Jordan';
nt1.fromAddress:='james_at_jordan.jam';
nt1.subject:='Ticket priority';
nt1.message:='How can I change the priority of one of my tickets?';
assert occurrence nt1;
}
ticket1:=nt1.createdTicket;

assert equals ticket1.number 1;
assert equals ticket1.ticketStatus #Open;
assert equals ticket1.subject 'Ticket priority';
assert equals ticket1.priority #Low;
assert true ticket1.assignedStaff->isEmpty();
assert equals ticket1.source #EMail;
assert equals ticket1.creationDateTime sys.currentDateTime;
assert true ticket1.dueDateTime.isUndefined();
assert true ticket1.lastResponseDatetime.isUndefined();
assert equals ticket1.assignedDepartment dptGeneral;

assert true [ticket1.ticketThreadMessage->one(m|m.datetime=sys.currentDateTime and m.text='How can I change the priority of one of my tickets?' and m.author='James Jordan')];

//autoresponses
assert true [EMail.allInstances()->exists(e|e.emailKind.oclIsTypeOf(NewTicketAutoresponse) and e.fromAddress='general_at_support.com' and e.toAddress='james_at_jordan.jam' and e.ticketNumber=1)];

//notice to department members
assert true [EMail.allInstances()->exists(e|e.emailKind.oclIsTypeOf(NewTicketAlertToStaff) and e.fromAddress='general_at_support.com' and e.toAddress='john_at_support.com' and e.ticketNumber=1)];
assert true [EMail.allInstances()->exists(e|e.emailKind.oclIsTypeOf(NewTicketAlertToStaff) and e.fromAddress='general_at_support.com' and e.toAddress='mary_at_support.com' and e.ticketNumber=1)];
assert false [EMail.allInstances()->exists(e|e.emailKind.oclIsTypeOf(NewTicketAlertToStaff) and e.fromAddress='general_at_support.com' and e.toAddress='david_at_support.com' and e.ticketNumber=1)];

}
//notice to department members
assert false [Email.allInstances()->exists(e|e.emailKind.oclIsTypeOf(NewTicketAlertToStaff) and
e.fromAddress='technical_at_support.com' and
e.toAddress='marta_at_johnes.mar' and
e.ticketNumber=1)];

//notice to department members
assert false [Email.allInstances()->exists(e|e.emailKind.oclIsTypeOf(NewTicketAlertToStaff) and
e.fromAddress='general_at_support.com' and
e.toAddress='john_at_support.com' and
e.ticketNumber=1)];

assert false [Email.allInstances()->exists(e|e.emailKind.oclIsTypeOf(NewTicketAlertToStaff) and
e.fromAddress='general_at_support.com' and
e.toAddress='mary_at_support.com' and
e.ticketNumber=1)];

assert false [Email.allInstances()->exists(e|e.emailKind.oclIsTypeOf(NewTicketAlertToStaff) and
e.fromAddress='general_at_support.com' and
e.toAddress='david_at_support.com' and
e.ticketNumber=1)];

}

test S11{
    load testConfiguration1;
    nt1:=new NewTicketOnline;
    nt1.fullName:='James Jordan';
    nt1.email:='james_at_jordan.jam';
    nt1.helpTopic:=helpTopicUse;
    nt1.subject:='Reopening ticket';
    nt1.message:='I do not know how to reopen one of my closed tickets';
    assert occurrence nt1;

    ticket1:=nt1.createdTicket;

    nt2:=new NewTicketOnline;
    nt2.fullName:='James Jordan';
    nt2.email:='james_at_jordan.jam';
    nt2.helpTopic:=helpTopicInstallation;
    nt2.subject:='Error operating system';
    nt2.message:='The installation process does not finish...';
    assert occurrence nt2;

    ticket2:=nt2.createdTicket;

    cts:=new DisplayTicketsAssociatedToEmail(email:='james_at_jordan.jam', ticketNumber:=2);
    assert occurrence cts;
    assert equals cts.answer() [Set<Tuple{createDate=1,department='General support',email='james_at_jordan.jam',number=1,status=#Open,subject='Reopening ticket'},Tuple{createDate=1,department='Technical support',email='james_at_jordan.jam',number=2,status=#Open,subject='Error operating system'}}];
}

test S12{
    load testConfiguration1;

    cts:=new DisplayTicketsAssociatedToEmail(email:='james_at_jordan.jam', ticketNumber:=2);
    assert non-occurrence cts;
}

test S13{
    load testConfiguration1;

    nt1:=new NewTicketOnline;
    nt1.fullName:='James Jordan';
    nt1.email:='james_at_jordan.jam';
    nt1.helpTopic:=helpTopicUse;
    nt1.subject:='Reopening ticket';
    nt1.message:='I do not know how to reopen one of my closed tickets';
assert occurrence nt1;
ticket1:=nt1.createdTicket;

cts:=new DisplayTicketsAssociatedToEmail(email:='james_at_jordan.jam', ticketNumber:=2);
assert non-occurrence cts;
}
test S14{
load testConfiguration1;
nt1:=new NewTicketOnline;
nt1.fullName:=‘James Jordan’;
nt1.email:=‘james_at_jordan.jam’;
nt1.helpTopic:=helpTopicInstallation;
helpTopicInstallation.autoresponse:=#Enabled;
nt1.subject:=“Reopening ticket”;
t11.message:=’I do not know how to reopen one of my closed tickets’;
assert occurrence nt1;
ticket1:=nt1.createdTicket;

cr:=new ReplyTicketByCustomer(ticket:=ticket1,replyText:’Please help me’);
assert occurrence cr;
assert true [EMail.allInstances()->exists(e|e.emailKind.oclIsTypeOf(NewMessageAlertToStaff) and
e.fromAddress=’general_at_support.com’ and
e.toAddress=’john_at_support.com’ and
e.ticketNumber=1)];
assert true [EMail.allInstances()->exists(e|e.emailKind.oclIsTypeOf(NewMessageAutoresponse) and
e.fromAddress=’technical_at_support.com’ and
e.toAddress=’james_at_jordan.jam’ and
e.ticketNumber=1)];
}
test S15{
load testConfiguration2;
nt1:=new NewTicketOnline;
nt1.fullName:=‘James Jordan’;
nt1.email:=‘james_at_jordan.jam’;
nt1.helpTopic:=helpTopicInstallation;
nt1.subject:=“Reopening ticket”;
t11.message:=’I do not know how to reopen one of my closed tickets’;
assert occurrence nt1;
ticket1:=nt1.createdTicket;

cr:=new ReplyTicketByCustomer(ticket:=ticket1,replyText:’Please help me’);
assert occurrence cr;
assert false [EMail.allInstances()->exists(e|e.emailKind.oclIsTypeOf(NewMessageAlertToStaff) and
e.fromAddress=’general_at_support.com’ and
e.toAddress=’john_at_support.com’ and
e.ticketNumber=1)];
}
test S16{
load testConfiguration1;
assert false generConsultant.isLoggedIn;
li := new LogIn(username:’mary’, password:’yyy’);
assert occurrence li;
assert true generConsultant isLoggedIn;
Department of Service and Information System Engineering

Development of the Conceptual Schema of the osTicket System by Applying TDCM

April 201

Research report

}{

\begin{verbatim}
} test S17{
  load testConfiguration1;
  li := new LogIn(username:='mary', password:='yyy');
  assert occurrence li;
  assert true generalConsultant.isLoggedIn;
  li := new LogIn(username:='mary', password:='yyy');
  assert non-occurrence li;
}

} test S18{
  load testConfiguration1;
  li := new LogIn(username:='patricia', password:='uuu');
  assert non-occurrence li;
  maximumPrivilegesGroup.status:=#Disabled;
  li := new LogIn(username:='mary', password:='yyy');
  assert non-occurrence li;
}

} test S19{
  load testConfiguration1;
  assert false generalConsultant.isLoggedIn;
  li := new LogIn(username:='mary', password:='yyy');
  assert occurrence li;
  assert true generalConsultant.isLoggedIn;
  lo := new LogOut(staffMember:=generalConsultant);
  assert occurrence lo;
  assert false generalConsultant.isLoggedIn;
}

} test S20{
  load testConfiguration1;
  li := new LogIn(username:='mary', password:='zzz');
  assert non-occurrence li;
}

} test S21{
  load testConfiguration1;
  minimumPrivilegesGroup.canCreateTickets:=true;
  li := new LogIn(username:='martin', password:='ttt');
  assert occurrence li;
  nt:=new NewTicketOffline;
  nt.fullName:='Mary Marnes';
  nt.email:='mary_at_marnes.mar';
  nt.telephone:='xxxxxxxx';
  nt.ext:='xxxxxxxx';
  nt.source:=#Phone;
  nt.assignedDepartment:=dptTechnical;
  nt.helpTopic:=helpTopicInstallation;
  nt.subject:='Error operating system';
  nt.message:='The installation process does not finish....';
\end{verbatim}


DEVELOPMENT OF THE CONCEPTUAL SCHEMA OF THE osTICKET SYSTEM BY APPLYING TDCM

April 2011

Research report

nt.internalNote:="It seems that the correct installer is being used";

dt2:=new Datetime(value:=[(sys.currentDateTime.value+2)]);
nt.dueDatetime:=dt2;
nt.priority:=#Normal;
nt.assignedStaff:=generalConsultant;
nt.creator:=technicalActive;
assert occurrence nt;
ticket1:=nt.createdTicket;
assert equals ticket1.number 1;
assert equals ticket1.ticketStatus #Open;
assert equals ticket1.subject 'Error operating system';
assert equals ticket1.priority #Normal;
assert equals ticket1.assignedStaff generalConsultant;
assert equals ticket1.creationDateTime sys.currentDateTime;
assert equals ticket1.lastMessageDateTime sys.currentDateTime;
assert equals ticket1.dueDatetime.value 3;
assert true ticket1.lastResponseDatetime.isUndefined();
assert equals ticket1.assignedDepartment dptTechnical;

assert true [ticket1.ticketThreadMessage->one(m|m.datetime=sys.currentDateTime and m.text='The installation process does not finish....' and m.author='Mary Marnes')];

assert equals ticket1.lastMessageDateTime sys.currentDateTime;

//no autoresponses
assert true [not(EMail.allInstances())->exists(e|e.emailKind.oclIsTypeOf(NewTicketNotice))];

//notice to administrator
assert true [EMail.allInstances()->exists(e|e.emailKind.oclIsTypeOf(NewTicketAlertToStaff) and e.fromAddress='general_at_support.com' and e.toAddress='system_at_support.com' and e.ticketNumber=1)];

//notice to department manager
assert true [EMail.allInstances()->exists(e|e.emailKind.oclIsTypeOf(NewTicketAlertToStaff) and e.fromAddress='general_at_support.com' and e.toAddress='john_at_support.com' and e.ticketNumber=1)];

//notice to department members
assert true [EMail.allInstances()->exists(e|e.emailKind.oclIsTypeOf(NewTicketAlertToStaff) and e.fromAddress='general_at_support.com' and e.toAddress='martin_at_support.com' and e.ticketNumber=1)];

//TICKET 2
ticketSettings.customersCanChangePriority:=true;

nt2:=new NewTicketOffline;
nt2.fullName:= 'James Jordan';
nt2.email:='james_at_jordan.jam';
nt2.telephone:='xxxxxxxx';
nt2.ext:='xxxxxxxx';
nt2.source:=#Other;
nt2.assignedDepartment:=dptGeneral;
nt2.priority:=#Low;
nt2.helpTopic:=helpTopicUse;
nt2.subject:= 'Reopening ticket';
nt2.creator:=technicalActive;
nt2.message:='I do not know how to reopen one of my closed tickets';
assert occurrence nt2;
ticket2:=nt2.createdTicket;
assert equals ticket2.number 2;
assert equals ticket2.priority #Low;
assert true ticket2.assignedStaff->isEmpty();

//autoresponses
assert true [EMail.allInstances()] -> exists (e| e.emailKind.oclIsTypeOf(NewTicketNotice) and
e.fromAddress='general_at_support.com' and
e.toAddress='james_at_jordan.jam' and
e.ticketNumber=2];

//notice to administrator
assert true [EMail.allInstances()] -> exists (e| e.emailKind.oclIsTypeOf(NewTicketAlertToStaff) and
e.fromAddress='general_at_support.com' and
e.toAddress='system_at_support.com' and
e.ticketNumber=2];

//notice to department members
assert true [EMail.allInstances()] -> exists (e| e.emailKind.oclIsTypeOf(NewTicketAlertToStaff) and
e.fromAddress='general_at_support.com' and
e.toAddress='john_at_support.com' and
e.ticketNumber=2];
assert true [EMail.allInstances()] -> exists (e| e.emailKind.oclIsTypeOf(NewTicketAlertToStaff) and
e.fromAddress='general_at_support.com' and
e.toAddress='mary_at_support.com' and
e.ticketNumber=2];
assert false [EMail.allInstances()] -> exists (e| e.emailKind.oclIsTypeOf(NewTicketAlertToStaff) and
e.fromAddress='general_at_support.com' and
e.toAddress='david_at_support.com' and
e.ticketNumber=2];

}

test S22{
load testConfiguration2;
minimumPrivilegesGroup.canCreateTickets:=true;

li := new LogIn(username:='martin', password:='ttt');
assert occurrence li;

nt:=new NewTicketOffline;
tt.fullName:='Mary Marnes';
nt.email:='mary_at_marnes.mar';
nt.telephone:='xxxxxxxx';
nt.ext:='xxxxxxxx';
nt.source:=#Phone;
nt.assignedDepartment:=dptTechnical;
nt.helpTopic:=helpTopicInstallation;
nt.subject:='Error operating system';
nt.message:='The installation process does not finish....';
nt.internalNote:='It seems that the correct installer is being used';
dt2:=new Datetime(value:=[(sys.currentDateTime.value+2)]);
nt.dueDatetime:=dt2;
nt.priority:=#Normal;
nt.assignedStaff:=generalConsultant;
nt.creator:=technicalActive;
assert occurrence nt;

ticket1:=nt.createdTicket;
assert equals ticket1.number 1;
assert equals ticket1.ticketStatus #Open;
assert equals ticket1.subject 'Error operating system';
assert equals ticket1.priority #Normal;
assert equals ticket1.assignedStaff generalConsultant;
assert equals ticket1.source #Phone;
assert equals ticket1.creationDateTime sys.currentDateTime;
assert equals ticket1.lastMessageDatetime sys.currentDateTime;
assert equals ticket1.dueDatetime.value 3;
assert true ticket1.lastResponseDatetime.isUndefined();
assert equals ticket1.assignedDepartment dptTechnical;

assert true [ticket1.ticketThreadMessage->one(m|m.datetime=sys.currentDateTime and
m.text='The installation process does not finish....' and
m.author='Mary Marnes')];
assert equals ticket1.lastMessageDatetime sys.currentDateTime;

//no autoresponses
assert false [EMail.allInstances()->exists(e|e.emailKind.oclIsTypeOf(NewTicketNotice))];

//no notice to administrator
assert false [EMail.allInstances()->exists(e|e.emailKind.oclIsTypeOf(NewTicketAlertToStaff) and
   e.fromAddress='general_at_support.com' and
   e.toAddress='system_at_support.com' and
   e.ticketNumber=1)];

//no notice to department manager
assert false [EMail.allInstances()->exists(e|e.emailKind.oclIsTypeOf(NewTicketAlertToStaff) and
   e.fromAddress='general_at_support.com' and
   e.toAddress='john_at_support.com' and
   e.ticketNumber=1)];

//no notice to department members
assert false [EMail.allInstances()->exists(e|e.emailKind.oclIsTypeOf(NewTicketAlertToStaff) and
   e.fromAddress='general_at_support.com' and
   e.toAddress='martin_at_support.com' and
   e.ticketNumber=1)];

}

test S23{
load testConfiguration4;
minimumPrivilegesGroup.canCreateTickets:=true;
helpTopicUse.status:=#Disabled;
helpTopicInstallation.status:=#Disabled;
li := new LogIn(username:='martin', password:='ttt');
assert occurrence li;
nt:=new NewTicketOffline;
nt.fullName:='Mary Marnes';
nr.email:='mary_at_marnes.mar';
nr.telephone:='xxxxxxxx';
nr.ext:='xxxxxxxx';
nr.source:'#Phone';
nr.assignedDepartment:=dptTechnical;
nr.subject:='Error operating system';
nr.message:='The installation process does not finish....';
nr.creator:=technicalActive;
nr.priority:=#Normal;
assert occurrence nr;
ticket1:=nt.createdTicket;
assert true ticket1.helpTopic->isEmpty();
}

test S24{
load testConfiguration9;
}
li := new Login(username:='martin', password:='ttt');
assert occurrence li;

nt:=new NewTicketOffline;
nt.fullName:='Mary Marnes';
nt.email:='mary_at_marnes.mar';
nt.telephone:='xxxxxxxx';
nt.ext:='xxxxxxxx';
nt.source:='#Phone';
nt.helpTopic:='helpTopicInstallation';
nt.assignedDepartment:='dptTechnical';
nt.assignedStaff:='generalConsultant';
nt.subject:='Error operating system';
nt.message:='The installation process does not finish....';
nt.priority:='#Normal';
tt.creator:='technicalActive';
assert non-occurrence nt;

}

test S25{
load testConfiguration9;

nt:=new NewTicketOffline;
nt.fullName:='Mary Marnes';
nt.email:='mary_at_marnes.mar';
nt.telephone:='xxxxxxxx';
nt.ext:='xxxxxxxx';
nt.source:='#Phone';
nt.helpTopic:='helpTopicInstallation';
nt.assignedDepartment:='dptTechnical';
nt.assignedStaff:='generalConsultant';
nt.subject:='Error operating system';
nt.message:='The installation process does not finish....';
nt.priority:='#Normal';
tt.creator:='technicalActive';
assert non-occurrence nt;

}

test S26{
load testConfiguration3;
load created_tickets;

li := new Login(username:='john', password:='xxx');
assert occurrence li;

dts:=new DisplayTicketsByStatus(consultant:='generalAdministrator', status:='#OpenTickets');
assert occurrence dts;
assert equals dts.answer() [Sequence[
    Tuple{createDate=1,department='Technical support',email='mary_at_marnes.mar',number=1,priority='#Normal',subject='Error operating system'},
    Tuple{createDate=1,department='General support',email='mary_at_marnes.mar',number=2,priority='#High',subject='Can I reply a ticket?'},
    Tuple{createDate=1,department='Technical support',email='martin_at_pope.mar',number=3,priority='#Low',subject='Error while login'},
    Tuple{createDate=2,department='General support',email='james_at_jordan.jam',number=4,priority='#Normal',subject='Reopening ticket'},
    Tuple{createDate=2,department='Technical support',email='marta_at_johnes.mar',number=5,priority='#High',subject='See my tickets'}}];

lo := new LogOut(staffMember:='generalAdministrator');
assert occurrence lo;

li := new Login(username:='martin', password:='ttt');
assert occurrence li;

dts:=new DisplayTicketsByStatus(consultant:=technicalActive, status:=#OpenTickets);
assert occurrence dts;
assert equals dts.answer() |Sequence{
    Tuple{createDate=1, department='Technical support', email='mary_at_marines.mar', number=1, priority=#Normal, subject='Error operating system'},
    Tuple{createDate=1, department='Technical support', email='martin_at_pope.mar', number=3, priority=#Low, subject='Error while login'},
    Tuple{createDate=2, department='Technical support', email='marta_at_johnes.mar', number=5, priority=#High, subject='See my tickets'}} |

} test S27{
    load testConfiguration11;

dts:=new DisplayTicketsByStatus(consultant:=generalAdministrator, status:=#OpenTickets);
assert non-occurrence dts;
}

test S28{
    load testConfiguration3;
    load created_tickets;
    li := new LogIn(username:='john', password:='xxx');
    assert occurrence li;
    stp:=new ChangeTicketPriority(staffMember:=generalAdministrator, ticket:=ticket1, newPriority:=#High);
    assert occurrence stp;
    assert equals ticket1.priority #High;
    assert true [ticket1.internalNote->one(i|i.datetime=sys.currentDateTime and
    i.text='The ticket priority has been changed' and
    i.subject='Ticket priority changed' and
    i.author='John')];
}

test S29{
    load testConfiguration4;
    load created_tickets;
    li := new LogIn(username:='martin', password:='ttt');
    assert occurrence li;
    stp:=new ChangeTicketPriority(staffMember:=technicalActive, ticket:=ticket2, newPriority:=#High);
    assert non-occurrence stp;
}

test S30{
    load testConfiguration4;
    load created_tickets;
    stp:=new ChangeTicketPriority(staffMember:=generalAdministrator, ticket:=ticket1, newPriority:=#High);
    assert non-occurrence stp;
}

test S31{
    load testConfiguration11;
    load created_tickets;
helpTopicInstallation.autoresponse:=# Enabled;

li := new LogIn(username:='john', password:='xxx');
assert occurrence li;

mto:=new MarkTicketOverdue(staffMember:=generalAdministrator, ticket:=ticket1);
assert occurrence mto;
assert true ticket1.isOverdue;
assert true [ticket1.internalNote->one(i|i.datetime=sys.currentDateTime and
i.text='Ticket flagged as overdue' and
i.subject='Ticket Marked Overdue' and
i.author='John')];

}

test S32{
load testConfiguration3;
load created_tickets;

li := new LogIn(username:='martin', password:='ttt');
assert occurrence li;

mto:=new MarkTicketOverdue(staffMember:=technicalActive, ticket:=ticket1);
assert non-occurrence mto;
}

test S33{
load testConfiguration4;
load created_tickets;

mto:=new MarkTicketOverdue(staffMember:=generalAdministrator, ticket:=ticket1);
assert non-occurrence mto;
}

test S34{
load testConfiguration3;
load created_tickets;

li := new LogIn(username:='mary', password:='yyy');
assert occurrence li;

at:=new AssignTicket(staffMember:=generalConsultant, ticket:=ticket1, assignee:=generalAdministrator, assignmentText:='This is
for you');
assert occurrence at;
assert equals ticket1.assignedStaff generalAdministrator;
assert true [ticket1.internalNote->one(i|i.datetime=sys.currentDateTime and
i.text='This is for you' and
i.subject='Ticket Reassigned' and
i.author='Mary')];

//notice sent to the assignee
assert true [EMAIL.allInstances()->exists(e|e.emailKind.oclIsTypeOf(TicketAssignedAlertToStaff) and
 e.fromAddress='general_at_support.com' and
e.toAddress='john_at_support.com' and
 e.ticketNumber=1)];
}
test S35{
load testConfiguration3;
load created_tickets;
li := new LogIn(username:='martin', password:='ttt');
assert occurrence li;

at:=new AssignTicket(staffMember:=technicalActive, ticket:=ticket2, assignee:=generalAdministrator, assignmentText:='This is for you');
assert non-occurrence at;
}

} test S36{
load testConfiguration3;
load created_tickets;
li := new LogIn(username:='mary', password:='yyy');
assert occurrence li;

at:=new AssignTicket(staffMember:=generalConsultant, ticket:=ticket1, assignee:=generalConsultantVacation, assignmentText:='This is for you');
assert non-occurrence at;
}

} test S37{
load testConfiguration3;
load created_tickets;

at:=new AssignTicket(staffMember:=generalConsultant, ticket:=ticket1, assignee:=generalAdministrator, assignmentText:='This is for you');
assert non-occurrence at;
}

} test S38{
load testConfiguration3;
load created_tickets;
li := new LogIn(username:='mary', password:='yyy');
assert occurrence li;

rt:=new ReleaseTicket(staffMember:=generalConsultant, ticket:=ticket1);
assert occurrence rt;
assert true ticket1.assignedStaff.isUndefined();

assert true [ticket1.internalNote->one(i| i.datetime=sys.currentDateTime and
i.text='Released ticket' and
i.subject='Ticket unassigned' and
i.author='Mary')];
}

} test S39{
load testConfiguration3;
load created_tickets;
li := new LogIn(username:='martin', password:='ttt');
assert occurrence li;
rt:=new ReleaseTicket(staffMember:=technicalActive, ticket:=ticket2);
assert non-occurrence rt;
}

} test S40{
load testConfiguration3;
load created_tickets;
li := new LogIn(username:='mary', password:='yyy');
assert occurrence li;
rt:=new ReleaseTicket(staffMember:=generalConsultant, ticket:=ticket4);
assert non-occurrence rt;
}

} test S41{
load testConfiguration3;
load created_tickets;
rt:=new ReleaseTicket(staffMember:=generalConsultant, ticket:=ticket2);
assert non-occurrence rt;
}

} test S42{
load testConfiguration3;
load created_tickets;
li := new LogIn(username:='mary', password:='yyy');
assert occurrence li;
dt3:=new Datetime(value:=(sys.currentDateTime.value+3));
et:=new EditTicket(staffMember:=generalConsultant, ticket:=ticket1, emailAddress:='mary2@marnes.mar', fullName:='Mary Marnes2', subject:='Error operating system2', telephone:='xxx2', ext:='xx2', dueDatetime:=dt3, priority:='#Low', helpTopic:=helpTopicUse, editionInternalNote:='The customer asks for this changes');
assert occurrence et;
assert equals ticket1.email 'mary2@marnes.mar';
assert equals ticket1.fullName 'Mary Marnes2';
assert equals ticket1.subject 'Error operating system2';
assert equals ticket1.telephone 'xxx2';
assert equals ticket1.ext 'xx2';
assert equals ticket1.priority #Low;
assert equals ticket1.helpTopic helpTopicUse;
assert true [ticket1.internalNote->one(i|i.datetime=sys.currentDateTime and i.text='The customer asks for this changes' and i.subject='Ticket updated' and i.author='Mary')];
}

} test S43{
load testConfiguration3;
DEVELOPMENT OF THE CONCEPTUAL
SCHEMA OF THE osTICKET
SYSTEM BY APPLYING TDCM

April 2011

load created_tickets;

li := new LogIn(username:='martin', password:='ttt');
assert occurrence li;
dt3:=new Datetime(value:=[(sys.currentDateTime.value+2)]);
et:=new EditTicket(staffMember:=technicalActive, ticket:=ticket1,
emailAddress:='mary2@marnes.mar',
fullName:='Mary Marnes2',
subject:='Error operating system2',
telephone:='xxx2',
ext:='xx2',
dueDatetime:=dt3,
priority:=#Low,
helpTopic:=helpTopicUse,
editionInternalNote:='The customer asks for this changes');
assert non-occurrence et;
}

test S44{
load testConfiguration3;
load created_tickets;

li := new LogIn(username:='martin', password:='ttt');
assert occurrence li;
dt3:=new Datetime(value:=[(sys.currentDateTime.value+2)]);
et:=new EditTicket(staffMember:=technicalActive, ticket:=ticket1,
emailAddress:='mary2@marnes.mar',
fullName:='John Johnes2',
subject:='Can I reply a ticket? Yes or no?',
telephone:='yyy2',
ext:='yy2',
dueDatetime:=dt3,
priority:=#Normal,
helpTopic:=helpTopicUse,
editionInternalNote:='The customer asks for this changes');
assert non-occurrence et;
}

test S45{
load testConfiguration3;
load created_tickets;
testConfiguration3;

li := new LogIn(username:='martin', password:='ttt');
assert occurrence li;
dt3:=new Datetime(value:=[(sys.currentDateTime.value+2)]);
et:=new EditTicket(staffMember:=technicalActive, ticket:=ticket1,
emailAddress:='mary2@marnes.mar',
fullName:='John Johnes2',
subject:='Can I reply a ticket? Yes or no?',
telephone:='yyy2',
ext:='yy2',
dueDatetime:=dt3,
priority:=#Normal,
helpTopic:=helpTopicUse,
editionInternalNote:='The customer asks for this changes');
assert occurrence et;
}

test S46{
load testConfiguration3;
load created_tickets;
dt3:=new Datetime(value:=[sys.currentDateTime.value+3]);
et:=new EditTicket(staffMember:=generalConsultant, ticket:=ticket1,
  emailAddress:='mary2@marnes.mar',
  fullName:='Mary Marnes2',
  subject:='Error operating system2',
  telephone:='xxx2',
  ext:='xx2',
  dueDatetime:=dt3,
  priority:=#Low,
  helpTopic:=helpTopicUse,
  editionInternalNote:='The customer asks for this changes');
assert non-occurrence et;
}

test S47{
  load testConfiguration1;
  helpTopicInstallation.autoresponse:=#Enabled;
  load created_tickets;
}

li := new LogIn(username:='mary', password:='yyy');
assert occurrence li;

rt:=new PostTicketReply(staffMember:=generalConsultant, ticket:=ticket1,
  response:='You should choose the installation executable...');
assert occurrence rt;

assert true [ticket1.ticketThreadMessage->one(m|m.datetime=sys.currentDateTime and
  m.text='You should choose the installation executable...' and
  m.author='Mary')];

assert equals ticket1.lastMessageDatetime sys.currentDateTime;

assert equals ticket1.lastRespondent generalConsultant;

//autoresponse
assert true [EMail.allInstances()->exists(e|e.emailKind.oclIsTypeOf(TicketResponseNotice) and
  e.fromAddress='technical_at_support.com' and
  e.toAddress='mary_at_support.com' and
  e.ticketNumber=1)];

//notice to assigned staff
assert true [EMail.allInstances()->exists(e|e.emailKind.oclIsTypeOf(NewMessageAlertToStaff) and
  e.fromAddress='general_at_support.com' and
  e.toAddress='mary_at_support.com' and
  e.ticketNumber=1)];

//notice to department manager
assert true [EMail.allInstances()->exists(e|e.emailKind.oclIsTypeOf(NewMessageAlertToStaff) and
  e.fromAddress='general_at_support.com' and
  e.toAddress='john_at_support.com' and
  e.ticketNumber=1)];

}

test S48{
  load testConfiguration4;
  helpTopicInstallation.autoresponse:=#Enabled;
  load created_tickets;
}

li := new LogIn(username:='mary', password:='yyy');
assert occurrence li;

rt:=new PostTicketReply(staffMember:=generalConsultant, ticket:=ticket1,
  response:='You should choose the installation executable...');
assert occurrence rt;
assert true [ticket1.ticketThreadMessage->one(m|m.datetime=sys.currentDateTime and m.text='You should choose the installation executable...' and m.author='Mary')];

assert equals ticket1.lastMessageDateTime sys.currentDateTime;
assert equals ticket1.lastRespondent generalConsultant;

// autoresponse
assert false [EMail.allInstances()]->exists(e|e.emailKind.oclIsTypeOf(TicketResponseNotice) and e.fromAddress='technical_at_support.com' and e.toAddress='mary_at_marnes.mar' and e.ticketNumber=1); 

// notice to assigned staff
assert false [EMail.allInstances()]->exists(e|e.emailKind.oclIsTypeOf(NewMessageAlertToStaff) and e.fromAddress='general_at_support.com' and e.toAddress='mary_at_support.com' and e.ticketNumber=1); 

// notice to department manager
assert false [EMail.allInstances()]->exists(e|e.emailKind.oclIsTypeOf(NewMessageAlertToStaff) and e.fromAddress='general_at_support.com' and e.toAddress='john_at_support.com' and e.ticketNumber=1); 

} }

test S49{
load testConfiguration4;
load created_tickets;
li := new LogIn(username:='martin', password:='ttt');
assert occurrence li;
rt:=new PostTicketReply(staffMember:=technicalActive, ticket:=ticket2, response:='You should choose the installation executable...');
assert non-occurrence rt; 
}

test S50{
load testConfiguration1;
load created_tickets;
rt:=new PostTicketReply(staffMember:=generalConsultant, ticket:=ticket1, response:='You should choose the installation executable...');
assert non-occurrence rt; 
}

test S51{
load testConfiguration1;
load created_tickets;
li := new LogIn(username:='martin', password:='ttt');
assert occurrence li;
pin:=new PostTicketInternalNote(staffMember:=technicalActive, ticket:=ticket1, title:='No tickets?', note:='It seems that she does not have tickets');
assert occurrence pin;
assert true [ticket1.internalNote->one(m|m.datetime=sys.currentDateTime and m.subject='No tickets?' and m.text='It seems that she does not have tickets' and m.author='Martin')];

// notice to assigned staff
assert true [EMail.allInstances()]->exists(e|e.emailKind.oclIsTypeOf(NewInternalNoteAlertToStaff) and e.fromAddress='general_at_support.com' and e.toAddress='mary_at_support.com' and e.ticketNumber=1);
test S52{
load testConfiguration4;
load created_tickets;
li := new Login(username:='martin', password:='ttt');
assert occurrence li;
pin:=new PostTicketInternalNote(staffMember:=technicalActive, ticket:=ticket5,
title:='No tickets?', note:='It seems that she does not have tickets');
assert occurrence pin;
assert true [ticket5.internalNote->one(m|m.datetime=sys.currentTime and
m.subject='No tickets?'and
m.text='It seems that she does not have tickets' and
m.author='Martin')];
//notice to assigned staff
assert true [EMail.allInstances()->exists(e|e.emailKind.oclIsTypeOf(NewInternalNoteAlertToStaff) and
 e.fromAddress='general_at_support.com' and
 e.toAddress='mary_at_support.com' and
 e.ticketNumber=5)];
//notice to department manager
assert true [EMail.allInstances()->exists(e|e.emailKind.oclIsTypeOf(NewInternalNoteAlertToStaff) and
 e.fromAddress='general_at_support.com' and
 e.toAddress='john_at_support.com' and
 e.ticketNumber=5)];
}

}
li := new LogIn(username:='mary', password:='yyy');
assert occurrence li;

 tt:=new TransferDepartment(staffMember:=generalConsultant, ticket:=ticket2,
department:=dptTechnical,
  note:='This is a technical question');
assert occurrence tt;

assert equals ticket1.assignedDepartment dptTechnical;
}

 test S56{
  load testConfiguration3;
tехnicalActive.staffGroup:=maximumPrivilegesGroup;
maximumPrivilegesGroup.departmentsAccess:=Set{dptTechnical};
load created_tickets;

li := new LogIn(username:='martin', password:='ttt');
assert occurrence li;

 tt:=new TransferDepartment(staffMember:=technicalActive, ticket:=ticket2,
department:=dptTechnical,
  note:='This is a technical question');
assert non-occurrence tt;

}

 test S57{
  load testConfiguration3;
load created_tickets;

li := new LogIn(username:='mary', password:='yyy');
assert occurrence li;

 tt:=new TransferDepartment(staffMember:=generalConsultant, ticket:=ticket2,
department:=dptGeneral,
  note:='This is a technical question');
assert non-occurrence tt;

}

 test S58{
  load testConfiguration3;
load created_tickets;

li := new LogIn(username:='martin', password:='ttt');
assert occurrence li;

 tt:=new TransferDepartment(staffMember:=generalConsultant, ticket:=ticket1,
department:=dptGeneral,
  note:='This is a technical question');
assert non-occurrence tt;

}

 test S59{
  load testConfiguration3;
load created_tickets;

 tt:=new TransferDepartment(staffMember:=generalConsultant, ticket:=ticket1,
department:=dptGeneral,
  note:='This is a technical question');
assert non-occurrence tt;

}

test S60{
load testConfiguration3;
load created_tickets;
li := new Login(username:='mary', password:='yyy');
assert occurrence li;

c:=new CloseTicket(staffMember:=generalConsultant, ticket:=ticket1);
assert occurrence ct;

assert equals ticket1.ticketStatus #Closed;

assert true [ticket1.internalNote->one(m|m.datetime=sys.currentDateTime and m.subject='Ticket closed' and m.text='Ticket closed without response' and m.author='Mary')];

c:=new CloseTicket(staffMember:=generalConsultant, ticket:=ticket1);
assert non-occurrence ct;
}

test S61{
load testConfiguration3;
load created_tickets;
li := new Login(username:='mary', password:='yyy');
assert occurrence li;

generalConsultant.staffGroup.departmentsAccess:=Set{dptGeneral};

c:=new CloseTicket(staffMember:=generalConsultant, ticket:=ticket3);
assert non-occurrence ct;

generalConsultant.staffGroup.departmentsAccess:=Set{dptGeneral,dptTechnical};
generalConsultant.staffGroup.canCloseTickets:=false;

c:=new CloseTicket(staffMember:=generalConsultant, ticket:=ticket3);
assert non-occurrence ct;
}

test S62{
load testConfiguration3;
load created_tickets;

c:=new CloseTicket(staffMember:=generalConsultant, ticket:=ticket3);
assert non-occurrence ct;
}

test S63{
load testConfiguration1;
load created_tickets;
helpTopicInstallation.autoresponse:=#Enabled;
li := new Login(username:='mary', password:='yyy');
assert occurrence li;

c:=new CloseTicketWithResponse(staffMember:=generalConsultant, ticket:=ticket1, response='Ticket solved');
assert occurrence ct;

assert equals ticket1.ticketStatus #Closed;

assert true [ticket1.internalNote->one(m|m.datetime=sys.currentDateTime and
assert true [EMail.allInstances()->exists(e|e.emailKind.oclIsTypeOf(TicketResponseNotice) and e.fromAddress='technical_at_support.com' and e.toAddress='mary_at_marnes.mar' and e.ticketNumber=1)];

assert true [ticket1.ticketThreadMessage->one(m|m.datetime=sys.currentDateTime and m.text='Ticket solved' and m.author='Mary')];

c:=new CloseTicketWithResponse(staffMember:=generalConsultant, ticket:=ticket1, response:='Ticket solved');
assert non-occurrence c;
}

test S64{
load testConfiguration4;
load created_tickets;

li := new LogIn(username:='mary', password:='yyy');
assert occurrence li;

c:=new CloseTicketWithResponse(staffMember:=generalConsultant, ticket:=ticket1, response:='Ticket solved');
assert occurrence c;
assert equals ticket1.ticketStatus #Closed;

assert true [ticket1.internalNote->one(m|m.datetime=sys.currentDateTime and m.subject='Ticket closed'and m.text='Ticket closed on reply' and m.author='Mary')];

assert false [EMail.allInstances()->exists(e|e.emailKind.oclIsTypeOf(TicketResponseNotice) and e.fromAddress='technical_at_support.com' and e.toAddress='mary_at_marnes.mar' and e.ticketNumber=1)];

assert true [ticket1.ticketThreadMessage->one(m|m.datetime=sys.currentDateTime and m.text='Ticket solved' and m.author='Mary')];
}

test S65{
load testConfiguration3;
load created_tickets;

li := new LogIn(username:='mary', password:='yyy');
assert occurrence li;

generalConsultant.staffGroup.departmentsAccess:=Set{dptGeneral};

c:=new CloseTicketWithResponse(staffMember:=generalConsultant, ticket:=ticket3, response:='Ticket solved');
assert non-occurrence c;

generalConsultant.staffGroup.departmentsAccess:=Set{dptGeneral,dptTechnical};
generalConsultant.staffGroup.canCloseTickets:=false;

c:=new CloseTicketWithResponse(staffMember:=generalConsultant, ticket:=ticket3, response:='Ticket solved');
assert non-occurrence ct;
}

test S66{
  load testConfiguration3;
  load created_tickets;

  ct:=new CloseTicketWithResponse(staffMember:=generalConsultant, ticket:=ticket1,
                   response:='Ticket solved');
  assert non-occurrence ct;
}

test S67{
  load testConfiguration3;
  load created_tickets;
  ticket1.ticketStatus:=#Closed;
  li := new LogIn(username:='mary', password:='yyy');
  assert occurrence li;
  rot:=new ReopenTicket(staffMember:=generalConsultant, ticket:=ticket1);
  assert occurrence rot;
  assert equals ticket1.ticketStatus #Open;
  assert true [ticket1.internalNote->one(m|m.datetime=sys.currentDateTime and
        m.subject='Ticket reopened'and
        m.text='Ticket reopened without comments' and
        m.author='Mary')];
  rot:=new ReopenTicket(staffMember:=generalConsultant, ticket:=ticket1);
  assert non-occurrence rot;
}

test S68{
  load testConfiguration3;
  load created_tickets;
  ticket2.ticketStatus:=#Closed;
  li := new LogIn(username:='martin', password:='ttt');
  assert occurrence li;
  rot:=new ReopenTicket(staffMember:=technicalActive, ticket:=ticket2);
  assert non-occurrence rot;
}

test S69{
  load testConfiguration3;
  load created_tickets;
  ticket1.ticketStatus:=#Closed;
  rot:=new ReopenTicket(staffMember:=generalConsultant, ticket:=ticket1);
  assert non-occurrence rot;
}

test S70{
  load testConfiguration1;
  helpTopicInstallation.autoresponse:=#Enabled;
  load created_tickets;
  ticket1.ticketStatus:=#Closed;
  li := new LogIn(username:='mary', password:='yyy');
assert occurrence li;

rot:=new ReopenTicketWithResponse(staffMember:=generalConsultant, ticket:=ticket1, 
  response:="The customer is not satisfied");
assert occurrence rot;

assert equals ticket1.ticketStatus #Open;

assert true (ticket1.internalNote->one(m|m.datetime=sys.currentDateTime and 
  m.subject="Ticket status changed to open" and 
  m.text="A staff member reopened the ticket on reply" and 
  m.author="Mary");

assert true (EMail.allInstances()->exists(e|e.emailKind.oclIsTypeOf(TicketResponseNotice) and 
  e.fromAddress=technical_at_support.com and 
  e.toAddress=mary_at_marnes.mar and 
  e.ticketNumber=1));

assert true (ticket1.ticketThreadMessage->one(m|m.datetime=sys.currentDateTime and 
  m.text="The customer is not satisfied" and 
  m.author="Mary");

rot:=new ReopenTicketWithResponse(staffMember:=generalConsultant, ticket:=ticket1, 
  response:="The customer is not satisfied");
assert non-occurrence rot;
}

test S71{
  load testConfiguration4;
  load created_tickets;
  ticket1.ticketStatus:=#Closed;

  li := new Login(username:='mary', password:='yyy');
  assert occurrence li;
  rot:=new ReopenTicketWithResponse(staffMember:=generalConsultant, ticket:=ticket1, 
    response:="The customer is not satisfied");
  assert occurrence rot;
  assert equals ticket1.ticketStatus #Open;
  assert true (ticket1.internalNote->one(m|m.datetime=sys.currentDateTime and 
    m.subject="Ticket status changed to open" and 
    m.text="A staff member reopened the ticket on reply" and 
    m.author="Mary");

assert false (EMail.allInstances()->exists(e|e.emailKind.oclIsTypeOf(TicketResponseNotice) and 
  e.fromAddress=technical_at_support.com and 
  e.toAddress=mary_at_marnes.mar and 
  e.ticketNumber=1));

assert true (ticket1.ticketThreadMessage->one(m|m.datetime=sys.currentDateTime and 
  m.text="The customer is not satisfied" and 
  m.author="Mary");

}
rot:=new ReopenTicketWithResponse(staffMember:=technicalActive, ticket:=ticket2, response:='The customer is not satisfied');
assert non-occurrence rot;
}

test S73{
load testConfiguration1;
helpTopicInstallation.autoresponse:=#Enabled;
load created_tickets;
ticket1.ticketStatus:=#Closed;
rot:=new ReopenTicketWithResponse(staffMember:=generalConsultant, ticket:=ticket1, response:='The customer is not satisfied');
assert non-occurrence rot;
}

test S74{
load testConfiguration3;
load created_tickets;
li := new LogIn(username:='mary', password:='yyy');
assert occurrence li;

cbt:=new BanEmailAndCloseTicket(staffMember:=generalConsultant, ticket:=ticket1);
assert occurrence cbt;
assert equals ticket1.ticketStatus #Closed;
assert true EmailSettings.allInstances->any(true).banList->includes(ticket1.email);

cbt:=new BanEmailAndCloseTicket(staffMember:=generalConsultant, ticket:=ticket1);
assert non-occurrence cbt;
}

test S75{
load testConfiguration3;
technicalActive.staffGroup.canBanEmails:=true;
load created_tickets;
li := new LogIn(username:='mary', password:='yyy');
assert occurrence li;

cbt:=new BanEmailAndCloseTicket(staffMember:=technicalActive, ticket:=ticket1);
assert non-occurrence cbt;
}

test S76{
load testConfiguration3;
lload created_tickets;
generalConsultant.staffGroup.canBanEmails:=false;
li := new LogIn(username:='mary', password:='yyy');
assert occurrence li;

cbt:=new BanEmailAndCloseTicket(staffMember:=generalConsultant, ticket:=ticket2);
assert non-occurrence cbt;
}

test S77{
load testConfiguration3;
load created_tickets;

cbt:=new BanEmailAndCloseTicket(staffMember:=generalConsultant, ticket:=ticket2);
assert non-occurrence cbt;
}
test S78{
  load testConfiguration3;
  load created_tickets;

  li := new Login(username:='mary', password:='yyy');
  assert occurrence li;

  dt:=new DeleteTicket(staffMember:=generalConsultant, ticket:=ticket1);
  assert occurrence dt;

  assert true [not(Ticket.allInstances()->exists(tt.number=1))];
}

test S79{
  load testConfiguration3;
  load created_tickets;
  technicalActive.staffGroup.canDeleteTickets:=true;

  li := new Login(username:='mary', password:='yyy');
  assert occurrence li;

  dt:=new DeleteTicket(staffMember:=technicalActive, ticket:=ticket2);
  assert non-occurrence dt;
}

test S80{
  load testConfiguration3;
  load created_tickets;
  technicalActive.staffGroup.canDeleteTickets:=false;

  li := new Login(username:='mary', password:='yyy');
  assert occurrence li;

  dt:=new DeleteTicket(staffMember:=technicalActive, ticket:=ticket1);
  assert non-occurrence dt;
}

test S81{
  load testConfiguration3;
  load created_tickets;

  dt:=new DeleteTicket(staffMember:=generalConsultant, ticket:=ticket1);
  assert non-occurrence dt;
}

test S82{
  load testConfiguration4;
  EmailSettings.allInstances()->any(true).banList:=Set{'hello_at_helloworld.hel'};

  li := new Login(username:='mary', password:='yyy');
  assert occurrence li;

  nt1:=new NewTicketOffline;
  nt1.fullName:='Mary Marnes';
  nt1.email:='hello_at_helloworld.hel';
  nt1.telephone:='xxxxxxxx';
  nt1.ext:='xxxxxxxx';
  nt1.source#:Phone;
  nt1.assignedDepartment:=dptTechnical;
  nt1.helpTopic:=helpTopicInstallation;
  nt1.subject:=Error operating system';
  nt1.message:=The installation process does not finish...';
  nt1.internalNote:=It seems that the correct installer is being used';
  dt2:=new Datetime(value:=[sys.currentDateTime.value+2]);
  nt1.dueDatetime:=dt2;
  nt1.priority#:Normal;
  nt1.assignedStaff:=generalConsultant;
}
nt1.creator:=generalConsultant;
assert non-occurrence nt1;

nt4:=new NewTicketOnline;
nt4.fullName:='James Jordan';
nt4.email:='hello_at_helloworld.hel';
nt4.helpTopic:=helpTopicUse;
nt4.subject:='Reopening ticket';
nt4.message:='I do not know how to reopen one of my closed tickets';
assert non-occurrence nt4;

nt5:=new NewTicketByEmail;
nt5.toAddress:='technical_at_support.com';
nt5.fromName:='Marta Johnes';
nt5.fromAddress:='hello_at_helloworld.hel';
nt5.subject:='See my tickets';
nt5.message:='Can I see my tickets?';
assert non-occurrence nt5;

}

test S83{
load testConfiguration3;
load created_tickets;

dt3:=new Datetime(value:=[sys.currentDateTime.value+2]);
sys.currentDateTime:=dt3;

cot:=new CheckOverdueTickets;
assert occurrence cot;

assert equals ticket1.isOverdue true;

//notice to assigned staff
assert false [EMail.allInstances()->exists(e|e.emailKind.oclIsTypeOf(NewInternalNoteAlertToStaff) and e.fromAddress='general_at_support.com' and e.toAddress='mary_at_support.com' and e.ticketNumber=1)];

//notice to department member
assert false [EMail.allInstances()->exists(e|e.emailKind.oclIsTypeOf(NewInternalNoteAlertToStaff) and e.fromAddress='general_at_support.com' and e.toAddress='john_at_support.com' and e.ticketNumber=1)];

}

test S84{
load testConfiguration1;
load created_tickets;

dt3:=new Datetime(value:=[sys.currentDateTime.value+2]);
sys.currentDateTime:=dt3;

cot:=new CheckOverdueTickets;
assert occurrence cot;

assert equals ticket1.isOverdue true;

//notice to assigned staff
assert true [EMail.allInstances()->exists(e|e.emailKind.oclIsTypeOf(OverdueTicketAlertToStaff) and e.fromAddress='general_at_support.com' and e.toAddress='mary_at_support.com' and e.ticketNumber=1)];

//notice to department member
assert true [EMail.allInstances()->exists(e|e.emailKind.oclIsTypeOf(OverdueTicketAlertToStaff) and e.fromAddress='general_at_support.com' and e.toAddress='john_at_support.com' and}
e.ticketNumber=1));
}
test S85{
    load testConfiguration3;
    load created_tickets;
    li := new LogIn(username:='john', password:='xxx');
    assert occurrence li;
    dts:=new DisplayTicketsByStatus(consultant:=generalAdministrator, status:=#OpenTickets);
    assert occurrence dts;
    assert equals dts.answer() [Sequence]
        Tuple{createDate=1, department='Technical support', email='mary_at_marnes.mar', number=1, priority=#Normal, subject='Error operating system'},
        Tuple{createDate=1, department='General support', email='mary_at_marnes.mar', number=2, priority=#High, subject='Can I reply a ticket?'});
    dts2:=new DisplayTicketsByStatus(consultant:=generalAdministrator, status:=#AssignedToMe);
    assert occurrence dts2;
    assert equals dts2.answer() [Sequence] ;
    at2:=new AssignTicket(staffMember:=generalAdministrator, ticket:=ticket4, assignee:=generalAdministrator, assignmentText:='This is for me');
    assert occurrence at2;
    dts3:=new DisplayTicketsByStatus(consultant:=generalAdministrator, status:=#AssignedToMe);
    assert occurrence dts3;
    assert equals dts3.answer() [Sequence] [Tuple{createDate=2, department='General support', email='james_at_jordan.jam', number=4, priority=#Normal, subject='Reopening ticket'} ];
    dts4:=new DisplayTicketsByStatus(consultant:=generalAdministrator, status:=#OverdueTickets);
    assert occurrence dts4;
    assert equals dts4.answer() [Sequence] ;
    dt3:=new Datetime(value:=[(sys.currentDateTime.value+3)]);
    sys.currentDateTime:=dt3;
    mto:=new MarkTicketOverdue(staffMember:=generalAdministrator, ticket:=ticket4);
    assert occurrence mto;
    cot:=new CheckOverdueTickets;
    assert occurrence cot;
    dts5:=new DisplayTicketsByStatus(consultant:=generalAdministrator, status:=#OverdueTickets);
    assert occurrence dts5;
    assert equals dts5.answer() [Sequence]
        Tuple{createDate=1, department='Technical support', email='mary_at_marnes.mar', number=1, priority=#Normal, subject='Error operating system'},
        Tuple{createDate=2, department='General support', email='james_at_jordan.jam', number=4, priority=#Normal, subject='Reopening ticket'} ];
    dts6:=new DisplayTicketsByStatus(consultant:=generalAdministrator, status:=#ClosedTickets);
    assert occurrence dts6;
    assert equals dts6.answer() [Sequence] ;
    ct:=new CloseTicket(staffMember:=generalAdministrator, ticket:=ticket4);
    assert occurrence ct;
    dts7:=new DisplayTicketsByStatus(consultant:=generalAdministrator, status:=#OverdueTickets);
    assert occurrence dts7;
    assert equals dts7.answer() [Sequence]
        Tuple{createDate=1, department='Technical support', email='mary_at_marnes.mar', number=1, priority=#Normal, subject='Error operating system'} ];
    dts8:=new DisplayTicketsByStatus(consultant:=generalAdministrator, status:=#ClosedTickets);
    assert occurrence dts8;
assert equals dts8.answer()[Sequence]
    Tuple(createDate=2, department='General support', email='james_at_jordan.jam', number=4, priority=#Normal, subject='Reopening ticket')
};

}