Designer: a tool to design and model workflows*

Camilo Ocampo, Pere Botella

Departament de Llenguatges i Sistemes Informàtics
Universitat Politècnica de Catalunya
Barcelona, Catalonia

cocampo@goliat.upc.es, botella@lsi.upc.es

November, 1996

Abstract
This work presents the methodological and technical issues for the Designer tool in the OBJECTFLOW\(^1\) Workflow Management System. This tool provides to the user the possibility to model and design workflow diagrams from Business Process of a corporation. This project is the result of an industry-university cooperation.

1 Introduction
Workflow Management (WM) is an emerging area that involves cross-disciplinary fields as Database, Software Engineering, Business Management, Human Coordination, and so on.

OBJECTFLOW is a project [1] comprised into the Pan European Project (PEP, a project from the company CCS), in which an integral business management system for medium or big corporations is created. Its main objective is the development of a business information system integrated and automated by workflow tools. This project has been the result of a cooperation between an industrial partner, the CCS company (a software factory located near Barcelona), and a research group from the Technical University of Catalonia. Main aim of OBJECTFLOW is the development of a Workflow Management System (WMS) that can be totally integrated with the existing software applications of a corporation.

In this work we treat the methodological and technical issues for the Designer tool in the OBJECTFLOW WMS, which is a tool for modeling, enacting and managing Business Processes (BPs), tools to interact with workflow agents and other software applications. The Designer is the component that deals with modeling BPs, such task is assisted by the tool and a graphical

\(^*\) This work is part of the OBJECTFLOW project, supported by the Centro de Cálculo de Sabadell and the Spanish Ministry of Industry and Energy under the PEIN program 1059/95. Submitted to CAISE '97.

\(^1\) Authors know that a commercial product from Digital Equipment Corporation with the same name already exists. The naming decision for the project is CCS responsibility.
language. Also the business rules can be graphically expressed using appropriated graphical items and written with the User Procedure Language (UPL) [2].

This document is organized as follows: Section 2 presents a background of the Workflow Management, Section 3 describes the method followed to model and depict any BP of a corporation using the Designer, e.g., a travel request process or a software development process. Section 4 describes the Graphical Language used by the analyst to map the reality into a workflow diagram. Section 5 describes the modules that integrate the Designer. Section 6 presents an example of a BP and its corresponding map designed using the tool. Finally Section 7 presents the future work and conclusions. Appendix A contains the corresponding UPL code generated for the client credit request workflow presented in Section 6.

2 Workflow Management

As defined by the Workflow Management Coalition [3], a workflow is “the automation of a BP, in whole or part, during which documents, information or tasks are passed from one participant to another for action, according to a set of procedural rules”. We understand a BP as a set of linked procedures focused to reach a business goal embedded in an organizational structure. In a BP, the work assignment follows rules which may be less or more complicated depending on the organization and the process involved, these are called business rules.

A WMS is a system that support the BP automation. A WMS normally has four software components used to define, to create, to manage and to enact the workflow, furthermore it may have an interface with the workflow participants and an interface with other tools and applications. The component that manages the workflow enactment is called workflow engine. In the OBJECTFLOW project, the Designer is the tool used to define and create workflow maps from BPs.

Workflow Management is a discipline that emerged first in industry and later as a research area. The need to improve the efficiency in the organization and the possibility to model organizations as process-centered, and automate these models, have promoted the development of WMSs.

There are several successful applications reported in industry [4] and more than one hundred commercial products [5,6]. Also there are projects with both industry and research participation [7]. As a research area, there are four main fields: technology [8], conceptual modeling [9,10], organization [11] and methodology [12].

Main roots of WM come from different fields: office automation [13], software process management, manufacturing, transaction processing, organizational and management science [14]. These roots have influenced on the type of the workflow application. So, there are workflows that are imaging document based, mail based, and server based. The OBJECTFLOW WMS is a server based workflow.

3 OBJECTFLOW Method

To automate a BP, a qualified person (from now on we call this person the analyst) has to analyze the situation identifying the relevant items that participate in the BP. A number of documents and tasks must be done before designing a workflow map using the Designer.

Organization. An organizational map must be created to depict the corporation hierarchy, indicating different positions and persons.

---

2This map is called Template in the OBJECTFLOW terminology.
Roles. A list containing the roles that participate within the BP and the people who fulfill them, must be created. There is a mapping among the roles and the organization positions. A role can be played by more than one worker who can have more than one role thus the worker can have different kinds of responsibilities.

Documents. A list of relevant documents that participate in the BP must be created, and those that will be electronically managed must be generated with a word processor or another kind of software. Hereafter, we understand as document any multimedia information exchange between two processes.

Legacy Systems. It is also important to consider that a WMS normally has to integrate existing software which can be very heterogeneous, including different kinds of software and hardware platforms. The analyst must list the systems that participate in the workflow, specifying which of them are assigned either as automatic workers or just as applications to help in the realization of an activity.

Existing hardware and software support. A WMS is a computer system that coordinates the work among people, so it is indispensable that the organization have a network of computers to deploy such kind of system. The OBJECTFLOW WMS underlies on the Client Server Paradigm [1].

Before starting the Template, the analyst must determine what kind of workflow application is the most suitable for the organization and specifically for the BP to be designed and enacted. In OBJECTFLOW we adopted the Workflow Application Classification Scheme (WACS) [15] to accomplish this task. WACS treats twelve topics to help the analyst positioning the application. The important issues mentioned in this scheme are:

Application Type. Based on the rules followed to assign the tasks to the workers, the kind of workflow can be ad hoc, business or production [16]. In the first one, there are no rules or they are informal, in the case of business workflow, there are policies to assign tasks, but some users could choose freely (or almost) who shall do the work. In production workflows, the rules and the process are well defined.

In OBJECTFLOW we adopted a mixed approach: on the one hand, the analyst can program who does the work or to assign directly a task to a worker with the UPL (see Section 5.3). On the other hand, the task is both associated with a responsibility and a role (or a group of users), in this way any worker that fits the role's profile (rights and liabilities) can take the responsibility to do the task. The later case is present at the office many times, e.g., a group of secretaries choose documents to do from a pool of pending tasks.

The Designer allows both the dynamic definition of new workflows and modification of existing ones. This issue is important for undefined workflows (ad hoc), where the business rules can change rapidly. This feature also helps the analyst to model a BP, compile the corresponding Template and enact it with the workflow engine observing the behavior of the workflow. If any change or adjust is necessary, the analyst simply must repeat the process.

Performers. There are different agents that participate in the workflow construction and deployment. As we mentioned before, one of them is the analyst. The analyst is the person that models a BP using the Designer. The administrator manages the WMS, doing tasks like software installation, user's account administration, backups. The workflow users do the job supported by the WMS.

Another important figure is the external software supplier. So long as a WMS can integrate different software platforms, it can be necessary that an external supplier do certain modules. Here, determine the type of the workflow is important because some ad
hoc workflows can be implemented by the same final user, but Information Technology knowledge is required for the development of mission critical workflow applications, which are quite more complex than the ad hoc ones.

**Distribution.** The simplest workflow configuration is a local area network inside a building with only one workflow engine server. More sophisticated configurations includes workflows where users are located in different buildings even not in the same geographical area, but in the same organization. The most complicated case are a workflow coordinating different people from different organizations distributed over the planet with different WMSs.

The *Designer* allows to model external participation, but the cross-organizational workflows are not taken into account for this version.

**Complexity.** There are levels of complexity for a workflow application. The basic one is a linear set of linked procedures. A next level of complexity can be the parallelism of some steps or conditioning the flow. A hierarchy of procedures can add more complexity as some procedures can contain sub-procedures and son on.

The *Designer* permits the analyst to model a BP with the required complexity providing graphical items to deal with all this issues.

**Data life time.** The amount of workflow information that must persist, depends on the necessity of data for control and monitoring. Each workflow instance generates a certain set of data. If it is not important to keep this information, the application will result simplified in terms of database support, but while more time the data must persist, more complex are the mechanism to support it, e.g., transaction models, interoperability, distribution.

Under the OBJECTFLOW point of view, a workflow is viewed by three layers:

- **Template.** It is the static definition of a workflow and contains all responsibilities. Afterwards it will be instanced by several cases, the dynamic part.

- **Responsibility.** It contains the task (process) to be done and the role (or a specific worker) who can take the responsibility.

- **Task.** It is a set of activities to be done by a worker (human or software) in a finite amount of time. During a task a set of documents is created or just reviewed.

Designing the BP, the analyst identifies some responsibilities. Associated to each responsibility there is a task a worker must accomplish using or generating documents (normally) in a defined amount of time.

The responsibility has an associated role that defines the profile of the worker who performs the task. When the *Template* is instanced and the corresponding case is enacted, the workflow engine assigns the responsibility to a worker either by rules or it waits one of the corresponding set of workers to take the responsibility, depending on what was specified by the analyst. Also, the analyst can assign directly at design time the responsibility to a single worker. As we mentioned before, a worker can be either human or automatic. The automatic worker can be a program or an information system.

A task always is related to one responsibility and can be decomposed in one or more activities. A task can have documents from other tasks as inputs that contain the information that will be used during the task execution. The modified document (or the new one) are passed to the next responsibility as an input.
A task can have a deadline associated which specifies the time expected to do the task. When the deadline is met, a programmed action can be done or the worker is notified by a message.

4 Graphical Language

This section describes the graphical language provided by the Designer to model a BP. Such graphical representation is called Template which is composed by seven items that are used to produce a kind of graph, i.e., the workflow map. Six of this items are nodes related by arcs following special constraints. The items are shown in Figure 1. These items have an icon to represent them and each one has particular information (properties) that the analyst can fill through a special module (Properties Editor, see Subsection 5.2). This information is used by the tool to generate a code representation of the UPL that is interpreted by the workflow engine.

Following we describe each item and their specific properties.

Responsibility. It is the main item of a Template. It represents a duty the worker must accomplish. There are three types of Responsibility items: Normal, Nested, and External.

A Responsibility has a main task to accomplish subdivided in several activities that can be carried out using associated applications. The addressees are the set of workers that will perform the task. The analyst can assign a default addressee to a Responsibility, this worker is used by the workflow engine when an exception occurs, e.g., nobody has taken the responsibility and the lapse of time to do it has expired. Furthermore, the task has associated some documents that represent the information that flows through the workflow.

Each Nested Responsibility corresponds to an entire Template, i.e., this item refers to another map. This is the way to reuse and structure a Template.

The External Responsibility represents steps which cannot be integrated into the workflow, e.g., an external client who is not physically connected with the organization.

Other information associated with a Responsibility is the role describing the kind of worker that is expected to perform the task. The role of a Responsibility is shown as a text in the graphical representation.
**Begin.** It is the initial symbol in a Template. This item cannot have precedent items and only one following it. There can be only one Begin at a time for each Template, i.e., one per level in the hierarchy.

**End.** It is the item expected to end a Template. A Nested Template also ends with an End item.

**Boolean Condition.** It is the representation of a decision step, i.e., the binary evaluation of a logical expression. It indicates where the flows will go depending on if the condition is either true or false. Each condition has a text associated that depicts the condition to evaluate. Only Responsibility, Boolean and General Condition items can precede and follow a Boolean Condition.

**Device.** It represents a hardware device used by the worker to perform a task. It is only for commentary purposes, i.e., no code is generated for this item when the Template is compiled.

**General Condition.** It is a multiple decision step. Its behavior is like a switch in C language. The flow is determined by the resulting value of the expression evaluation, if this one not correspond to any specified value, a default flow that will be taken.

**Link.** It relates two or more items and represents the flow in a Template. The analyst can define different kind of Link items depending on its type:

- **Normal Link.** It is the simplest type of Link. Both source and target of the Link is only one item respectively.
- **And Link.** It means that it is absolutely necessary for the next step that all the tasks involved in the source of the Link are finished.
- **Or Link.** It means that it is only needed one of the previous tasks has finished to start the next step of the workflow.
- **Exclusive Or Link.** It is used when the analyst wants to model the fact that for a new task to start and finish, it is necessary that only and only one of the preceding tasks is finished.

### 5 Architecture

This section presents the modules that constitute the tool. The Designer is composed by four modules: Template Editor, item Properties Editor, UPL Code Editor & Compiler, and Documentation Generator. In Figure 2 we show the relations among modules.

The Template Editor interacts with the analyst through a Graphical User Interface (GUI). The Properties Editor is used to fill the information for each graphical item located in the Template Editor window. The UPL Code Editor & Compiler verifies at any moment the correctness of the pieces of code typed by the analyst and generates the intermediate code used by the workflow engine (UPL). When the analyst wants to document the work, the Documentation Generator takes the Template designed with the Template Editor and it produces a technical specification of the template.
5.1 Template Editor

The Template Editor is the component used to define workflows instead of directly programming them with UPL. This module provides a GUI which facilitates the description of the Template.

5.1.1 User Interface

The Designer has a GUI based on menus and toolbars as shown in Figure 3. It is an MDI windows application, i.e., it can handle multiple sub-windows at the same execution time.

Two toolbars are provided: one fixed in the menu for quick operations, e.g., printing a Template, the other one floating over the Designer and called the items palette. It allows to select or insert new graphical items.

5.1.2 Creating a Template

The first step in order to design a Template is create a new project. After that, the analyst selects an item, from the items palette, which has a button icon associated to each item. A project can contain several Templates displayed in their own window and can be saved in a file in a binary format.

Each incorrect operation the analyst performs is warned. There are wrong actions that can be detected at design time, e.g., trying to start a flow from an End item. Other errors can only be found when the analyst compiles the Template, e.g., it does not have a Begin item.

The analyst can define variables globally to the Template and use them in any procedure or function, as well as the analyst can define private variables with a local scope. Also, the OBJECTFLOW WMS variables can be referenced in any line of code.

Each graphical item and type of instances (from Templates, Cases, and Tasks) the engine manage has programmable events [2]. The analyst can associate UPL code to each event executed by the engine when the event occurs. The analyst can compile a Template and generate the UPL code at any stage of the design.
5.2 Properties Editor

This module allows to fill the properties of a graphical item. This information is used to generate the corresponding UPL code. The Properties Editor appears as a floating window over the selected item; depending its type, it shows properties of the item in different pages and allows to update them.

5.3 UPL Code Editor & Compiler

The UPL Code Editor allows the analyst to define the UPL functions or procedures associated to an event selecting the appropriated event depending on the working item. Also, it allows the analyst to generate the intermediate UPL code used by the workflow engine. The input for this module is a Template located in a Template Editor window.

The UPL Compiler checks the code. If an error exists, a message window is displayed to warn the analyst, indicating the reason and the place where the error occurred. This checking process occurs at design time and whenever the UPL code is generated.

5.4 Documentation Generator

At any step of the design, the analyst can generate documentation from the working Template. This module interprets the Template and produces information about each graphical item as well as general information (variables, function, user procedures).

The result is a file in Rich Text Format (RTF) that can be edited with any word processor that recognizes this format, e.g., Microsoft Word.
5.5 Implementation Issues

The Rumbaugh Object Oriented methodology was used to design and implement the tool. The platform for this version is Windows 95. We use Microsoft Visual C++ as programming language and the Microsoft Foundation Class to program all the GUI.

6 Example

6.1 The Client Credit Request BP

This example describes the process followed by a client who requests a credit from a bank. The corresponding UPL code generated by the tool is shown in Appendix A. The scenario begins with the client who does a credit requests to the Credit clerk. The clerk has to receive the documentation delivered from the client and registers the credit request. He delivers to the Credit Analysts the documentation needed, i.e., documents specifying data about the request (date, amount of money, name of the client, kind of client, etc.).

Next, Credit Analysts verify whether the client is a credit subject with a restriction of time: two days at most. A decision must be taken after this task has finished: Is the credit granted? If it does, another decision must be taken: if the credit amount is greater than $500 but less than $1,000, the Credit Manager must review the credit; if it is greater than $1,000, the Bank Manager must approve or deny the credit. Otherwise the process continues, the Credit Department has to update the appropriated database, deliver to the Credit clerk and the Financial clerk the client data and the credit amount. After this step, parallel tasks must be done: The Credit clerk must prepare the document which will be given to the client, and the Financial Functionary must give the money check to the client.

After these tasks have been done, the client finally receives the credit and the workflow finishes.

If the credit is not granted, the Credit Functionary has the task of notifying the client the situation, and then the workflow finishes.

The corresponding Template to this BP is shown in Figure 4.

7 Conclusions and Future Work

The WM is an enabler for human work coordination and useful tool for Business Process Reengineering. It relates disciplines from different science fields and first emerges in the industry as a mean to improve the efficiency at the organization.
The tool presented here is a part of the OBJECTFLOW project, product of the collaboration between university and industry. The OBJECTFLOW Designer is a flexible tool to model and design BPs through a friendly GUI dealing with the definition of the static part of a workflow.

As a new generation workflow tool it allows to depict different elements that participate in a BP: roles, users, documents, organizational positions, legacy and end user applications. Also it allows business rules programming through the UPL. It is flexible with the work assignment: it permits to predetermine who performs the job, or indicates the necessary role to take a responsibility, or program the work assignment with the UPL. It provides items to handle any level complexity of BPs, taking into account issues as concurrence, parallelism and synchronization.

The file format of the project is binary and Microsoft Visual C++ proprietary. The information for each graphical item is tightly coupled with the object oriented (OO) model of the Designer. This issue complicates constructing other applications to read the Template in an easy way because the OO model of the Designer must be well known to do this task. Also, any minimum change in the model cause that all previous project files cannot be read any more with the new release of the Designer.

For a new version of the Designer a different format will be implemented in order to separate the graphical information of the items from its attributes allowing to program a simpler and language independent mechanism to reconstruct the Template. In this way OBJECTFLOW WMS modules will able to handle Template files, displaying and interpreting them.

Roles and users definition are encapsulated in other tool. At this moment there is not interaction between the Designer and this one. It is important that the analyst can easily associate roles in the organization with responsibilities in the Template.

In the present version the scope of user's variables associated with a Template is local to this. For future versions this issue will be integrated to the tool allowing the analyst referencing variables from different Templates.

As we mentioned earlier, an important issue in WM are the cross-organizational workflows, that connect workflows from independent organizations. At the moment this feature is not covered but it is desirable that future versions could do it.

Acknowledgments

The authors are very grateful to CCS staff: Joan Canal, J.M. Espejo, J.R. Freixanet and J.L. Albero for their cooperation and participation in the development of the OBJECTFLOW project. We also thank M. Barceló, X. Burgués, C. Guberna and J. Carpintero for the UPC part. Finally we thank E. Pasarella and S. Silva for their helpful commentaries to this work.

References


Appendix A. UPL code corresponding to the example

' Template included in project: Central Bank
' Template contained in file: C:\Workflows\Central.p
' Template created: Sunday, November 24, 1996
' Template last modification made: Monday, November 25, 1996
' Template Client Credit Request
Const True = 1
Const False = 0
Const Default = 0
' Variables of the template
Dim counter As Integer
Dim day As Date
' Functions of the template
Sub Inc
counter = counter + 1
End Sub
Sub Init
counter = 0
day = #today()#
End Sub
' Internal Functions of the template
Sub ofte69960()
Init()
End Sub
Sub ofbe69960()
Inc()
End Sub
' Functions of Boolean Conditions of the template
Function EvalGrant?() As Integer
If Then
EvalGrant? = True
Else
EvalGrant? = False
End If
End Function
Function EvalApprove?() As Integer
If Then
EvalApprove? = True
Else
EvalApprove? = False
End If
End Function
' Functions of General Conditions of the template
Function EvalAmount() As Integer
Select Case
Case <1000
EvalAmount = 1
Case <500 & >1000
EvalAmount = 2
Case Else
EvalAmount = Default
End Select
End Function
' Responsibilities of the template
Responsibility Responsibility1
Source BEGINITEM, Responsibility2
Role Clerk
Task Task1
Target Responsibility3
End Responsibility
Responsibility Responsibility2
Source Responsibility1
Role Credit Analyst
Task Task2
Target Scanner, Grant?
End Responsibility
Responsibility Responsibility3
Role Client
Task Task3
Target Responsibility1
End Responsibility
Responsibility Responsibility4
Source Amount
Role Manager
Task Task4
Target Approve?
End Responsibility
Responsibility Responsibility5
Source Amount
Role Credit Manager
Task Task5
Target Responsibility6
End Responsibility
Responsibility Responsibility6
Source Responsibility5,
Responsibility10, Approve?
Role Financial Dept.
Task Task6
Target Responsibility7
End Responsibility
Responsibility Responsibility7
Source Responsibility6
Role Accountant
Task Task7
Target ENDITEM
End Responsibility
Responsibility Responsibility8
Source Responsibility6
Role Clerk
Task Task8
Target Responsibility9
End Responsibility
Responsibility Responsibility9
Source Responsibility7
Role Grant?
Task Task9
Target Responsibility3
End Responsibility
Responsibility Responsibility10
Source Amount
Role Credit Clerk
Task Task10
Target Responsibility6
End Responsibility
Responsibility Responsibility11
Source Responsibility8
Role Client
Task Task11
End Responsibility
Responsibility Responsibility12
Source Responsibility9
Role Client
Task Task12
End Responsibility
' Tasks of the template
Task Task1
End Task
Task Task2
End Task
Task Task3
End Task
' Boolean Conditions of the template
BoolCondition Grant?
Source Responsibility2
Target (True) Amount
Target (False) Responsibility9
End BoolCondition
BoolCondition Approve?
Source Responsibility4
Target (True) Responsibility6
Target (False) ENDITEM
End BoolCondition
' General Conditions of the template
GenCondition Amount
Source Grant?
Target (1) Responsibility10
Target (2) Responsibility5
Target (Default) Responsibility12
End GenCondition
' Devices of the template
Device Scanner("Clerk's scanner","Workstation1","scanner ")
' Begin item of the template
BeginItem Responsibility1
' End item of the template
EndItem Approve?, Responsibility7, Responsibility8, Responsibility9
' Template events of the template
Template.Activate = ofte69960()
' Begin events of the template
BeginItem.Init = ofbe69960()
' GeneralCondition events
Amount.Evaluate = EvalAmount()
' Begin events of the template
BeginItem.Init = ofbe69960()
' GeneralCondition events
Amount.Evaluate = EvalAmount()
' BooleanCondition events
Grant?.Evaluate = EvalGrant?()
Approve?.Evaluate = EvalApprove?()