Using Goals and Quality Models to Support the Matching Analysis During COTS Selection

Carina Alves[1], Xavier Franch[2], Juan P. Carvallo[2], and Anthony Finkelstein[1]

[1] Department of Computer Science,
University College London, London, UK
{C.Alves, A.Finkelstein}@cs.ucl.ac.uk
[2] Universitat Politècnica de Catalunya (UPC)
Barcelona, Catalunya, Spain
{carvallo, franch}@lsi.upc.es

Abstract. The selection process is a crucial activity of the development of COTS-based systems. A key step of the evaluation of COTS components carried out during selection is the matching between user requirements and COTS features. We propose a goal-based approach to guide the matching process, using quality models for leveraging goals and COTS features. The different mismatch situations that may arise are reasoned by means of exploratory scenarios. We demonstrate the approach with the mail server case study.

1 Introduction

The growing importance of COTS components (throughout the paper, we use the noun "COTS" as an abbreviation of "COTS component") requires the definition of processes, methods, models and metrics aimed at supporting COTS acquisition. One of the most important activities taking place in this context is *COTS selection* [3, 14]. For COTS selection to be successful (i.e., reliable and as less time-consuming as possible), many factors need to be taken into account, among which we mention: requirements shall play a prominent role during the process; a well-defined process shall be followed; selection usually involves multiple components; and knowledge about the COTS market shall be deep enough and shall be expressed properly. Our paper tackles these fundamental issues as follows.

Requirements. When selecting COTS, stakeholder requirements have to be assessed and matched against product features. In our approach, we employ a goal-oriented requirements engineering strategy [9].

Process. The evaluation of COTS usually reveals some mismatches that demand an extensive negotiation of requirements in order to accept products limitations [1]. In contrast with other proposed methods, our work aims at by supporting the matching process as a way to guide COTS selection.

Multiple Components. In real-world applications, selection of one component will usually require selection of others [6]. As a result, the process delivers an ensemble of components forming a configuration of the prospective system.

X. Franch and D. Port (Eds.): ICCBSS 2005, LNCS 3412, pp. 146 – 156, 2005.

Knowledge of the COTS Market. In this paper, we propose the notion of quality model [7] as a means to support the uniform description of quality features of components in the COTS market, as well as an essential aid for leveraging user requirements. This decision conforms to one of the lessons enumerated in [12], about making requirements as measurable as possible.

Summarizing, we propose a process based on goals and quality models to support the matching between COTS features and stakeholder needs. To facilitate the process we defined some matching patterns. The decision-making is based on concepts from utility theory [11] to measure to which extent COTS alternatives satisfy or not goals. We underline the importance to identify and tackle mismatches as early as possible. For that, we defined exploratory scenarios that help reasoning about mismatches and examine possible resolutions.

We use as case study some requirements for the selection of *mail servers systems*. Mail servers are a good case study not only for their strategic importance, but also because of their own nature (see [2] for details). Mail servers provide a lot of functionalities and exhibit a great deal of quality features which can be hard to analyze. In particular, features such as security control and operability shall demand additional COTS components to be selected and connected, e.g. anti-virus and backup and recovery tools. In order to demonstrate our approach in a practical fashion, we have defined a goal specification that we will use in the rest of the paper (see Table 1).

Table 1. Goal specification for the mail server case study

High level goals	Operational goals
g1 Ensure and communicate message delivery	g1.1 Configure number of delivery retries g1.2 Configure time between retries g1.3 Provide message delivery notification
g2 Ensure that messages never get lost	g2.1 Messages must never get lost if mailbox runs out of space g2.2 Messages must never get lost if a failure happens g2.3 Messages must never get lost if they cannot be delivered
g3 Ensure fast message delivering	g3.1 The average response time should not exceed 1 minute g3.2 Message throughput should be less than 5 minutes per Mb
g4 Support collaborative work	g4.1 Provide integrated document management g4.2 Provide instant messaging g4.3 Provide voice and video conferencing
g5 Ensure data security	g5.1 Provide authentication of users g5.2 Ensure data integrity
g6 Support protection against external attacks	g6.1 Provide anti-spam filters g6.2 Provide anti-virus scanning

The remainder of the paper is structured as follows. Sections 2 and 3 introduce the key concepts of goal and quality model and their relationships. Section 4 describes our proposal to guide the matching process. Section 5 introduces the notion of satisfaction function as the cornerstone of the measurement strategy. Section 6 shows the use of scenarios as a way to manage mismatches. Finally, we discuss related work and conclusions.

2 Specifying Goals to Evaluate COTS

The specification of stakeholder needs is generally the first activity of any system development. New challenges faced by COTS-based systems demand the definition of more flexible requirements statements in which stakeholder needs should be continuously negotiated and changed against the features offered by COTS. Based on that, we believe that *goal-oriented requirements engineering* is a suitable approach to specify genuine stakeholder needs without imposing unnecessary constraints. Goal-oriented requirements engineering is concerned with the formulation of requirements as goals to be achieved [9]. Goals can be specified in different levels of abstraction, ranging from high level, strategic objectives (such as "Support collaborative work") to low level operational concerns (such as "Provide voice and video conferencing").

High level goals capture the overall organizational objectives and key constraints; therefore they represent stable needs that are unlikely to change. Given that product capabilities change constantly affecting some previously defined requirements that will no longer be satisfied, requirements engineers should not spend too much time and effort to capture a complete set of goals. The initial set of goals will guide the definition of the system scope and the identification of COTS packages that might satisfy them. The specification of goals should be done in parallel with the evaluation of products. In fact, the analysis of features can help stakeholders to clarify vague goals as well as reveal desired functionalities that were not discovered with traditional elicitation techniques. Depending on the complexity of the application domain and the scope of available products, it is possible to find different COTS solutions ranging from a single, large package or several specific packages that once integrated will provide the desired capabilities. The next step of the goal specification process is the refinement of high level goals into more concrete subgoals until it is possible to objectively measure the satisfaction of subgoals that at this stage are called *operational goals*.

The prioritization of goals is particularly important when developing COTS-based systems because a number of goals might not be satisfied by any available product. Therefore, the assignment of priority helps to distinguish core goals (i.e. critical needs that should always be satisfied) from irrelevant goals (i.e. the ones that could be traded off with little trouble for stakeholders). We propose the assignment of normalized weights in order to guide the decision-making. For a detailed explanation on how to obtain goal priorities using utility theory, we refer to the systematic technique developed by Yen [13]. In particular, goal weights facilitate the identification of tradeoffs that stakeholders are willing to make. Tradeoff analysis involves the balancing of what stakeholders would like to get against what is possible to achieve with COTS products. Therefore, when performing tradeoffs, stakeholder goals should be continuously negotiated and priorities reassessed.

3 Quality Models and Goal Acquisition

In order to assess how well COTS alternatives meet operational goals, it is necessary to obtain precise metrics to quantify the satisfaction of each operational goal. We propose to use *quality models* for making goals operational. According to [8], a

quality model is "the set of characteristics and the relationships between them which provide the basis for specifying quality requirements and evaluating quality". Quality models are structured in a hierarchical way by refining the *quality factors* therein. The leafs of the hierarchy represent quality factors that can be directly measured and assessed; also, other derived metrics can be bound to quality factors represented by inner nodes of the hierarchy.

Quality models shall be built selectively, as required by the particular selection process at hand [5]. This is useful not only for limiting the effort invested in building them, but also for refining some goals and subgoals, for making them measurable and even for identifying new ones. Also, some new domains to be considered in the resulting COTS configuration may be discovered. We illustrate these situations by means of some examples in the mail server case.

Subgoal Identification. The initial form of g_3 expresses a very general goal that clearly demands some clarification. An initial approach we considered was to refine this goal into a subgoal such as "*Message transmission time shall take less than 1 minute*". However, building the part of quality model corresponding to the "*Time Behaviour*" quality factor provided us with a deeper knowledge. The quality model showed that in fact there are mainly two features that influence message transmission time, message throughput and average response time. This knowledge guided us to split the original goal into two subgoals, one for each feature.

Subgoal Refinement. To provide a measurable expression of the two identified subgoals, definition of the feature units (i.e., their metrics) becomes crucial. Consider, for instance, the subgoal concerning message throughput. We analyzed the information coming from a lot of sources, including widespread benchmarks such as the Microsoft [10]. All the benchmarks that we examined provide different efficiency tables for different messages sizes (among other information). For this reason, we were able to formulate a more accurate definition of the subgoal $g_{3.2}$ taking this factor into account, as "*Message throughput should be less than 5 minutes per megabyte"*.

Dependent Features. Not surprisingly, some of the subgoals depend on factors that are external to the system being developed. Subgoal $g_{3.2}$ is an example. The benchmarks showed that besides message size, some organizational aspects (e.g., number of registered users and expected concurrent access rate to the mail server) and platform components and policies (e.g., number and characteristics of hosts and protocols used) influence message throughput. Consequently, some subgoals will be said to be conditionally fulfilled, i.e. they will be attained just for particular values of these organizational aspects and particular configurations of these platform components and policies.

New Domains of Interest. Goal g_6 refers to system security. Again we built selectively the piece of the quality model related to this quality factor. In this case, one of the quality features that influences security is protection against virus attacks. In fact, we decided to define a subgoal ($g_{6.2}$) bound to just this attribute. However, market studies show that virus detection and removal is not a feature generally offered by mail server packages; instead, mail servers incorporate (mail-specific) anti-virus tools.

Consequently, goal attainment requires an anti-virus tool to be integrated in the final COTS configuration and anti-virus domain must be incorporated into the discussion.

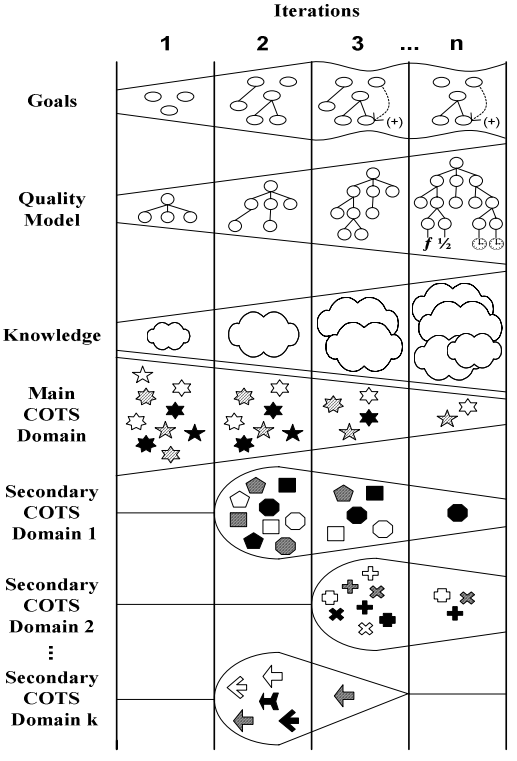

Fig. 1. Multiple COTS selection process using goals and quality models

In summary, goal specification, knowledge of the domain and quality model construction are activities closely related in our approach. Figure 1 shows in a graphical form the evolution of concepts and the solution space through time. This figure is inspired by the characterization of the PORE methodology [12], but takes multiple selection and quality models into account. In the beginning, the departing system goals and the initial set of candidate components for the domain of interest are determined; the departing quality model is also included at this initial stage. Whilst the process proceeds, goals may slightly change, some candidates are eliminated and the quality model is built selectively refining just those parts directly related to the goals; also, new domains may show up as part of the selection process. It may also happen that all the candidates for a particular domain are discarded, which means that bespoke software must be developed for covering this part of the system. At the end, some particular configurations emerge as the solutions to be proposed to the management.

4 Matching Goals and COTS Features

We have defined a set of matching patterns to help decision-makers to classify the matching between COTS functionalities and goals in a systematic way. We present below these patterns and provide examples.

Fulfill - The operational goal is fully satisfied by the product, which means that the goal is achieved at the target level. This is the usual case in operational goal $g_{5.1}$. Most mail servers available in the market provide reliable and sufficient authentication facilities and then the operational goal is fulfilled.

Differ - The operational goal is partially satisfied by the product. The *differ* pattern occurs when the satisfaction of the operational goal is within the acceptable interval but not optimal. For example, consider the feature *Delivery retries configuration* that maps the operational goal $g_{1.1}$. We have analyzed a particular mail server *Foo* that does not allow full configuration of number of delivery retries, but just allows users to configure delivery retries during the first 24 hours. Therefore we say that this particular mail server differs from the desired operational goal.

Fail - The satisfaction degree of the operational goal is below the worst level of the acceptable interval. The *fail* mismatch occurs in two situations: when the COTS product does not meet the operational goal at the requested level or when it does not exhibit the desired functionality. Some evaluated mail servers fail to satisfactorily support anti-virus facilities. In this situation, a potential alternative to satisfy $g_{6.2}$ could be acquiring a specific anti-virus tool, yielding a new candidate COTS configuration composed by mail server and anti-virus tool.

Extend - This case occurs when the COTS product provide functionalities that are not requested by the stakeholders. The *extend* pattern can give rise to the following interaction situations:

- *Hurtful* - The extra feature has a negative impact over stakeholder goals, so that it might interfere with other functions of the system (e.g. automatic data backup facility can affect the response time goal);
- *Helpful* - The extra feature is accepted, such that it might be included in the goal specification as part of the feedback mechanism;
- *Neutral* - The extra feature does not interfere with the achievement of any goal nor it is a desired functionality.

Last, we remark that in some situations, evaluators may not have sufficient information about packages features to classify the matching. Therefore, further clarification is needed in order to verify the matching. In other words, the pattern is *unknown*.

To measure the degree to which COTS candidates satisfy each operational goal, it is necessary to define the interval of acceptable values in terms of quality model elements.

5 Defining a Measurement Strategy

We use concepts from utility theory [11] to obtain the satisfaction function of operational goals. We assume that an operational goal expresses a condition over a quality factor, that we call its *underlying quality factor*. The *satisfaction function* of an operational goal g_i is defined as:

$$\mathrm{Sat}_{gi}: \mathrm{M} \rightarrow [0, 1] \tag{1}$$

where M is the set of values that the underlying quality factor q_i of g_i may take.

Each of the matching patterns defined in the last section corresponds to different degrees of goal satisfaction (see Table 2). Note that the extend pattern is not applicable since it expresses something that is not a goal, it may become a goal (helpful extend and then other pattern would be applied) or not. The unknown pattern requires further exploration using scenarios, as explained in the next section. The acceptable interval ranges from the *target level*, i.e. the highest desirable value of the underlying quality factor q_i that fully satisfies the goal, to the *worst level*, i.e. the minimum level that a goal would be considered satisfied. These two levels are the boundaries for the application of the fulfill and fail patterns.

Table 2. Satisfaction value for each matching pattern

Matching pattern	Satisfaction function value
Fulfill	1
Differ	0.9, …, 0.1
Fail	0
Extend	Not applicable

Given the acceptable interval to satisfy each operational goal g_i, we can determine the satisfaction function of g_i. Consider that x_{target} and x_{worst} are respectively the target and worst values that q_i may take to satisfy g_i. Then, we have that $\mathrm{Sat}_{gi}(x_{target}) = 1$ and $\forall x$: $x < x_{worst}$: $\mathrm{Sat}_{gi}(x) = 0$. For simplicity reasons, we assume that all goals have a linear satisfaction function in the form:

$$\mathrm{Sat}_{gi}(x_k) = a_k x_k + b_k \tag{2}$$

where a_k and b_k are constants defined as:

$$a_k = 1 / (x_{target} - x_{worst})$$

$$b_k = -x_{worst} / (x_{target} - x_{worst})$$

to make sure that the satisfaction function is continuous. Then we have:

$$\mathrm{Sat}_{gi}(x_k) = 0, \text{ if } x_k < x_{worst}$$

$$\mathrm{Sat}_{gi}(x_k) = a_k x_k + b_k, \text{ if } x_{worst} \leq x_k < x_{target}$$

$$\mathrm{Sat}_{gi}(x_k) = 1, \text{ if } x_k \geq x_{target}$$

Consider, for example, that the acceptable interval for the operational goal $g_{3,2}$ ranges finally from 4 minutes/Mb to 6 minutes/Mb. These values are respectively the worst and target levels. Figure 2 shows the goal refinement tree for the high level goal e*nsure fast message delivering*, the diagrammatic acceptable interval for the operational goal *message throughput*, and its satisfaction function.

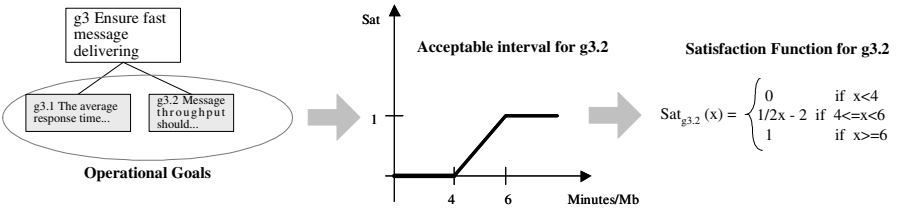

Fig. 2. Defining the acceptable interval and satisfaction functions

By solving these linear equations, we can determine the satisfaction function of each operational goal. The next step is to measure how each COTS satisfies operational goals. More formally, consider that COTS A satisfies goal g_i at level x_k, i.e. the underlying quality factor q_i of g_i has a value x_k in A, denoted by $A_{qi} = x_k$. Since we already know the satisfaction function of g_i we can easily obtain $Sat_{gi}(A_{qi})$ Then, the overall COTS satisfaction is obtained by aggregating individual preferences. We use the weighted summation to aggregate individual preferences, which is a well-known and simple aggregation operator. Then we have:

$$Sat(A) = \sum_{i=1}^{n} w_{gi} \times Sat_{gi}(A_{qi}) \quad (3)$$

where w_{gi} is the weight of goal g_i (see section 2) and $Sat_{gi}(A_{qi})$ represents the satisfaction degree that COTS A meets with operational goal g_i.

6 Scenarios to Manage Mismatches

The overall satisfaction that each COTS meet operational goals allows decision-makers to compare different COTS products. In order to perform wise decisions we still need to handle mismatches and analyse tradeoffs. Given that mismatches represent non-adherence of COTS packages to operational goals, a fundamental need to handle mismatches is the capacity to systematically structure tradeoffs.

This section describes how mismatches can be tackled using exploratory *scenarios*. The benefits of using scenarios to deal with conflicts are as follows: (*i*) to explore resolution alternatives and highlight products limitations; (*ii*) to identify associated risks with each COTS; (*iii*) to explicit evaluate the impact of decisions. Once mismatches are detected, we aim at exploring the possible conflicting situations through the combination of different scenarios. By identifying scenarios that lead to

unwanted situations, evaluators can clearly reason about why a conflict has arisen and which are the consequences of the conflict. We use semi-structured textual form to represent the scenarios. Table 3 depicts an example of exploratory scenario where we investigate the involved conflicts detected on a particular COTS configuration composed by mail server and anti-virus products. The first helps to solve an unknown pattern. The second is concerned with priorisation, which affects the weighting factor in the satisfaction function. The third relates to goal decomposition. By choosing potential resolutions, more information for COTS selection becomes available.

Table 3. Exploratory scenarios

Scenario 1. Selection of COTS configuration composed by mail server and anti-virus tool.	
Conflicting situation 1	Subgoals $g_{3.1}$ $g_{3.2}$ are difficult to evaluate.
Involved issues	Efficiency benchmarks available are not trustable. Performing test cases demands great effort.
Fixed parameters	Number of users, average message size (the latter obtained from estimation)
Negotiable parameters	Communication protocol, platform
Potential resolutions	1. Increase server resources, add servers to clusters, activate load balancing in order to ensure higher system efficiency. 2. Put more human resources to obtain more trustable information.
Involved risks	High cost due to acquisition of servers or man power.
Conflicting situation 2	Conflict between $g_{2.2}$ and $g_{3.1}$.
Involved issues	Negotiate the efficiency of message delivery against the availability and recoverability capabilities.
Fixed parameters	Recovery process strategy
Negotiable parameters	Level of concurrency, maximum allowed size of messages
Potential resolutions	1. Relax g_3 since the tradeoff decision is to favor reliability with loss of efficiency.
Involved risks	1. Loose data if a failure happens 2. Sacrifice message response time
Conflicting situation 3	The definition of subgoal $g_{6,2}$ is not sufficient to choose which is the best anti-virus tool.
Involved issues	The selected anti-virus tool must be compliant to the mail server
Fixed parameters	License agreement, platform (dependent on mail server)
Negotiable parameters	Not identified
Potential resolutions	1. Refine the goal 2. Gather more information about available anti-virus tools
Involved risks	Due to the high contribution of $g_{6,2}$ the satisfaction of g_6 might be compromised if no anti-virus is selected.

7 Discussion and Related Work

In this paper we have discussed the importance to analyse how well COTS features match stakeholder needs. Our motivation has been to provide a framework for

supporting the matching process and managing conflicts in COTS-based development. We have demonstrated the suitability to combine goals and quality models as both approaches represent knowledge in a hierarchical fashion. We may say that our work joins two lines of research: how to operationalize goals in a methodical way (using quality models) and how to drive quality model construction (through goal identification and refinement). Both concepts fit in a very smoothly way. The definition of matching patterns provides a qualitative and well-defined basis to assess the satisfaction of goals in terms of COTS features. Utility theory is a suitable decision-making approach to capture the notion of satisfaction degrees. Finally, exploratory scenarios provide an effective mechanism to explicitly reason about mismatches and manage risks. Most selection methods present in the literature overlooked the matching problem. One of the few works that covers these issues is provided by Wallnau et al. In [14], he proposes the use of utility techniques also in the field of COTS-based system development. They identify two situations in the matching process, fit and misfit, and for misfits they propose to quantify the costs and risks for assessing the final decision. Although the underlying ideas of their approach and ours have similarities, we emphasize the model aspects covered by quality models, matching patterns and scenarios, which are dealt with in an ad-hoc manner in their proposal. Also, the relationships among requirements and COTS feature evaluation seem not to be explicitly addressed in the matching process. In [4] Chung provides an approach called CARE that emphasizes the importance of bridging the gap between the sets of native (i.e. requirements) and foreign requirements (i.e. COTS features). As main drawback, the approach does not provide or suggest any effective solution to support the possible mismatching between both specifications.

Acknowledgement. This work is partially supported by CICYT TIC2001–2165.

References

1. C. Alves, A. Finkelstein. Investigating Conflicts in COTS Decision Making. *International Journal of Software Engineering and Knowledge Engineering.* World Scientific Publishing Company, 2003.
2. 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 Systems* (ICCBSS), Ottawa (Canada), LNCS 2580, 2003.
3. S. Comella-Dorda, J. Dean, E. Morris, P. Oberndorf. A Process for COTS Software Product Evaluation. In *Proceedings of the 1st International Conference on COTS Based Software Systems* (ICCBSS), Orlando (USA), LNCS 2255, 2002.
4. L. Chung and K. Cooper. A Knowledge-based COTS-aware Requirements Engineering Approach. *In Proceedings of the 14th International Conference on Software Engineering and Knowledge Engineering* (SEKE), 2003.
5. X. Franch, J.P. Carvallo. Using Quality Models in Software Package Selection. *IEEE Software*, 20(1), 2003.

6. X. Franch, N. Maiden. Modelling Component Dependencies to Inform their Selection. In *Proceedings of the 2nd International Conference on COTS-Based Software Systems* (ICCBSS), Ottawa (Canada), LNCS 2580, 2003.
7. ISO/IEC Standard 9126-1: *Software Engineering – Product Quality – Part 1: Quality Model*, 2001.
8. ISO International Standard 8402: *Quality management and quality assurance-Vocabulary*, 1986.
9. A. Lamsweerde. Goal-Oriented Requirements Engineering: A Guided Tour. *Invited mini-tutorial paper 5th IEEE International Symposium on Requirements Engineering*. 2001.
10. http://www.microsoft.com/exchange/techinfo/planning/2000/mmb2desc.asp
11. R. Keeney and H. Raiffa. Decision with Multiple Objectives: Preferences and Value Tradeoffs. Wiley, New York, 1993.
12. N. Maiden, C. Ncube. Acquiring Requirements for COTS Selection. *IEEE Software* 15(2), 1998.
13. J. Yen, W. Tiao. A Systematic Tradeoff Analysis for Conflicting Imprecise Requirements. *IEEE 4th International Conference on Requirements Engineering*. 1997.
14. K. Wallnau, S. Hissam, R. Seacord. *Building systems from commercial components*. Addison-Wesley Longman Publishing, 2002.