TOWARDS THE DEFINITION OF A TAXONOMY
FOR THE COTS PRODUCT ´S MARKET
Ayala, Claudia P.
Research Report LSI-04-3-R
Towards the definition of a Taxonomy for the COTS product’s market
GUIDED RESEARCH PROJECT

Director: Dr. Pere Botella
Table Of Contents

INTRODUCTION ........................................................................................................................................... 3

THE GESSION PROJECTS ............................................................................................................................... 4

A TAXONOMY PROPOSAL BY GESSION ........................................................................................................ 7
   A CONCEPTUAL MODEL FOR COTS TAXONOMIES ..................................................................................... 12
   A SET OF CHARACTERIZATION ATTRIBUTES FOR BUSINESS APPLICATIONS ........................................ 13
   IDENTIFYING DEPENDENCIES AMONG BUSINESS APPLICATIONS ........................................................ 16
   MAIN CONTRIBUTIONS .......................................................................................................................... 18

TOWARDS THE ENHANCING OF GESSION PROPOSAL ................................................................................ 20
   1st METHODOLOGY [19]............................................................................................................................ 21
      The Strategic Dependency (SD) model .................................................................................................... 22
      The Strategic Rationale Model.............................................................................................................. 22
   GOAL-BASED REQUIREMENTS ANALYSIS METHOD [20] ..................................................................... 23
      Goal Analysis ......................................................................................................................................... 24
      Goal Evolution ....................................................................................................................................... 24
      Goal Analysis and Evolution ................................................................................................................ 24

REFERENCES .................................................................................................................................................. 28
Introduction

The GESSI (Software Engineering Group for Information Systems) research group of the Software Department at the UPC conducts research in many fields of software engineering, with particular emphasis on procurement and implementation of COTS components, requirements engineering, construction of quality models for software domains and software process modeling and enactment; also software certification is a target for the near future. We are also focusing in some particular kinds of software components and packages, remarkably ERP systems and container libraries.

In the field of selection of COTS components we are studying many concerned aspects addressed to build a proposal which is described and applied in [7]. This proposal has been well accepted in the international community (that paper was awarded the best in the International Conference on COTS-Based Software Systems 2004).

This proposal represents a good contribution to the area of COTS selection mainly because of the existence of such taxonomy provides a framework for the whole selection process and structures knowledge on the field. It means that there is a wide area of research.

Due to this, the core of this research is aimed at improving the building taxonomies issues in the context of the GESSI proposal, through the start of a new line of research for analyzing in depth those aspects concerned with formal methodologies for supporting and enhancing our proposal. This is the scope of this work of research, to show many topics that we think that can be helpful for our research work in the framework of the projects that are on going in GESSI.

In the first chapter, we show the projects ongoing in GESSI and their main scopes as a context to justify our activities and research lines (specifically this new line).

The second chapter is focused on showing the main aspects of the proposal, contributions and importance. The third chapter presents a framework of the related work that we are studying in depth.
The GESSI Projects

The GESSI (Software Engineering Group for Information Systems) research group of the Software Department at the UPC conducts research in many fields of software engineering, with particular emphasis on procurement and implementation of COTS components, requirements engineering, construction of quality models for software domains and software process modelling and enactment; also software certification is a target for the near future. We are also focusing in some particular kinds of software components and packages, remarkably ERP systems and container libraries.

In the software engineering field, those aspects concerning component-based software development currently play a prominent role, both in the case of components to be integrated in host applications and in the case of components that, properly aligned and implemented in the target company, become the application itself.

The statement of methodologies and the construction of tools to support these development paradigms can be a key point to enhance: the efficiency of the software process, through the reduction of production and maintenance costs, the accuracy and reliability of the decisions relative to the acquirement, integration and use of applications in the most critical points of the target companies from a strategical perspective.

In this context, the GESSI group has been working many years in projects that are concerning with these issues; the DALI (methodologies and tools for the Development, Acquisition, evaluation and Integration on software components) with government financial, and has presented the proposal of the UPIC (towards a Unified approach to the Procurement and Implementation of information system Components) project (the first is in the ending, the second is a continuation of the work related to the first - it is in approval stage -).

The main goal of DALI consisted on supplying a methodological and technological platform for component-based software development, stressing on the one hand, the case
of OO classes, and on the other hand, the case COTS software packages; and taking into account those aspects concerning the acquired components and the application development and maintenance processes.

Regarding the components, the goals were [1]:

1) defining a notation for stating software quality characteristics, specially non-functional ones, and using it for defining the metrics which are appropriate for the component-based software development framework;

2) building tools for helping the evaluation and selection of components described with this notation;

3) studying different aspects related to the construction of reusable components.

Concerning the software development process, the goals were:

1) defining a notation for modeling the software development process. This notation would be used in the particular case of component-based software development;

2) studying and defining flexible processes for the acquisition, evaluation and integration of software packages of different kinds and scale;

3) building tools for the definition and enactment of the software process models based on workflow technologies.

Regarding the topic of component selection that is the core of this work (we are focusing only in the issues concerning to this subject, being aware that many others are hidden behind the curtain), it is evident that the growing importance of COTS requires adapting certain software engineering practices, such as requirements elicitation to this emergent framework. Also some specific activities arise, among which COTS selection plays a prominent role.

The first approximation of GESSI to the topic of classifying COTS showed that there are proposals of specific taxonomies where the different domains and categories are classified [2], [3], [4]. But there are not enough deep studies about identification of properties that can help to organize the taxonomies. An outstanding exception are the works of the Politecnico di Torino Group about characterization attributes [5]. The need of to have these taxonomies is evident; we can see that it arises in very different contexts, from huge consultant organizations like Gartner to web sites as www.componentsource.com. Also websites like eCots of Thales enterprise [6] (it shows an example of the use of a taxonomy to organize descriptions of COTS components).
The main contribution of GESSI in this topic, is summarized and exemplified in [7] (this paper will be deeply analyzed in a next chapter, since it is the basis of the research approach that we want to present in this work). We use the concept of characterization attribute cited above to create a taxonomy of components for business applications as an example.

As a continuation of the works related on that paper [7], we are focusing our activities on the UPIC project, which in this sense, has among its objectives (we will cite only these concerning to the specific topic of Classifying COTS): to acquire, classify and characterize the knowledge that exists in the organizations involved in the Information Systems component market.

In this context, the scope of this work of research is to show the main elements of knowledge for approach the COTS selection problem exposed above from a formal perspective, in order to obtain a solid basis for a new line of research in the GESSI group.
A Taxonomy Proposal by GESSI

An increasing number of organizations are procuring off-the-shelf systems from commercial suppliers. However, successful selection of off-the-shelf system to fit customer requirements remains problematic. There are many well-known examples of systems failure due, at least in part, to poor product selection. New methods and techniques for requirements acquisition and product selection are needed.

We take as a basis the paper [7], which presents a proposal of taxonomy for classifying COTS business applications, i.e. products that are used in the daily functioning of all types of organizations worldwide. It proposes the identification of characterization attributes to arrange the domains which these products belong to, and also group these domains into categories. It defines questions and answers as a means for browsing the taxonomy during COTS selection. It shows the need of identifying and recording the relationships among the domains and proposes the use of actor-oriented models for expressing these relationships as dependencies.

In this chapter, we focus on a deeply explication of the aim of this proposal, because our objective in future research is to enrich it with a more formal method to establish the taxonomies; in this sense we related in following chapters the different aspects that we consider have to research for intending a solid base for our future research.

The amount of Commercial, Off-The-Shelf (COTS) products [8] available in the market are growing more and more. This situation requires adapting some software engineering practices, such as requirements elicitation to this emergent framework; and there is an increasing need for arranging the types of available COTS products to improve the efficiency and reliability of selection processes.

COTS selection poses some questions to be addressed such as [9]:

- How COTS components can be arranged in categories and domains for knowing which is the current state of the COTS market?
• How COTS components from a given domain are described, to make feasible their comparison when selection is required?

• How features of COTS components may be reconciled with requirements on them?

• Is it possible and realistic to describe COTS components and requirements in a structured and even formal manner?

It was mentioned above that the market of COTS components is huge and highly dynamic. On the one hand, new types of COTS components, i.e. COTS domains, appear day by day. On the other hand, new COTS components embrace often capabilities from more than one type, especially when they evolve through the years, making more difficult their analysis; this is the case of e-mail client packages, which often offer also functionalities for chatting or scheduling meetings, for instance.

For these reasons, the GESSI group advocates than improving the effectiveness and confidence of COTS selection requires:

• Having a taxonomy for arranging COTS domains. The existence of such taxonomy provides a framework for the whole selection process and structures knowledge on the field. The intermediate nodes of the taxonomy stand for general COTS categories: they are just classification means, not real COTS domains. Fig 1 shows an excerpt of how this taxonomy may look like. It shows that as many levels as needed may be introduced to catch similarities in the right point.
Fig 1. An excerpt of the ongoing taxonomy for COTS market

- Identifying the features that characterize each of these COTS categories and domains. These features capture the similarities of all COTS components belonging to the same COTS domain and also those of all COTS domains belonging to the same COTS category. Features are hierarchically inherited.

- Making explicit the relationships among COTS domains. It was proposed to use a i* SD-model [10] to visualize these relationships; if enough knowledge exists, parts of the SD-model may be refined into SR-ones. Fig. 2 shows an i* SD-model to make explicit a few relationships among the mail server and mail client COTS domains, and also the meeting scheduler one. Domains are modeled using the i* notion of intentional agent, while relationships take the form of dependencies. It becomes clear that scheduling meetings requires the ability of sending messages (goal dependency) and access to address books (resource dependency); also the mail client relies on the mail server on sending those messages. The i* SD-model shows that a COTS meeting scheduler not providing mailing facilities, requires a mail client COTS product to exist or to be also acquired. This kind of multiple selection has been addressed in [11].
• Classifying COTS components as belonging to one or more of these domains.

Key success factors in defining this taxonomy are:

• Define the right COTS categories and their proper decomposition. An appropriate number of categories and levels of the hierarchy is needed to present a good trade-off between knowledge structure and taxonomy management. Also, future evolution of the hierarchy must not be compromised by too early decisions.

• Define the right COTS domains. The granularity must be fine-grained enough to avoid failure of COTS components classification and wide-grained enough to avoid proliferation of artificial domains.

• Focus on functional dependencies among COTS domains, not on non-functional ones. If a functional dependency exists from one COTS component to another, for sure other non-functional ones exist, but we feel it is not necessary to reflect them at this stage. For instance, if the meeting scheduler needs a mail client to send messages, this implies that the reliability of the scheduler depends partly on the reliability of the mail client. This decision avoids proliferation of dependencies in the i* SD-model that are not useful in this context.

• Identify the appropriate set of features for each COTS category and domain. This set should be kept minimal to avoid having useless or meaningless features at any place of the hierarchy.

• COTS components classification is currently a focus of interest in many contexts, both purely academic and commercial. Concerning academic proposals [12], identify some relevant criteria for building a COTS market classification. This approach is more general than the GESSI proposal regarding criteria, because they
are not restricted to quality; but on the other hand, classification is a goal by itself, 
while in this proposal is a starting point for quality model definition (being aware 
that this aspects are hidden behind the curtain in this report, so that, we show only 
the issues concerned with building a taxonomy for COTS domains). 

In the paper [7], the proposal of taxonomies made by the GESSI group is showed by an 
example, which purpose is to provide specific support for improving the Business 
Applications (BA) selection processes. In order to achieve it, is built upon a taxonomy 
(described above), precisely, they consider that BA belong to one or more BA domains 
which appear as leaves in the taxonomy.

As showed before, a domain encloses a significant group of functionality. Domains are 
grouped into categories, and also categories can be grouped themselves to form a multi-
level taxonomy. Dependencies among domains that belong to the taxonomy are included 
in the hierarchy itself. The taxonomy can be incorporated to the selection practices, which 
are based on the use of quality models to assess the adequacy of components with respect 
to requirements [13], [14]. Those quality models are attached to nodes in the taxonomy, 
supporting model reuse by inheriting quality models downwards the BA hierarchy. 

The fundamental elements of a taxonomy are presented in the figure 3

![Diagram of a taxonomy](image)

*Fig 3. The fundamental elements of a taxonomy.*

It is important to remark that we do not take into account the aspect of quality models; we 
will focus only on taxonomy issues.

There are a lot of taxonomies not only for BAs but in general for other kind of domains 
[2], [3], [4]. But more important than the concrete form that a taxonomy takes, is the 
rationale behind its construction, i.e. which are properties that may help to arrange the 
BAs and how the taxonomy can be searched. This is especially true when considering not 
just the construction of the taxonomy, but its evolution. It is proposed the use as rationale
the notion of characterization attribute as introduced in, [5], [12]. Categories and domains may be arranged taking into account various characterization attributes.

A conceptual Model for COTS Taxonomies

A conceptual model, written in UML,[15] presents the formally concepts that appear in a valid taxonomy (figure 4).

A taxonomy is composed by categories and domains, generalized as scopes. Domains are grouped into categories and categories on their turn are grouped into other categories, forming a multi-level structure. A category C shall be partitioned into subcategories based on the values of a characterization attribute (attribute, for short) that applies to C. Also dependencies of four dependencies of four different types may be declared between domains.

Considering the intended use of the GESSI proposal, i.e. support for BA selection, some classes were added to the conceptual model in order to aid the identification of the domains of interest during selection processes. Thus, we associate one or more questions to attributes in each category. Each question explores the attributes with different criteria. People involved in the selection processes will answers these questions and since each answer is associated with an attribute value, they will browse the taxonomy at a more abstract level that using directly characterization attributes and their values. Examples can be associated to questions to clarify their understanding.

Finally, quality models are defined for each scope in the taxonomy. A quality model [16], [17] is defined as a multi-level hierarchy of quality features. Quality features are measurable and their values are computed using some metric. The quality models associated to each sub-category must differ from each other at least by one quality feature. If a partition of a
subcategory does not enrich the quality models then it only adds “noise”. That is, it makes the taxonomy more complex and adds a question that has no purpose, therefore the Occam’s razor (“one should not increase, beyond what is necessary, the number of entities required to explain anything”) is applied.

Attributes in general are not orthogonal, i.e. an attribute used to split a subcategory depends on the attribute used to split the super-category. In figure 3, attribute A1 is used to split category C1, it makes sense because attribute A0 has value v0.1 while it could be meaningless if applied to category C2 for which A0 equals v0.2.

A Set of Characterization Attributes for Business Applications

Since reuse is one of the main motivations of the work of GESSI group, it is applied in this method. Instead of starting from scratch and defining a brand new taxonomy, it was preferred to take an existing one as starting point and then refactoring it to obtain a valid taxonomy according to the criteria defined in the previous section. Given the particular type of systems that are addressed to, the focus is on the way that professional software consultant companies organize the BA’s services that they offer to their customers. In particular, the Gartner group’s [18] classification was selected, which is uniform enough and application-oriented.

The refactoring process that was performed was based on the identification of a set of characterization attributes and their associated values, questions and answers. Once the attributes were defined, they were used as the rationale to rearrange the BA categories and domains defined in the original taxonomy, with the goal of obtaining a final taxonomy as close as possible to the original one.

An excerpt of the taxonomy is presented in table 1. As first step during the refactoring, is was noticed that although the root of the original taxonomy was partitioned into eight sub-categories, it could not find an attribute that could be used to support it. Therefore intermediate categories were introduced. At this point an attribute able to discriminate among the original subcategories was needed. It was found that a good option was looking at the number of users; therefore two main categories were considered, corresponding to two values of the attribute number of users: single user and multi-user. Single user systems are used typically by one person to work on his/her own data, while multi-user systems operate on information shared by several people. Two different questions to elucidate the attribute’s values were identified:

- How many users has the system? Answers One or More than one.
- Does the system reconcile the interests of many stakeholders? Answer yes or no
<table>
<thead>
<tr>
<th>Categories</th>
<th>Domains</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Level 1</strong></td>
<td></td>
</tr>
<tr>
<td>a. Single-User Systems</td>
<td></td>
</tr>
<tr>
<td>b. Multi-User Systems</td>
<td></td>
</tr>
<tr>
<td>c. Management Software</td>
<td>h. ERP Software</td>
</tr>
<tr>
<td>d. Operational</td>
<td>i. Capture and Forms Processing Software</td>
</tr>
<tr>
<td>e. Internal Software Systems</td>
<td>f. Information Systems</td>
</tr>
<tr>
<td>g. Collaboration Software</td>
<td>j. Knowledge Management Software</td>
</tr>
<tr>
<td></td>
<td>m. Knowledge Presentation Software</td>
</tr>
<tr>
<td></td>
<td>n. Document Management</td>
</tr>
<tr>
<td></td>
<td>o. Content Management</td>
</tr>
<tr>
<td></td>
<td>p. Records Management</td>
</tr>
<tr>
<td></td>
<td>q. Web Content Management</td>
</tr>
<tr>
<td></td>
<td>r. Virtual Classrooms</td>
</tr>
<tr>
<td></td>
<td>s. Information Access</td>
</tr>
<tr>
<td></td>
<td>t. Workflow</td>
</tr>
<tr>
<td></td>
<td>u. Versioning &amp; Concurrency Control</td>
</tr>
<tr>
<td></td>
<td>v. Off-Line</td>
</tr>
</tbody>
</table>

Table 1. Partial view of the BA taxonomy.

Starting from this classification, subsequently other characterization attributes were identified, their corresponding values, their questions and their answers (see table 2). The questions are applied in more than one branch (see rows 3 and 7 in table 2). Note the correspondence among the questions’ level and the taxonomy structure.
Table 2. An excerpt of the attributes, questions and answers to browse the taxonomy.

After the process is finished, the differences among the original and resulting taxonomies were identified. The summary of the results are presented below. It includes two situations that we have not come across yet, but that could appear in the general case.

- **Identification of new scopes.** This is the most usual action we have taken, due to the nature of our activity. Many examples exist: categories such as Single User Systems and its heir Management and Operational Tools or domains as Workflow.

- **Division of existing scopes.** Occasionally, some original scopes were too coarse-grained mixing different concepts belonging to different characterization attributes. This is the case for instance of Collaboration and Knowledge Management Software which has been split into Collaboration Software and Knowledge Management Software (the latter as subcategory of Information Systems).
• *Join of existing scopes.* The other way round, some scopes are too similar that their differentiation does not really make sense (e.g. the Business Metrics and the Corporate Performance Management Software domains).

• *Promotion of domains to categories.* Some proposed domains could be actually categories, although we have not found examples of this situation.

• *Degradation of categories to domains.* Complex domains such as Enterprise Content Management Software were originally considered as categories although there exist products covering their intended functionalities (which makes them domains).

• *Removal of existing scopes.* Not very often, but some scopes appearing in the original taxonomy have been removed. Three main reasons behind:

  - The scope is not a software domain, e.g. the Voice and Call Processing Equipment category.
  - The scope does not really add value to the taxonomy.
  - The scope does not really belong to the category of BAs. For instance, the Instant Messaging or Audio and Video Conferencing domains are technical means for some team work domains.

It is important to remark that the taxonomy that was obtained is just one of the possible taxonomies for BAs. The goal here is not to find the “best” (if any exists) taxonomy but show how it is possible to build one [7].

**Identifying Dependencies among Business Applications**

The analysis of any segment of the COTS market shows that COTS products are not designed to operate isolated; instead, they work together and therefore some relationships may be established between them. This is especially true in COTS products of a large granularity.

Commonly, COTS products of certain domains depend on other with different aims [9], [11]. Among others we mention:

- *Enabling their functionality.* A product from a domain requires a product from other domain to provide a given functionality. E.g. in order to follow document life-cycles, document management tools need workflow technology to define them.
• Complementing their functionality. A product from a domain requires a product from other domain to offer an additional feature, not originally intended to be part of its suitability. For instance, a web page edition tool can be complement of a web browser to facilitate the edition and modification of web pages.

• Enhancing their quality attributes. A product from a domain requires a product from other domain to improve somehow its quality of service. For instance, consider the improvement of resource utilization that compression tools offer to other tools of any kind.

In the context of COTS selection, these dependencies among COTS domains have been dealt with in a case-by-case basis, i.e. they have been discovered in each new selection process, with the only exception of product lines architectures; it is important to remark that in the GESSI proposal does not include these relationships.

The strong belief of the GESSI proposal is that dependencies among domains shall be identified and recorded explicitly for their repeated use during different selection processes. Specifically, these dependencies help organizations involved in a selection process to find out that some goals that they want to achieve with a COTS of a domain will not be satisfied if they do not have or procure COTS from other domains. For instance, an organization selecting a document management tool would discover quickly that they need a document imaging tool for scanning and storing paper documents.

Furthermore, it was thought that domain taxonomies provide a great opportunity for including this information as an additional element for structuring COTS domains. In fact, taxonomies are arranged along two different dimensions, one using characterization attributes and the other using dependencies. The selection process can use the first dimension to determine the domain of interest through a repeated question-answer pattern, and then the second dimension to determine the additional products required.

Actor-oriented approaches allow the optimal identification and formal representation of dependency relationships between COTS products from different domains in a taxonomy. Specifically it is proposed the use of i* SD models [19], where actors represent COTS products and dependencies are established among them. In a dependency relationship, a depender COTS product from a domain depends on dependee COTS products from other domains.

The four types of dependencies of i*, namely goals, soft goals, tasks and resources, allow us to represent different relationships between domains:

• Goals: The depender depends on the dependee to achieve a new functionality. For the fact of being a goal dependency, the additional functionality can be provided in different ways. For example, the COTS products in the Document Management domain depend on COTS products of the Web Content Management domain to
offer the functionality of Visualization of Managed Documents through Web Pages.

- Soft goals: The depender depends on the dependee to enhance some of its non-functional attributes. For example, the Efficiency of COTS products in the Virtual Classroom domain depends on others that in fact are not BAs but technologic components (i.e., belonging to a different domain).

- Resource: The depender depends on the dependee for its access to some resource. For example, the Collaborative Engineering COTS products need the Definition of Resources like users, user types and document types, to define the life cycle and business processes of documents in an organization. This definition will be provided by a Document Management COTS products.

- Tasks: The depender depends on the dependee to carry out a task. For example the Document Management COTS products depend on COTS products from the Content Management domain to Store Documents in a Certain Format.

Figure 5 shows an i* SD diagram that includes a subset of the dependencies identified in the taxonomy of BA during the analysis of the document management domain carried out during a COTS selection process.

![Fig. 5. i* SD diagram with dependencies involving the Document Management domain](image)

**Main contributions**

- The concept of characterization attribute as a rationale for building a taxonomy in any category of COTS components, together with the questions, answers and examples as assets for making their use easier. Although introduced in the particular context of business applications, this approach can be applied to any other broad category of COTS products, such as scientific packages or life-cycle support tools.
The explicit representation of relationships among business applications domains using the concept of dependencies present in actor-based models. As a result, the implications of the use of a particular business application become clearer. As far as we know, this information is not included in other taxonomy proposals.
Towards the Enhancing of GESSI proposal

The GESSI group are detecting that the proposal exposed in the last chapter, can be enhanced with a more formal method for building the taxonomy. Therefore, ongoing work is focusing our activities to study, analyze and evaluate many topics related to what we think that can be applied.

We are doing an exhaustive study of the literature concerned with different methods of classifying, software reuse, goal-oriented idea for requirements analysis, i* methodology, and many proposals on COTS classifying.

The application of goal-oriented [24] approaches could be helpful to enhance our proposal, because it focus on why systems are constructed, providing the motivation and rationale to justify software requirements that we need for formalizing the building of taxonomies.

A critical factor in successful projects is often that developers not only understand what they are developing, but why they are developing a given system, goal driven approaches focus on this issues.

The general areas of work in the requirements engineering literature which are important to consider when discussing goal-based requirements analysis are: goal-driven approaches, inquiry-driven analysis, scenarios, viewpoints and negotiation. It should be noted that the analysis of goals is not unique to software; goals are also addressed in non-computing intensive arenas such as goal-based learning and strategic planning.

We are studying in deep the i* methodology proposed by Yu [19] and Goal Based Requirements Analysis Method (GBRAM) by Anton[ 20].

I* is a methodical approach to designing business processes in the context of information systems development. This approach offers a formal representation of goals and their
behaviors with a formal decomposition structure, treating non-functional requirements. The GBRAM is effective for treating functional requirements.

Yu’s strategic dependency model supports the process of suggesting, exploring and evaluating alternative solutions, providing the rationale for networks of actors in which agents depend on each other to achieve goals, perform tasks and furnish resources. The model facilitates the identification of what is at stake for whom, and what impacts are likely if a dependency fails. GBRAM discusses four types of dependencies: goal, precedence, agent, and contract dependencies. The approach adopted in GBRAM differs from Yu’s model in that dependency relations are used primarily to order goals so that they may be subsequently refined and it is focuses on aspects other than specific representations, emphasizing analysts tendencies to work with different sources of knowledge possessing various semantic properties.

In the field of classifying, we are studied many proposals [2],[5],[12],[21],[22],[23], they strengthen the main concepts of our approach ad enable to the aim that there are not enough deep studies about identification of properties that can help to organize the taxonomies (except [5], due to we are working together in this issues).

A summary of the most important topics is done:

**i* methodology [19]**

The i* framework was developed for modelling and reasoning about organizational environments and their information systems, as it attempts to articulate a notion of “distributed intentionality”. It consists of two models components: the Strategic Dependency Model (SD) and the Strategic Rationale Model (SR).

Some of the features of i*, using primarily a graphical representation. The i* framework has also been applied to business process modelling and redesign and to software process modelling.

The central concept in i* is that of the intentional actor. Organizational actors are viewed as having intentional properties such as goals, beliefs, abilities, and commitments. Actors depend on each other for goals to be achieved, tasks to be performed, and resources to be furnished. By depending on others, an actor may be able to achieve goals that are difficult or impossible to achieve on its own. On the other hand, an actor becomes vulnerable if the depended-on actors do not deliver. Actors are strategic in the sense that they are concerned about opportunities and vulnerabilities, and seek rearrangements of their environments that would better serve their interests.
The Strategic Dependency (SD) model

The Strategic Dependency (SD) model is used to describe the dependency relationships among various actors in an organizational context. It is a network of dependency relationships among actors. The intuitive meaning of a dependency is that a depender, by depending on someone else (the dependee) for something (the dependum), can accomplish some goal or objective that it would otherwise be unable to achieve (or not as well). If the dependum is not forthcoming from the dependee, the depender would suffer as a result, i.e., its attempt to accomplish the objective may fail or may be compromised.

The SD model therefore aims to capture the intentional structure of a process, instead of the usual non-intentional, and non-strategic process models of activities and entities. It is a higher level characterization of a process because it captures what matters to the actors, while leaving out non-essential details.

The model distinguishes among several types of dependencies based on how agents constrain each others freedoms, and the extent to which they are vulnerable in their dependencies. Dependencies are threaded through roles and positions, as well as physical agents, creating an intricate web of relationships.

The SD model supports analysis of who depends on whom for what, directly or indirectly. One can use the model to explore opportunities that are open to each actor, by matching the dependums that dependees offer and those that dependers want. One can analyze vulnerability, by tracing chains of dependencies. How far down a chain one might be concerned about vulnerability is based on a distinction of the degree of dependency into three categories—open, committed or critical. Using the SD model, one can identify who are the stakeholders, and what are their stakes. To validate a model, one can compare answers to various types of queries to see if they agree with what is expected intuitively.

The Strategic Rationale Model

In the Strategic Rationale (SR) model, the rationales behind process configurations can be explicitly described, in terms of process elements and relationships among them. The main types of relationships are represented as means-ends links and task-decomposition links. Means-ends links are seen as applications of generic rules in particular contexts. Process elements include subgoals, subtasks, resources and softgoals. The model is strategic in that elements are included only if they are considered important enough to affect the achievement of some goal. Agents may be able to accomplish something by themselves, or by depending on other agents. An interconnected collection of process elements serving some purpose for an agent is called a routine. An agent often has more than one
routine for accomplishing something. Process reengineering involves modeling existing routines (e.g., by asking “why” and “how” questions) and discovering new and better routines.

Beyond basic queries about nodes and links, the SR model offers four levels of analysis at a more aggregate level. An actor has the ability to accomplish something if it has a routine for it (“knowing how”). Next, one can check if the routine is workable, i.e., whether it is reducible to workable elements, through task-decomposition and means-end links, or workable dependencies. Thirdly, one can check if a routine is viable with respect to desired qualitative criteria. Finally, one can check whether the assumptions involved in reasoning about the routine and believable, i.e., sufficiently justified.

The framework provides support for raising issues, addressing them, identifying correlated issues, identifying assumptions and justifying them and settling issues and accepting assumptions.

The framework is intended to provide interactive support for an argumentative style of reasoning, not to fully automate the reasoning. It is assumed that the type of strategic reasoning being supported is largely a judgemental, iterative process, frequently based on incomplete knowledge. The aim of the framework is to provide modeling features, which can lead to semi-automated support facilities to help human users express, manipulate, organize, manage and draw conclusions from this knowledge.

The SD model and the SR models were built on a knowledge representation approach to information system development.

The i* framework was given as an example in which agent and goal-oriented concepts and techniques were adapted to address some of the early-phase of requirements engineering.

**Goal-Based Requirements Analysis Method** [20]

Goal-based methods stress the need to characterize, categorize, decompose and structure goals as requirements, but usually fail to offer strategies to identify goals, taking it for granted that the goals have already been documented. If goals have not been previously documented, how do we first identify them and know when they are all completely specified? the Goal-Based Requirements Analysis Method (GBRAM) has been developed in this context. The method is useful to identify, elaborate, refine and organize goals for requirements specification.

This work consider the goals from the perspective of two themes: goal analysis and goal evolution.
Goal Analysis

Goal analysis is the process of exploring gathered documentation, ranging from information about the organization (i.e. enterprise goals) to system specific information (i.e. requirements), for the purpose of identifying, organizing and classifying goals. It is often assumed that software systems are constructed with some goal(s) or purpose in mind. However, what happens when the goal or purpose is not clear? Goals are often not given, so where do they originate? Enterprise goals do not always reflect what actually takes place. Thus, it is important to gather as much information as possible in order to obtain a broad understanding of the domain, organizations, process and system. The starting point is always vague. Given different types of documentation, how should analyze them in order to identify and construct goals? Several techniques like: scenario analysis, identification of goal obstacles and constraints, and goal operationalization are useful not only for analyzing goals but also for elaborating and refining them.

Goal Evolution

From the developer’s viewpoint, we wish for goals and software requirements to remain as stable as possible. Although it is true that requirements are volatile and constantly changing, much iteration in the refinement process is due to the requirements simply being misunderstood and/or misinterpreted. Each stakeholder has different, and sometimes conflicting, requirements and priorities. Often the strategies for conflict identification and resolution are inadequate.

Goals are characteristically more stable than processes, organizational structures and operations of a system which continuously evolve; this is why we emphasize them. Nevertheless, goals evolve gradually and informally depending on the changing needs, circumstances and goal priorities of stakeholders.

Goal Analysis and Evolution

Some key concepts and terminology are presented in this section for provide a brief overview of GBRAM.
CONCEPTS AND TERMINOLOGY.

We define the following concepts:

• Goals are high level objectives of the business, organization or system. They capture the reasons why a system is needed and guide decisions at various levels within the enterprise.

• A requirement specifies how a goal should be accomplished by a proposed system.

• Operationalization is the process of defining a goal with enough detail so that its subgoals have an operational definition.

• Achievement goals are objectives of some enterprise or system. For example, a university course registration system may need to satisfy the goal of enrolling students in courses before the first day of class each semester. The object of the goal is course registration, which is exactly what the stakeholders believe the purpose of the system to be.

• Maintenance goals are those goals that are satisfied while their target condition remains true. They tend to be operationalized as actions or constraints that prevent certain states from being reached.

• Agents are the entities or processes that seek to achieve goals within an organization or system based on the implicit responsibility that they must assume for the achievement of certain goals. For example, given an Electronic Meeting System (EMS), a meeting initiator is the agent responsible for calling, or initiating a meeting.

• Constraints are requirements that must be met for goal completion. A constraint places a condition on the achievement of a goal.

• Goal decomposition is the process of subdividing a set of goals into a logical subgrouping so that system requirements can be more easily understood, defined and specified.

• Scenarios are behavioral descriptions of a system and its environment arising from restricted situations. They exemplify behaviors enabling hidden needs to be uncovered and are useful for evaluating design alternatives and validating designs.

• Goal obstacles are behaviors or other goals that prevent or block the achievement of a given goal. Abstracting and identifying goal obstacles allows one to consider the possible ways for goals to fail and anticipate exception cases.
**Process Elements**

Goal analysis concerns the exploration of documentation (for goal identification) followed by the organization and classification of goals. Goal evolution concerns the way goals change from the moment they are first identified to the moment they are operationalized in a system specification.

**Goal Analysis**

Goals may be extracted from various types of gathered information including process descriptions such as flow charts or Entity Relationships (ER) diagrams. It is useful to identify goals from process descriptions by searching for statements which seem to guide design decisions at various levels within a system or organization. When used as the exclusive source of information, process descriptions are insufficient for achieving thoroughness and completeness. Therefore, if possible, the practitioner should consider other possible sources such as transcripts of interviews with stakeholders. However, stakeholders tend to express their requirements in terms of operations and action rather than goals. Thus, searching for action words is a useful way to extract goals from stakeholder descriptions. For example, in a meeting scheduler, stakeholders may use action words such as 'schedule' and 'reserve' which give rise to goals such as: Schedule Meeting and Reserve Room.

In addition to goals, the agents, stakeholders and constraints must also be identified. How and when do we go about this? The most sensible approach is to identify the responsible agents as early as possible (when each goal is first identified) by determining what agents are ultimately responsible for the achievement or maintenance of a goal. For example, the goal Schedule Meeting is the responsibility of the Meeting Scheduler. Constraints are useful because they provide additional information regarding requirements that must be met in order for a given goal to be completed. How are such constraints identified? As a general rule, we identify constraints by searching for temporal connectives, such as during, before and after, or any variants thereof. Constraints may also be identified by looking for dependency relations. Consider the goal Meeting Arranged in the meeting scheduler system with the constraint: Meeting room must be available during the meeting date/time. Once the goals, agent responsibilities and stakeholders are identified and specified, the goals are then classified according to their target conditions and begin to evolve.

**Goal Evolution**

Goals evolve because stakeholders change their minds and refine and operationalize the goals into behavioral requirements. Over the course of time, a stakeholder’s goals may change or, at a minimum, their goal priorities are likely to change. The first type of change is elaboration and the second is refinement.

Goal evolution is thus effected via goal elaboration and refinement. Useful techniques for goal elaboration are: identifying goal obstacles, analyzing scenarios and constraints, and
operationalizing goals. Identifying goal obstacles in order to consider the possible ways for goals to fail enables one to anticipate exception cases. When goal priorities change, scenarios facilitate the evaluation of these new priorities. Goals are further elaborated by considering the possible ways in which goals can be blocked and by identifying scenarios to develop an understanding of how the goals can be operationalized.

Goal refinement occurs when synonymous goals are reconciled, when goals are merged into a subgoal categorization, when constraints are identified, and when goals are operationalized. In the GBRAM, achievement goals are merged and listed according to their precedence relations and dependencies. This ordering enables us to determine a goals’s preconditions and postconditions. It is beneficial to consider goal precedence relations such as “Which goals need to be fulfilled first? (this approach differs from Yu’s model in that dependency relations are used primarily to order goals so they can be subsequently refined).

Goals are refined by eliminating redundancies and reconciling synonymous goals. For example, the goals Meeting Arranged and Meeting Scheduled are synonymous and can be reconciled. In practice, the best approach is to eliminate redundancies after the goals (extracted from each source) have been merged into one ordered goal set. It is then easier to identify synonymous goals because they typically are listed adjacent to each other (or clustered) in the ordered set since they tend to share common precedence relations. Goals are also refined via elaboration. The operationalized goals, responsible agents, stakeholders, constraints and scenarios are ultimately consolidated into a set of goal schemas that can be easily translated into a requirements specification. The resulting artifact, while not formal in the strict sense, provides a textual representation of system requirements organized according to system goals.

An helpful example is done in [25]

This chapter confirms the areas of work concerned to improve the building of taxonomies that we are focusing. We are working for consolidating this new line of research in the GESSI group, and our tasks are addressed to analyze in depth those aspects concerned with formal methodologies for supporting and enhancing our proposal (the cited above in this chapter are the earliest lines that we are exploring).
References

1. GESSI Report “Memoria-CICYT-DALI”


Departament de Llenguatges i Sistemes Informàtics
Universitat Politècnica de Catalunya

Research Reports - 2004

- LSI-04-2-R: *Comparison of Methods to Predict Ozone Concentration*, Orozco, J.
- LSI-04-3-R: *TOWARDS THE DEFINITION OF A TAXONOMY FOR THE COTS PRODUCT’S MARKET*, Ayala, Claudia P.

Hardcopies of reports can be ordered from:

Núria Sanchez
Departament de Llenguatges i Sistemes Informàtics
Universitat Politècnica de Catalunya
Campus Nord, Mòdul C6
Jordi Girona Salgado, 1-3
03034 Barcelona, Spain
nurias@lsi.upc.es

See also the Departament WWW pages, http://wwwlsi.upc.es/