International Journal of Software Engineering
and Knowledge Engineering
Vol. 13, No. 5 (2003) 1–21

World Scientific
www.worldscientific.com

INVESTIGATING CONFLICTS IN COTS DECISION-MAKING

CARINA ALVES* and ANTHONY FINKELSTEIN†

Department of Computer Science, University College London,
London WC1E 6BT, UK
**c.alves@cs.ucl.ac.uk*
†*a.finkelstein @cs.ucl.ac.uk*

The development of COTS-based systems largely depends on the success of the selection process. This activity encompasses the evaluation of COTS packages against customer requirements, so that it is necessary to perform a complex decision-making process to select the most suitable package. We argue that analysing the matching between features and requirements is a core task of COTS decision-making. In particular, it is very likely that mismatches occur, as COTS are designed to meet very general requirements rather than specific needs. In this paper, we investigate the different types of conflicts that can arise from these mismatches. We propose a goal driven framework to deal with such conflicts. We demonstrate the approach with the mail server case study.

Keywords: COTS selection; conflict management; goal-based requirements engineering.

1. Introduction

The prospect of reducing time and cost associated with software development has led organizations to an increasing interest in acquiring and integrating commercial products instead of developing systems from scratch. The wide availability of both generic and domain-specific COTS (Commercial-Off-The Shelf) packages is a key-driving factor to support the development of COTS-based systems. The selection of suitable COTS products is often a non-trivial task that requires a careful consideration of multiple criteria [15]. In practice, most selection decisions are based on subjective judgments, such as current partnerships, commercial profits, and successful vendor marketing. Moreover, organizations usually operate in a very rigid development schedule, on which their competitiveness depends. Selection is a time consuming activity, where a considerable amount of time is necessary to search and carefully screen potential COTS candidates.

It is widely accepted that COTS procurement must be an interleaved process with requirements specification [9][15][11]. In particular, requirements statements need to be much more flexible than in the traditional requirements engineering (RE) process. Another major difference between traditional RE and COTS-based RE is that the latter does not need to be complete. Instead, initial incomplete

requirements can be progressively refined and detailed as soon as product features are assessed. An interesting approach is to let available COTS features influence requirements [3]. Consequently, it is necessary to achieve the best balancing of requirements precision and flexibility.

The use of COTS involves many challenges and risks. For instance, organizations have very limited access to the product's internal design and the typical description of commercial packages is an incomplete and confused textual description. On the other hand, when buying COTS products, users can take advantage of buying a product that has been tested many times by other users with consequent improvement in software quality. An additional complication is that vendors have full control over the product releases and upgrades. Therefore, users are put into unexpected situations over which they have no control. In fact, COTS packages usually include more functionalities than the customer really needs. Therefore, these extra capabilities impose constraints and limitations over requirements creating different circumstances for the requirements engineer who now has to deal with undesired features. Thus, this new situation leads to a continuous process of requirements negotiation.

Because of the uncertain nature of COTS capabilities, the evaluation of how customer requirements can be met by products is too complex to be performed. Analysing the matching between features and requirements is a way to identify possible inconsistencies between what is wanted and what is possible to meet. In other words, the selection process encompasses the balancing of conflicting interests between customers and vendors. Current COTS selection methods fail to support the effective comparison of features against requirements. In particular, the evaluation process demands some form of inexact matching between these specifications, where a range of conflicts can arise from these mismatches. It is also necessary to engage in an extensive process of requirements negotiation in which the requirements of the organization have to be balanced against the capabilities of the package. Our work aims at developing a better understanding of how these conflicts should be identified and managed in order to support COTS decision-making.

In a previous paper [8], we have discussed the main challenges and problems that may arise in COTS decision-making. The contribution of the present paper is the development of a goal-oriented approach to manage conflicts in COTS-based development. The motivation to follow a goal-based approach is that high-level goals represent requirements as strategic needs, giving the right abstraction level to evaluate the different alternatives that operationalise such goals. Note that, in our approach COTS features are considered ways to operationalise goals. Moreover, high-level goals are very stable as they represent the real needs of stakeholders; as far as goals are decomposed, different refinements are proposed which complies with the necessity to have flexible requirements stated before. Rather than eliminating conflicts, what is practically impossible given the lack of control over the COTS behavior, we aim at proposing a course grain framework that handles conflicts in order to achieve the best balancing between COTS constraints and customer goals.

This paper is organized as follows. In Sec. 2, we review the related work done in the areas of COTS selection, multi-criteria decision-making and conflict resolution. In Sec. 3, we describe the main characteristics of the COTS decision-making process. Section 4 presents the mail server case study. Section 5 describes how the specification and refinement of goals are performed in the context of COTS-based systems. Section 6 describes our proposal to support the matching process and analyses the types of mismatches that can arise between goals and features. In Sec. 7, we classify the conflicts and propose the conflict management framework. Section 8 concludes the paper and presents some future work.

2. Related Work

The selection process is a critical activity of COTS-based development when the quality and suitability of the product have to be verified with respect to organizational requirements and business expectations. We describe the main approaches to support the selection of COTS products including a discussion about how the requirements process is covered in each method.

The *OTSO* (*Off-The-Shelf Option*) method [15] was one of the first COTS selection methods proposed in the literature. It is a well-defined process that covers the whole selection process. The definition of hierarchical evaluation criteria is the core task of this method, which consists of the hierarchical set of functionalities, architectural constraints, and organizational characteristics. The evaluation activity identifies four different subprocesses: search criteria, definition of the baseline, detailed evaluation criteria definition, weighting of criteria. Surprisingly the method considers quality aspects (e.g. reliability, portability) as extra factors that may influence the decision but are not necessarily included in the evaluation criteria. However, we consider such characteristics fundamental aspects to assess COTS products, for example, if you are integrating several components into a single system, you should verify how well they interoperate before buying them.

OTSO also supports the cost estimation of each alternative using cost models. The approach breaks down the acquisition cost into three main classes: acquisition costs, further development costs and integration costs. Although having proved to be successful in building the evaluation criteria, this approach has limitations on how to conduct the requirements acquisition process. The method assumes that requirements already exist since it is based on a fixed and pre-defined requirements specification that will be part of the evaluation criteria. Moreover, OTSO just mentions the possibility to have unrequired features but do not provide any strategy on how to deal with them. Note that such situations are very common and need to be properly examined as they can determine the success of COTS-based systems. The method relies on the use of AHP (*Analytic Hierarchy Process*) technique [25] for conducting the evaluation of products in order to support the decision-making. Although suffering from some weaknesses as those stated above, the OTSO method served as an initial step for further approaches.

Another important contribution for COTS selection is the *PORE* (*Procurement-Oriented Requirements Engineering*) [11]. The method is a template-based approach to support selection that is based on the iterative process of requirements acquisition and product evaluation. At the beginning of the process there are few requirements specified and a large number of candidate products. Using the templates several times, it is possible to refine the product list until the most suitable product is selected. In particular, the templates are derived from empirical studies about current processes and problems encountered during the selection activity. PORE integrates various techniques, methods and tools, such as knowledge engineering techniques, multi-criteria decision-making methods, and requirements acquisition techniques. It also provides guidelines for designing product evaluation test cases which guide the evaluation team to acquire product information to select or reject products. The method suggests the use of fit criteria to determine whether or not a solution satisfies the original requirements. According to the method, the compliance between features and requirements is a fundamental step for effective product selection. However, the compliance process is not examined in sufficient detail. For instance, it is not clear how the comparison between features and requirements is performed and how products that do not satisfy the requirements are eliminated from the candidate list.

CRE (*COTS-based Requirements Engineering*) [10] is a method that highlights the importance of non-functional requirements as decisive criteria to evaluate alternative products. In particular, quality attributes are very difficult to be verified mainly because suppliers do not provide a complete description of quality aspects. Furthermore, non-functional requirements can often be interacting, such that attempts to achieve one requirement can hurt or help the achievement of another. The method proposes the use of the NFR Framework [18] to model non-functional requirements. CRE supports the evaluation of candidate products through the definition of systematic criteria that includes a clear description of quality attributes that potential candidates have to meet. Moreover, the method provides guidelines on how to acquire and specify non-functional requirements. CRE emphasizes that evaluating and analysing all relevant quality features of COTS candidates takes a great amount of time, typically more than the organization has. Therefore, it is both necessary and cost-effective to select the most promising candidates for detailed evaluation. As a drawback, the method does not address quality testing issues and it is not clear how the product's quality issues are verified with regard to non-functional requirements. Another problem with the method is the lack of support in cases when non-functional requirements are not properly satisfied.

CARE (*COTS-Aware Requirements Engineering*) [19] is a goal- and agent-oriented requirements engineering approach that explicitly supports the use of off-the-shelf products. CARE points out the importance to keep requirements flexible since they have to be constrained by the capabilities of available COTS. In this approach requirements are classified as: native (requirements acquired from cus-

tomers) and foreign (requirements of the COTS components). The method considers that bridging the gap between the sets of native and foreign requirements is a critical task. Although the approach points out the importance of mapping system requirements and products specification, it does not provide or suggest any systematic solution to support the possible mismatching between both specifications. CARE is based on the i* notation [13] to describe the process methodology. The process model ontology includes actors, goals, resources, dependencies and relationships. The authors developed a prototype called CARE assistant tool as a first attempt to model the interdependencies between native and foreign requirements.

It is interesting to note that most selection methods assume the existence of a predefined and fixed set of requirements (i.e. the evaluation criteria) in which the candidate products must be assessed against it. We believe that following such approaches developers take advantage of having a well-defined and systematic process. However, it also implies the definition of strict requirements, which means that either promising candidate products have to be eliminated because they do not meet the stated requirements or that large product modifications will be needed in order to satisfy such restrictive requirements. Moreover, many methods propose the prioritization of requirements in order to conduct the COTS evaluation but do not properly tackle the complex matching between requirements and COTS features. We consider the matching process a critical issue where features should be mutually balanced against requirements in a very interactive way.

It is practically impossible to conduct a comprehensive discussion about COTS evaluation without mentioning multi-criteria decision-making techniques since they have been widely used in many selection methods. The basic concepts of MCDM (*Multi Criteria Decision-Making*) approaches are establishing a set of criteria that products should meet, assigning scores to each criterion based on its relative importance in the decision and then ranking products based on their total scores. Figure 1 depicts an overview of the selection decision-making, where a three-levels hierarchy represents: at the first level, the main goal for the decision-making process; the evaluation criteria at the second level; and finally at the third level the alternative candidates. The two most used approaches are the AHP (*Analytic Hierarchy Process*) [25] and WSM (*Weighted Scoring Method*) [15]. Using the WSM technique, the overall score of each alternative is calculated using the following formula:

$$\text{Score} = \max \sum_{j=1}^{n} (\text{weight}_j * \text{score}_{ij}) \qquad \text{for } i = 1, 2, \ldots, m$$

where score_{ij} is the value of the ith alternativein terms of the jth criterion and weight_j is the weight of the jth criterion.

Weights are assigned by stakeholders and represent the importance given to a particular requirement to be met. The score represents the compliance of the attributes to be evaluated. WSM can be confusing since the resulting scores only represent the relative ranking of alternatives and the differences in their value do not give any indication of their relative superiority. For more complex decision-making

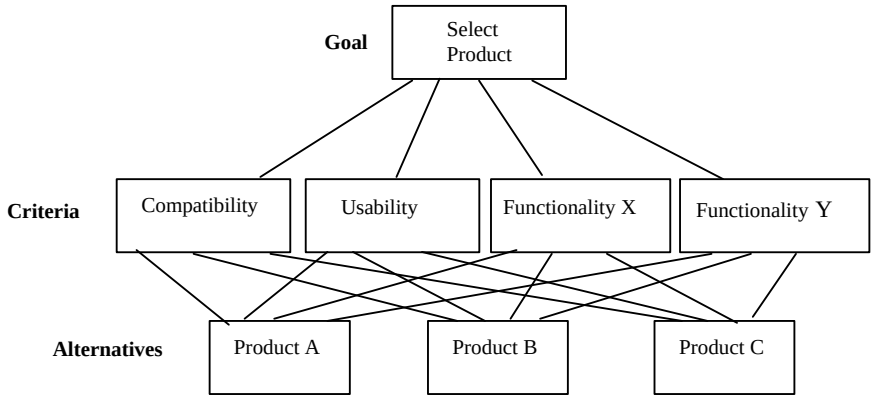

Fig. 1. Overview of decision-making process.

processes, a more effective technique is the AHP. This technique provides a hierarchical approach for consolidating information about alternatives using pair-wise comparisons and the overall priorities are computed using an eigenvalue technique of matrix comparison. The technique suggests that comparing criteria in pairs brings more reliable results.

WSM technique has some limitations when applied in COTS assessment, for instance: (i) this approach produces real numbers as results so they can be misinterpreted as the real differences between the alternatives rather than the relative ranking; (ii) difficulty in assigning weights when the number of criteria is large. Ryan [16] has performed some industrial case studies on requirements prioritization using AHP technique. However, as pointed out by Sivzattian [23], the main limitations of AHP technique for requirements prioritization are: (i) they assume that requirements are independent; (ii) the calculation model involves a very high number of pair-wise comparisons. These techniques are also weak in supporting comparison between interdependent requirements. Note that interactions between requirements are very usual, in particular non-functional requirements are well known as interacting both in conflict or synergy.

Another relevant topic is the research on conflict resolution. Robinson has an extensive work in this area [26, 27]. He identifies the notion of requirement restructuring by modifying requirements and generating alternatives that remove stakeholder conflicts. In [27] a domain independent restructuring approach called CORA (*Conflict-Oriented Requirements Analysis*) is proposed. This approach is a paradigm of requirements development that incorporates requirement ontology and strategies for restructuring requirements transformations as a mean to reduce or remove conflicts.

Similarly, Win Win [2] is a groupware support system for determining software requirements as negotiated win conditions. A key principle of the approach is that your project will succeed "if and only if you make winners of all the critical

stakeholders". The main purpose of the Win-Win negotiation model is to provide a stepwise approach for stakeholders to use in reconciling their individual win conditions. The model includes tailorable domain taxonomy of common requirements conflicts that assist in conflict resolution generation. In [22], Easterbrook reviews the research on conflict from a social perspective, representing common beliefs and assumptions from various disciplines that study the conflict problem. The Viewpoints framework [4] has been proposed as a way of managing inconsistent and incomplete information gathered from multiple perspectives. The approach handles the descriptions provided by each stakeholder and supports the identification and resolution of conflicts among them. Another work in inconsistency management, Lamsweerde [5] proposes a formal framework to tackle various types of inconsistency that can arise among stakeholders requirements. Special attention is given to divergence conflicts, which arise when a particular combination of circumstances can cause requirements conflicts. A systematic approach is proposed for identifying and resolving divergences using various heuristics and formal techniques. A key principle of the approach is to manage conflicts at the goal level so that developers can have a suitable abstraction level to handle conflicts.

3. COTS Decision-Making

The decision-making supported by most selection methods is based on a quantitative approach. It compares all features against the selection criteria. Note that when the number of criteria to be compared is large, the entire process may become impractical. Suppose that AHP is used, the number of all possible comparisons is equal to $n(n-1)/2$, where n is the number of attributes to be compared. Suppose that the selection criteria have 50 requirements to be compared (i.e., $n = 50$), the decision-maker would have to make 1225 pairwise comparisons; which can be considered impractical. Even though, some methods have been proposed in order to reduce the number of pairwise comparisons [24], an even more problematic issue is that individuals cannot simultaneously compare such a large number of elements and recognize the real differences among the alternatives.

We argue that the COTS decision-making should focus on a more qualitative analysis. The motivation for that is due to the following issues:

- It can be difficult to measure COTS attributes as some aspects are uncertain by the time products are being evaluated, for example the level of interoperability between the package and other system might only be known after the product has been integrated.
- The satisfaction of features is not always associated with a single requirement because it is unreliable to assign, for example, a conformance number on how the security attribute is met by the product (i.e. measuring how users accept the security level provided by the product). Note that security is a very broad attribute and therefore has different ways to be achieved.
- The assignment of criteria scores is not always straightforward, mainly because

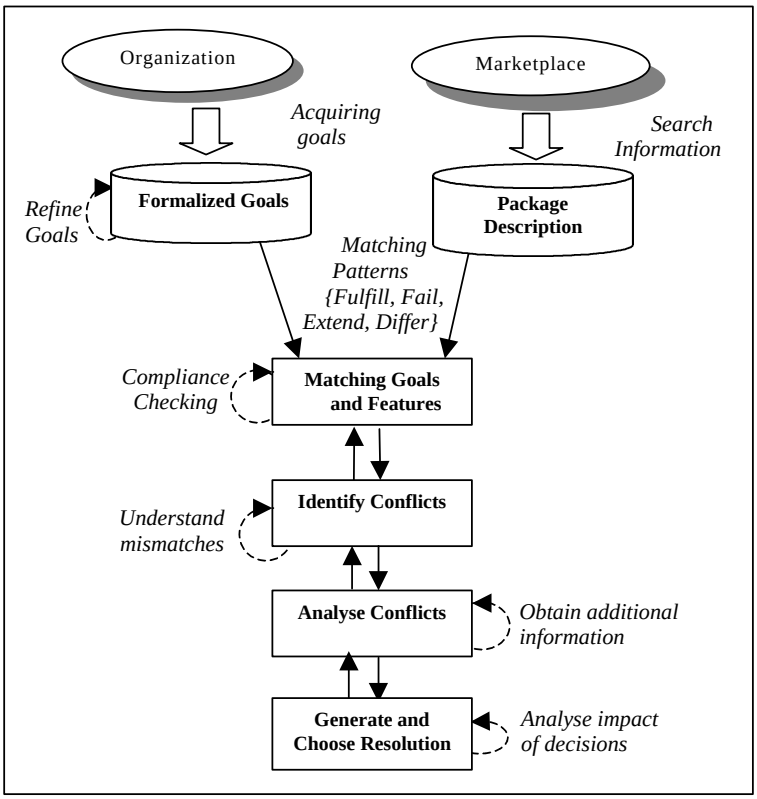

Fig. 2. Overview of conflict management process.

requirements and features are not identically specified. Moreover, it requires a deep understanding about the domain to "translate" different vocabularies.

- Features can satisfy requirements in different ways, not only with different compliance levels but also extending or even modifying the selection criteria. In fact, there are simultaneous interactions between requirements and features.

In order to tackle the difficulties described, we propose an approach to evaluate COTS capabilities in terms of how well they match customer requirements; assuming that mismatches are very likely to arise, potential conflicts have to be identified and managed. The conflict management framework proposed herein supports well justified decisions based on resolution proposals and risk evaluation. Figure 2 gives an overview of the conflict management framework; in the following sections we describe each process in detail.

4. Case Study

The selection of a mail server [28] is used to illustrate our proposal. Mail servers are the core of the communication and coordination infrastructure of organizations. A

successful mail service deployment depends on its right selection and our framework can be employed to assist this process. Mail servers are a good case study not only for their strategic importance, but also because of their own nature. Mail servers provide a lot of functionalities and exhibit a variety of quality features that can be hard to analyze. 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.

5. Goal Driven Specification

As our approach relies on goal driven requirements engineering, it is necessary to clarify some concepts about goals. Goals have been recognized as a leading concept in the RE process [1], [18]. According to Lamsweerde [6], "*A goal is an objective which the system under consideration should achieve. They may be formulated at different levels of abstraction, ranging from high-level, strategic concerns to low-level, technical concerns.*" The modelling of goals has many benefits, such as to represent goals explicitly, to identify interdependencies among goals, to support qualitative reasoning.

Instead of evaluating products based on detailed and fixed selection criteria like those proposed by other selection methods [15, 11], in our approach we follow the strategy to treat features as alternatives to operationalise goals. In this way features might influence the refinement of goals which leads to a very interactive and collaborative process. The reasons for this are due to the fact that COTS features will hardly meet very specific goals. Therefore, the goal specification process should be done in parallel with the evaluation of products. The modeling of goals starts with the elicitation of high-level goals that represent the strategic concerns from stakeholders. Each goal can be refined into satisfying subgoals represented by a graph structure inspired by the AND/OR trees used in problem solving. Links AND refine a goal into a set of subgoals; which means that only if all subgoals are met then the overall goal is achieved. Links OR refine a goal into an alternative set of refinements which means that satisfying one of the subgoals is sufficient to satisfy the parent goal. More formally, goal refinement can be defined as follow:

Definition 1 (Goal Refinement) Let $G = \{g_1, \ldots, g_n\}$ be a finite set of goals and $SG = \{g_{i,1}, \ldots, g_{i,n}\}$ the set of subgoals of goal g_i with $i \in N^+$.

Definition 2 (AND Refinement) Let $g_{i,1}, \ldots, g_{i,n}$ be a set of subgoals of g_i ***if*** $[(g_{i,1})_{AND}(g_{i,2})_{AND} \ldots_{AND} (g_{i,n})] :=$ *satisfied* ***then*** g_i is *satisfied*.

Definition 3 (OR Refinement) Let $g_{i,1}, \ldots, g_{i,n}$ be a set of subgoals of g_i ***if*** $\exists\ [(g_{i,1})_{OR}(g_{i,2})_{OR} \ldots_{OR} (g_{i,n})] :=$ *satisfied* ***then*** g_i is *satisfied*.

The refinement process continues until the abstraction level is sufficient to map possible features that can satisfy the refined subgoal. As shown in Table 1, we have refined the goal "*g2 Ensure that messages must never get lost*" into three subgoals through an AND link "*g2.1 Ensure that messages must never get lost* if mailbox runs

Table 1. Goal specification for the mail server case study.

Goal description	Subgoals
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 must never get lost	g2.1 Messages must never get lost if mailbox run 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 rate should be less than 5 minutes per megabyte
g4 Support collaborative work	g4.1 Integrated document management g4.2 Instant messaging g4.3 Voice and video conferencing
g5 Support commonly communication protocols	g5.1 Support POP 3 g5.2 Support IMAP 4 g5.3 Support HTTP g5.4 Support SMTP
g6 Support for web access	g6.1 Support Webmail g6.2 Support Web browser
g7 Ensure data security	g7.1 Authentication of users g7.2 Data integrity
g8 Support automatic subscription to mail lists	None Subgoal
g9 Support protection against external attacks	g9.1 Provide anti-spam filters g9.2 Provide anti-virus scanning
g10 Support middleware	g10.1 Support DCOM g10.2 Support CORBA g10.3 Support RMI
g11 Verify the maturity of the product in the market	g11.1 Version Stability g11.2 Vendor reputation
g12 Installation and administration facilities	g12.1 Adm tools and wizards g12.2 Documentation available

out of space", "*g2.2 Ensure that messages must never get lost if a failure happens*" and "*g2.3 Ensure that messages must never get lost if they cannot be delivered*". At this point, decision-makers have a general understanding of the stakeholders goals and can start looking for products that might support these goals. After screening the products, some features (i.e. alternative operationalizations) are identified.

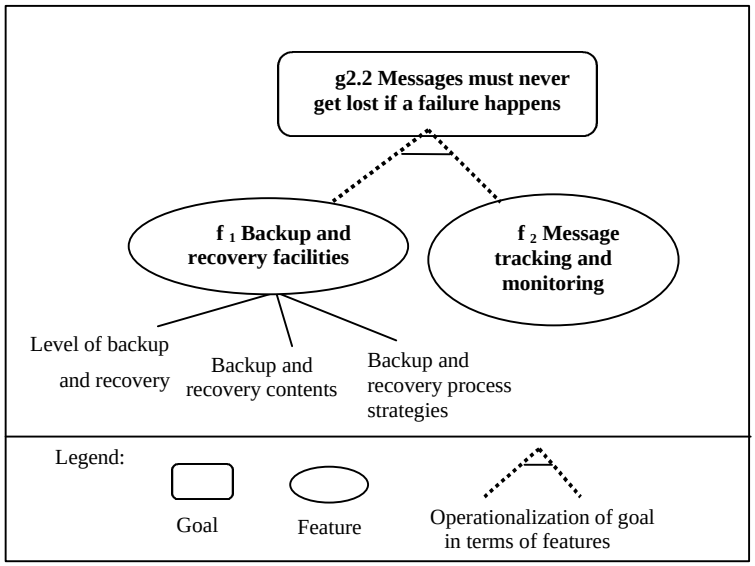

Fig. 3. Alternatives to operationalise the goal "Message must never get lost if a failure happens".

Definition 4 (COTS Feature) Let $F^{\rho} = \{f_1, \ldots, f_m\}$ be a set of features implemented by COTS product ρ.

Consider, for example, that after examining product A we observed that goal g2.2 could be achieved by means of the features "*f1 Backup and recovery facilities*" and "*f2 Message tracking and monitoring*" (see Fig. 3). The next step consists of asking the stakeholders if these attributes are sufficient to ensure that goal 2.2 is achieved at an acceptable level. To deal with this important aspect of decision-making in a systematic way, we propose the *acceptability* attribute to evaluate each operationalization alternative in which stakeholders accept or reject it.

Definition 5 (Acceptability) An operationalization alternative A consists of a set of features F^{ρ} that satisfy a subgoal $g_{i,j}$ such that stakeholders can evaluate the acceptability of the alternative $A(_{\text{acceptability}}) == \langle accept,\ reject \rangle$.

The evaluation of some subgoals in terms of operationalizable features can be very time-consuming. For example, the verification of the goal "*g3.1 the average response time should not exceed 1 minute"* demands the analysis of parameters such as platform, number of users, average message size. In the goal specification, these parameters are called dependencies that constraint the achievement of goals. In order to measure the response time of message delivery, detailed test cases must be performed which can be a complex task. As a way to tackle this problem, we assign a desirability level to goals, which indicates the stakeholders' preferences to achieve a particular goal. For instance, the decision-maker can judge if it is worth or not, spending a considerable effort evaluating low priority goals that in turn give little return to the overall satisfaction of the system. When assigning

the desirability of goals, stakeholders should have in mind that even very critical goals might not be achieved; therefore, stakeholders have to be prepared to perform extensive negotiation of goals. Table 2 shows how the desirability levels are assigned and briefly explains their meaning.

Definition 6 (Desirability) Given a goal g_i, we define a desirability value that is assigned to g_i and represents the stakeholders' preferences for g_i to be satisfied such that $g_i(_{\text{desirability}}) == \langle$*very high, high, medium, low, very low*$\rangle$.

During the specification process, as far as the goals are refined and we get a better understanding about the domain, it is very likely that interactions between non-related goals might be found. These interactions can be either positive or negative, for example, goal "*g2 Messages must never get lost*" has a negative impact over "*g3 Ensure fast message delivery*", and similarly "*g9 Protection against external attack*" interacts positively with "*g7 Ensure security*". These interactions are very important for the decision process, because when we make choices in order to achieve a particular goal, it is also necessary to verify how it can affect other goals.

Table 2. Goal desirability assignment.

Desirability	Explanation
5 — Very High	Very critical goal that must be fulfilled otherwise the success of the system will be strongly compromised
4 — High	Critical goal that must be fulfilled otherwise the success of the system will be compromised
3 — Medium	Important goal that should be fulfilled in order to ensure that significant functionalities will be satisfied
2 — Low	Desirable goal that could be interesting to have but that does not affect the success of the system
1 — Very Low	Slightly desirable goal that does not affect the success of the system

Definition 7 (Interaction) Given two goals g_i and g_j, we shall say that $[g_i(+)g_j]$ if there exist a positive interaction between g_i and g_j. Similarly, we shall say that $[g_i(\text{-})g_j]$ if there exist a negative interaction between g_i and g_j.

We show an example of goal description as follows:

Goal g3 Fast message delivery
Subgoals g3.1 The average response time should not exceed 1 minute
g3.2 Messages throughput should be less than 5 MB/min
Desirability High
Interactions g2 (-) g3
Dependencies platform, number of users, average message size

From our experience, when gathering information about product features, we may use various sources including: package description, demo sessions, third-party

evaluators, specialized magazines, etc. It is very likely that information about products is not always reliable, as we have encountered biased descriptions that favor a particular product. Therefore, in order to perform evaluation with an acceptable degree of reliability, we propose the use of the *trustability* value to be associated with the information acquired.

Definition 8 (Trustability) Given a feature f_i, we define a trustability value associated with f_i representing the reliability of the information source that describes f_i such that $f_i(_{\text{trustability}}) == \langle 5$ — *very trusted, 4* — *trusted, 3* — *medium trusted, 2* — *low trusted, 1* — *not trusted* $\rangle$.

In situations where we have to decide between two alternatives of operationalization, we can judge if it is worthwhile to take the risk of relying on information about features that have a low trustability level. For example, decision-makers have to judge whether it is appropriate to accept the promised benefits of the administrative support (related to goal $g12.1$) available with the mail server package or acquiring a specific tool developed for this purpose. Note, however, that trustability of information is assigned by evaluators who have expertise in the domain; therefore, the trustability level corresponds to their own judgments.

6. Analyzing the Matching Process

In our approach, the matching process consists of evaluating how well goals are satisfied by features. In other words, it is a kind of conformance-checking mechanism, as stakeholders have to reach an agreement on how their goals are sufficiently achieved in terms of features. This follows the ideas of goal-oriented RE which emphasizes that goals have degrees of satisfaction instead of a one-to-one basis (i.e. satisfied or not). In our case, goals can be satisfied in many different ways by COTS capabilities. We have specified a set of matching patterns to help decision-makers to classify the matching in a more formal way.

Fulfill — In this situation there is a mapping between goal and feature. The matching is complete as the feature fully satisfies the requested goal. A special case of fulfill pattern is called *conditional fulfill*, it happens when the product requires the acquisition of an extra software in order to totally satisfy the goal. Here there is a *constraint* that has to be accomplished in order to fully fulfill the initial goal. For example, apparently Product A supports goal "$g4.2$ *Instant Messaging*". However, after a careful analysis on the documentation, we verify that capability imposes the constraint of requiring the conferencing server version in order to provide the promised functionality.

Differ — This situation is the most complex and probably the most common case. In this pattern, there is a partial mapping between goals and features. However, the feature does not fully satisfy the goal. In particular, three mismatch cases can occur. In the first case, the feature partially satisfies all the constraints imposed by the goal. For instance, Product B provides documentation that apparently meets

goal "*g12.2 Documentation available*" but after examining the installation manuals, we verified that it does not cover the topic "*migration from other mail servers*"; as for the stakeholders it is considered an important subgoal. The second case happens when the feature meets some aspects of the goal but the available information is not sufficient to ensure that it completely meets (i.e. fulfill) the goal. In this case the resolution should be to obtain more information about this functionality. The last case corresponds to a situation where there is a semantic inconsistency when comparing goals against features, therefore it is uncertain whether the goal is satisfied or not. For example, goal "*g9.1 Provide anti-spam filters*" refers to the support of filters to handling junk mail. All products provide filters for incoming messages, which partially has the same syntax as the requested goal but the meaning is that filters are based on criteria defined by the user and the products do not automatically detect spam from DNS blacklist sites that is a capability needed to handle junk mail.

Extend — This situation occurs when products provide features that are not requested by stakeholders, more precisely, there are extra features not mapped in the goal specification. The *Extend* pattern can lead to the following interaction cases:

Hurtful — when an extra feature has a negative impact over a particular goal;
Helpful — when an extra feature is wanted and it can be included in the goal specification as part of the feedback mechanism;
Neutral — when an extra feature does not interfere with the achievement of any goal nor the stakeholders want it.

Fail — In this situation stakeholders require a goal that cannot be satisfied by the evaluated products. More precisely, a *fail* matching happens when there is no way to operationalise a specified goal.

Unknown — Available information is insufficient to classify the matching. Further clarification is necessary to verify whether the goal is satisfied or not.

For every matched goal, a satisfiability level is computed based on the degree goals are satisfied by features. Note that one or more features can satisfy a single goal as well as one feature can operationalise many goals. For example, Product A functionality "*f5 Allow users to fully configure maximum delivery retries*" supports the achievement of both goals "*g1.1 Configure number of delivery retry*" and "*g2.3 Messages must never get lost if they cannot be delivered*". The correspondent matching pattern for both goals is fulfilled as the product allows full configuration of delivery retries. The satisfiability level is defined in the following way:

Definition 9 (Satisfiability) Let $g_{i,1},\ldots,\ g_{i,n}$ be a set of subgoals and $F^{\rho}=\{f_1,\ldots,\ f_n\}$ be a set of features implemented by COTS product ρ. We shall say that a feature f_k can satisfy a subgoal $g_{i,j}$ in different degrees within the interval [0..1]. More specifically:

iff $f_{kFulfill}g_{i,j}$ ***then*** Satisfiability:: =1
iff $f_{kDiffer}g_{i,j}$ ***then*** Satisfiability:: = [0.1..0.9]

Table 3. Satisfiability level of each matching pattern.

Matching pattern	Satisfiability level
Fulfill	1
Differ	0.9 ... 0.1
Fail	0
Extend	No value

iff $f_{kFail}g_{i,j}$ ***then*** Satisfiability:: = 0
iff $f_{kExtend}g_{i,j}$ ***then*** Satisfiability:: = no value

For $\langle fulfill \rangle$ and $\langle fail \rangle$ situations, the assignment of satisfiability value is straightforward, whereas the $\langle differ \rangle$ cases admit a range of values where the higher the value, the higher the goal satisfaction. If the matching is $\langle extend \rangle$, no satisfiability can be assigned as the provided feature is not mapped to any specified goal. In these situations, potential conflicts can arise; however it is necessary to have a detailed examination of each case to determine if there is a real conflict. Table 4 shows the matching between two Mail Server products and the stakeholders goals. Note that matching patterns are associated to subgoals rather than to goals, as goals usually describe too abstract needs; in this way we assume that goals are hardly measured in terms of operationalization alternatives provided by products. The $\langle unknown \rangle$ is an interesting situation that occurs when extra information is needed to classify the matching, for example, the satisfaction of subgoal "*g*3.1 *Messages throughput rate should be less than 5 minutes per megabyte*" cannot be evaluated at the present time without first considering some dependencies like platform, number of users, and average message size. In fact, depending on other factors such as the desirability to achieve the parent goal, that undetermined subgoal can be properly evaluated in a later stage, making the matching a very interactive process.

As illustrated in Sec. 4, goals are refined into subgoals representing a more specific description of how to achieve a higher-level goal. In this way, we can say that subgoals contribute to the satisfaction of the parent goal. In order to measure the relative contribution each subgoal gives for the satisfaction of the parent goal, we propose the use of normalized weights associated to each subgoal.

Definition 10 (Contribution) Let $g_{i,1},\ldots,\ g_{i,n}$ be a set of subgoals of parent goal g_i. We shall say that each subgoal $g_{i,j}$ has a contribution value $\text{cont}_j$to achieve g_i so that:

$$\sum_{j=1}^{n} g_{ij}(\text{cont}_j) = 1 \text{ where } n ::= n^{\circ} \text{ subgoals}$$

Table 4. Matching patterns associating goals and features.

	Matching between goals and features	
Goal description	Product A	Product B
g1 Ensure and communicate message delivery		
g1.1 Configure number of delivery retries	*Fulfill*	*Differ*
g1.2 Configure time between retries	*Fulfill*	*Differ*
g1.3 Provide message delivery notification	*Fulfill*	*Fulfill*
g2 Ensure that messages must never get lost		
g2.1 Messages must never get lost if they cannot be stored	*Fulfill*	*Fulfill*
g2.2 Messages must never get lost if a failure happens	*Fulfill*	*Unknown*
g2.3 Messages must never get lost if they cannot be delivered	*Fulfill*	*Fulfill*
g3 Ensure fast message delivering		
g3.1 The average response time should not exceed 1 minute	*Unknown*	*Unknown*
g3.2 Message throughput rate should be less than 5 minutes per megabyte	*Unknown*	*Unknown*
...	...	...
g9 Support protection against external attacks		
g9.1 Provide anti-spam filters	*Differ*	*Differ*
g9.2 Provide anti-virus scanning	*Fail*	*Fail*

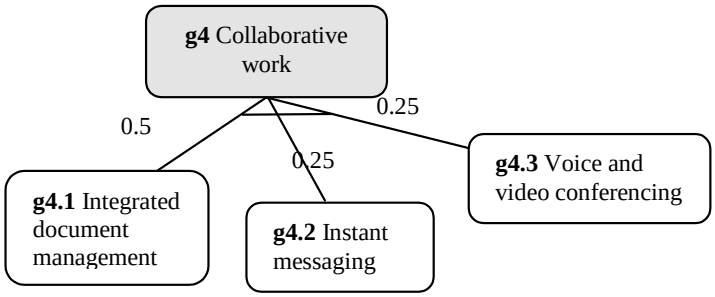

Fig. 4. Subgoals contribution to achieve parent goal.

In Fig. 4, the goal "*g*4 *Collaborative work*" is refined into three subgoals. The subgoal "*g4.1 integrated document management*" contribution is 0.5, whereas "*g*4.2 *instant messaging*" and "*g*4.3 *voice and video conferencing*" have the equal contribution of 0.25. Note that the decision to operationalising *g*4.1 gives higher contribution to achieve *g*4 compared to the other two subgoals. Consider the following situation to see how the contribution value supports the decision process. Suppose that the achievement of subgoals *g*4.2 and *g*4.3 impose the acquisition of conferencing server version for both product alternatives resulting in extra costs.

On the other hand, subgoal *g*4.1 can be easily satisfied by the product version

being evaluated. In this case, $g4.1$ should be preferred as it gives a higher contribution to achieve the parent goal without any constraints attached. Suppose now that $g4.3$ has the highest contribution value, here decision-makers have to perform negotiations to deal with the conflict. Moreover, it is necessary to judge which are the real benefits of operationalising $g4.3$ in order to achieve the parent goal as well as assessing the associated risks and costs of acquiring a more expensive and complex package version.

7. Dealing with Conflicts

In the previous section, we have illustrated with some examples that a conflict can arise when different alternatives can be chosen to satisfy stakeholders goals, so that evaluators have to investigate the risks and trade-offs associated with each alternative. These alternatives are generated from the combination of interacting and interdependent issues. The satisfaction of goals depends on the following factors: desirability to achieve the goal; subgoals contribution; attached constraints; interactions with other goal and subgoals; satisfiability of features. A conflict arises due to association of disputing factors. For instance, a typical conflicting situation happens when a highly desirable goal is affected by negative interactions and is under hard constraints. The conflict management aims at finding solutions that are sufficiently good, even if they are not optimal [18].

7.1. *Categories of conflicts*

In particular, the following kinds of conflict may be generated:

Intra conflicts — conflicts originated between goals when there is a negative interaction between them. For example, goal "$g2.2$ *messages must never get lost if a failure happens*" has a negative impact on "*g3 fast message delivery*" as backup and recovery strategies, which are possible operationalizations of $g2.2$ (see Fig. 3), induce extra overhead delaying the time of message delivery. In this case, decision-makers have to deal with questions like "Which goal should be satisfied?" or "Which goal brings more benefits?" in order to solve intra conflicts.

Inter conflicts — conflicts found between goals and product features. This type of conflict occurs in cases of mismatch, in other words, when there is one of the patterns $\langle conditional\_fulfill \rangle$ $\langle differ \rangle$ $\langle fail \rangle$ or $\langle extend \rangle$. In particular, the $\langle unknown \rangle$ situation does not lead to mismatches as it represents the lack of information to classify the matching. As example of inter conflict consider the mismatch shown in Table 5 where "*g9.2 Provide anti-virus scanning"* is not satisfied (i.e. matching $\langle fail \rangle$) by any of the evaluated packages. Let us assume that the desirability of goal "*g9 Support protection against external attacks*" is very high and the contribution of $g9.2$ to its parent goal is 0.8, which means that operationalising g9.2 represents a very important contribution to the overall satisfaction of the system. However, as that subgoal cannot be achieved by any package, a conflict arises. On the other hand, a mismatch between goals and features does not necessarily

Table 5. Conflict management framework.

Situation	Matching Pattern	Example	Rationale	Potential Resolutions	Risks
Product can fully satisfy the goal, but it is necessary to accept some constraints	Fulfill Conditional	g4.2 Instant messaging	This feature imposes the constraint of requiring the conferencing server version	Investigate if the constraints will hurt any goal and also verify the goal desirability	Acquiring that specific version might increase costs
Goal satisfaction is difficult to test	Differ	g 3.1 The average response time should not exceed 1 minute	Testing the feature demands the analysis of parameters such as platform, number of users, average message size	— If desirability is high and the test cases easy to perform then test the feature. — If desirability is high and test cases are difficult to perform then try to discard the goal, otherwise if users do not accept, run the tests.	Testing might demand extra time and resources
Feature partially meets all the constraints of the wanted goal	Differ	g12.2 Documentation available	The installation manuals do not cover the topic "*migration from other mail servers*" that is a very desirable attribute	— Investigate if it is possible to obtain that information from other sources — Try to negotiate the goal and accept packages limitations	The mail server installation might be compromised
Feature partially meets the wanted goal but it is necessary extra information to ensure whether the matching is complete	Differ	g1.2 Configure time between retries	At the moment it is not possible to ensure that Product B allows fully configuration of time between retries	— Examine the feature in more detail and verify the trustability of information. — Analyse if the goal satisfiability is acceptable	If the feature trustability is low it might compromise the reliability of the evaluation
Feature apparently meets the goals but there is a semantic inconsistency between goal and feature	Differ	g9.1 Provide anti-spam filters	Both products provide filters for incoming messages, which partially meets the requested goal but the filters are based on criteria defined by the user, it does not automatically detect spam from DNS blacklist	First examine if the feature satisfies the goal in an acceptable level: — If not acceptable then try to relax the goal or if stakeholders do not accept then analyse the feasibility of acquiring a specific tool	Acquiring extra tools might cause interoperability problems as well as increase cost and integration time
Feature extend the goal specification so that it has a negative impact over a particular goal	Extend	f20 Presence information, enables one user to see whether another user is currently logged	This feature may hurt some privacy concerns of stakeholders	— Investigate the impact of the conflict, if it is high and not acceptable then develop wrappers to hide the unwanted features	Developing wrappers might be complex and time demanding
Goal cannot be satisfied by the product	Fail	g9.2 Provide anti-virus scanning	None product has anti-virus scanning	— Try to discard the goal, if not possible, analyse the feasibility of acquiring a specific tool	Acquiring extra tools might cause interoperability problems as well as increase cost and integration time

originate a conflict as some potentially conflicting situations can be accepted without major consequences. For example, when a feature ⟨*extend* ⟩ the specified goals, stakeholders might find that unforeseen functionality helpful.

7.2. *Conflict management*

The conflict management involves the following processes: understanding the nature of conflicts; analysing the causes of conflicts and other involved factors; and finally, exploring potential resolutions that might be a compromise among all these concerns. The attributes we have presented in the previous sections are at the core of the decision-making process as they support well-justified resolutions. Our approach proposes the strategy of minimizing the number of conflicts so that decision-makers should explore alternatives to reduce potential conflicting situations. In particular, stakeholders should weak goals or accept some restrictions imposed by the packages. However, the risk of these strategies has to be carefully considered, as they should not compromise the overall satisfaction of the system.

When managing conflicts, we aim at achieving a balancing among all involved factors that originated the conflict. In Table 5, our conflict management framework is described. Note that the resolution process largely depends on the stakeholders' judgment whether the proposals are acceptable or not.

8. Conclusion

COTS selection is a complex task that requires careful decision-making among candidate products where the most suitable one might be selected, in other words the product that best meets users' needs. Note, however, that when acquiring COTS, users have to balance their requirements in order to accept product limitations. We have proposed a set of matching patterns to support the evaluation of how features match requirements. We have demonstrated that conflicts may arise when there is a mismatch between requirements and features.

This paper presented a goal-oriented approach to manage conflicts in COTS-based development. In order to tackle the conflict problem we showed that it is necessary to understand the nature of conflict, analyse the causes of conflict, explore potential resolutions and examine the associated risks of resolution alternatives. We argue that conflict management should start as early as possible. However, in some situations conflicts may remain even after the package has been acquired. When to apply one or another resolution may depend on the specific package domain, on the kind of conflict, and on the seriousness of the consequences of the conflict.

We argue that requirements engineering process is in the center of COTS selection. Note that in the traditional RE, goals are elicited and then decomposed into a concise and clear description of the system to be implemented. After the system has been developed, analysts can trace it backwards to the goals and ensure that goals are met. In COTS-based requirements engineering, the process is similar with the difference that features are alternatives to operationalise goals.

As future work, we plan to enrich the negotiation heuristics, investigate in more detail the proposed resolutions and evaluate the risks associated with each proposal. Moreover, we aim at examining the types of conflict that may arise when multiple packages are selected and integrated into a composed system.

Acknowledgements

We would like to give special thanks to Xavier Franch and Juan Carvallo for the valuable discussions about the mail server case study. This work is partially supported by CAPES grant — Brazil.

References

1. A. Anton, W. Cracken and C. Potts, "Goal decomposition and scenario analysis in business process reengineering", in *Proc. CAISE'94*, Utrecht, Netherlands, 1994.
2. A. Egyed and B. Boehm, "Comparing software system requirements negotiation patterns", *Systems Engineering Journal* **6**(1) (1999) 1–14.
3. A. Finkelstein, G. Spanoudakis and M. Ryan, "Software package requirements and procurement", in *8th Int. Workshop on Software Specification and Design*, Paderborn, Germany, 1996.
4. A. Finkelstein, D. Gabbay, A. Hunter, J. Kramer, and B. Nusheibeh, "Inconsistency handling in multi-perspective specifications", *IEEE Trans. on Software Engineering* **20**(8) (1994).
5. A. Lamsweerde, R. Dairmont, and E. Letier, "Managing conflicts in goal-driven requirements engineering", *IEEE Trans. on Software Engineering* **24**(11) (1998).
6. A. Lamsweerde, "Goal-oriented requirements engineering: A guided tour", invited mini-tutorial paper for *5th IEEE Int. Symp. on Requirements Engineering*, Toronto, Canada, 2001.
7. B. Nuseibeh and S. Easterbrook, "Requirements engineering: A Roadmap", *The Future of Software Engineering*, ACM Press, 2000.
8. C. Alves and A. Finkelstein, "Challenges in COTS decision-making: A goal-driven requirements engineering perspective", in *Workshop on Software Engineering Decision Support*,in conjunction with SEKE'02, Ischia, Italy, 2002.
9. C. Alves and A. Finkelstein, "Negotiating requirements for COTS selection", in *8th Int. Workshop on Requirements Engineering: Foundation for Software Quality*, in conjunction with RE'02, Essen, Germany, 2002.
10. C. Alves and J. Castro, "CRE: A systematic method for COTS selection", in *XV Brazilian Symp. on Software Engineering*, Curitiba, Brazil, 2001.
11. C. Ncube and N. Maiden, "PORE: Procurement-oriented requirements engineering method for the component-based systems engineering development paradigm", in *Int. Workshop on Component-Based Software Engineering*, Los Angeles, USA, 1999.
12. D. Carney, "Requirements and COTS-based systems: A thorny question indeed", *SEI Interactive*, Carnegie Mellon University, 1999.
13. E. Yu, P. Du Bois, E. Dubois, and J. Mylopoulos, "From organization models to system requirements: a 'cooperating agents' approach", in M. Papazoglou and G. Schlageter (eds)., *Cooperative Information Systems*, Academic Press, 1998, pp. 293–312.
14. G. Kotonia, I. Sommerville, *Requirements Engineering: Processes and Techniques*, John Wiley & Sons, 1998.

15. J. Kontio, "A COTS selection method and experiences of its use", in *Proc. 20th Annual Software Engineering Workshop*, 1995.
16. K. Ryan and J. Karlsson, "Prioritizing software requirements in an industrial setting", in *19th Int. Conf. on Software Engineering*, Boston, Massachusetts, 1997.
17. K. Wallnau, S. Hissam, and R. Seacord, "Building systems from commercial components", *SEI Series in Software Engineering*, Addison Wesley, 2002.
18. L. Chung, B. Nixon, E. Yu, and J. Mylopoulos, *Non-Functional Requirements in Software Engineering*, Kluwer Academic Publisher, 2000.
19. L. Chung and K. A. Cooper, "Knowledge-based COTS-aware requirements engineering approach", in *14th Int. Conf. on Software Engineering and Knowledge Engineering*, Ischia, Italy, 2002.
20. L. Karlsson, A. Dahlstedt, J. Dag, B. Regnell, and A. Persson, "Challenges in market-driven requirements engineering — An industrial interview study", in *8th Int. Workshop on Requirements Engineering: Foundation for Software Quality* in conjunction with RE'02, Essen, Germany, 2002.
21. P. Zave, "Classification of research efforts in requirements engineering", in *2nd Int. Symp. on Requirements Engineering*, York, England, 1995.
22. S. Easterbrook, E. Beck, J. Goodlet, L. Plowman, M. Sharples and C. Wood, "A survey of empirical studies of conflict", in S. M. Easterbrook (ed.), *CSCW: Cooperation or Conflict?*, Springer-Verlag, London, 1993.
23. S. Sivzattian and B. Nuseibeh, "Linking the selection of requirements to market value: A portfolio-based approach", in *7th Int. Workshop on Requirements Engineering: Foundation for Software Quality*, Interlaken, Switzerland, 2001.
24. T. Evangelos, "Multi-Criteria Decision-Making Methods: A Comparative Study", Kluwer Academic Publishers, 2000.
25. T. Saaty, *The Analytic Hierarchy Process*, McGraw-Hill, New York, 1990.
26. W. Robinson, "Negotiation behaviour during requirements specification", in *12th Conf. on Software Engineering*, New Orleans, USA, 1990.
27. W. Robinson and S. Volkov, "Conflict-Oriented Requirements Restructuring", GSU CIS Working Paper 99-5, Georgia State University, 1999.
28. X. Franch and J. Carvalo, "Defining a quality model for mail servers", in *2nd Int. Conf. on Component-Based Software Systems*, Ottawa, Canada, 2002.