DesCOTS: A Software System for Selecting COTS Components

Gemma Grau, Juan Pablo Carvallo, Xavier Franch, Carme Quer
Universitat Politècnica de Catalunya (UPC)
C/Jordi Girona 1-3, UPC-Campus Nord (C6), Barcelona (Spain)
{ggrau, carvallo, franch, cquer}@lsi.upc.es
http://www.lsi.upc.es/~gessi/

Abstract

Selection of commercial-off-the-shelf software components (COTS components) has a growing importance in software engineering. Unfortunately, selection projects have a high risk of ending up into abandonment or yielding an incorrect selection. The use of some software engineering practices such as the definition of quality models can reduce this risk. We defined a process for COTS components selection based on the use of quality models and we started to apply it in academic and industrial cases. The need of having a tool to support this process arose and, although some tools already exist to partially support the involved activities, none of them was suitable enough. Because of this we developed DesCOTS, a software system that embraces several tools that interact to support the different activities of our process. The system has been designed taking into account not only functional concerns but also non-functional aspects such as reusability, interoperability and portability. We present in this paper the different subsystems of DesCOTS and discuss about their applicability.

1. Introduction

Software system procurement based on *commercial-off-the-shelf software components* (hereafter *COTS components*) is becoming a central task in software engineering. The huge offer of COTS components available in the market forces to choose the most suitable for each project. Unfortunately, this selection has a high risk of failure due to the lack of accurate and complete component information and the existence of ill-defined user requirements. To minimize this risk, several software engineering practices can be applied. In the last years, some methods have been proposed for dealing with COTS component selection [1, 2, 3, 4]. In all of them, a key point is the comparison of the user requirements that drive the selection process with the capabilities of the evaluated COTS components.

We have been involved in several academic and industrial selection projects that include the domains of mail servers, document management tools and requirements management tools, among others. Based on the methods mentioned above, we have followed a COTS component selection process with three clearly distinguished activities. In the first one, the domain of the COTS component that we need to select (hereafter, *COTS domain*) is analysed and a *quality model*, i.e. "a structured set of quality factors that describe the relevant features that software exhibits" [5], is built. The second activity consists in evaluating COTS components belonging to the domain in terms of the quality factors defined in the quality model. The third activity is the selection process itself in which the user defines his requirements and the evaluated COTS components are compared to them searching for the most suitable for being selected. According to the COTS-based development life cycle, these three activities are not sequential but iterative and can be overlapped as required.

Once we completed our first project, we realized the need of having a software tool to support the selection processes in which we were involved. We studied several existing tools with similar objectives such as miniSQUID [6], OPAL [7], the eCOTS portal [8] and IRqA [9], but none of them satisfied our needs (see section 9 for a more detailed analysis). Thus, we designed the DesCOTS system (**D**escription, **e**valuation and **s**election of **COTS** components). We refer to it as "system" because, as the selection process is divided into different activities, we decided to implement it as a set of tools that interoperate to support the whole process: the *Quality Model Tool* allows to define quality models; the *COTS Evaluation Tool* allows the evaluation of components; the *COTS Selection Tool* allows the definition of requirements that drive the COTS component selection; and the *Taxonomy Tool* allows to organize COTS domains as a taxonomy supporting reuse of quality models.

The paper is organized as follows. In section 2 we give an outline of the concept of quality model for software domains and we formulate a method for guiding its construction. In section 3 we present the specification of DesCOTS. The goals of every particular tool and their main functionalities are described in sections 4 to 7. In section 8 we propose some business issues for the system and the future work. Last, in section 9 and 10, we compare DesCOTS with other tools and we provide the conclusions.

2. Quality Models: The Basis of the Selection Process

Our selection process is based on comparing user requirements with the evaluation of COTS components. We focus in one particular case of user requirements, namely quality requirements. Quality requirements are often difficult to check because of their nature but also because of the lack of structured and widespread descriptions of COTS domains. This absence hampers the accurate description of software COTS components and the precise statement of quality requirements. As a consequence, the selection is damaged, and confidence on the result of the process diminishes. In our selection process, we propose the adoption of a *quality model* as an essential aid for solving this drawback.

Quality models are built from a catalogue of *quality factors* which are the basis for describing the quality domain that is addressed. Many catalogues are proposed that can be used as a departing catalogue. One of the most widespread existing quality frameworks is the ISO/IEC 9126-1 quality standard [5]. This standard organizes quality factors (referred to as *quality entities*) into six very high-level quality *characteristics* (functionality, reliability, usability, efficiency, maintainability and portability) that are decomposed into a first level of *subcharacteristics* (such as security, interoperability, etc., up to more than 20 high-level subcharacteristics). This hierarchical structure is generic enough to be adapted to any specific quality context and can be used as a departing catalogue in which other subcharacteristics and measurable attributes featuring an specific domain are added to complete its description, yielding to a multilevel hierarchy. We need to mention that the standard is not precise enough in some points and therefore some decisions have been taken such as [10]:

- Hierarchies of subcharacteristics and attributes are allowed without any restriction about their number of levels.
- An attribute may be associated to several subcharacteristics, as the standard does not forbid overlapping of quality factors.

As COTS domains are very different in their nature (e.g., ERP systems from requirements management tools; anti-virus tools from GUI libraries; etc.), quality models may dramatically differ from one domain to another. This fact implies that, in the construction of a quality model, the domain has to be deeply studied in order to establish the quality factors that best describe it. Thus, quality model construction may be a complex, time-consuming and cumbersome activity. For this reason, this activity requires adopting a method to build reliable quality models in an efficient manner. Some methods aimed at supporting this activity have been proposed. In particular, we have formulated elsewhere IQMC [11], an ISO/IEC 9126-1-based method that uses this standard as a starting point; it consists of the following steps (intertwined and iterated as needed):

- Add new subcharacteristics specific to the domain, refine the definition of some existing ones, or even eliminate some.
- Decompose the subcharacteristics into subcharacteristics. Subcharacteristics are high level quality factors used as an organizational level for classifying attributes and, if needed, they can be decomposed into other subcharacteristics.
- Decompose subcharacteristics into quality attributes. Attributes are quality factors that allow the evaluation of observable properties of the software components that belong to the COTS domain.
- Decompose derived attributes in terms of the basic ones. Some attributes can be measured in a direct way but others need to be further decomposed until being defined in terms of others.
- Determine metrics for attributes. Metrics for allowing the evaluation of both basic and derived attributes have to be established.
- State relationships between quality factors. Defining explicit relations between quality factors allow a better understanding of the model and a more accurate analysis during the selection activity.
- Identify requirement patterns for quality factors. During the definition of the quality model, some generic requirements can be defined in order to provide a pattern for the user requirements definition activity.

3. The DesCOTS System Description

In this section we describe the architecture of the DesCOTS system. We represent it using an *i** actor-based model [12], in particular, a Strategic Dependency (SD) model. An SD model describes the elements of a socio-technical system as actors and makes explicit the relationships among them. Actors are intentional entities, characterised by a rationale lying behind the activities that they carry out. Relationships represent dependencies among actors.

DesCOTS actors fall mainly into two categories: software and human. Software actors are the 4 tools composing DesCOTS that have already been introduced in section 1. Human actors are defined considering the three activities that are carried out in the context of our process for COTS components selection:

- Domain Expert: Studies a COTS domain and defines its quality model using the Quality Model Tool.
- Component Supplier: Evaluates COTS components belonging to a certain domain using the COTS Evaluation Tool.
- End Organization: Has the need of selecting a COTS component.
- Requirements Engineer: Under the supervision of the End Organization, defines requirements in order to perform the selection of a COTS component using the COTS Selection Tool.

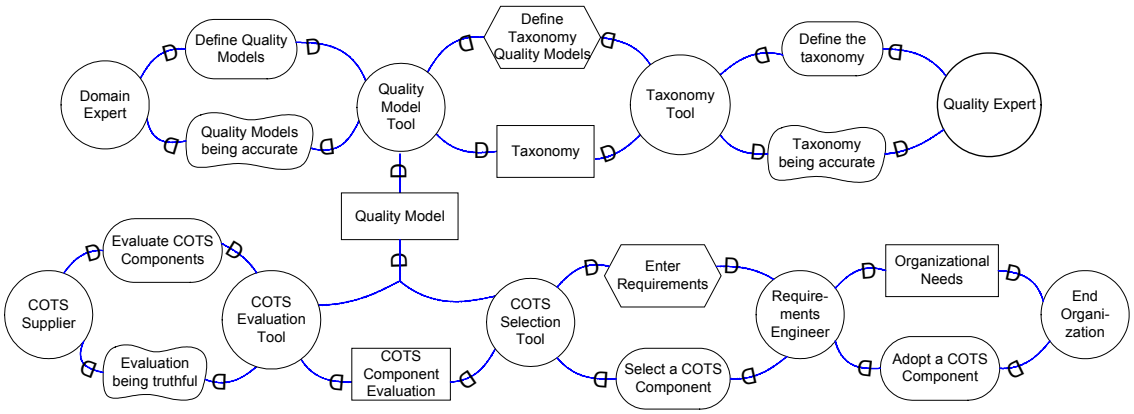

Figure 1. *i** SD model for the DesCOTS system (excerpt)

- Quality Engineer: Arranges the domains into a taxonomy using the Taxonomy Tool.

Figure 1 shows the *i** SD model for the DesCOTS system. There we can find the main goals of the experts that depend on tool support to be attained; these goal dependencies are represented by a curved rectangle in *i**: the Domain Expert depends upon the Quality Model Tool for defining quality models, the COTS Supplier depends upon the COTS Evaluation Tool for evaluating COTS components, whilst the Quality Expert depends on the Taxonomy Tool for defining the taxonomy of domains. The End Organization depends on the Requirements Engineer to adopt a COTS Component whilst the Requirements Engineer depends on the COTS Selection Tool for selecting a component. The softgoal dependencies (represented by a twisted rectangle) refer to non-functional requirements and, as it is showed, the tools depend on the corresponding experts for their data being correct. The resources dependencies (rectangles) express that the COTS Evaluation Tool and the COTS Selection Tool depend upon the Quality Model Tool for obtaining the quality model; the COTS Selection Tool depends on the COTS Evaluation Tool for obtaining the evaluations; and the Quality Model Tool depends on the Taxonomy Tool for getting the taxonomy. Finally the model includes two task dependencies (hexagons) modelling concrete ways to undertake activities: the Taxonomy Tool depends upon the Quality Model Tool for providing the starting quality model, whilst the COTS Selection Tool depends upon the Requirements Engineer for entering the requirements.

Some other dependencies could be added to obtain a more detailed description of the system, but we have omitted them in order to simplify the concepts. Some of these hidden dependencies would reflect that all the tools and experts depend on a User Management actor (that takes the form of an internal component in DesCOTS) for accessing the system and maintaining their personal data. Also, they would reflect questions such as traceability (external –where the knowledge comes from- and internal –how model elements are related-), validation protocols (e.g., a quality model cannot be used by the COTS Evaluation and Selection Tools until the Domain Expert has validated it), version management and administration (e.g., definition of methodologies for model construction).

4. The Quality Model Tool

In this section, we present the Quality Model Tool, which is aimed to support the method for the construction of quality models presented in section 2. The Quality Model Tool provides functionalities to define software quality factors, to reuse them among different quality models, to state relationships among them, to assign metrics for their future evaluation and to define requirement patterns, among others.

4.1. Defining the hierarchy of quality factors

The ISO/IEC 9126-1 standard fixes six quality characteristics and a first level of refinement into subcharacteristics. It is necessary to determine if this first structure is appropriate for the COTS domain under analysis by adding, redefining and/or deleting some of these characteristics and subcharacteristics provided as a departing point. In the general case, subcharacteristics may be further decomposed with respect to some factors, yielding thus to a hierarchy of them. As subcharacteristics are in a very high level they cannot be measurable, so it is necessary to decompose them into quality attributes. An attribute keeps track of a particular observable feature of the COTS components that belongs to the domain. Some attributes may be directly measurable but others may be still abstract enough to require further decomposition. Therefore, we distinguish between *derived* and *basic* attributes. Derived attributes should be decomposed over and over until they are completely expressed in terms of basic ones.

To help the understanding of a quality model, the tool shows its quality factors organised as a tree (see Figure 2). Some quality factors may appear in more than one branch due to overlapping. The tool allows constructing the model by browsing this tree and follows the concepts defined by the method, thus: only subcharacteristics can be added to the characteristics decomposition; it is not allowed to decompose

one subcharacteristic into subcharacteristics and attributes at the same time; and only derived attributes can be decomposed, not the basic ones.

In Figure 2 we show an screenshot of the Quality Model Tool where part of the hierarchy of a quality model for the domain of mail server systems is showed in the left. In the inner screen a derived attribute is added to the quality model and four buttons are provided in order to facilitate the traversal of the tree during the definition of the hierarchy.

4.2. Defining relationships among quality factors

To obtain a really complete quality model, relationships between quality factors should be explicitly defined. The Quality Model Tool allows stating these relationships between factors labelling them with values from a defined scale, such as "synergy" or "conflict". These scales are completely customisable, so new relationships of any elaborated type can be created. Once the relationships are stated, it is also possible to define intensities between them: a relationship may refine another one with a certain degree. We propose the adoption of the relationship style defined in [13].

4.3. Defining metrics for quality factors

To allow further evaluation of COTS components from the domain being modelled, we shall define metrics for quality factors. A quality factor may have more than one metric bound and, if the quality factor appears more than once in the hierarchy, it may also have different metrics in each appearance.

Taking into account some widespread proposals (e.g., [14, 15]) and also our own experiences, we have established an exhaustive categorization of metrics based on the interpretation of the measures that takes places in the evaluation process. We made a first distinction between *subjective* and *objective* metrics. A metric is subjective when it is not possible to establish a precise, non-ambiguous measurement procedure to get the value of the quality factor that it evaluates, but it is possible to give an appreciative value (subjective). Otherwise, the metric is objective and can be of various types. We have simple types such as boolean, string, integer, real and enumeration; and structured types such as set (e.g., for the set of protocols supported) and function (e.g., for stating that the response time of a component depends on the platform characteristics).

A metric can be bound to quality factors belonging to different quality models. Thus, in order to help reusability, the tool collects the metrics in a catalogue that may be browsed in every quality model under construction.

In order to ensure that component evaluations can be considered reliable, it is necessary to provide a rationale that explains how the metric must be evaluated. Thus, the tool asks for the definition of a measurement protocol when its users define a metric.

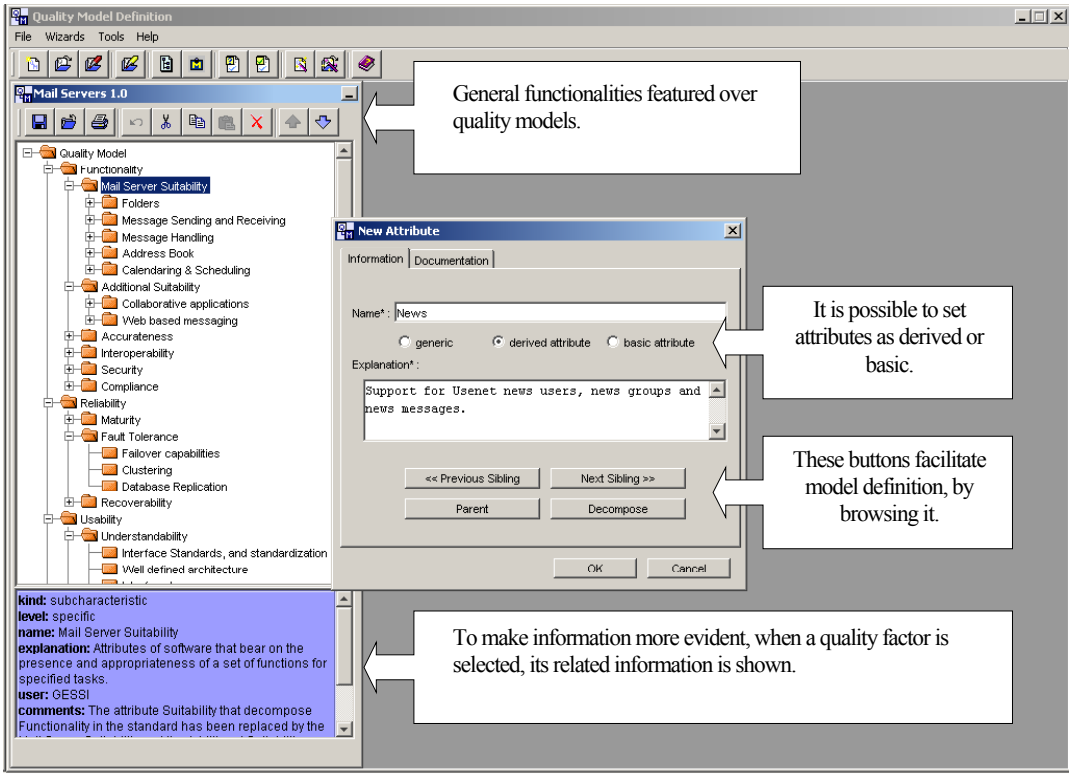

Figure 2. Quality Model Tool snapshot: adding a new attribute to the quality model hierarchy

4.4. Defining requirements patterns

As an advanced feature with the aim of facilitating later selection activities, the Quality Model Tool makes possible to identify some typical requirements related to the quality factors of the model, and construct a requirements patterns catalogue to facilitate requirements elicitation during the final activity of the selection processes. An example of requirement pattern will be "The response time when the mail server is requested for connection should not exceed [x] ms. for the [y]% of requests" where *x* and *y* will be customized by the user of the COTS Selection Tool in a particular selection project.

4.5. Other aspects

Some other functionalities have been added in order to obtain:

- **Reusability**: Although quality factors can vary from one domain to another, in practice we have observed that some of them appear over and over, which means that quality models are not completely different and some hierarchy parts can be reused. We will refer to these parts as quality patterns: pieces of quality models that appear in many other quality models. The tool promotes reusability by allowing the construction of a catalogue of quality patterns in which users can store their own patterns and look up all existing patterns (even those defined by other users). Quality patterns appear related to concepts such as user authentication, user interface description, etc. Thus, it is possible to copy hierarchical fragments from the quality patterns or from other existing quality models and paste them into the current model.
- **Composition:** As an extreme condition of the reusability situation, it may be the case that the modelled domain depends on other domains. For instance, when defining the security quality factors for the mail servers domain, the part concerning virus detection and repair can be defined in terms of the quality model of the anti-virus tools domain. The Quality Model Tool supports inclusion of quality models to deal with this situation.
- **Methodology.** We have motivated in the introduction the need of adopting a method for the construction of quality models. The Quality Model Tool has been built having in mind the method introduced in section 2 but it does not enforce its adoption. This has two different implications. On the one hand, the tool allows using methods in an informative rather than prescriptive way. On the other hand, other methods can be defined and adopted.

5. The COTS Evaluation Tool

The COTS Evaluation Tool supports the evaluation of candidate COTS components using a quality model defined with the Quality Model Tool for the domain the components belong to. Its main functionalities are the management of catalogues of COTS components and the management of their evaluations. So, the tool allows the registration, modification and removal of the different COTS components and the management of their evaluations.

As stated in the *i** SD model of Figure 1, the evaluations are entered by the Component Supplier by giving values to the basic attributes defined in the quality model of the domain. The values of derived attributes (defined as a formula in terms of the values of other attributes) are computed automatically as soon as the value of its operands are available; if any of the values of the operands change, the tool updates automatically the value of the derived attribute.

As evaluating COTS components is the main goal of this tool, the user interface was designed in order to facilitate it. We can see this fact in Figure 3 where the COTS Evaluation Tool shows all the information defined in the quality model: the quality model hierarchy (left), and the definition of the metrics and the measurement protocols for the Boolean evaluation (middle). On the other hand, it allows evaluating the model with the minimum interaction. Thus it is possible to browse the metrics of the quality model and to make searches both by quality factor and by metric.

6. The COTS Selection Tool

The COTS Selection Tool is in charge of mapping two different processes. On the one hand it supports the definition of selection requirements and, on the other hand, it analyses the stated requirements and the COTS component evaluations in order to inform the selection of a COTS component belonging

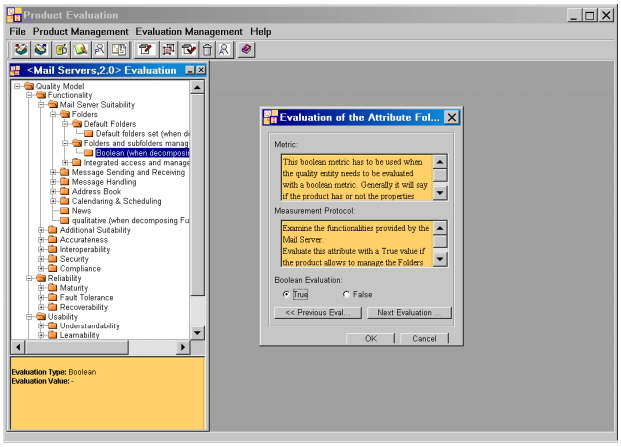

Figure 3. COTS Evaluation Tool snapshot: evaluating an attribute with a Boolean Metric

to an specific domain. Both processes are related to each other and can be applied in a cyclic way as many times as needed in order to effectively support requirements negotiation.

Selection requirements are defined upon the quality model of an specific COTS domain. Some requirements are obtained from the requirements patterns already defined in the quality model, while others are introduced as new. In both cases the requirements will be defined in terms of the quality factors of the model and this definition will be done in a formal way in order to facilitate the matching of the requirements with the evaluations of the COTS components. The tool allows the management of the list of requirements (creation, modification and deletion) and supports assignment of priorities, either using a user-definable scale (e.g., mandatory, important, optional) or the AHP multicriteria decision making technique [16]. With the aim of structuring knowledge, it also allows the decomposition of requirements. Thus, a requirement can be decomposed into others defining a hierarchy of requirements.

Once the list of requirements can be considered as definitive, the selection process takes place giving as a result one, several or none candidate components. The COTS Selection Tool allows some exploration by changing requirements priorities and defining filters over the results (e.g., show just the COTS components that satisfy more than 50% important requirements).

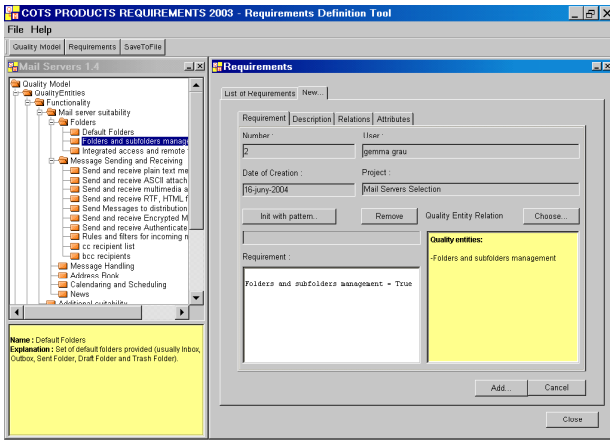

Figure 4. COTS Selection Tool snapshot: Defining a user requirement.

7. The Taxonomy Tool

As we have already explained, a quality model is never build from the scratch because we use the quality model provided by the standard ISO/IEC 9126-1 as the starting point. Despite of this, building a quality model can be a time-consuming activity, because the domain must be deeply studied, choosing which quality factors are the most appropriate.

In our practices, we have observed that some quality factors are not bound to individual COTS domains but to some of them, which form a kind of category. This is the case of the domains of mail servers and application servers, which share quality factors such as Failover Capabilities, Database Replication and Load Balancing, among many others. There are many other cases of those COTS *categories*, and this fact yields to the believe that reuse can be done better than just copying-and-pasting quality factors and defining quality patterns, that are the two form of reuse already mentioned in section 4. The key point is that the quality factors that these domains have in common can be inherited from a more generic quality model. In the case of mail servers and application servers, we can define a server category registering the mentioned shared factors. This classification is used to build a taxonomy of COTS categories and domains that is used both during the definition of quality models and the selection of COTS components.

The Taxonomy Tool is integrated with Quality Model Tool for supporting the reuse of quality models. When a new COTS domain is considered, first it is placed in the appropriate place of the taxonomy (eventually, this process can give light to new COTS categories). Once placed, the Quality Model Tool constructs the starting quality model by inheriting all the quality factors that appear in the quality models of the categories the model belongs to. Afterwards, the Quality Model Tool can proceed the usual way.

Also the COTS Selection Tool uses the Taxonomy Tool, because when the Requirements Engineer starts to determine the selection requirements, he or she can browse the taxonomy to find out which is the type of COTS component that is needed.

Figure 5 shows a snapshot of the Taxonomy Tool for the *business applications* that we propose in [17]. In the left, the tool shows the taxonomy in its hierarchical form where categories and domains are arranged with respect to various characterization attributes (e.g., number of users, objective, orientation). Overlapping of categories is permitted. In the right, the tool shows how these characterization attributes can be declared. Each characterization attribute has its corresponding values, questions and answers in order to make easier the use of the hierarchy. The questions can be applied at different levels in the taxonomy and some of them are applied in more than one branch.

8. Usage of DesCOTS

The DesCOTS system was designed with the purpose of being used by different kind of users that will probably not be working in the same location. To allow data interchange between these different users, we decided to design the system following a client/server architectural pattern. Each tool has its client program which can be installed independently from the others and just needs to access the server program to get and store the shared data using HTTP/XML, JAVA™ servlets and MySQL. All the libraries used are open source, following the aim of getting openness and flexibility on the system development and distribution. The client parts of the tools are small applications of 20 Mb on average size, plus 35Mb of the

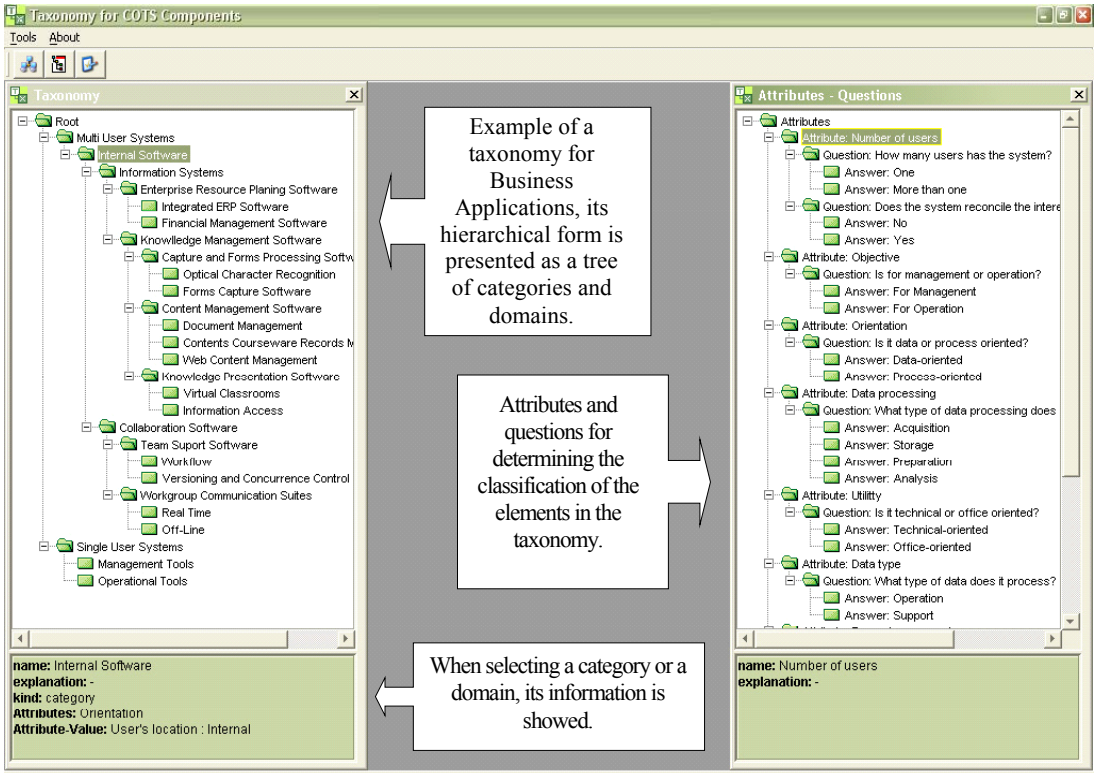

Figure 5. Taxonomy Organization snapshot: defining the taxonomy

Java Virtual Machine (JVM™) so it is easy to download and distribute. They run on a low-profile hardware and are easy to install.

The distribution of DesCOTS may follow any of the business issues discussed in table 1. We observe that the chosen client/server architecture facilitates an open collaboration between different communities by allowing to share the introduced data. This collaboration is controlled in different measure depending on the goal to achieve.

9. Related work

There exist some other tools in the market that can be used for the construction of quality models. We analysed four of them that are really new and representative of the types of tools that can be used for our purposes: miniSQUID [6] as a tool for defining metrics and quality factors; OPAL [7] as a tool for supporting a COTS selection process; eCOTS [8] as a platform for sharing massive information about COTS domains and components; and IRqA [9] as a typical requirements management tool.

The miniSQUID tool has been developed to store complex software metrics data sets and the metadata that describe them. MiniSQUID uses a *development model* as a framework for the development of the structure in which metrics will be set. This development model can be used as a quality model although the tool does not structure it in a hierarchical way. There is not distinction between characteristics, subcharacteristics, and attributes, and overlapping of quality factors is not permitted. Despite of this, it allows to state if an attribute is internal or external and to establish its type: Date, Identifier, Measure, String and Text. The units for the attributes can be of one of the types defined in by [13]: Absolute, Interval, Nominal, Ordinal, Ratio and an additional Undefined Ordinal Type.

The second analysed tool is OPAL, which is oriented to the selection of COTS components making easier their comparison; it offers and supports the construction of a hierarchical structure to organise user requirements. This structure can be used as a quality model and its items are customizable, so it is possible to include the information needed. This customization allows defining the elements in the hierarchy as quality factors, metrics, requirements or all of them; no more than one metric or requirement for quality factor can be defined.

The eCots portal is a platform that supports a catalogue of organization descriptions, COTS component descriptions, and COTS versions descriptions. The COTS are described in *detailed description templates*, which are based in industry standards. These templates are a list of criteria, not structured in a hierarchical way. A *detailed description instance* contains the data that describe a COTS component with respect to the description template. Although the data is available to the users,

Business Issue	Description	Goal
Personal Use	As it has a client/server architecture the system can also be installed in the same local machine for a personal use	Personal evaluation of the system
University Community Distribution	The clients parts of the tools can be installed in other universities using the same databases in order to share the work.	Provide to the university community facilities for applying our process
Corporative	Installation of the system in an organization that will use it for its own benefit without publishing any of its data	Commercial use of the system
Registered Use	For allowing to share the data generated with the tool among different organizations, it will be possible to give wide access to the tool to registered users	Allow data sharing between different organizations in a controlled way
Controlled Use	Give the privileges of a certain domain to the community that has a better knowledge of it	Use of the system for certification
Open Source	As the system has been developed in a open-source environment and all the data can be obtained in XML format, interested communities could reuse our code for supporting their own processes	Reuse

Table 1. Table with the different business issues of the tool

it does not provide any real selection facility such as comparing evaluations for a given factor.

Also Requirement Management Tools can be adapted to support the quality models hierarchy. Among them we have studied IRqA, which provides a very flexible framework to the definition and organization of requirements. In this tool customizable items call *facets* are used to structure the requirements and define aspects related to them. This facets, originally created to structure the requirements, can be used to define the characteristics and subcharacteristics related to the ISO/IEC-9126-1 and some metrics. The attributes are stated as requirements which are then associated to this facets, emulating a quality model structure.

Table 2 presents the comparison of DesCOTS and these three tools with respect to seven relevant criteria. As a general comment, we observe that DesCOTS is a more comprehensive system than the others, embracing the activities defined in the introduction as crucial in COTS selection. Concerning the particular criteria, we observe that the most salient features of DesCOTS with respect the others are: arranges quality factors into a hierarchy; organizes the COTS domains into a taxonomy; metrics are bound to particular quality models but stored in a repository; requirements are maintained close to quality factors; there is tool support for assisting the selection process; reuse is not limited to copy&paste and reuse of previous projects, because DesCOTS provides artefacts such as taxonomies of domains, patterns both of quality factors and requirements, and inheritance of quality models.

10. Conclusions

In this paper we have presented DesCOTS, a system for supporting COTS selection processes based on the use of software quality models. DesCOTS has been used in several experiences, in the fields of mail server systems, ERP systems, document management systems and others [18, 19, 20]. Our system presents some salient features:

- **Comprehensiveness.** It supports the definition of selection criteria, the classification of COTS domains, the evaluation of COTS components, the management of requirements and the selection process itself.
- **Foundations.** We use models and techniques such as quality models and AHP that are well-known in the field.
- **Decomposability.** DesCOTS' tools can be used as independent tools, which gives the chance to use them to support other software engineering practices such as software quality assurance during software development [21, 22] or as a baseline for arranging the criteria used during a software component evaluation [23].
- **Openness.** As the different tools share data in XML format, it is possible to interconnect them with other external tools, for instance using a repository or blackboard architectural style with quality models, quality taxonomy and COTS components evaluations at the heart of the resulting system. The other way round, it is feasible to get data from other systems and integrate it into DesCOTS, for instance evaluations of COTS components from other databases.
- **Flexibility.** We have defined several business models to be possible for exploiting the system.
- **Methodology.** The system provides facilities for defining methods of use, declaring its steps and linking them to the particular features offered by the tool.

Acknowledgments

The work reported in this paper has been partially supported by the CICYT project TIC2001-2165. Gemma Grau work has been partially supported by an UPC scholarship.

DesCOTS has been developed by Daniel Aut, Ana Candelario, Mark Foster and Gemma Grau.

Criteria	DesCOTS	MiniSQUID	OPAL	eCOTS	IRqA
Quality Factors Definition	ISO/IEC-compliant hierarchy of quality factors	List of quality factors	Hierarchy of quality factors	List of quality factors	Defined as facets
Domain Organization	Taxonomy of COTS domains	List of Development Models	List of Projects	List of Description Templates	List of projects
Metrics Definition	User-definable catalogue usable in various quality models	Choose from the provided types.	Customizable templates, units in terms of intervals.	Textual	Defined by facets
Evaluation Support	Assignment of values to quality factors	Supported, Allows generic evaluations.	Supported	Supported	Not supported
Requirements Integration	Req. management; Req. bound to quality factors	Not Supported	Supported	Not Supported	Req.management; Req. bound to facets
Selection Support	Matching requirements-COTS components	Not Supported	Matrix with evaluations of the products.	Not Supported	Not Supported
Reuse	Copy&Paste, Project Reuse, Patterns, Inheritance	Copy&Paste, Project Reuse, Definition of generic attributes	Copy&Paste Project Reuse	Copy&Paste Project Reuse	Copy&Paste Project Reuse

Table 2. Table comparing the DesCOTS system to the other tools

11. References

[1] J. Kontyo. "A Case Study in Applying a Systematic Method for COTS Selection". In *Proceedings of the 18st International Conference on Software Engineering* (ICSE), March, 1996.

[2] N. Maiden, C. Ncube. "Acquiring Requirements for COTS Selection". *IEEE Software*, 15(2), 1998.

[3] X. Burgués, C. Estay, X. Franch, J. Pastor, C. Quer. "Combined Selection of COTS Components". In *Proceedings of the 1st International Conference on COTS-Based Software Systems* (ICCBSS), LNCS 2255, February 2002.

[4] N.A.M. Maiden, H. Kim. "SCARLET: Light-Weight Component Selection in BANKSEC". In *Business Computer-based Software Engineering*, Kluwer Academic, v. 705, 2002.

[5] *ISO/IEC Standard 9126-1 Software Engineering – Product Quality – Part 1: Quality Model*, 2001.

[6] B.A. Kitchenham. *MiniSQUID handbook*. 2002. http://www.keele.ac.uk/depts/cs/se/e&m/handbook.pdf

[7] M. Krystkowiak, B. Bucciarelli, E. Dubois. "COTS Selection for SMEs: a report on a case study and on a supporting tool". *Proceedings of the 1st International Workshop on COTS and Product Software: Why Requirements are so Important* (RECOTS), September 2003, available at http://www.lsi.upc.es/events/recots/papers/Krystkowiak.pdf.

[8] J.C. Mielnik, B. Lang, S. Laurière, J.G. Schlosser, V. Bouthors. "eCots Platform: An Inter-industrial Initiative for COTS-Related Information Sharing". In *Proceedings of the 2nd International Conference on COTS-Based Software System* (ICCBSS), September, 2003.

[9] TCP Systemas e Ingenieria, Integral Requisite Analizer IrQA, available at: http://www.irqaonline.com/index_irqa.htm

[10] P.Botella, X. Burgués, J.P. Carvallo, X. Franch, G. Grau, J. Marco, C. Quer. "ISO/IEC 9126 in Practice: What Do We Need to Know?". In *Proceedings of the 1st Software Measurement European Forum* (SMEF), January 2004.

[11] X. Franch, J.P. Carvallo. "Using Quality Models in Software Package Selection". *IEEE Software*, 20(1), January/February 2003, pp. 34-41.

[12] E. Yu, *Modelling Strategic Relationships for Process Reengineering*, PhD. thesis, University of Toronto, 1995.

[13] L. Chung, B. Nixon, E. Yu, J. Mylopoulos. *Non-Functional Requirements in Software Engineering*. Kluwer Academic Publishers, vol. 5, 2000.

[14] N.E. Fenton, S.L. Pfleeger. *Software Metrics: A Rigorous and Practical Approach*. PWS, 1998.

[15] B.A. Kitchenham, R.T. Hugues, S.G. Linkman. "Modeling Software Measurement Data". *IEEE Transactions on Software Engineering*, 2001.

[16] T.L. Saaty. *The Analytic Hierarchy Process*. McGraw-Hill, 1990.

[17] J.P. Carvallo, X. Franch, C. Quer, M. Torchiano. "Characterization of a Taxonomy for Business Applications and the Relationships among them". In *Proceedings of the 3rd International Conference on COTS-Based Software System* (ICCBSS), LNCS 2959, available at http://www.springerlink.com/openurl.asp?genre=issue&issn=0302-9743&volume=2959&issue=preprint, February 2004.

[18] J.P. Carvallo, X. Franch, C. Quer. "Defining a Quality Model for Mail Servers". In *Proceedings of the 2nd International Conference on COTS-Based Software System* (ICCBSS), LNCS 2580, September 2003.

[19] P. Botella, X. Burgués, J.P. Carvallo, X. Franch, J.A. Pastor, C. Quer. "Towards a Quality Model for ERP System Selection". Chapter in *Component-Based Software Quality: Methods and Techniques*, A. Cechich, M. Piattini, A. Vallecillo (eds.), LNCS 2693, 2003.

[20] J.P. Carvallo, X. Franch., C. Quer. "A Quality Model for Requirements Management Tools". Chapter of the book *Requirements Engineering for Socio-Technical Systems*, A. Silva and J. L. Maté editors. To be published in 2004.

[21] R.G. Dromey. "Cornering the Chimera". *IEEE Software*, volume 20, 1996.

[22] J. Boegh, S. Depanfilis, B. Kitchenham, A. Pasquini. "A Method for Software Quality Planning, Control, and Evaluation". *IEEE Software*, 23(2), 1999.

[23] L. Beus-Dukic, J. Boegh. "COTS Software Quality Evaluation". In *Proceedings of the 2nd International Conference on COTS-Based Software System* (ICCBSS), LNCS 2580, February 2003.