Analyzing Component Based System Specification

Return to Published Papers

Sajjad Mahmood and Richard Lai*
Department of Computer Science and Computer Engineering,
La Trobe University, Melbourne, Victoria 3086, Australia.
Email: - lai@cs.latrobe.edu.au

Abstract

Component Based System (CBS) development is integration centric with a focus on selecting components that match stakeholder requirements. Recent research suggests that CBS development needs a quantitative analysis model that evaluates candidate components and facilitates informed decisions during CBS development. To date, there has been little work done on building CBS assessment models that can help quantify components features against stakeholder requirements, and thus, select suitable components. This paper describes a CBS requirements analysis model that can be used to quantify component alternatives and select suitable components based on their satisfaction and risk measures. The framework also performs a relevance analysis for candidate components in a conflicting scenario and uses resolution selection rules to negotiate mutually acceptable agreement during component selection.

1. Introduction

CBS requirements analysis and component selection is widely recognized as an interrelated process which plays a central role in overall CBS development. Software literature [1-4] show that CBS success depends on the ability to select suitable components. An inappropriate component selection can lead to adverse affects such as short listing components that barely fulfill the needed functionality or they introduce extra costs in integration and maintenance phases [3]. Individual components usually provide fix capabilities that might not satisfy all system requirements and some of them may be unnecessary in a given system. This reduces the chance of a match between a component and stakeholder requirements. Therefore, it is difficult to find a supplier who can meet all stakeholder requirements [4].

Fundamental to CBS success is the need for a collaborative process in which both stakeholders and candidate components balance the conflicting interests between what is needed and what is available [2, 5]. This collaborative process needs to focus on how to share knowledge between stakeholders and components and facilitate negotiation of individual interests during component selection. Component selection involves a continuous process of requirements negotiation and for an effective requirements negotiation, it is necessary to analyze the impact of change [6]. This involves looking at balancing a component's satisfaction against the involved risks. Risk assessment is another important attribute affecting overall CBS development, as it provides a basis for comparing candidate components by focusing on their risk profiles.

In this paper, we present a quantitative requirements analysis framework for a CBS. The first step is to characterize requirements by eliciting stakeholder demands and rank them based on their priority, view and matching potential. Further, our framework combines a top-down and trade-off approaches to analyze requirements and select suitable components.

2. Approach Overview

CBS development is a complex and risk prone process which needs a flexible requirements analysis technique that provides an opportunity to both stakeholders and component vendors to reach a mutually acceptable agreement. We believe that a CBS requirements model needs to address three key issues: understand stakeholder requirements and component features; quantify alternatives based on satisfaction and risk analysis; and balance a component's satisfaction against the involved risk during conflict resolution. By analogy with Mendonca et al. measurement framework

[7], we propose a CBS requirements analysis approach to address these issues, as shown in Figure 1.

We divide our approach into three main phases, namely, requirements characterization, top-down analysis and trade-off analysis. The first phase - requirements characterization - starts with a process to elicit stakeholder requirements. We propose the use of a goal oriented requirements engineering process to specify stakeholder demands. After this process, requirements are ranked according to their priority, view and matching potential. The second phase - top-down analysis - presents a metrics hierarchy to quantify requirements matching with component features. It uses satisfaction and risk metrics to select suitable components for each requirement. The third phase - trade-off analysis - is executed to perform a relevance assessment for candidate components in a conflicting scenario and we propose a set of resolution selection rules for a trade-off between requirements and components.

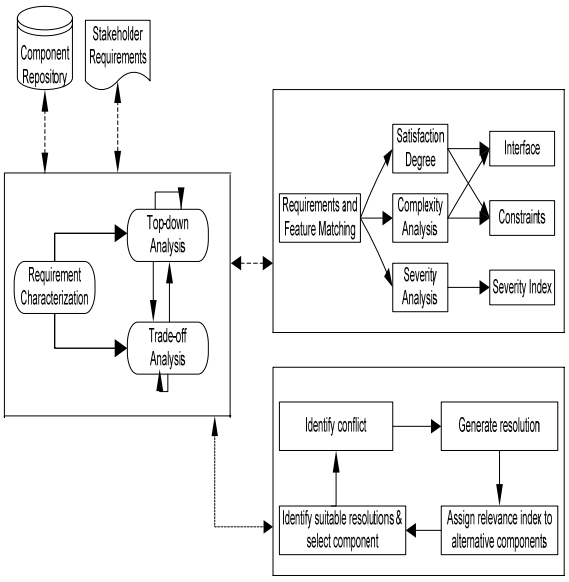

Figure 1 CBS requirements analysis approach

3. Requirements Characterization

Requirements characterization is used to identify and rank stakeholder requirements and how they relate to each other. We discuss requirements characterization in details as follows.

3.1 Requirements Elicitation

CBS requirements need to start with a less specific and more flexible definitions [2]. Flexible requirements definition increases the probability of finding matching components because rigid requirements either exclude the use of components or require a large modification as they are less stringent. We propose to elicit stakeholder demands using goal - scenario coupling approach [8]. In this approach, a goal discovery and a scenario authoring are complementary activities. After goal discovery, scenario authoring is initiated, followed by goal discovery. The "goal-discovery, scenario-authoring" sequence is repeated to incrementally elicit requirements [8].

Goal-scenario approach [8] is used to elicit CBS requirements because it allows requirements to be represented at different abstraction levels. This provides a systematic process of refining high-level requirements into objectively measurable sub-requirements. The aim of these abstraction levels is to identify different component alternatives which satisfy stakeholder's requirements. We propose to organize CBS requirements, as shown in Figure 2, into two hierarchy levels, namely, high requirements level and aggregate requirements level. The high requirements level reflects an abstract set of requirements indicating overall business goals of an organization. The aggregate requirements level represents a set of services which can be used to achieve the high level requirements. These abstraction hierarchy levels are used to elicit requirements into concrete sub-requirements which can be objectively measured against the component features. Requirements are elicited into these two levels based on refinement rules defined in [8].

High Requirements level (HRL): - Aim of HRL is to identify an initial set of minimum CBS requirements which corresponds to a given business objective. This initial set of requirements represents a possible methodology for fulfilling overall the business goal of a CBS. This emphasis on HRL means the selection process relies less on pre-emptive decisions about the candidate component. HRL captures the requirements as a pair <Gh, Sh> where Gh is a high-level goal and Sh is a high-level scenario. A high-level goal in our approach describes each mandatory requirement which acts as a guide to component identification. Staring component selection by investigating high-level goals also assures the satisfaction of mandatory requirements. Similarly, high-level scenario represents the process for achieving a high-level goal.

Aggregate Requirements level (ARL): - At the ARL, the focus is on refining a high-level goal into sub-goals that can be used to quantify relevance of candidate components. High-level requirements are refined by considering the interaction between the

system and the users. These interactions represent a possible method of achieving a high-level goal defined at HRL. ARL specifies requirements as a pair <Ga, Sa> where Ga is an aggregate-level goal and Sa is an aggregate-level scenario. An aggregate level goal expresses the manner of realizing a high-level requirement. The associated aggregate scenario describes a flow of interactions between a system and its user to fulfill the aggregate goal.

Further, it is important to classify requirements to enable distinction between core and peripheral requirements. By analogy with Lee et al. [9] facet classification model, we classify requirements under three facets: priority, matching potential and view. This classification helps in performing an objective measure of requirements against component features.

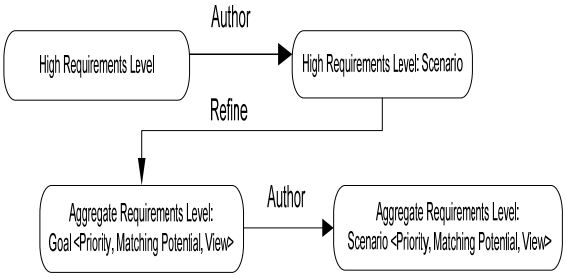

Figure 2 Requirements Elicitation Model

Priority

Facet priority represents stakeholder's desire for a ARL satisfaction. We propose to classify each ARL priority as mandatory, very important, important or optional. A mandatory ARL represents a minimum set of requirements that need to be satisfied for the success of the system. Very important ARL represents requirements that ensure significant functionality of the system. Important ARL represents requirements that ensure sufficient significant functionality of the system. Similarly, optional ARL represents desirable requirements that do not affect the success of the system.

Matching Potential

Facet matching potential represents ARL prospective to find a matching component in the repository. We define matching potential as the percentage of the number of candidate components with the potential to match the ARL to the total number of components in a repository. Low value of matching potential indicates a high number of constraints associated with the ARL realization. High value of matching potential indicates a low number of constraints associated with the ARL realization.

View

Facet view classifies a ARL as either actor specific or system specific. Actor specific ARL are the objectives of users who interact with the proposed system. System specific ARL are the objectives of entities that do not interact directly with the system but will have a stake hold in the system requirements. System ARL provide a mechanism for expressing organizational goals, project concerns and regulator requirements that apply to system as a whole.

3.2 Requirements Ranking

In the second step, each ARL goal is ranked according to its priority, view and matching potential. We rank ARL goals in a descending order of their priority. Requirements with the same priority are classified based on their view. Actor specific requirements get precedence on system level requirements. Further, requirements with the same priority and view are further classified based on their matching potential values. Requirements with lower 'matching potential' are ranked higher than requirements with low 'matching potential' because we initially aim to select a component for a requirement that has limited choices rather than those with a higher number of choices. We argue that by selecting components for requirements with higher choices after component selection for requirements with lower choice, we are given a better chance of providing alternatives if a conflict occurs.

4. Top-Down Analysis

The top-down approach uses matching criteria between component features and requirements to calculate a component's satisfaction degree and associated risk. The high-level requirements are refined into sub requirements and analyzed against candidate component features to obtain satisfaction metrics. We use severity analysis and complexity metrics to derive component risk assessment. We choose a suitable component from a range of candidate components for each requirement by defining a decision metrics denoted by D (r). Decision metrics value is defined as a percentage of component satisfaction degree and associated risk, as shown in equation 1.

$$D(r) = (\text{Satisfaction}(C) / \text{Risk}(C)) \times 100 \qquad (1)$$

where *r* is the requirement under consideration. We discuss satisfaction degree and risk metrics in detail as follows.

4.1 Satisfaction Degree

A key challenge in CBS requirements engineering is to reconcile stakeholder's demands with available component capabilities. The component matching involves the evaluation of a component satisfaction degree to a requirement. The first step is to identify component attributes that contribute to the satisfaction assessment. A component is characterized according to its characteristics and context dependencies [10]. Fundamental to a component is its characteristics which specify the functionality provided. These characteristics define only the individual elements of a component, mainly in syntactic terms. However, a component is subject to configuration dependencies on its use. These dependencies are both on individual elements as well as on the relationship among the elements [11]. Thus, it is essential for a component user to understand the constraints, to enable its proper use.

Table 1 Suitability criteria

Criterion	Measurement Scale
Compatible features – CF	{1,2, …n} where n is the number of compatible features.
Missing features – MF	{1, 2,…n} where n is the number of the ARL scenarios not met by a component.
Impact of missing features – IMF	{negligible, little, moderate, considerable, excessive}
Added features – AF	{1,2,… n} where n is the number of added functionality provided by components and not required by the requirement.
Impact of added functionally – IAF	{negligible, little, moderate, considerable, excessive}
Adaptation effort required – AER	{negligible, little, moderate, considerable, excessive}
Adhere to system architecture – ASA	{poor, fair, good, very good, excellent}
Non development cost – NDC	{within budget, < 5 %, 5 – 7 %, 7 – 10 %, > 10 %}

Recently, Cechich et al. [12] defined a measure for early detection of component functional suitability as a function of number of compatible functionality, missing functionality and added functionality. We believe that in addition to these three attributes, it is important to consider the impact of missing and added features introduced by candidate components. Further, configuration constraint is another primary attribute which directly affects the satisfaction degree of a component with reference to a requirement. After a detailed analysis, we identify a set of attributes, as shown in Table 1, which need to be considered during satisfaction degree measurement.

We commence the satisfaction measure by considering requirements at ARL level. The satisfaction of a requirement is realized at ARL scenario level and each ARL scenario is objectively measured against component features. Candidate component features are identified from their interface specification and information provided in information brochures, evaluation downloads, user documentation, tutorials and manuals. Table 1 shows the check list of selection criteria that is used to the measure satisfaction degree of a component with respect to a requirement (ARL goal).

We classify our check list into two main groups: component characteristics and configuration constraints. The first group aims at measuring suitability of a component from its characteristic point of view. It consists of five selection criteria: Compatible Features (CF), Missing Features (MF), Impact of Missing Features (IMF), Added Features (AF) and Impact of Added Features (IAF). CF measures the number of component features that contribute in fulfilling the requirement. Table 2 shows matching metrics where ARL scenarios are listed in rows and component features are arranged in columns. A '✓' at the intersection of an ARL scenario and a component feature indicates that the ARL scenario is satisfied by the corresponding feature. Similarly, MF and AF are the number of ARL scenarios not fulfilled by a component and the number of added features introduced by the component, respectively. We introduce the notion of IMF, which quantifies the impact of the missing features as either 'negligible, little, moderate, considerable or excessive'. Similarly, IAF quantifies the impact of extra features introduced by the component.

Table 2 Matching Metrics

Component A	ARL Goal 1: Sc 11	Sc 12	Sc 13	Sc 14
Feature A1	✓			
Feature A2		✓		✓
….				
Feature An			✓	

The second group evaluates the compatibility of a component from a context dependency point of view. First, we consider Adaptation Effort Required (AER), which quantifies the amount of adaptation development required to use the component. It is important to consider adaptation cost as, although it usually accounts for less than half the total CBS development effort, the effort per line of adaptation averages is three times the effort per line of traditional development code [13]. Adhere to System Architecture

(ASA) analyses the compatibility of a component regarding its operating system for development, architecture of product and pre-requires. Non Development Cost (NDC) assesses the cost associated with licensing administration [13].

We define a utility function uf1, based on multi-attribute utility theory, to classify IMF, IAF and AER. The function uf1 is transformed onto the unit interval {-5, 5} such that, uf1 (excessive) = -5, uf1 (considerable) = -2, uf1 (moderate) = 1, uf1 (little) = 3 and uf1 (negligible) = 5. Similarly, we define utility function uf2 to classify ASA. The function uf2 is transformed onto the unit interval {-5, 5} such that, uf2 (poor) = -5, uf2 (fair) = -2, uf2 (good) = 1, uf2 (very good) = 3 and uf2 (excellent) = 5. Finally we define utility function uf3 based on US Department of Defense (DOD) risk management guide [14] to classify NDC. The function uf3 is transformed onto the unit interval {-5, 5} such that, uf3 (within in budget) = 5, uf3 (< 5 %) = 3, uf3 (5 – 7 %) = 1, uf3 (7 – 10 %) = -2 and uf3(> 10 %) = -5.

We define characteristics measure for a component c, denoted as CM (c), as

$$CM(c) = CF + (MF \times IMF) + (AF \times IAF) \quad (2)$$

We define configuration constraints measure for a component c, denoted as CCM (c), as

$$CCM(c) = AER + ASA + NDC \quad (3)$$

Finally, satisfaction degree of a component c, denoted by SD (c), is defined as

$$SD(c) = CM(c) + CCM(c) \quad (4)$$

4.2 Risk Assessment

We propose to use risk assessment to quantify the degree of uncertainty associated with selection of a component during a CBS development. NASA Safety Manual NPG 8715.3 [15] defines risk as a combination of probability of malfunctioning (failure) and the consequence of malfunctioning (severity). Probability of a failure depends on the probability of occurrence of a fault combined with the likelihood of exercising that fault in a scenario in which a failure will be triggered [16]. Since it is difficult to find exact estimates for the probability of failure of a component in the early phases of CBS development, we propose to use complexity to estimate fault proneness of a component. Component complexity is chosen as a quantitative factor because it has a proven impact on fault proneness [16, 17]. Further, we perform severity analysis to quantify the impact of the probable failure. Thus, we define risk for a component c as

$$Risk(c) = Complexity(c) \times Severity\_Index(c) \quad (5)$$

We perform a complexity measure for a component specification with a focus on identifying component attributes that affect its complexity at both component and intra-component levels. We identify that for a component specification, the overall complexity is attributed mainly to component interface, constraints and interaction. For a detail discussion on the complexity metrics, please refer to our previous work [10].

In addition to the estimate of the fault proneness of each component based on a component's complexity, we need to consider the severity of the consequences of potential failures [16]. For example, a component may have low complexity, but its failure may lead to catastrophic consequences. Therefore, our risk assessment takes into consideration the severity associated with each component, based on how its failure affects the requirement satisfaction. We identify that Volatility of Component (VC) and Supplier Creditability (SC) that help to estimate consequence of malfunction of a component. We use VC and SC to estimate the severity of the component, based on the experience recorded by other users of the component, and stored in component repositories.

Table 3 VC Severity Classification

Volatility of component - VC	Severity Index – SI (VC)
VC <= 1.22 versions per year	1
1.22 < VC <= 1.38 versions per year	2
1.38 < VC <= 1.54	3
1.54 < VC < =1.70 versions per year	4
> 1.70 versions per year	5

We calculate VC of a component as the ratio of the number of component releases to the total number of years from the first release. For example, 'Rapid Spell'[1] component has five releases since its launch from 2003. Therefore, its VC value is 1.25 (five divided by four). We classify and calculate the severity index of VC by calculating confidence interval based on a sample size of 80 components available at component source repository. We calculate the confidence interval of VC with the confidence coefficient of 95%. Further, we assumed that the severity index of a component c with the VC (c) >= mean is positive. The results of the analysis indicated

[1] www.componentsource.com

that, for 95 % of components, their VC value was in the interval [1.22, 1.54]. The mean value of VC was 1.38 releases per year. Table 3 shows the severity index for VC based on our confidence interval analysis.

Table 4 SC Severity Classification

Supplier creditability – SC	Severity Index – SI (SC)
1	5
2	4
3	3
4	2
5	1

SC is the ranking given by the reviewers of the component. The rating reflects the creditability of the component. Component source repository provides a rating of components, based on the reviews, by rating components on the scale of one to five. A component with rating five indicates that the customers of the component are completely satisfied by its functionality, provided documentation and support. A component with rating one indicates that the customer is not satisfied with provided functionality. Table 4 shows the severity index for a component on a scale of one to five. We propose to assign a severity index (SC) of five to a component which has a rating of one. This indicates that the component is less credible and thus, will have a high degree of severity associated with its use. Finally, we define the overall severity index of a component c, denoted by Severity_Index (c) as

Severity_Index (c) = SI (VC) + SI (SC) (6)

5. Trade-off Analysis

Trade-off analyses aims at balancing the conflicting interests between stakeholder requirements and at negotiating resolutions during component selection. In this paper, we only consider interaction conflicts. Requirements are said to be in interaction conflict, if the satisfaction of one requirement may impair or cease the satisfaction of another requirement. The key feature of trade-off analysis in our approach is a shift from requirements driven conflict analysis to a collaborative process in which both requirements and component features balance the conflicting interests.

In CBS development, there is a need to find a balance between requirements and available component features. Requirement driven approaches alone are usually not sufficient as they do not support a collaborative process of negotiating individual interests of stakeholders against a component's features. We propose a trade-off analysis which attempts to find a mutually acceptable resolution to a conflicting situation by providing an opportunity to both stakeholders and candidate components to trade-off individual interests. The trade-off analysis consists of four stages ranging from conflict identification, potential resolution generation, measuring component's relevance to each potential resolution and selection of suitable resolution. In the first step, interaction conflict is identified among requirements. In the second step, we propose a set of rules to generate potential resolution to a conflict. In the third step, we calculate the relevance index for each component with reference to the potential resolutions. Finally, suitable resolution and components are selected using the relevance measure.

5.1 Conflict Identification

For a given requirement, trade-off analysis starts with investigating its relationships with other requirements. We adopt the concept of *Generic Relationship Question (GRQ)* [7] to represent the set of relationships to be investigated for identifying possible conflicts. We use the following template to define a GRQ.

How does 'Attribute Class X' relate to 'Attribute Class Y'? (7)

An attribute class defines a set of attributes grouped together according to certain features relevant to a requirement. 'Attribute class X' represents requirement X with priority i, whose relationship is being investigated to identify conflicts. We define 'Attribute class X' as a set of ARL scenarios for requirement X. Similarly, 'Attribute class Y' represents requirement Y with priority i + 1, with whom the relationship is being investigated. Since requirement Y has a higher priority than the requirement X, we assume that a component has been selected to satisfy it. Thus, we define 'Attribute Class Y' as a set of component features which has been selected to satisfy requirement Y.

After establishing the GRQ, we propose to start the conflict identification by doing a pair-wise comparison between 'Attribute Class X' and 'Attribute Class Y'. Table 5 shows a conflict identification metrics where all the attributes of 'Attribute Class X' (ARL scenarios of requirement X) are listed in rows, and all the attributes of 'Attribute Class Y' (features of components that satisfies requirement Y) are arranged in columns. A 'X'at the interaction of a component feature and ARL scenario indicates that ARL scenario has a negative interaction with the component feature. We propose to consider two requirements in conflict if there is at least one negative interaction between them.

Table 5 Conflict identification metrics

Requirement X	Requirement Y – Component A			
	F1	F2	F3	F4
Scenario 1		X		
Scenario 2		X	X	
….				
Scenario n				

5.2 Resolution Generation

The next step in trade-off analysis is to generate potential resolutions to a conflict. We adopt the concept of knowledge based negotiation [18] to propose a set of rules for resolution generation. Knowledge based negotiation proposes concepts of compensation and dissolution. For compensation, the strategy is to satisfy requirements outside the conflict to increase the overall desirability of the resolution by adding new issues to negotiations. For dissolution, the strategy is to allow conflicting designs to be part of the specification, but remove their negative interaction. Dissolution replaces conflicting items with potentially less contentious items. In our approach, we propose the following rules as a guideline to generate resolutions.

Rule1: For each conflict, introduce a new scenario for the requirement with lowest priority in the conflict, such that it applies a set of constraints which result in conflict resolution.
Rule2: Delete a scenario or group of scenarios from the requirement with lowest priority in the conflict such that it removes the conflict.
Rule3: Write an adaptation code for interoperability between selected components such that it masks the conflict.

5.3 Relevance Index

In order to effectively select a suitable resolution to a conflict, there is a need to analyze the impact of each resolution and quantify candidate components relevance with respect to each resolution. We introduce the notion of relevance index which provides a mechanism to understand and balance the satisfaction, risk and impact of the selecting each resolution. The relevance index of a component is defined as a function of a component satisfaction degree, associated risk and impact of each resolution. We calculate relevance index as follows.

- *Calculate satisfaction and risk metrics for the requirement.* We use the set of metrics defined in section 4 to measure candidate components satisfaction degree and associated risk with reference to the requirement. These measures indicate the potential value for each metrics for the case when there is no conflict between the requirement and its predecessor requirements. We calculate these values to help us in identifying the impact on component satisfaction and risk caused by the conflict. We propose to denote these values as SP and RP respectively.
- *Calculate component satisfaction degree and risk for each resolution.* Similarly, we use the set of metrics defined in section 4 to measure component satisfaction and associated risk with reference to each conflict resolution. These measures indicate the actual value for each metrics with reference to a conflict resolution. We propose to denote these values as SA and RA respectively.
- *Calculate impact of resolution for each metrics.* Whenever a conflict is detected and a set of resolutions are generated, it is necessary to take into account the impact of resolution on the candidate component's satisfaction. When a satisfaction metric value decreases against its potential value, we assign an impact factor to component's satisfaction degree to quantify the impact introduced by the conflict resolution. We define satisfaction impact factor Is as

  if (SA < 0)
  then $I_S = (SA - SP) / SA$
  else if (SA >= 0) (8)
  then $I_S = (SP - SA) / SA$
  else
  $I_S = 0$

- *Calculate relevance index (RIc) for each component.* Relevance percentage for a component in a resolution is defined as the ratio of component's satisfaction degree and risk; subtracted by the associated impact factor.

  $RIc = ((SA / RA) – Is) \times 100$ (9)

5.4 Suitable Resolution Selection

The last step of trade-off analysis is to identify suitable resolution and select relevant components. We propose to identify suitable resolutions and select relevant components by comparing the relevance index measures. However, it is important to note that using only the final value of relevance index cannot be employed as a sole criterion for selecting suitable resolutions and relevant components. For example, same relevance measure can be achieved for two components, but the individual attributes which contribute to the measure can be different from one another. Hence, we define the following rules as a

guide on selecting suitable resolutions and components.

Rule1: *Calculate suitability of each resolution.* Suitability of each resolution is defined as the sum of relevance indexes of all components for the resolution; divided by the total number of components.

$$\text{Suitability} = \sum_{i=0}^{C} \text{RIc} / C \qquad (10)$$

where C is the total number of components.
Rule 2: Identify the most suitable resolution by selecting the one with highest value of suitability.
Rule 3: Form the most suitable resolution, select the component with the highest value of relevance index.
Rule 4: If two or more than two components have same relevance index, select the component with highest value of satisfaction.
Rule 5: If two or more than two components have same value for relevance index and satisfaction, then select the component with lowest value of risk.

6. Conclusion and Future Work

In this paper, we have presented a new quantitative framework for a CBS to manage requirements analysis and suitable component selection. Our framework guides stakeholders through a collaborative process of requirements elicitation, matching and negotiation. For future work, we plan to empirically investigate the benefits associated with using our framework during a CBS development. Further, we plan to extend UML to incorporate CBS requirements analysis and component selection and thus provide a standard notation for all phases of a CBS development.

References

[1] N. A. Maiden and C. Ncube, "Acquiring COTS Software Selection Requirements," *IEEE Software*, vol. 15, pp. 46-56, 1998.

[2] C. Alves and A. Finkelstein, "Investigating Conflicts in COTS Decision-Making," *International Journal of Software Engineering and Knowledge Engineering*, vol. 13, pp. 1-21, 2003.

[3] K. R. P. H. Leung and H. K. N. Leung, "On the Efficiency of Domain based COTS Product Selection Method," *Information and Software Technology*, vol. 44, pp. 703 - 715, 2002.

[4] S. Lauesen, "COTS Tenders and Integration Requirements," In proceedings of 12th IEEE international requirements engineering conference, Kyoto, Japan, 2004, pp. 166 - 175.

[5] B. Boehm and C. Abts, "COTS Integration: Plug and Pray?," *IEEE Computer*, vol. 32, pp. 135 - 138, 1999.

[6] C. Alves and A. Finkelstein, "Requirements Negotiation for COTS-based Systems: Challenges and Open Issues," In proceedings of second international workshop on requirements and COTS components (RECOTS'04), Koyoto, Japan, 2004.

[7] M. G. Mendonca and V. R. Basili, "Validation of an Approach for Improving Existing Measurement Frameworks," *IEEE Transactions on Software Engineering*, vol. 26, pp. 484 - 499, 2000.

[8] C. Rolland, C. Souveyet, and C. B. Achour, "Guiding Goal Modeling Using Scenarios," *IEEE Transactions on Software Engineering*, vol. 24, pp. 1055 - 1071, 1998.

[9] J. Lee and N.-L. Xue, "Analyzing User Requirements by Use Cases: A Goal-Driven Approach," *IEEE Software*, vol. 16, pp. 92 - 101, 1999.

[10] S. Mahmood and R. Lai, "A Complexity Measure for UML Component System Specification," *Software-Practice and Experience*, John Wiley & Sons, In Press.

[11] J. Han, "A comprehensive interface definition framework for software components," In proceedings of 1998 Asia pacific software engineering conference, Taipei, Taiwan, 1998, pp. 110 - 117.

[12] A. Cechich and M. Piattini, "Early Detection of COTS Component Functional Suitability," *Information and Software Technology*, In Press.

[13] V. R. Basili and B. Boehm, "COTS-based Systems Top 10 List," *IEEE Computer*, vol. 34, pp. 91-93, 2001.

[14] "Risk Management Guide for DOD Acquisition," *Department of Defense*, 2003.

[15] "NASA Safety Manual NPG 8715.3," 2000.

[16] K. Goseva-Popstojanova, A. Hassan, A. Guedem, W. Abdelmoez, D. E. M. Nassar, H. Ammar, and A. Mili, "Architectural-level risk analysis using UML," *IEEE Transactions on Software Engineering*, vol. 29, pp. 946-960, 2003.

[17] Khaled El Emam, Walcelio Melo, and J. C. Machado, "The Prediction of Faulty Classes Using Object Oriented Design Metrics," *Journal of Systems and Software*, vol. 56, pp. 63 - 75, 2001.

[18] W. N. Robinson and S. Fickas, "Supporting Multi-Perspective Requirements Engineering," In proceedings of first international conference on requirement engineering, Silver Spring, MD, 1994, pp. 206 - 215.