ISSN: 2278 – 1323
International Journal of Advanced Research in Computer Engineering & Technology (IJARCET)
Volume 2, Issue 4, April 2013

Conceptual Dependency Based Requirements Clustering for Component Selection

Siba A.[1], R. Subha[2], S. Palaniswami[3]

[1]PG Scholar, Department of Computer Science and Engineering
Sri Krishna College of Technology, Kovaipudur, Coimbatore-641 042, Tamil Nadu, India

[2]Assistant Professor, Department of Computer Science and Engineering
Sri Krishna College of Technology, Kovaipudur, Coimbatore-641 042, Tamil Nadu, India

[3]Principal,
Govt. College of Engineering, Bodinayakanur-625 528, Tamil Nadu, India

***Abstract*-Component Based System (CBS) development is used to integrate existing components to build a software system, with the potential benefits of reducing development time and delivering quality system by using quality components. Component selection process is used to provide guidelines for CBS developers to consolidate related requirements into clusters and shortlist components that best match the required functionalities of CBS. The component selection process helps a developer to elicit stakeholder needs, analyze their interdependencies and select components for a CBS. In this a goal model is created to denote various Concrete Level Goals and using these goals a conceptual dependency analysis with the Schank's notation is created to model interdependencies of CBS.The conceptual dependencies represents AND, OR relationship between various goals and a matching index of each cluster of related goals is used as a criterion to identify a portfolio of candidate components for a CBS, providing multiple solutions for the component selection problem whenever possible.**

Keywords-Component based software system; Software component selection; Conceptual dependency analysis; Matching index; Requirements engineering.

I. INTRODUCTION

Component-based Software Engineering (CBSE) (also known as component-based development (CBD)) is a branch of software engineering that emphasizes the separation of concerns in respect of the wide-ranging functionality available throughout a given software system. It is a reuse-based approach to defining, implementing and composing loosely coupled independent components into systems. This practice aims to bring about an equally wide-ranging degree of benefits in both the short-term and the long-term for the software itself and for organizations that sponsor such software.Software Reuse may be defined as the process of reusing the software with existing components .In most engineering disciplines, systems are designed by composition (building system out of components that have been used in other systems).Software engineering has focused on custom development of components to achieve better software quality, more quickly, at lower costs, software engineers are beginning to adopt systematic reuse as a design process.

A goal model is represented based on a Meeting Scheduling System (MSS). CBS requirements is classified into two refinement levels, namely, High-Level Goal (HLG) and Concrete-Level Goal (CLG).An HLG represents CBS requirements that contribute to achieve business objectives but are not detailed enough to be directly used for component selection and evaluation. A CLG represents a finer-grained goal that can be directly used for component selection. The goal model identifies the CLGs that can be used for component selection. The component selection process needs to consider the semantic dependencies between CLGs. The relationships between concrete-level goals are classified into three types of semantic dependencies, namely, usage, non-functional and threat dependencies. Using these dependencies a conceptual analysis is made and it is represented using Schank's notation [2]. A conceptual graph (CG) is a graph representation for logic based on the semantic networks. Conceptual Dependency originally developed to represent knowledge acquired from natural language input. The proposed method supports the conceptual dependencies between the various CLG's and cluster those goals based on the local optimization algorithm. Each CLG in the cluster is identified as a keyword and is used to search for off-the-shelf components that satisfy it. The off-the-shelf components that satisfy all the CLGs in a cluster are candidates for the cluster (consolidated CLG). The rest of the paper is organized as follows. In Sect. 2, we take a look at related literature. In Sect. 3, we have various methodologies to support the component selection process In Sect. 4, we discuss the performance of the proposed method and conclude in Sect. 5.

II. RELATED WORK

Ivica Crnkovi´c, S´everine Sentilles, Aneta Vulgarakis, and Michel R. V. Chaudron presented a classification framework for software component models. In the last decade a large number of different software component models have been developed, with different aims and using different principles and technologies. This has resulted in a number of models which have many similarities and many unclear concepts. Currently, there exist many component models. Some component models target specific application domains, ranging from embedded systems (automotive software, consumer electronics) to particular business domains (finance, telecommunications, healthcare, transportation).Other component models are based on certain technological platforms (Enterprise Java Beans, DCOM).Component-based development has not succeeded in providing standard principles, as has, for example, object-oriented development. In order to increase the understanding of the concepts, and to differentiate component models more easily, this paper identifies, discusses and characterises fundamental principles of component models, and provides a Component Model Classification Framework based on these principles. This paper describes a group of method to classify component models.

Lawrence Chung and Kendra Cooper analysed a knowledge-based cots-aware requirements engineering approach [13]. A COTS-Aware Requirements Engineering (CARE) approach explicitly supports the use of COTS components is used. CARE approach is knowledge based, has a defined process, and is agent- and goal-oriented. In particular, a meta-model for the knowledge base and a prototype of the CARE Assistant Tool is also implemented. The effective use of COTS component provides a set of concepts for modelling the subject matter, a set of guidelines for using such concepts, and tool support to assist the developer.

Abdallah Mohamed, Guenther Ruhe and Armin Eberlein presented a MiHOS: an approach to support handling the mismatches between system requirements and COTS products**.** In the process of selecting Commercial Off-The-Shelf (COTS) products, it is inevitable to encounter mismatches between system requirements and COTS products [12]. These mismatches occur as a result of an excess or shortage of the COTS attributes. In this paper a decision support approach, called MiHOS (Mismatch Handling for COTS Selection), that aims at addressing COTS mismatches during and after the selection process. MiHOS can be integrated with existing COTS selection methods at two points: (1) for evaluating COTS candidates: MiHOS estimates the anticipated fitness of the candidates if their mismatches are resolved [6]. This helps to base COTS selection decisions on the fitness that the candidates will eventually selected. (2) Mismatch resolution after selecting a COTS product: MiHOS suggests alternative plans for resolving the most appropriate mismatches using suitable actions.

Jong Kook Lee, Seung Jae Jung and Soo Dong Kim introduced a component Identification method with Coupling and Cohesion. Component-Based Development (CBD) is an effective component identification technique. A component consists of one or more related objects, carrying out a homogeneous functionality. Most of the CBD methodologies utilize UML as the basic notational convention. Especially the component diagram or its variation is used to depict components.CBD methodologies is having lack of systematic component identification algorithm that can be effectively used to group related use-cases and classes into components. In this paper, component granularity is the most important factor for designing business component-based systems. Typically, the time and platform resources of inter-component communications are typically expensive. Thus, components must be larger rather than smaller.

Christoph Becker and Andreas Rauber introduced improving component selection and monitoring with controlled experimentation and automated measurements. It describes an evidence-based approach for evaluating component which can improve repeatability and reproducibility of component selection under the following conditions: (1) Functional homogeneity of candidate components and (2) High number of components and selection problem instances. In this a framework is introduced and a method is used for evaluating and selecting components. A number of approaches have been proposed for the general problem of software component evaluation and selection. Most approaches come from the field of Component-Based Software Development (CBSD), tackle the problem of Commercial-off-the-shelf component selection and use goal-oriented requirements modelling and multi-criteria decision making techniques.

Ivica Crnkovic and Magnus Larsson introduced challenges of component-based development.System evolution, maintenance, migration and compatibilities are some of the challenges met with developing a component-based software system. Since most systems evolve over time, components must be maintained or replaced. Increased complexity is a consequence of different components and systems having different lifecycles. In component-based systems it is easier to replace part of system with a commercial component. This process is however not straightforward and different factors such as requirements management, marketing issues, etc., must be taken into consideration. In this various issues and challenges encountered when developing and using an evolving component-based software system is depicted. The reuse orientation provides many advantages, but it also requires systematic approach in design planning, extensive development, support of a more complex maintenance process, and in general more consideration being given to components.

III. METHODOLOGY

The component selection process consists of the following methodologies: namely a goal model construction, conceptual dependency analysis, CLG clustering, component selection and matching index. Goals are objectives that a system should achieve through cooperation of actors in the intended software and in the environment [4]. A goal model is represented based on a Meeting Scheduling System (MSS). CBS requirements is classified into two refinement levels, namely, High-Level Goal (HLG) and Concrete-Level Goal (CLG).In the next method a conceptual dependency is analysed between various CLG's.The dependency is classified mainly into usage, non functional and threat dependencies.The conceptual dependencies may be analysed based on the schank's notation. It includes the logical conjunction, logical disjunction and the negation between the various dependencies.

A. *Construction of a Goal Model*

Goals are objectives that a system should achieve through cooperation of actors in the intended software and in the environment. Goal modeling is useful in the early phase of a project. It expresses the relationships between a system and its environment. Goal models are effective in capturing large numbers of alternative sets of low- level tasks, operations, and configurations that can fulfil high-level stakeholder requirements.This module contains basic goals that are represented using GR tool.

GR Tool is the goal reasoning tool [15]. The GR-Tool is graphical tool in which it is possible to draw goal models and run the algorithms and tools for forward and backward reasoning. Goal modelling can identify and help to resolve tradeoffs between cost, performance, flexibility, security and other goals. It expresses the relationships between a system and its environment. Goal models have been found to be effective in concisely capturing large numbers of alternative sets of low-level tasks, operations, and configurations that can fulfil high-level stakeholder requirements. The algorithms for the forward reasoning have been fully developed in java and are embedded in the GR-Tool.

A goal model is represented based on a Meeting Scheduling System (MSS). CBS requirements is classified into two refinement levels, namely, High-Level Goal (HLG) and Concrete-Level Goal (CLG).An HLG represents CBS requirements that contribute to achieve business objectives but are not detailed enough to be directly used for component selection and evaluation. A CLG represents a finer-grained goal that can be directly used for component selection. The lists of concrete level goals are in Table 1 and a goal model is depicted in Fig.1

TABLE I
LIST OF CLG

1	Constraints requested
2	Constraints entered
3	Network communication working
4	Meeting scheduler
5	Scheduled meeting notification
6	Schedule conflict free meeting
7	Promptness
8	Minimum interaction
9	User friendly data input
10	Store participant constraints

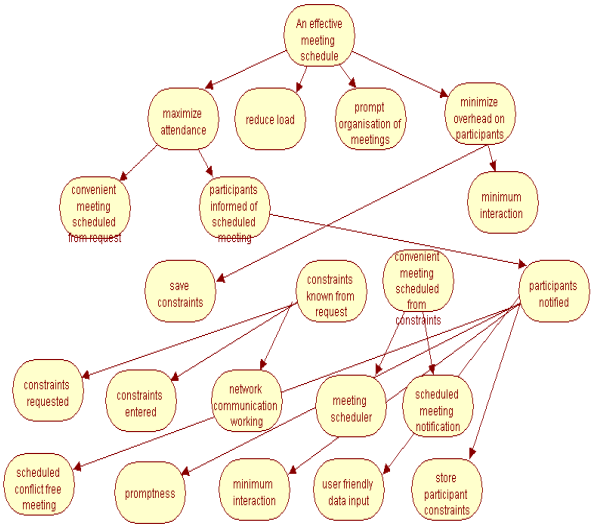

Fig. 1. A goal model

B. *Conceptual Dependency Analysis*

The goal model identifies the CLGs that can be used for component selection. The component selection process needs to consider the semantic dependencies between CLGs. The relationships between concrete-level goals are classified into three types of semantic dependencies, namely, usage, non-functional and threat dependencies.

1. Usage dependency

Usage dependencies specify the usage associations between functional concrete-level goals such that one concrete-level goal requires another concrete-level goal for its correct functionality or implementation.

Example, 'meeting scheduler' functional CLG depends on 'schedule conflict free meeting' for its correct functionality.

2. Non-functional dependency

Non-functional dependency is an association between a non-functional concrete-level goal and a functional concrete-level goal such that the non-functional concrete-level goal is used as one of the criteria for identifying candidate components for the functional concrete-level goal.

Example, 'user-friendly data input' is a non-functional CLG that directly impacts the 'constraint entered' functional CLG.

3. Threat dependency

Threat dependency as an association between two functional concrete-level goals A and B such there exists a threat T that affects both concrete-level goals. These common threats need context specific solutions at different points in a system. Hence, it is important to analyze the threats between different concrete-level goals.

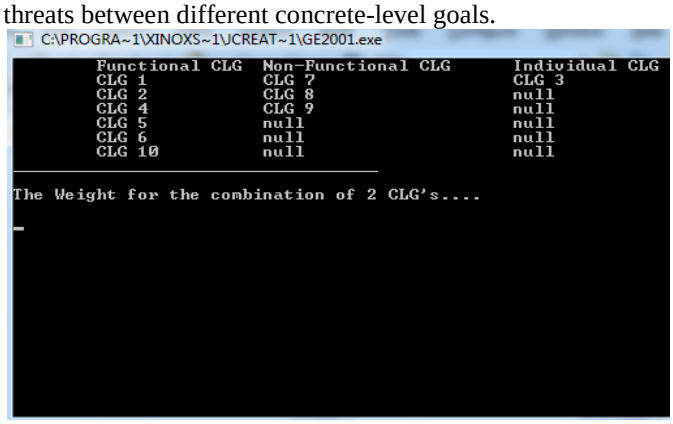

Fig. 2. Dependency analysis of CLG.

A conceptual graph (CG) is a graph representation for logic based on the semantic networks [3]. Conceptual Dependency originally developed to represent knowledge acquired from natural language input.

The goals of this theory are:

- To help in the drawing of inference from sentences.
- To be independent of the words used in the original input.
- For any 2 (or more) sentences that are identical in meaning there should be only one representation of that meaning.

It has been used by many programs that portend to understand English. CD was developed by Schank *et al.*

a. Logical negation

Logical negation is an operation on one logical value, typically the value of a proposition that produces a value of true if its operand is false and a value of false if its operand is true.

b. Logical conjunction

Logical conjunction is an operation on two logical values, typically the values of two propositions, that produces a value of true if both of its operands are true.

c. Logical disjunction

Logical disjunction is an operation on two logical values, typically the values of two propositions, that produces a value of true if at least one of its operands is true.

In order to differentiate between the three dependences, three weight values w1, w2 and w3 such that w1, w2 > 0 (as the usage and non-functional dependencies are positive in nature) while w3 < 0 (as the threat dependency is a negative relationship) are chosen. We propose to assign weights of 1, 0.1 and _1 to the non-functional, usage and threat dependencies, respectively. The choice of weights is based on the two case studies and guided by the following rules: _ since non-functional CLGs are rendered meaningless for component selection without their associated functional CLGs, it is far more important to keep CLGs with a non-functional dependency in the same cluster (component) than with a usage dependency. As a result, a non-functional dependency should be assigned a much higher positive weight than a usage dependency to ensure that the clustering algorithm prefers non-functional dependencies over usage dependencies [1]. We therefore assign weights of 1 and 0.1 respectively to non-functional and usage dependencies.

C. Goal Cluster Analysis

Goal cluster analysis consists of two steps: cluster formation in the various CLG's followed by component selection based on the matching index of each cluster.

1. Cluster formation

Clustering is the process of grouping the data into classes or clusters, so that objects within a cluster have high similarity to one another but are dissimilar to objects in other cluster. In this step, the CLG's are clustered based on their interdependencies. We use a local optimization algorithm for clustering [8]. Our objective is to cluster the concrete-level goals with non-functional and usage relationships together while separating concrete-level goals with threat dependencies. If the goal dependency graph G is not clusterable, the local optimization algorithm finds a partition that minimizes the clustering error. The clustering error Er is defined as Er=pos+|neg|, where | | denotes absolute value [9]. The negative error of a partition is the sum of weights of all negative values that lie inside clusters and the positive error can be defined as the sum of weights of positive values joining different clusters.

D. Component Selection and Matching Index

In this phase a set of components is analysed from various software requirement specifications and drupal modules. Off-the shelf components are the prebuilt component usually from a third party vendor. Each CLG in the cluster is identified as a keyword and is used to search for off-the-shelf components that satisfy it. The off-the-shelf components that satisfy all the CLGs in a cluster are candidates for the cluster (consolidated CLG).

The matching index of a cluster is defined in (1)

$$\frac{\text{Number of components satisfying all CLGs in C}}{\text{Number of components satisfying at least one CLG in C}} = M(C) \qquad (1)$$

The matching index of a cluster consisting of a single non functional CLG is necessarily 'undefined', as a component cannot be selected for such a CLG. When a non-functional CLG is clustered with a related functional CLG it acts as a criterion for selecting the component for the functional CLG. Therefore, to satisfy a cluster, a component must satisfy all the functional and non-functional CLGs in that cluster. The matching index decreases with increasing cluster size.

IV CONCLUSION

The component selection process helps system analysts to model interdependencies of CBS requirements and facilitates the selection of a portfolio of candidate components that satisfy system requirements [11]. The component selection process uses goal-oriented requirements specification techniques to elicit CBS-to-be requirements at two abstraction levels, namely, high-level and concrete-level goals. We also identify three types of semantic dependencies between concrete-level goals, namely, usage, non-functional and threat. The component selection process uses a conceptual dependency analysis by using Schank's notation with the help of clustering algorithm and matching index to select CBS components and presents multiple candidates for each clustered requirement of the CBS.

REFERENCES

[1] Muhammad Ali Khan , Sajjad Mahmood,"A graph based requirements clustering approach for component selection," Advances in Engineering Software 54 (2012) 1–16

[2] Li J, Conradi R, Siyngstad OPN, Bunse C, Torchiano M, Morisio M,"Development with off-the-shelf components"10 facts. IEEE Software 2009;26(2):80–7.

[3] Rolland C, Souveyet C, Achour CB,"Guiding goal modeling using scenarios,"IEEE Trans Software Eng 1998;24(12):1055–71.

[4] van Lamsweerde A," Requirements engineering from system goals to UML models to software specifications," Wiley; 2009.

[5] Doreian P, Mrvar A,"Partitioning signed social networks," Soc Networks 2009;31(1):1–11.

[6] Cechich A, Piattini M," Early detection of cots component functional suitability,"Inf Software Technol 2007;49(2):108–21.

[7] Harman M, Skaliotis A, Steinhofel K. Search-based approaches to the component selection and prioritization problem. In: Proceedings of the 22nd IEE international conference on software maintenance. 2006. p. 176–85.

[8] Birkmeier D, Overhage S. "On componenet identification approaches classification, state of the art, and comparision," In: Lewis GA, Poernomo I,Hofmeister C. editors. 12th international symposium on component-based software engineering, vol. LNCS 5582. 2009. p. 1–18.

[9] Crnkovic I, Larsson M,"Challenges of component based development,"J Syst Software 2002;61(3):201–12.

[10] Sommerville I,"Integrated requirements engineering,": a tutorial. IEEE Software 2005;22(1):16–23.

[11] Mohamed A, Ruhe G, Eberlein A," Mihos: an approach to support handling the mismatches between system requirements and cots products," Requirements Eng 2007;12:127–43.

[12] Hope P, McGraw G, Anton AI," Misuse and abuse cases: getting past the positive," IEEE Secur Privacy 2004;2(3):90–2.

[13] Sindre G, Opdahl AL,"Eliciting security requirements with misuse cases,"Requirements Eng 2005;10:34–44.

[14] de Nooy W, Mrvar A, Batageli V,"Exploratory social network analysis with Pajek," Cambridge University Press; 2005.

[15] http://www.tropos project.org