RE-UML: An extension to UML for specifying Component-Based Software System

Sajjad Mahmood
Information and Computer Science Department
King Fahd University of Petroleum and Minerals
Dhahran 31261, Saudi Arabia
smahmood@kfupm.edu.sa

Richard Lai
Department of Computer Science
and Computer Engineering
La Trobe University, Melbourne
Victoria, 3086, Australia

Abstract

Unified Modeling Language (UML) is a de-facto industry standard for analysis and design modeling of software systems. However, it has been rarely used for specifying the process of balancing stakeholder requirements against candidate components which play a central role in Component Based System (CBS) development. One of the reasons is that UML lacks the features for supporting CBS requirements analysis. In this paper, we present an extension to UML, RE-UML (Requirements Engineering-UML), which supports CBS analysis and assessment process. RE-UML enables a system analyst to specify satisfaction and risk assessment for evaluating customer demands against component features.

1 Introduction

The ability to select suitable components that match customer requirements is fundamental to CBS success. Individual components are usually designed for general purposes that might not satisfy stakeholder requirements and some of them may be unnecessary in a given system. We have developed Requirements Analysis and Assessment Process (RAAP) framework [14], as shown in Figure 1, that guides stakeholders through a process of balancing user expectations against available components. First, stakeholder requirements are elicited and ranked based on their priority, view and matching potentials. Second, satisfaction and risk metrics are used to quantify requirements matching with component features. Finally, trade-off analysis is applied to identify and resolve the potential conflicts in requirements. For a detailed discussion, please refer to [14].

A domain expert's perceptions play a central role in processes such as requirements analysis and component selection [4]. This becomes more significant in CBS development because component information is usually available in an adhoc manner, ranging from natural language descriptions to sample evaluations prototypes. This creates ambiguity in the CBS requirements analysis process and the domain expert plays a certain role in how requirements are perceived and component information is analysed.

A standard notation, such as UML [16], tends to help decrease the ambiguity associated with software processes. UML is a de-facto industry standard for analysis and design modeling of software systems. However, it has been rarely used to specify the collaborative process of balancing stakeholder demands against candidate components. One of the main reason is the lack of support for specific CBS development phases; in particular for CBS requirements analysis and component selection. The ability to model the CBS requirements analysis in the UML also offers the potential for the tool support. There is therefore a need to develop a notation (for example, an extension to UML[16]) to represent the CBS requirements analysis and component selection processes.

UML allows defining profiles to extend standard notations in order to meet specific needs in different domains [3]. For example, standard UML specification and its extensions [9, 12, 6, 1] have been used to represent requirements analysis phases of a software system. Similarly, Cheesman et al. [5] use UML to specify CBS development and divide the whole process into three phases: component identification; component interaction and component specification. However, there is not much work done on modeling CBS requirements analysis; in particular there is little work done on developing a notation to quantify and assess component alternatives during CBS requirements analysis.

This paper describes an extension to UML, RE-UML (Requirements Engineering-UML), which supports each phase of RAAP. RE-UML enables a system analyst to specify satisfaction and risk assessment for evaluating customer requirements against component features. RE-UML ex-

1530-0803/09 $25.00 © 2009 IEEE
DOI 10.1109/ASWEC.2009.28

tends UML class diagrams to specify stakeholder requirements and component features; and UML sequence diagrams for matching criteria between stakeholder demands and component features. It introduces a set of associations to model the top-down and trade-off analyses. RE-UML also uses logic programming rules to specify formal semantics of proposed extensions to UML. The notation enables a system analyst to specify satisfaction and risk assessment which evaluates stakeholder requirements against component features.

2 Related Work

Several extensions [1, 18, 17, 2, 7] to UML specifications have been proposed in literature to model different application domains, ranging from mobile systems to data warehouse. Standard UML specification and its extensions have also been used to represent requirements analysis phases of a software system. UML use case diagrams based approaches [9, 13] have been successfully used in eliciting, analyzing and documenting functional requirements for a software system.

Lee et al. [13] have proposed a goal-driven use cases approach which extends UML use case diagrams with goals concepts. They use goals to derive use cases models and analyse interactions between requirements by investigating the relationship between goals and use cases. Further, it uses the concept of rigid and soft goals to handle non-functional requirements. Heaven et al. [6] have proposed a UML profile to model the KAOS goal-oriented requirements engineering approach [10, 11]. This extension introduces stereotyped classes and associations to model the KAOS semantic network graphically in UML. Further, it introduces a set of tags to represent the informal and formal descriptions associated with stereotyped classes and associations. However, these traditional requirements analysis techniques cannot be directly used for CBS requirement process because they do not support the collaborative process needed to match stakeholder requirements with candidate components.

Standard UML and its extensions have been used to represent different phases of CBS development life cycle. Recently, Hussein et al. [7] have proposed a UML profile to specify intrusion detection facilities during CBS development. It extends UML use case, class, package, state machine and component diagrams to specify intrusion scenarios. Cheesman et al. [5] use UML to specify CBS development and divide the whole process into three phases: component identification; component interaction and component specification. However, they do not discuss how to quantify and assess component alternative during CBS requirements analysis.

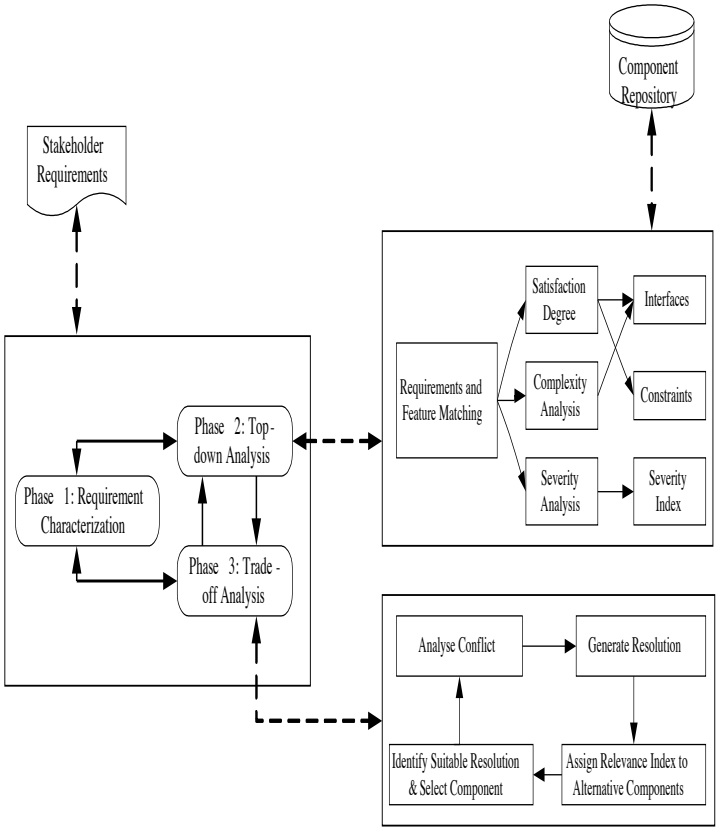

Figure 1. CBS Requirements Analysis Approach

3 The RAAP Framework and UML

We believe that in order to support CBS requirements analysis and component selection, the notation needs to have three key attributes: specify stakeholder requirements and component features; specify top-down analysis to evaluate stakeholder requirements against component features; and specify relationships between requirements and potential resolutions in case of conflicts. RAAP introduces Requirements Graph (RG) concept to represent the ranked requirements and the relationship between them. The node represents requirements and associated selected components; and the directed edge represent the association relationship between requirements. The RAAP framework is suitable for using UML as a notation because RAAP three phases have an analogy with the meta-model layer, model layer and user model layer of the UML. Furthermore, The RG concepts also has parallels with the UML analysis and association notations.

The UML is extended by introducing new stereotypes to the language [6]. These stereotypes are applied to existing

UML entities, such as classes and associations. RE-UML extends UML class diagrams to specify stakeholder requirements and component features; and UML sequence diagrams for matching criteria between stakeholder demands and component features. RE-UML also introduces a set of associations to model the top-down and trade-off analyses. RE-UML uses logic programming rules to specify formal semantics of proposed extensions to UML. In the following sections, we describe the formal syntax for RE-UML to facilitate each phase of RAAP.

4 RE-UML Requirements Characterization

Requirements characterization process is used to represent stakeholder requirements and the way they relate to each other. RE-UML supports requirements and component feature specification by introducing following notations to UML.

4.1 Rclass Diagram

Rclass, an extension of UML class diagram, specifies stakeholder requirements. Rclass is modeled as a rectangle with four sections, as shown in Figure 2. The top section represents the name of the class. The name is specified as a combination of unique stereotypes, the requirements abstraction level and a unique number. Rclass is identified with stereotype $<<$*requirements*$>>$. The abstraction level is defined as a compulsory tag which is associated with each Rclass and is defined as either high-level requirements or concrete-level requirements.

The goal section of Rclass represents the objective to be achieved by the Rclass. The third section of Rclass, scenario, represents the set of interactions performed to achieve the goal. Finally, the fourth section represents the ranking of the Rclass which is calculated using the rules defined in [14]. Formally, we define the requirements structural relationship as:

Predicate 1 *member* (X_r, X_i)

This binary predicate specifies that the interaction X_i belongs to the requirement X_r.

4.2 Cclass Diagram

Cclass, an extension of UML class diagram, specifies candidate components. Cclass is modeled as a rectangle with three sections, as shown in Figure 3. A Cclass is identified by a stereotype $<<$*component*$>>$ and the name of the component. Feature specifies the functionality provided by a component. These features define only the individual elements of a component, mainly in syntactic terms. However, a component is subject to configuration dependencies on its use. Thus, the third section in a Cclass symbol represents a component's dependencies on both individual elements as well as the relationship among the elements. This aids in understanding the constraints associated with its use. Formally, we define the component structural relationship as:

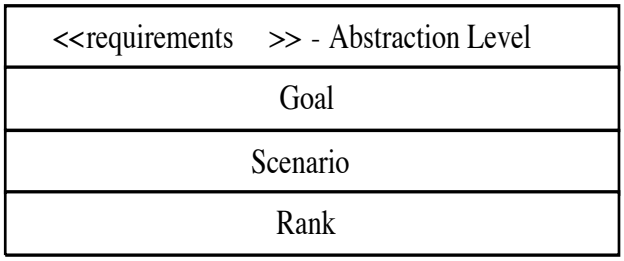

Figure 2. Rclass Notation

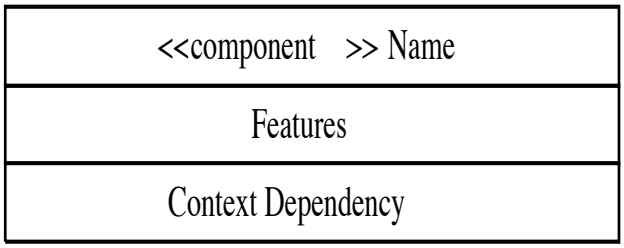

Figure 3. Cclass Notation

Predicate 2 *member* (X_c, X_cf)

This binary predicate specifies that the feature X_cf belongs to the component X_c.

4.3 Associations

Stakeholder requirements and components are often associated with, or relate to, other requirements and components. RE-UML models associations as an association class connecting two classes involved in the relationship. These association classes have a label which describe the association. Furthermore, a filled triangle at an end of a line indicates the direction of the association.

RE-UML classifies associations into two types: (1) interaction relationship and (2) mapping relationship. The interaction relationship defines the association between two Rclass, as shown in Figure 4. Two requirements that are in

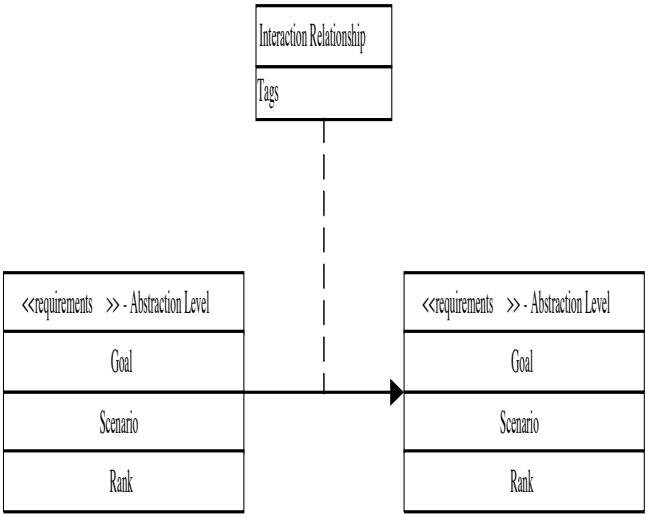

Figure 4. Interaction Relationship

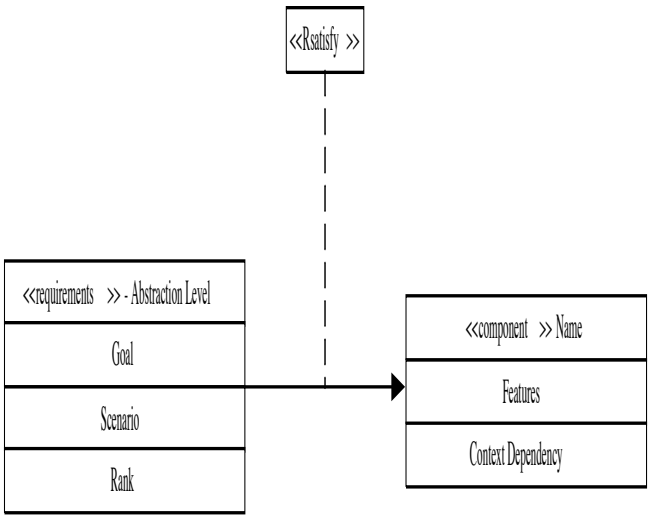

Figure 5. Satisfy Mapping Relationship

conflict with each other is an example of an interaction relationship. Similarly, the mapping relationship defines the association between a Rclass and a Cclass, as shown in Figure 6. A component satisfying a requirement is an example of a mapping relationship. We define the satisfy mapping relationships required to model requirements characterization process of RAAP in detail as follow:

4.3.1 Satisfy Mapping

A component that satisfies a requirement is modeled with stereotype <<Rsatisfy>> specifies that a Cclass C1 satisfies a Rclass R1. Figure 5 shows the notation of <<Rsatisfy>> mapping relationship. Formally, we define the satisfy mapping as:

Predicate 3 *member* (X_r, X_c)

This binary predicate specifies that the component X_c satisfies the requirement X_r.

5 RE-UML Top-down Analysis

The second phase of RAAP, top-down analysis, uses a metrics hierarchy to quantify stakeholder requirements against component features. RE-UML introduces the following notations that allows the top-down analysis to be represented in the UML.

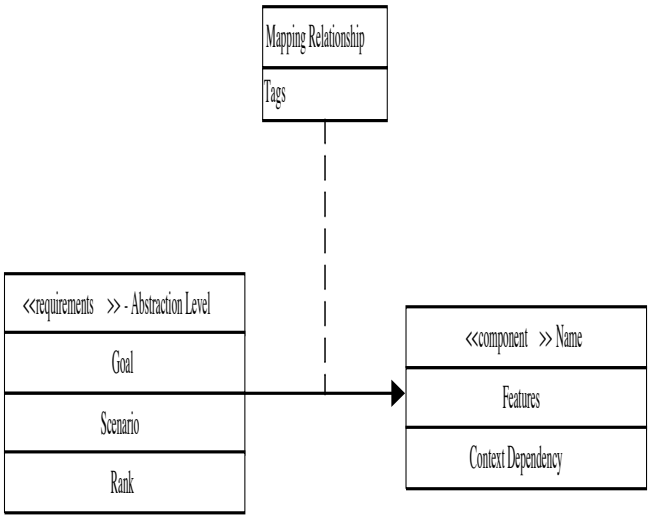

Figure 6. Mapping Relationship

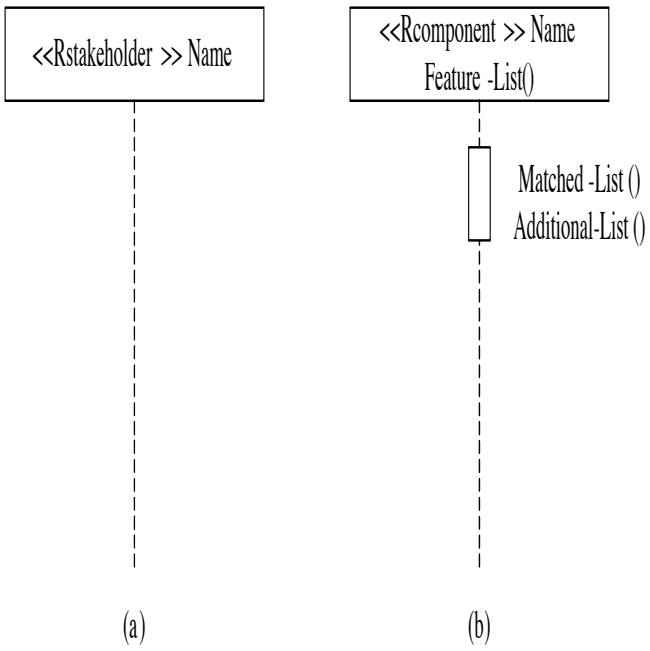

Figure 7. <<Rstakeholder>> and <<Rcomponent> Notation

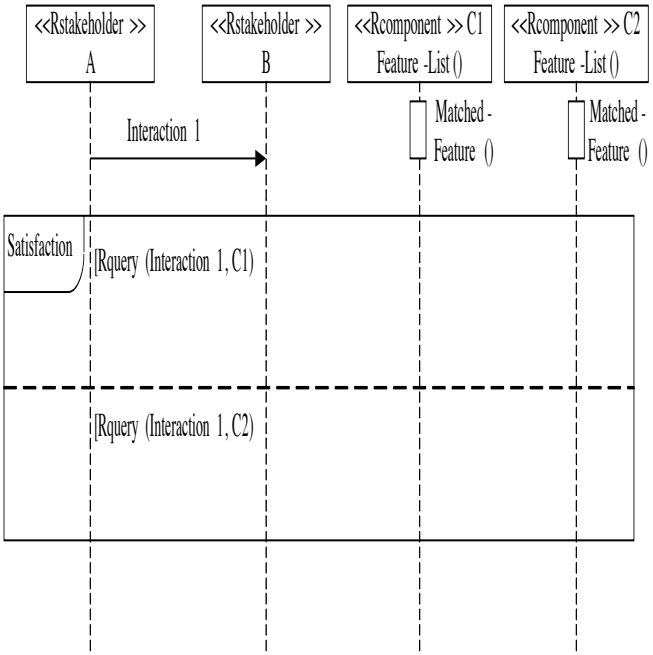

Figure 8. Sequence Diagram

5.1 Rsequence Diagram

RSequence diagram is an extension of UML sequence diagram [16] to model and document component matching which involves an evaluation of the degree to which a component satisfies a requirement. We identify that there are two types of participants, namely *stakeholder* and *component*, that interact during satisfaction analysis of a requirement with candidate components.

A *stakeholder* participant represents a role that an actor plays within a CBS. In Rsequence diagram, a *stakeholder* participant is modeled as a rectangle with the stereotype <<Rstakeholder>>, as shown in Figure 7 (a). Similarly, a *component* participant is modeled as a rectangle with the stereotype <<Rcomponent>>, as shown in Figure 7 (b). The dashed lines hanging from the rectangles, called lifelines, represents the life span of the participants in a RSequence diagram. Furthermore, we use three tags of the <<Rcomponent>> stereotype: *Feature-List*, *Matched-List* and *Additional-List*. The tag *Feature-List* represents the list of features supported by a component. The tag *Matched-List* represents the list of features that match a concrete-level requirements interaction. Finally, the tag *Additional-List* represents the list of additional features supported by a component which are not required for satisfaction of a requirement.

UML specification [16] introduces the concept of frames to model conditional behavior. Rsequence diagram extends this concept to introduce a new frame with stereotype <<Rsatisfaction>> to model matching criteria between stakeholder requirements and component features, as shown in Figure 8. A <<Rsatisfaction>> frame is associated with each interaction between two <<Rstakeholder>> participants. A <<Rsatisfaction>> frame can be of one or more fragments. The number of fragments in a <<Rsatisfaction>> frame is equal to the number of <<Rcomponent>> participants in a RSequence diagram.

Furthermore, each fragment has a guard condition which is used to specify the relationship between an interaction and a component. Formally, we define guard condition as

Predicate 4

Rquery (Interaction X_i, Component X_c)

This guard condition is used to model the conditional expression, 'does component X_c satisfies the interaction X_i'.

5.2 Compatible Feature

Compatible feature is the number of component features that contribute in fulfilling a concrete-level requirements interaction. Compatible features are identified by

analyzing Rsequence diagram interactions with candidate component features. If a component has a feature or set of features which satisfy an interaction, it is added to the matched-feature list and, compatible feature counter is incremented. RE-UML models compatible feature relationship as an association between <<Rstakeholder>> and <<Rcomponent>> participants of a Rsequence diagram.

Figure 9 shows the notation of compatible feature association which starts at <<Rcomponent>> C1 and links to <<Rstakeholder>> A in the <<Rsatisfaction>> frame associated with interaction I1. This association indicates that component C1 satisfies the interaction I1. Formally, we define the compatible feature association as

Rule 1

CF (X_i, X_c, $X_c$on):
member (X_i, X_c),
add (X_cf,Matched-List),
$X_c$on is $X_c$on + 1

The rule specifies that the component X_c satisfies the interaction X_i. The CF rule returns the variable $X_c$on which is the compatibility feature counter for the component X_c. The 'match' and 'add' rules are defined as follows.

Rule 2

Match (X_ir,X_c):
member (X_i, Feature-List)

The rule specifies that the component X_c supports the interaction X_i if the interaction matches the component-feature list.

Rule 3

Add(Xcf, Matched-Feature):
insert (Xcf, Matched-Feature),
member (Xcf, Matched-Feature)!

denotes that a component feature $X_c f$ is inserted into matched-list, if the $X_c f$ is not already a member of the list.

5.3 Missing Feature

Missing feature is a measure of concrete-level requirements interactions not fulfilled by a component. Missing features are identified by analyzing a RE-UML sequence diagram interaction with candidate component features. If there are no component features that satisfy a RE-UML sequence diagram interaction, missing feature

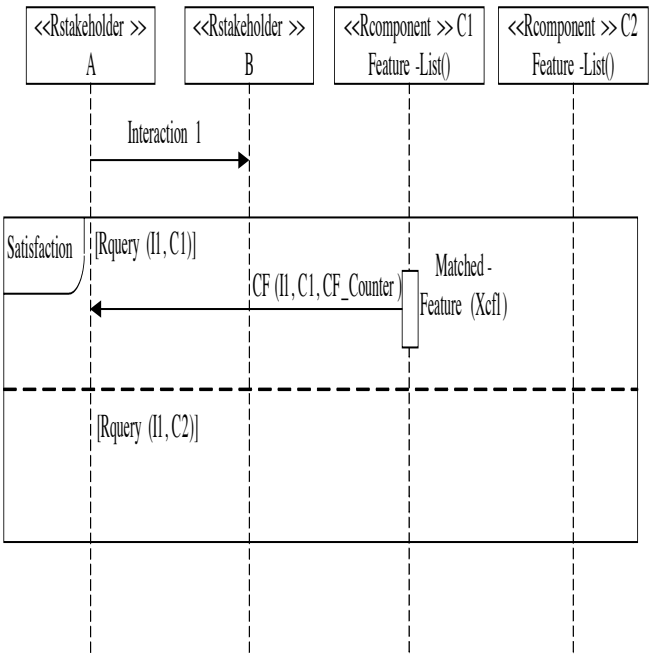

Figure 9. Compatible Feature Relationship

counter is incremented by one. RE-UML models missing feature as an association between a <<Rstakeholder>> and <<Rcomponent>> participants of a Rsequence diagram.

Figure 10 shows the notation of missing feature association which starts at <<Rcomponent>> C2 and links to <<Rstakeholder>> A in the <<Rsatisfaction>> frame associated with interaction I1. This association indicates that component C2 does not satisfy the interaction I1. Formally, we define the missing feature association as:

Rule 4

MF (X_i, X_c, $X_m f$):
match (X_i, X_c) !,
$X_m f$ is $X_m f$ + 1.

The rule specifies that the interaction X_i is not satisfied by the component X_c and returns the value of $X_m f$ which is a missing feature counter for the component X_c.

5.4 Additional Feature

Added feature is a measure of component features that are not required for the satisfaction of a concrete-level requirements interaction. Additional features are identified by comparing a <<Rcomponent>> participant's feature

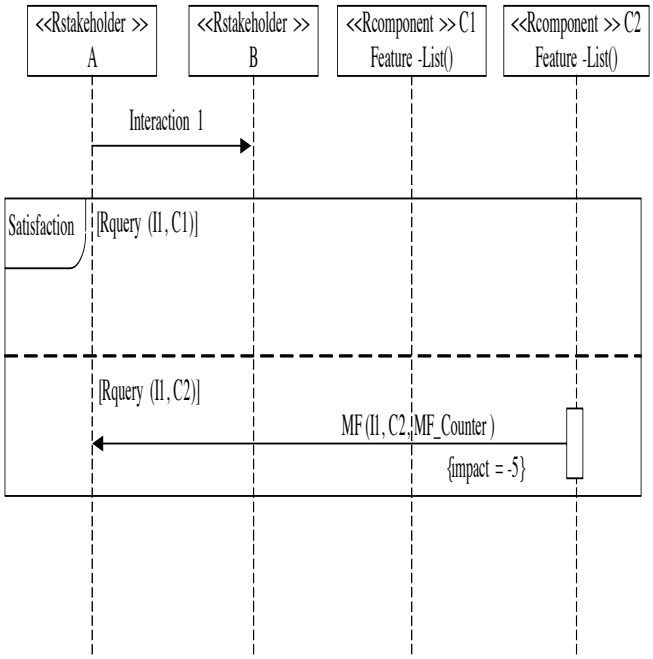

Figure 10. Missing Feature Relationship

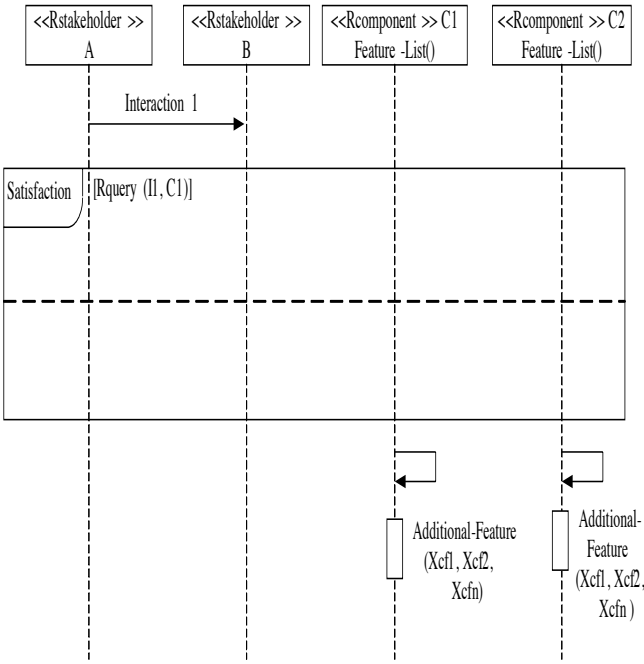

Figure 11. Additional Feature Relationship

list with matched list. In RE-UML, added feature is represented as a stereotype <<additional-feature>> and modeled using a self stimulus symbol, as shown in Figure 11. Formally, we define the <<additional-feature>> as:

Rule 5

$$AF\,(X_i, X_c,\ X - af):$$

insert (member (X_c, Feature-List)
- member (X_c, Matched-List),Additional-List)
X_af is length (Additional-List)

The rule specifies that for the interaction X_i, number of added features provided by component X_c is X_af.

5.5 Impact constraint

Finally, impact constraint specifies the impact of missing and added features introduced by a component during characteristic analysis of a requirement. The impact constraint is modeled as a stereotype <<impact>> and has an associated tagged value. The tag value quantifies the impact as either 'negligible', 'little', 'moderate', 'considerable' or 'great'. We assign impact indices of 5, 3, 1, -3 and -5 to 'negligible', 'little', 'moderate', 'considerable' and 'great' respectively.

5.6 Risk Mapping

Risk assessment is used to quantify the degree of uncertainty associated with the selection of a component. The risk assessment is modeled as a special mapping association with stereotype <<Rrisk>> together with three tag values: complexity, VS and SC. Figure 12 illustrates a risk-analysis mapping relationship between a requirement Rclass R1 and a candidate component Cclass C1. The complexity tag value specifies the interface complexity associated with the component. The VS tag specifies the volatility of a component; and the SC tag specifies the supplier creditability. The VS and SC are quantified using the rules defined in [14].

6 RE-UML Trade-off Analysis

The third phase of RAAP, trade-off analysis, is executed to identify and resolve the potential conflicts in requirements. The conflict analysis is modeled as a special interaction association with stereotype <<Rconflict>>. The conflict mapping specifies that there is a negative association between two requirements. Figure 13 shows the graphical representation of <<conflict>> mapping. Furthermore, we propose to model each resolution by defining a generalization relationship denoted as stereotype <<Rresolution>>. Each <<Rresolution>> is related to a Rclass which indicates that it is one possible resolution to a conflict between Rclass R2 and Rclass R1, as shown in

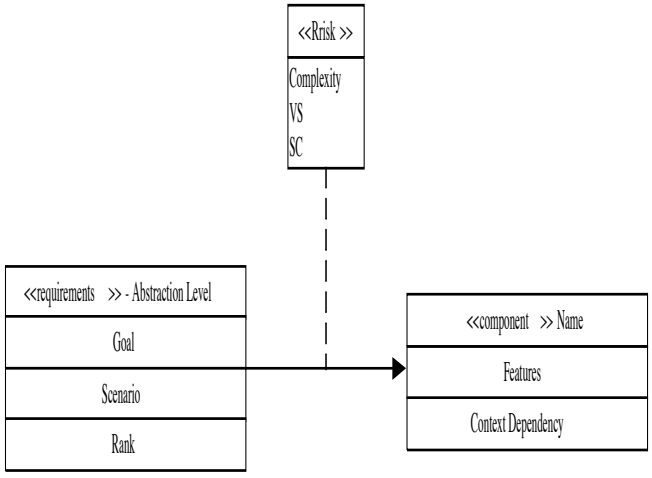

Figure 12. Risk Mapping Relationship

Figure 13. For a detailed discussion on conflict indentification, resolution generation and suitable resolution selection, please refer to our previous work [14].

7 Conclusions

In this paper, we have presented RE-UML, an extension to UML, to support each phase of RAAP. RE-UML extends the UML class diagram with two specialized classes, namely, *Rclass* and *Cclass*, to specify stakeholder requirements and component features respectively. Similarly, the UML sequence diagram is extended with <<Rsatisfaction>> frame to model matching criteria between stakeholder demands and component features. RE-UML introduces a set of associations to model the top-down and trade-off analyses. RE-UML also uses logic programming style predicates and rules to specify formal semantics of proposed extensions to UML.

The development of RE-UML helps specify RAAP in a notation that is compatible with industry standards. RE-UML removes the need for a system analyst to learn a new notation to model CBS requirements and component selection process. Further, we believe that RE-UML reduces the dependency on a system analyst's perceptions to represent the CBS requirements analysis and the component selection. For future work, RE-UML needs to be extended to support non-functional requirements analysis and assessment. Furthermore, the ability to model CBS requirements analysis and assessment in UML also offers the potential for tool support.

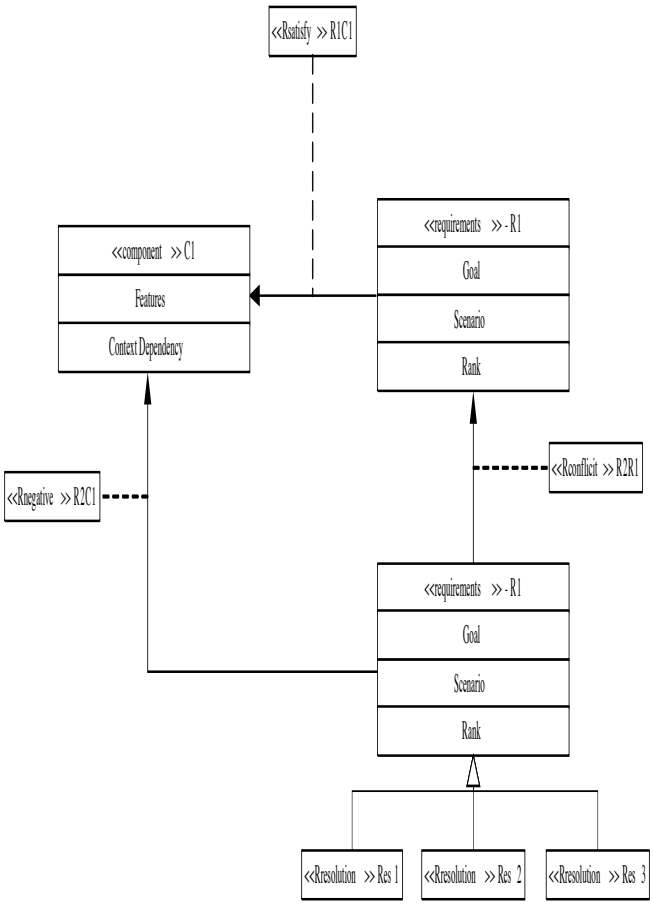

Figure 13. Trade-off analysis specifications

References

[1] Khaled Alghathbar. Validating the enforcement of access control policies and separation of duty principle in requirement engineering. *Information and Software Technology*, 49(2):142 – 157, 2007.

[2] Ludovic Apvrille, Jean-Pierre Courtiat, Christophe Lohr, and Pierre de Saqui-Sannes. Turtle: A real-time uml profile supported by a formal validation toolkit. *IEEE Transactions on Software Engineering*, 30(7):473 – 487, 2004.

[3] Grady Booch, James Rumbaugh, and Ivar Jacobson. *The Unified Modeling Language User Guide*. Addison-Wesley, 1999.

[4] Alejandra Cechich and Mario Piattini. Early detection of cots component functional suitability. *Information and Software Technology*, 49(2):108 – 121, 2007.

[5] John Cheesman and John Daniels. *UML Components A Simple Process for Specifying Component Based Software*. Addison-Wesley, 2001.

[6] William Heaven and Anthony Finkelstein. Uml profile to support requirements engineering with kaos. *IEE Proceeding Software*, 151(1):10 – 27, 2004.

[7] Mohammed Hussein and Mohammad Zulkernine. Intrusion detection aware component-based systems: A specification-based framework. *The Journal of Systems and Software*, 80(5):700 – 710, 2007.

[8] Akira Ishikawa and Tai Nejo. *The Success of 7-Eleven Japan: Discovering the Secrets of the World's Best-Run Convenience Chain Stores*. World Scientific Publishing Co., 1998.

[9] J.Rumbaugh. Getting started: Using use cases to capture requirements. *Journal of Object Oriented Programming*, 7(5):8 – 12, 1994.

[10] Axel van Lamsweerde, Robert Darimont, and Emmanuel Letier. Managing conflicts in goal-driven requirements engineering. *IEEE Transactions on Software Engineering*, 24(11):908 – 926, 1998.

[11] Axel Van Lamsweerde and L Willemet. Inferring declarative requirements specifications from operational scenarios. *IEEE Transactions on Software Engineering*, 24(12):1089 – 1114, 1998.

[12] Jonathan Lee and Nien-Lin Xue. Analyzing user requirements by use cases: A goal-driven approach. *IEEE Software*, 16(4):92 – 101, 1999.

[13] Sang Duck Lee, Young Jong Yang, Fun Sook Cho, Soo Dong Kim, and Sung Yul Rhew. Como: a uml-based component development methodology. In *Proceedings of Sixth Asia Pacific Software Engineering Conference (APSEC '99)*, pages 54–61, 1999. TY - CONF.

[14] Sajjad Mahmood and Richard Lai. Analyzing component based system specification. In *proceedings of 11th Australian Workshop on Requirements Engineering (AWRE 2006)*, Adelaide, Australia, 2006.

[15] Sajjad Mahmood and Richard Lai. A complexity measure for uml component system specification. *Software-Practice and Experience*, 38(2):117–134, 2008.

[16] James Rumbaugh, Ivar Jacobson, and Grady Booch. *The Unified Modeling Language Reference Manual*. Addison-Wesley, 2005.

[17] Kassem Saleh and Christo El Morr. M-uml: An extension to uml for the modeling of mobile agent-based software systems. *Information and Software Technology*, 46(4):219 – 227, 2004.

[18] Rodolfo Villarroel, Eduardo Fernandez-Medina, Mario Piattini, and Juan Trujillo. A uml 2.0/ocl extension for designing secure data warehouses. *Journal of Research and Practice in Information Technology*, 38(1):31 – 43, 2006.