GEA: A Goal-Driven Approach to Discovering Early Aspects

Jonathan Lee, *Senior Member, IEEE* and Kuo-Hsun Hsu, *Member, IEEE*

Abstract—Aspect-oriented software development has become an important development and maintenance approach to software engineering across requirements, design and implementation phases. However, discovering early aspects from requirements for a better integration of crosscutting concerns into a target system is still not well addressed in the existing works. In this paper, we propose a Goal-driven Early Aspect approach (called GEA) to discovering early aspects by means of a clustering algorithm in which relationships among goals and use cases are utilized to explore similarity degrees of clustering goals, and total interaction degrees are devised to check the validity of the formation of each cluster. Introducing early aspects not only enhances the goal-driven requirements modeling to manage crosscutting concerns, but also provides modularity insights into the analysis and design of software development. Moreover, relationships among goals represented numerically are more informative to discover early aspects and more easily to be processed computationally than qualitative terms. The proposed approach is illustrated by using two problem domains: a meeting scheduler system and a course enrollment system. An experiment is also conducted to evaluate the benefits of the proposed approach with Mann-Whitney U-test to show that the difference between with GEA and without GEA is statistically significant.

Index Terms—Early aspects, goals, goals interaction, fuzzy logic, use cases, goal cluster

1 Introduction

Aspect-oriented software development (AOSD) has become an important development and maintenance approach to software engineering across requirements and design [1], [2], and coding [1], [3], [4], [5], [6], [7], [8]. It provides explicit means to model important stakeholders' concerns that tend to crosscut multiple system components. However, identifying concerns that crosscut the systems in the early stages of software development are still hindered from the difficulty that most programming and modeling formalisms enforce a dominant decomposition that allows only a few concerns to be separated [9].

To address this issue, many researches [10], [11], [12], [13], [14] have suggested that the coupling of goal-based and user-centered approaches is a good way to elicit user requirements that contain recurring properties or important stakeholders' concerns. It is generally agreed that the occurrence of crosscutting is not limited to non-functional requirements but also to functional requirements [15], [16], [17]. The tenet of goal-based approaches focuses on why systems are constructed, which provides the motivation and rationale to justify software functional and non-functional requirements [18]. User-centered approach is also useful in elicitation, analysis and requirements documentation [19]. In [20], discovering of concerns in early phases of software development process has been deemed as a way to achieve untangled and non-scattered designs and codes and accomplish a common agreement on the views of the involved.

In this work, we propose a goal-driven approach, called Goal-driven Early Aspect (GEA), to discovering early aspects through goals interactions by means of a clustering algorithm for grouping goals, in which early aspectual candidates are derived as a basis for finding early aspects. An early aspectual candidate refers to a set of goals with a higher frequency of being grouped together. Early aspects are defined as crosscutting concerns that are discovered in the early life cycle phases of a software systems' development, including requirements analysis, domain analysis and architecture design phases [21]. This work is an extension to our previous research on goal-driven use case model [13], [22], [23], [24], [25], in which use cases are derived based on the analysis of goals interactions. Introducing the notion of goals clustering enables the goal-driven use case model to address crosscutting concerns in the early stage of software development. GEA consists of three main features:

1) To analyze a system by formulating goals and use cases. Goals are represented by an extended goal structure to consider various facets of requirements, such as functional or non-functional, rigid or soft, and actor-specific or system-specific. Use cases are identified to achieve the goals. Developers can assign relationships between goals and use cases numerically and in a pairwise manner.
2) To discover early aspects by means of a clustering algorithm that organizes goals into goal clusters for discovering early aspectual candidates through the exploration of interactions among goals and use cases. The clustering algorithm engages similarity degree among goals for clustering goals, and total

- *J. Lee is with the Department of Computer Science and Information Engineering, National Taiwan University, Taipei, Taiwan. E-mail: jlee@csie.ntu.edu.tw.*
- *K.-H. Hsu is with the Department of Computer Science, National Taichung University of Education, Taichung, Taiwan. E-mail: glenn@mail.ntcu.edu.tw.*

Manuscript received 3 Mar. 2013; revised 22 Apr. 2014; accepted 1 May 2014. Date of publication 28 May 2014; date of current version 19 June 2014. Recommended for acceptance by M. Dwyer. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below. Digital Object Identifier no. 10.1109/TSE.2014.2322368

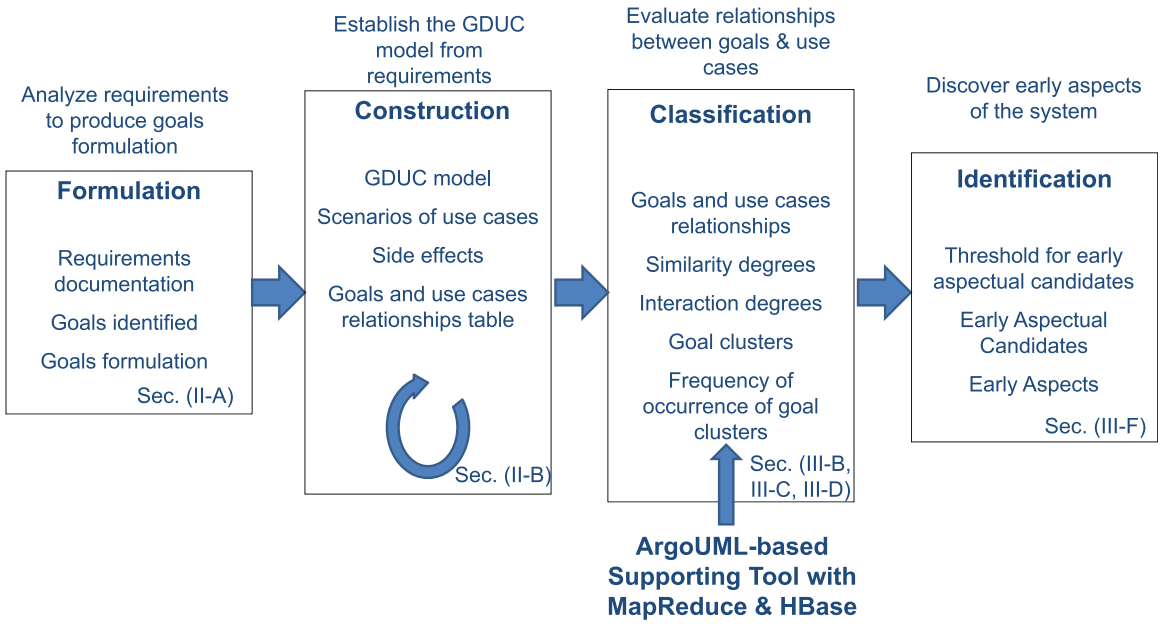


Fig. 1. Goal-driven early aspect process.

interaction degree for checking the validity of the formation of each cluster grouped. Through the use of numerical representation of relationships between goals and use cases, the relationships among goals can be more easily processed computationally.

3) To ease up the discovery of early aspects in the early stage of software development, an ArgoUML-based supporting tool is developed and integrated with MapReduce and HBase as a means to improve the performance of the proposed clustering algorithm. Adopting MapReduce enhances the clustering algorithm by executing grouping procedures concurrently. Using HBase helps manage the storage of intermediate data and final results generated by Mappers during the grouping of goals.

There are four phases in the Goal-driven Early Aspect process (see Fig. 1): formulation, construction, classification, and identification, to facilitate the discovery of early aspects. In formulation, requirements are analyzed in order to identify and formulate goals based on the concept of goal structure. In construction, a goal-driven use case model is established together with scenarios of use cases, side effects and a table of relationships among goals and use cases. Noted that a more fine-grained goal-driven use case model is obtained in an iterative manner. In classification, relationships between goals and use cases are evaluated. Our ArgoUML-based supporting tool generates the following artifacts: similarity degrees, interaction degrees, goal clusters, and the frequency of occurrence of goal clusters. Finally, in identification, a threshold is set for determining which of these goal clusters will be treated as early aspectual candidates, and to be identified as early aspects by a 4-step guideline suggested in the proposed approach.

As pointed out in [26], using a benchmark helps examine the research contribution rigorously, and improve the tools and technique being developed. In this work, the meeting scheduler system [27] is chosen as an illustrative example throughout this paper to demonstrate the proposed approach as it has been adopted as a benchmark problem in requirements engineering [28]. In addition to the meeting scheduler system, the proposed approach is also applied to another application domain, a course enrollment system developed at National Central University to further illustrate the feasibility of the proposed approach.

An experiment is conducted to evaluate the benefits of the proposed approach by inviting three groups of people, including 15 college professors (P group), 15 graduate students (S group) and 15 software engineers from a software R&D institute in Taiwan (E group), to a survey: P and S groups apply the GEA to the course enrollment system to discover early aspects, while the E group discovers early aspects without using the GEA. The result of the survey is further validated through Mann-Whitney U-test [29] to show that the difference between with GEA and without GEA is statistically significant.

In the rest of the paper, we first discuss how to identify goals through the use of an extended goal structure based on our previous work in Section 2. In Section 3, details on how to discover early aspects by goals interactions and the clustering algorithm are fully described. The course enrollment system is illustrated in Section 4 along with an experiment to further illustrate the benefits of the proposed approach. Related work on early aspects are discussed in Section 5. Finally, we conclude by outlining benefits of this work and our future research plan in Section 6.

2 Goal-Driven Use Case Model

As a starting point to identify early aspects, it is crucial to clarify the relationships among system functional and non-functional requirements. Use case driven analysis focuses the expression of requirements on users, beginning with the perspective that a system is built first and foremost for its users, which offers an important benefit that helps manage complexity as it focuses on one specific usage at a time. Our previous work [22], [24], goal-driven use case

model is developed upon the benefit of use case modeling to address the interactions among goals and use cases to provide valuable information in identifying, organizing and justifying software requirements, and are served as a background knowledge for the discovery of early aspects in this work.

2.1 Formulation: Goal Identification and Formulation

Goals identification plays a pivotal role in the elicitation of software requirements. In [30], C. Rolland et al. proposed a goal structure to analyze the requirements based on a verb and its parameters. To better capture users' intention of a system, an extension to the goal structure [31] is used to make easy the capturing of software requirements. The extended goal structure is developed with two features: (1) classify a verb from two viewpoints: content and competence, to distinguish different types of requirements based on the notion of requirements satisfiability, that is, a requirement that needs to be satisfied utterly or can be satisfied to a certain degree; and (2) add two new types of parameters: view and constraints, to offer separate views in the analysis of the requirements.

In the extended goal structure, a goal is expressed as a clause with a verb and a number of parameters, where each parameter plays a different role with respect to the verb. To be more specific, the verb used in a requirements document pinpoints a way that helps developers identify the types of the goals. For example, in a requirements document, if it states "Initiator plans a meeting with date and location by asking participants." A goal can then be identified by using its verb "plan" to depict that the system should provide a function for an initiator to plan a meeting.

In terms of the parameters, there are four types in the extended goal structure: *view*, *target*, *direction*, and *constraints*. The *view* concerns whether a goal is actor-specific or system-specific. An *actor-specific* view is an objective of an external entity that uses a system, meanwhile, a *system-specific* view is a requirement for the services that a system provides. *Target* is an entity affected by a goal and can be further distinguished into two types: *object* and *results*. An *object* is supposed to exist before a goal is achieved. *Results* can be of two kinds: (1) entities that do not exist before a goal is achieved, or (2) abstract entities that exist but are made concrete as a result of achieving a goal. The two types of *directions*: *source* and *destination*, identify the direction of the action, which is to or from the objects to be communicated with, respectively. *Constraints* represent the *pre-/post-condition* that must be satisfied before or after achieving a goal or *Invariant* that stands for conditions that always hold before and after achieving a goal.

A goal is thus represented as follows:

Action: [Actor, Target, Source, Destination, Condition, Competence].

Action is the verb from requirements documents that a goal intends to achieve, which could be either functional or non-functional. *Actor* represents *view* and refers to people who use this system or the system itself that performs the action. *Target* is an entity affected by the goal. *Source*, *destination*, and *condition* are optional. *Source* and *destination* are both related to *direction* to indicate the initial and final locations of objects.

Condition represents the *constraints* and serves two purposes: to describe the situation prior to or after performing an action and the invariant that the system must keep before and after performing the action. *Competence* can be either rigid or soft to show whether the goal must be satisfied utterly or to some extent.

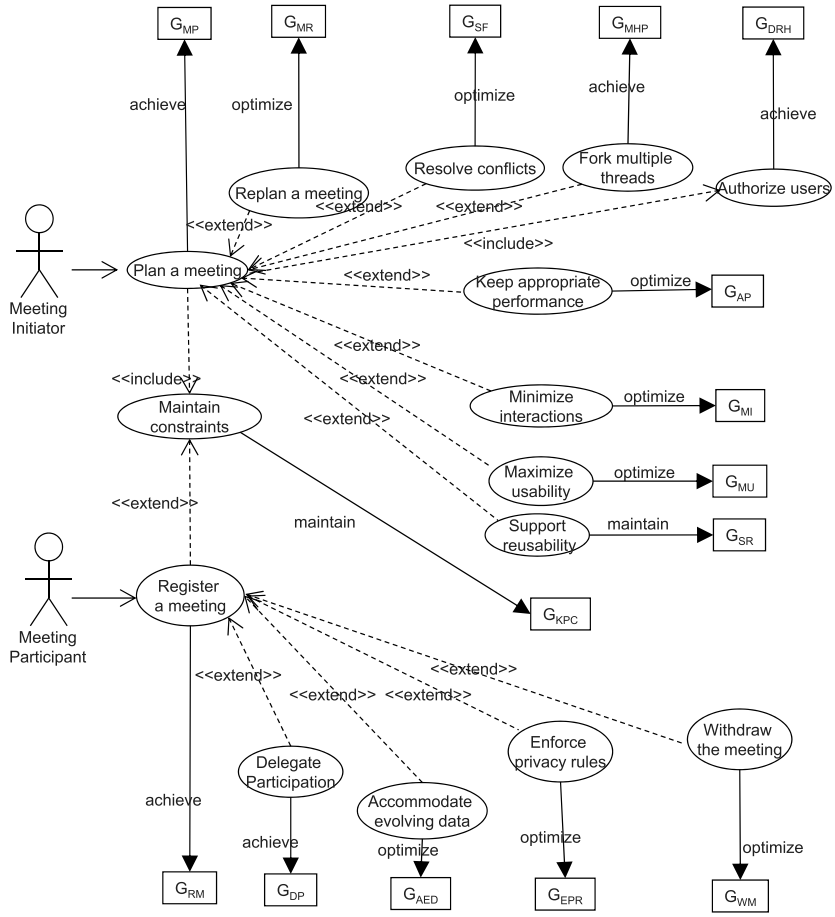
In the meeting scheduler system, 15 goals are identified by applying the extended goal structure below:

- G_{MP}: Plan: [initiator, meeting date and location, initiator, participants, $\emptyset$, rigid]
- G_{MR}: Replan: [initiator, meeting date and location, initiator, participants, support flexibility, soft]
- G_{SF}: Support: [system, conflicts resolution, initiator, participants, support flexibility, soft]
- G_{MI}: Manage: [system, interactions, participants, participants, as small as possible, soft]
- G_{MHP}: Handle: [system, plan meetings, initiator, $\emptyset$, in parallel, rigid]
- G_{DRH}: Accommodate: [system, decentralized requests, initiator, $\emptyset$, been authorized, rigid]
- G_{KPC}: Maintain: [system, physical constraints, $\emptyset$, $\emptyset$, not to be broken, soft]
- G_{AP}: Provide: [system, performance, $\emptyset$, $\emptyset$, an appropriate level, soft]
- G_{RM}: Register: [participant, a meeting, participant,$\emptyset$, $\emptyset$, rigid]
- G_{DP}: Delegate: [participant, participation, participant, participants, $\emptyset$, rigid]
- G_{AED}: Accommodate: [participant, evolving data, participant, $\emptyset$, $\emptyset$, soft]
- G_{WM}: Withdraw: [participant, the meeting, participant, $\emptyset$, $\emptyset$, soft]
- G_{EPR}: Enforce: [system, privacy rules, participants, $\emptyset$, $\emptyset$, soft]
- G_{SR}: Support: [system, reusability, $\emptyset$, $\emptyset$, $\emptyset$, soft]
- G_{MU}: Maximize: [system, usability, $\emptyset$, $\emptyset$, non-experts, soft]

It is noted that not all verbs in a requirements document are identified as goals. The extended goal structure is provided as a guideline to assist developers to capture software requirements based on verbs in the requirements document. It is up to the developers to identify verbs as goals directly from the requirements document or to paraphrase the requirements document for identifying goals.

2.2 Construction: Goal-Driven Use Case Diagram

After goals are identified, a goal-driven use case diagram can then be established. In the goal-driven use case approach [24], each use case is viewed as a process that can be associated with a goal to be achieved, optimized, or maintained by the use cases. To start with, original use cases guaranteeing that a target system meets the minimum requirements are first addressed. Each original use case is associated with an actor to describe the process to achieve an original goal that is rigid, actor-specific, and functional. Building original use cases by investigating all original goals makes the use case model satisfy at least all actors' rigid goals. To extend the original use cases to take into



Acronyms	Goal Names
G_{MP}	Meeting Planned
G_{MR}	Meeting Re-planned
G_{SF}	Support Flexibility
G_{MI}	Minimal Interactions
G_{MHP}	Meeting Handle in Parallel
G_{DRH}	Decentralized Requests Handled
G_{KPC}	Keep Physical Constraints
G_{AP}	Appropriate Performance
G_{RM}	Register a Meeting
G_{DP}	Delegate Participation
G_{AED}	Accommodate Evolving Data
G_{WM}	Withdraw from a Meeting
G_{EPR}	Enforce Privacy Rules
G_{SR}	Support Reusability
G_{MU}	Maximal usability

Fig. 2. Goal-driven use case model of meeting scheduler system.

account various types of goals, extension use cases are created for various situations: optimizing/maintaining a soft goal, achieving a system-specific goal, or achieving a non-functional goal.

Refer to the meeting scheduler system, use cases are established according to the system goals identified, and the goal-driven use case diagram is then constructed as shown in Fig. 2. A detail discussion of our goal-driven use case model can be found in [24].

3 Early Aspects Discovery

An early aspect is a crosscutting concern that scatters over various modules or components in the early life cycle phase of a software system [8], [21]. It is pinpointed out in [32] that identifying clusters that aggregate these modules or components in spotting early aspects can be beneficial for software development. As a continuation of our previous work on goal-driven use cases model [22], [24], [33] and early aspects identification method [33], we focus our attention in this work, called Goal-driven Early Aspect, on the improvement of the following three main features: reduce the time complexity from $O(n^n)$ to $O(n!)$ of our clustering algorithm, streamline the GEA process as shown in Fig. 1, and enhance the supporting tool by integrating MapReduce and HBase for increasing the performance of our clustering algorithm and an ArgoUML plug-in for drawing or importing use case diagrams. In addition to the above-mentioned features, an experiment to evaluate the benefits of the proposed approach is also conducted.

Fig. 3 illustrates the concept of goals with and without crosscutting concerns. In GDUC modeling, each use case is viewed as a process that can be associated with a goal to be achieved, ceased, impaired, optimized, or maintained by the use case. In addition to the direct associated goal, a use case can also have side effects to achieve, cease, impair, optimize or maintain other goals in the target system. Consequently, if goals G_1 and G_2 have crosscutting concerns, these goals will share common properties influenced by the

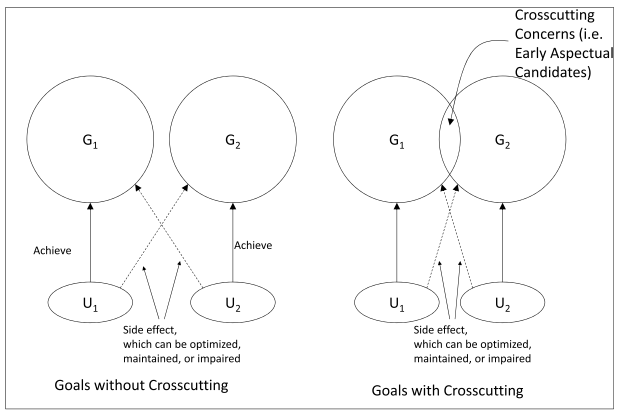


Fig. 3. Illustration of the concepts of goals, use cases and crosscutting concerns.

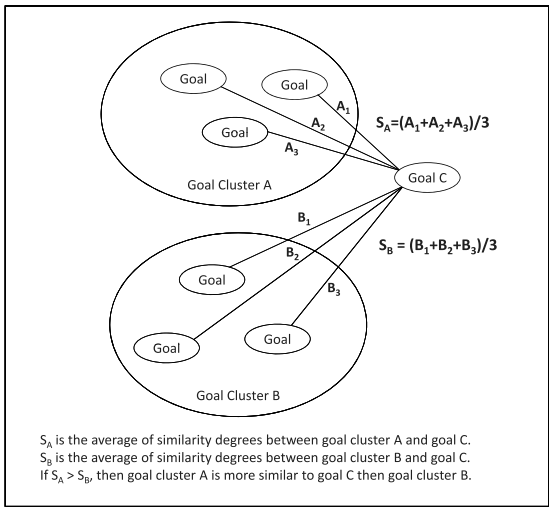


Fig. 4. Illustration of the concept of similarity.

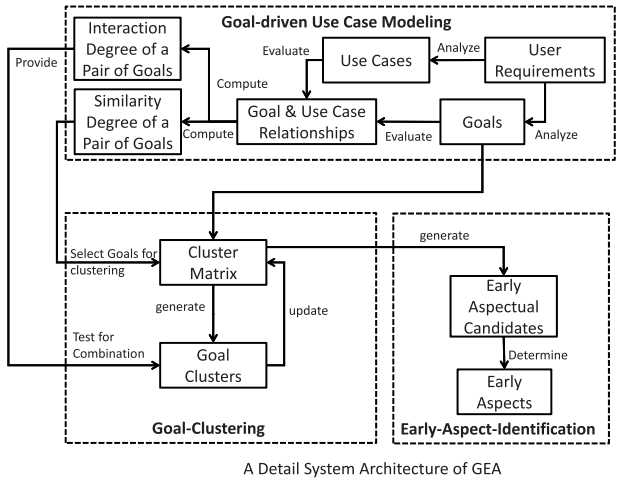


Fig. 5. GEA system architecture.

same set of effects that are contributed by their corresponding use cases, and behave similarly as shown on the right hand side of Fig. 3. Therefore, it is our aim to find out those goals that share common properties for discovering early aspects. In the proposed approach, the discovery of early aspects is achieved by a clustering algorithm to locate groups of goals in which goals in each group response to a set of use cases in a similar way.

The grouping of a goal into a goal cluster is determined by the values measured by the goal's similarity to the goals in the cluster (see S_A and S_B in Fig. 4). The higher a similarity degree from a goal cluster to a goal, the higher the likelihood that the goal behaves similarly to the goals in the goal cluster. Fig. 4 depicts the concept of similarity used in the proposed approach, in which the similarity degree is measured by subtracting the summation of the difference between the score of goals wrt a designated use case by 1 (detail definition can be found in Section 3.3.) In this work, the relationships among goals such as interaction and similarity are represented numerically to make easy the processing of all the relations computationally, and to facilitate the discovery of early aspects by investigating the common properties shared by goals.

In what follows, to further elaborate the GEA process, we first outline the system architecture of GEA to give an overview of the key components in this work in Section 3.1. The details on how to evaluate the relationships among use cases and goals, to obtain goal relationships, and to establish goal clusters are depicted in Sections 3.2, 3.3, and 3.4, respectively. The design of our ArgoUML-based supporting tool that eases up the discovery of early aspects is described in Section 3.5. In Section 3.6, the identification of the early aspectual candidates is presented along with a 4-step guideline for determining early aspects. Finally, the comparison of our previous work and the proposed approach is discussed in Section 3.7.

3.1 System Architecture of Aspect-Enhanced GDUC

In GEA (see Fig. 5), there are three subsystems, including Goal-driven Use Case Modeling, Goal-Clustering, and Early-Aspect-Identification. User requirements are analyzed and modeled as goals and use cases to enable the evaluation of goal and use case relationships, which are further transformed into interaction degree and similarity degree in a pairwise manner for all the goals in the Goal-driven Use Case Modeling subsystem. The interaction and similarity degrees are used as inputs to the Goal-Clustering subsystem where cluster matrices are established as a basis for generating goal clusters by means of similarity degree, whilst the interaction degree is served as a validation criteria for the formation of each goal cluster generated. In the end of the clustering algorithm, early aspectual candidates are high-lighted in a frequency-based manner for developers to identify early aspects in Early-Aspect-Identification.

In Goal-driven Use Case Modeling, the user requirements are analyzed and expressed as goals by using the goals structure in a six-tuple form. Use cases are established to achieve/optimize these goals. Relationships between goals and use cases are characterized as predicates: satisfied, satisfiable, denied, deniable, and independent, to describe the effects of use cases on goals. The computation of the relationships among goals are re-formulated by changing the predicates from qualitative terms, such as fully satisfied, largely satisfied, partially satisfied, not affected, partially denied, largely denied, and fully denied, to numerical representation of relations among goals ranging from -5 to 5 as shown in Table 1.

Interaction degree and similarity degree are calculated in a pairwise manner. The interaction degree of a pair of goals is defined as a fuzzy union of cooperative and conflicting degrees between goals. Meanwhile, the similarity degree of a pair of goals is defined as the summation of the satisfying and denying degrees that each use case contributes to the two goals in each pair. The larger the result of summation, the more different the two goals are.

In Goal-Clustering, the clustering algorithm begins with initializing a cluster matrix of goals, which is followed by selecting goals to be grouped into a goal cluster with the use of similarity degrees, and finally to validate the formation of each grouping by using interaction degree. A goal cluster groups goals/goal clusters with the highest similarity degree. The grouping of goals/goal clusters is then validated with the interaction degree, which is defined as the

TABLE 1
Relationships between Use Cases and Goals

Score	Explanation
5	The goal is fully satisfied after the use case is performed
3	The goal is largely satisfied after the use case is performed
1	The goal is partially satisfied after the use case is performed
0	The goal is not affected after the use case is performed
-1	The goal is partially denied after the use case is performed
-3	The goal is largely denied after the use case is performed
-5	The goal is fully denied after the use case is performed
2, 4, -2, -4	Represent the degrees between scores listed above

sum of interaction degrees between goals in a goal cluster. That is, the interaction degree of the to-be-grouped goal cluster is compared with goals/goal clusters before actually performing the grouping procedure to serve as a criteria for the validation of the formation of the clustering. In the end of the clustering, if there is no more grouping procedure can be processed, goal clusters can then be obtained.

In Early-Aspect-Identification, as a result of each round of the clustering, it is possible that multiple groupings can occur due to a same similarity degree, which causes each of these groupings to continue to derive its own cluster matrices with respect to each round of the clustering until there is no more goals/goal clusters to be grouped. In the end, it is likely that there will be multiple groupings of goals/goal clusters. Early aspectual candidates can be identified based on the frequency of the occurrence of a specific goal cluster appearing in all groupings. It is up to the developers to set a threshold to determine how strong the likelihood is of an early aspectual candidate containing an early aspect. The higher the threshold set by the developers, the less the number of early aspects we will obtain, and vice versa.

3.2 Classification: Evaluate the Relationships among Use Cases and Goals

The first step in the classification phase is to evaluate the relationships among goals and use cases, in which the effect of performing a use case to its directly associated goal, including achieved, ceased, impaired, optimized, realized, or maintained, and to all other goals, called side effects, are considered.

A pairwise evaluation of how a use case affects a goal is adopted. The evaluation of the achievement of a goal is rated from -5 to 5 to represent the degree to which the goal is achieved while performing a use case. The score can be given by following Table 1 as suggested in Satty's work [34]. A similar scoring strategy is also adapted in Brito's work [35] as Satty's scale is based on psychological theories and experiments that point to the use of nine unit scales as a reasonable set that allows humans to perform discrimination between preferences for two items. In Table 1, 5 means the goal can be fully satisfied by a use case; -5 means the goal is fully denied by a use case; and 0 means a use case does not have any effect on the goal. Details of the rating of satisfaction degree can be found in Table 1.

TABLE 2
Numerical Score for the Relationships between Goals and Use Cases of the Meeting Scheduler System

	G_{MP}	G_{MR}	G_{SF}	G_{MI}	G_{MHP}	G_{DRH}	G_{KPC}	G_{AP}	G_{RM}	G_{DP}	G_{AED}	G_{WM}	G_{EPR}	G_{SR}	G_{MU}
U_{PAM}	5	4	0	0	0	2	2	0	0	0	0	0	0	1	1
U_{RAM}	4	5	3	-2	0	0	0	-2	0	0	0	0	0	0	0
U_{RC}	3	3	5	-3	0	0	0	-2	0	0	0	0	0	0	0
U_{MI}	0	-2	-2	5	0	0	0	3	0	0	0	0	0	0	0
U_{FMT}	0	0	2	0	5	4	0	-1	0	0	0	0	0	0	0
U_{AU}	0	1	2	0	4	5	0	-1	1	0	0	0	0	2	0
U_{MC}	2	3	3	-1	0	0	5	-3	1	0	0	0	0	2	0
U_{KAP}	2	-2	-3	0	0	0	0	5	0	0	0	0	0	-2	-1
U_{RM}	0	0	0	0	0	0	0	1	5	2	2	2	2	1	1
U_{DP}	0	0	0	0	0	0	0	0	1	5	1	0	0	0	0
U_{AED}	0	0	0	0	0	0	0	0	1	1	5	0	0	0	0
U_{WM}	0	0	0	0	0	0	0	0	1	0	0	5	0	0	0
U_{EPR}	0	0	0	0	0	0	0	0	1	0	0	0	5	0	0
U_{SR}	0	0	0	0	0	0	0	-2	0	0	0	0	0	5	-1
U_{MU}	0	0	0	0	0	0	0	-1	0	0	0	0	0	-1	5

To better model the relationships among use cases and goals, two membership functions are proposed for representing the satisfying and denying degrees of goals with respect to use cases from two viewpoints: satisfying and denying. They are defined as

Definition 1. *Let $u_{Sat(U_i,G_j)}(x)$ be a membership function for describing the satisfying degree with respect to a rating x of the base set X (i.e., from -5 to 5). Then,*

$$u_{Sat(U_i,G_j)}(x) = \begin{cases} 0, & \text{if } x < 0; \\ 0.2x, & \text{if } x \geq 0. \end{cases}$$

where $u_{Sat(U_i,G_j)}(x)$ represents the degree that goal G_j is satisfied by use cases U_i wrt score x and $0 \leq u_{Sat(U_i,G_j)}(x) \leq 1$.

Definition 2. *Let $u_{Den(U_i,G_j)}(x)$ be a membership function for describing the denying degree with respect to a rating x of the base set X (i.e., from -5 to 5). Then,*

$$u_{Den(U_i,G_j)}(x) = \begin{cases} 0.2x, & \text{if } x \leq 0; \\ 0, & \text{if } x > 0. \end{cases}$$

where $u_{Sat(U_i,G_j)}(x)$ represents the degree that goal G_j is denied by use cases U_i wrt score x and $0 \leq u_{Den(U_i,G_j)}(x) \leq 1$.

In the meeting scheduler system, the relationships among goals and use cases are evaluated in a pairwise manner based on Table 1. For example, goal G_{MP}($meeting\ planned$) not only can be achieved by the use case U_{PAM}($plan\ a\ meeting$), but can also be achieved by the use case U_{RAM}($replan\ a\ meeting$); therefore, a score of 4 is given to indicate that the goal G_{MP} can be largely to fully satisfied by the side effect of performing U_{RAM}. By applying the membership functions, we can obtain the satisfying degree between G_{MP} and U_{RAM} as $u_{Sat(U_{RAM},G_{MP})}(4) = 0.8$ and denying degree $u_{Den(U_{RAM},G_{MP})}(4) = 0$, which is paired as a tuple $-(u_{Sat(U_{RAM},G_{MP})}(4), u_{Den(U_{RAM},G_{MP})}(4)) = (0.8, 0)$. Another example is that goal G_{MI} ($min.\ interactions$) is partially to largely denied by use case U_{RAM}($replan\ a\ meeting$) since replanning a meeting increases the frequency of communication among participants, which results in a negative number of rating, -2. The result of the evaluation of relationships is shown in Table 2. By applying the membership functions to the

TABLE 3
A Tuple Form Representation of Relationships between Goals and Use Cases of the Meeting Scheduler System

	G_{MP}	G_{MR}	G_{SF}	G_{MI}	G_{MHP}	G_{DRH}	G_{KPC}	G_{AP}	G_{RM}	G_{DP}	G_{AED}	G_{WM}	G_{EPR}	G_{SR}	G_{MU}
U_{PAM}	(1, 0)	(.8, 0)	(0, 0)	(0, 0)	(0, 0)	(.4, 0)	(.4, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(.2, 0)	(.2, 0)
U_{RAM}	(.8, 0)	(1, 0)	(.6, 0)	(0, .4)	(0, 0)	(0, 0)	(0, 0)	(0, .4)	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 0)
U_{RC}	(.6, 0)	(.6, 0)	(1, 0)	(0, .6)	(0, 0)	(0, 0)	(0, 0)	(0, .4)	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 0)
U_{MI}	(0, 0)	(0, .4)	(0, .4)	(1, 0)	(0, 0)	(0, 0)	(0, 0)	(.6, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 0)
U_{FMT}	(0, 0)	(0, 0)	(.4, 0)	(0, 0)	(1, 0)	(.8, 0)	(0, 0)	(0, .2)	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 0)
U_{AU}	(0, 0)	(.2, 0)	(.4, 0)	(0, 0)	(.8, 0)	(1, 0)	(0, 0)	(0, .2)	(.2, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(.4, 0)	(0, 0)
U_{MC}	(.4 0)	(.6 ,0)	(.6, 0)	(0, .2)	(0, 0)	(0, 0)	(1, 0)	(0, .6)	(.2, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(.4, 0)	(0, 0)
U_{KAP}	(.4 ,0)	(0, .4)	(0, .6)	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(1, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(0, .4)	(0, .2)
U_{RM}	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(.2, 0)	(1, 0)	(.4, 0)	(.4, 0)	(.4, 0)	(.4, 0)	(.2, 0)	(.2, 0)
U_{DP}	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(.2, 0)	(1, 0)	(.2, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 0)
U_{AED}	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(.2, 0)	(.2, 0)	(1, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 0)
U_{WM}	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(.2, 0)	(0, 0)	(0, 0)	(1, 0)	(0, 0)	(0, 0)	(0, 0)
U_{EPR}	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(.2, 0)	(0, 0)	(0, 0)	(0, 0)	(1, 0)	(0, 0)	(0, 0)
U_{SR}	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(0, .4)	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(1, 0)	(0, .2)
U_{MU}	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(0, .2)	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(0, .2)	(1, 0)

relationships among goals and use cases in Tables 2, and 3 can be obtained in a tuple form for each entry.

3.3 Classification: Obtain Goals Relationships

In order to facilitate the clustering of goals, two factors are further explored: one is the similarity degree of goals for grouping a goal clusters, and the other is the interaction degree for evaluating the validity of the grouping procedure of goals from a system-wise point of view, where cooperative and conflicting degrees are introduced at use case level to serve as a basis for measuring interaction degrees among goals at the system level.

It is noted that the larger the summation of the differences between the scores of two goals wrt a designated use case, the more different the two goals are. Therefore, the function to represent the similarity degree between two goals is defined by subtracting the summation by 1 (see Definition 3 below).

Definition 3. *Let Similarity(G_i, G_j) be a function for representing the similarity degree between two goals: Then,*

$$Similarity(G_i, G_j) = 1 - \left(\sum_{k=0}^{n} \left(\left| u_{Sat(U_k,G_i)}(x) - u_{Sat(U_k,G_j)}(x) \right| + \left| u_{Den(U_k,G_i)}(x) - u_{Den(U_k,G_j)}(x) \right| \right) \right) / S_{i,j},$$

where $i \neq j$, n is the number of use cases in the system, $-5 \leq x \leq 5$, and $S_{i,j}$ is an adjusting factor, which is the counts of satisfying/denying degrees of G_i and G_j with respect to U_k that are not equal to 0.

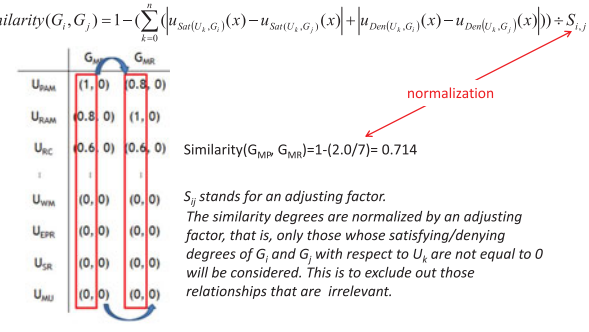


Fig. 6. Illustration of computing similarity degree.

TABLE 4
Similarity Degrees among Goals of the Meeting Scheduler System

	G_{MP}	G_{MR}	G_{SF}	G_{MI}	G_{MHP}	G_{DRH}	G_{KPC}	G_{AP}	G_{RM}	G_{DP}	G_{AED}	G_{WM}	G_{EPR}	G_{SR}	G_{MU}
G_{MP}	x	0.71	0.5	0.1	0.29	0.34	0.4	0.4	0.55	0.4	0.4	0.34	0.34	0.47	0.43
G_{MR}	0.71	x	0.7	0.11	0.33	0.38	0.51	0.25	0.55	0.44	0.44	0.4	0.4	0.56	0.5
G_{SF}	0.5	0.7	x	0.11	0.4	0.43	0.48	0.18	0.55	0.44	0.44	0.4	0.4	0.6	0.51
G_{MI}	0.1	0.11	0.11	x	0.33	0.37	0.28	0.68	0.56	0.46	0.46	0.4	0.4	0.5	0.56
G_{MHP}	0.29	0.33	0.4	0.33	x	0.73	0.2	0.4	0.55	0.32	0.32	0.2	0.2	0.53	0.49
G_{DRH}	0.34	0.38	0.43	0.37	0.73	x	0.3	0.42	0.56	0.37	0.37	0.28	0.28	0.53	0.49
G_{KPC}	0.4	0.51	0.48	0.28	0.2	0.3	x	0.49	0.6	0.4	0.4	0.3	0.3	0.57	0.53
G_{AP}	0.4	0.25	0.18	0.68	0.4	0.42	0.49	x	0.57	0.55	0.55	0.53	0.53	0.44	0.53
G_{RM}	0.55	0.55	0.55	0.56	0.55	0.56	0.6	0.57	x	0.69	0.69	0.66	0.66	0.65	0.67
G_{DP}	0.4	0.44	0.44	0.46	0.32	0.37	0.4	0.55	0.69	x	0.47	0.45	0.45	0.56	0.57
G_{AED}	0.4	0.44	0.44	0.46	0.32	0.37	0.4	0.55	0.69	0.47	x	0.45	0.45	0.56	0.57
G_{WM}	0.34	0.4	0.4	0.4	0.2	0.28	0.3	0.53	0.66	0.45	0.45	x	0.33	0.53	0.53
G_{EPR}	0.34	0.4	0.4	0.4	0.2	0.28	0.3	0.53	0.66	0.45	0.45	0.33	x	0.53	0.53
G_{SR}	0.47	0.56	0.6	0.5	0.53	0.53	0.57	0.44	0.65	0.56	0.56	0.53	0.53	x	0.51
G_{MU}	0.43	0.5	0.51	0.56	0.49	0.49	0.53	0.53	0.67	0.57	0.57	0.53	0.53	0.51	x

In the definition of similarity function, the computation of the similarity degrees is normalized, that is, only those whose satisfying/denying degrees of G_i and G_j with respect to U_k are not equal to 0 will be considered. This is to exclude out those relationships that are rated as irrelevant (i.e., the goal is not affected after the use case is performed). For example (see Fig. 6), to compute the similarity degree of G_{MP} and G_{MR}, $Similarity(G_{MP}, G_{MR}) = 1 - (2.0/7) = 0.714$, where 7 is the adjusting factor to indicate the counts of satisfying/denying degrees of G_{MP} and G_{MR} with respect to use cases whose values are not equal to 0.

The pair-wise similarity degrees among goals in the meeting scheduler system in shown in Table 4, in which the larger the value is, the more similar the two goals are. For example, G_{MR} is more similar to G_{MP} ($Similarity(G_{MR}, G_{MP}) = 0.71$) than G_{SF} ($Similarity(G_{SF}, G_{MP}) = 0.5$) is.

To compute the interaction degree between two goals, two kinds of interaction relationships, cooperative and conflicting degrees, are introduced at the use case level to serve as a basis for measuring interaction degrees among goals at the system level. Definitions 4 and 5 show the definitions of the two degrees.

Definition 4. *Let $Cooperative_{U_k}(G_i, G_j)$ be a function for representing the cooperative degree of two goals G_i and G_j wrt a designated use case U_k. Then,*

$$Cooperative_{U_k}(G_i, G_j) = (u_{Sat(U_k,G_i)}(x) \cap u_{Sat(U_k,G_j)}x(x)) \cup (u_{Den(U_k,G_i)}(x) \cap u_{Den(U_k,G_j)}(x)),$$

where $\cap$ stands for fuzzy AND representing the intersection operation, $\cup$ stands for fuzzy OR representing the union operation, and $-5 \leq x \leq 5$.

Definition 5. *Let $Conflicting_{U_k}(G_i, G_j)$ be a function for representing the conflicting degree of two goals G_i and G_j wrt a designated use case U_k. Then,*

$$Conflicting_{U_k}(G_i, G_j) = (u_{Sat(U_k,G_i)}(x) \cap u_{Den(U_k,G_j)}(x)) \cup (u_{Den(U_k,G_i)}(x) \cap u_{Sat(U_k,G_j)}(x))$$

where $\cap$ stands for fuzzy AND representing the intersection operation, $\cup$ stands for fuzzy OR representing the union operation, and $-5 \leq x \leq 5$.

TABLE 5
Interaction Degrees among Goals of the Meeting Scheduler System

	G_{MP}	G_{MR}	G_{SF}	G_{MI}	G_{MHP}	G_{DRH}	G_{KPC}	G_{AP}	G_{RM}	G_{DP}	G_{AED}	G_{WM}	G_{EPR}	G_{SR}	G_{MU}
G_{MP}	x	0.4	0.2	-0.6	0	0.4	0.4	0	0.2	0	0	0	0	0	0
G_{MR}	0.4	x	0.6	-0.6	0.2	0.4	0.6	-0.6	0.2	0	0	0	0	0.4	0.2
G_{SF}	0.2	0.6	x	-0.6	0.4	0.4	0.6	-0.6	0.2	0	0	0	0	0.4	0.2
G_{MI}	-0.6	-0.6	-0.6	x	0	0	-0.2	0.6	-0.2	0	0	0	0	-0.2	0
G_{MHP}	0	0.2	0.4	0	x	0.8	0	-0.2	0.2	0	0	0	0	0.4	0
G_{DRH}	0.4	0.4	0.4	0	0.8	x	0.4	-0.2	0.2	0	0	0	0	0.4	0.2
G_{KPC}	0.4	0.6	0.6	-0.2	0	0.4	x	-0.6	0.2	0	0	0	0	0.4	0.2
G_{AP}	0	-0.6	-0.6	0.6	-0.2	-0.2	-0.6	x	0	0.2	0.2	0.2	0.2	-0.2	0
G_{RM}	0	0.2	0.2	-0.2	0.2	0.2	0.2	0	x	0.4	0.4	0.4	0.4	0.2	0.2
G_{DP}	0	0	0	0	0	0	0	0.2	0.4	x	0.4	0.4	0.4	0.2	0.2
G_{AED}	0	0	0	0	0	0	0	0.2	0.4	0.4	x	0.4	0.4	0.2	0.2
G_{WM}	0	0	0	0	0	0	0	0.2	0.4	0.4	0.4	x	0.4	0.2	0.2
G_{EPR}	0	0	0	0	0	0	0	0.2	0.4	0.4	0.4	0.4	x	0.2	0.2
G_{SR}	0	0.4	0.4	-0.2	0.4	0.4	0.4	-0.2	0.2	0.2	0.2	0.2	0.2	x	0
G_{MU}	0	0.2	0.2	0	0	0.2	0.2	0	0.2	0.2	0.2	0.2	0.2	0	x

The interaction degree between two goals at the system level is then defined as the difference between the sum of cooperative degrees and that of conflicting degrees (see Definitions 4 and 5) with respect to all use cases in a target system, which can be obtained by applying fuzzy union operation to the cooperative and conflicting degrees with respect to all use cases and is shown in Definition 6.

Definition 6. *Let $Sys(G_i, G_j)$ be a function for representing the interaction degree between two goals at the system level. Then,*

$$Sys(G_i, G_j) = \bigcup_{k=1}^{n} Cooperative_{U_k}(G_i, G_j) - \bigcup_{k=1}^{n} Conflicting_{U_k}(G_i, G_j),$$

where n is the number of use cases in the system.

Table 5 shows the interaction degrees between goals in the meeting scheduler system. In this table, the larger the value is, the higher cooperative level the two goals are. For example, G_{MP} has higher cooperative degree with respect to G_{MR} ($Sys(G_{MP}, G_{MR}) = 0.4$) than G_{SF} ($Sys(G_{MP}, G_{SF}) = 0.2$). Noted that Tables 4 and 5 are symmetric matrices since the relationship between any two goals is bidirectional.

3.4 Classification: Establish Goal Clusters

In developing a target system, it is usually desirable to aggregate goals with a high cooperative degree, which makes the system conform to the principle of high cohesion in software design. On the other hand, a goal cluster with a large number of goals would probably violate the principle of high cohesion, which may lead to a problematic design.

Based on this belief, the clustering begins with the checking of similarity degree to find out whether two goals should be grouped into a goal cluster, which is followed by the checking of total interaction degree of all goal clusters to validate the clustering.

In the clustering, the grouping procedure of goals/goal clusters is based on similarity degrees. To validate the clustering, a total interaction degree is proposed to control the progress of the grouping procedure. A total interaction degree is defined as a summation of interaction degrees for all goal clusters.

Definition 7. *Let $TotalInteractionDegree(systemstate)$ be a function for representing the total interaction degree as a summation of interaction degrees in all goal clusters. Then,*

$$TotalInteractionDegree(systemstate) = \sum_{k=1}^{m} \sum_{i=j+1}^{n} \sum_{j=1}^{n} Sys(G_i, G_j),$$

where m is the number of goal clusters in a systemstate, n is the number of goals in the kth goal cluster, and $i \neq j$. A system state is a snapshot of the grouping of goals.

The clustering algorithm and its activity diagram for discovering the aspectual candidates are described below.

Algorithm 1 (Clustering Algorithm)

1) Initialization
   a) Construct a similarity matrix (SM) and an interaction relation matrix (IR) for evaluating clustering of goals/goal clusters.
   b) Construct a starting clustering matrix CM_1 of size N * N, where N is the number of goals and all the goals are labelled on the Top-most row and Left-most column.
   c) Compute similarity degrees in CM_1 using SM.
   d) Create a queue (Q) for storing unprocessed CMs, and put CM_1 into Q.
2) Clustering
   a) Pick CM_i from Q. If Q is empty, then go to step 2f.
   b) Mark CM_i as processing, and find the highest similarity score goal pairs in CM_i, named H_j.
      i) If there is no more highest score, mark CM_i as finished and go to step 2a.
      ii) If there are more then one highest scores H_j, go to step 2d.
      iii) otherwise, go to step 2c.
   c) Examine whether the goal/goal cluster associated with the highest score H_j can be grouped.
      i) If yes, group the goals or goal clusters to form a new CM_{i+1}, and compute similarity degrees in CM_{i+1}. Then go to step 2b.
      ii) If no, set the highest score entries to zero in CM_i, then go to step 2b.
   d) If any one of the goals/goal clusters associated with H_j is to be grouped, group those goals/goal clusters to form new CMs and add them to Q. Then go to step 2a.
   e) If none of the goals/goal clusters associated with H_j is to be grouped, set the highest score entries to zero in CM_i, then go to step 2b.
   f) Goal clusters in CMs that marked as finished are the final result of the clustering algorithm.

Based on this algorithm, goals are grouped into goal clusters as a basis for discovering early aspects. Each goal cluster with a frequency indicates the odds of goals sharing a same common property, namely, the early aspect. Fig. 7 shows the activity diagram of the clustering algorithm for a better understanding of the process.

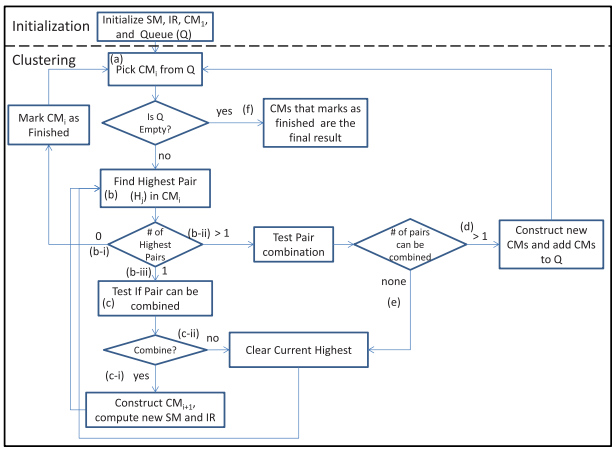


Fig. 7. Activity diagram of the clustering algorithm.

3.5 Design of the Supporting Tool

To ease up the discovery of early aspectual candidates, an ArgoUML-based supporting tool with MapReduce and HBase is developed to integrate our relationships evaluation mechanism plug-in into ArgoUML together with MapReduce programming model and HBase data sets. Thirty four newly added classes and three modified classes to ArgoUML with a total of 5,764 lines of code (LOC), and five more new classes with a total of 1,278 LOC are added for MapReduce.

The design of the supporting tool is shown in Fig. 8. The *UMLUseCaseDiagram* in the upper left corner of the figure represents the GDUC model that diagramed using ArgoUML and provides information needed in evaluating the relationship between goals and use cases, such as goal name, use case name, and association between goals and use cases. The relationship between goals and use cases represented by *GDUCRelation* (A) is determined by developers and is treated as an input source to the *ClusteringAlgorithm* (B) through *GDUCDirectReader* (C). The clustering algorithm then constructs two matrices, similarity matrix (*SimilarityMatrix*) (D) and interaction relation matrix (*InteractionRelation*) (E), based on the relationships determined by developers, which is stored in *UseCaseGoalRelationship*. In the clustering algorithm, the clustering matrix (CM) represented by *CompoundSimilarityMatrix* (F) continues grouping

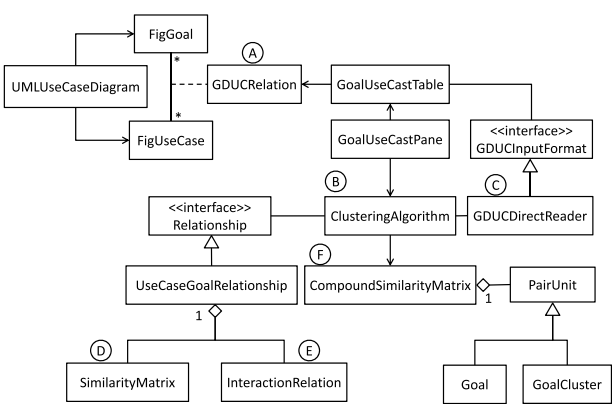


Fig. 8. Design of the supporting tool.

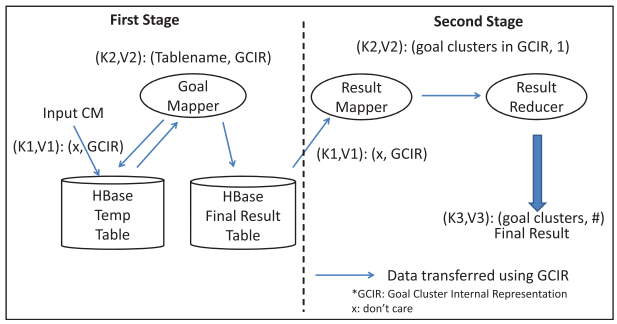


Fig. 9. Design of using MapReduce and HBase for the clustering algorithm.

goals/goal clusters to derive new clustering matrices (CMs) until there is no further round of clustering left to be processed.

To enhance the computation performance of the clustering algorithm, MapReduce programming model [36] is introduced to accelerate the process of all clustering matrices (*CMs*) in the algorithm. In MapReduce, each input is represented as a key-value pair and is denoted as $<K1, V1>$. A Mapper takes $<K1, V1>$ as its input and transforms the key-value pair into $<K2, V2>$, which is then handled by Reducers that reduce $<K2, V2>$ to $<K3, V3>$ to obtain the final results. The mapping of key-value pairs, from $<K1, V1>$ to $<K2, V2>$ and to $<K3, V3>$, is defined first in order to serve as a basis for establishing transformation rules in Mappers and Reducers, respectively. Furthermore, as these key-value pairs may occupied a vast amount of space, HBase [37] is also used to store intermediate data and final results, since HBase is designed for large data sets and its key-value storage machinery makes it decent for handling sparse data sets.

By combining MapReduce and HBase, we proposed a two-stage MapReduce processing model to handle the computation of the algorithm as shown in Fig. 9. The first stage focuses on performing the grouping procedure to group goals, and the second stage counts the occurrence of each goal cluster in the final results stored in HBase Final Result Table. To achieve this, two types of Mappers are developed, Goal Mapper and Result mapper, and two tables are created for storing outputs generated from the two Mappers respectively. One is HBase Temp Table for storing goals/goal/clusters in *CMs* that have not yet been fully grouped using goal cluster internal representation form, called GCIR, the other is the HBase Final Result Table for keeping the final grouping results.

The input Clustering Matrix (*CM*) is a matrix of size N * N, where N is the number of goals. To represent the matrix in the clustering algorithm using MapReduce, we construct a representation form called GCIR to represent goals and goal clusters in the clustering algorithm. GCIR represents goals and goal clusters in a numerical form such as $1, 2, 3, 4, \ldots$. Goals are separated by using letter **'P'** and goals in a same goal cluster are separated by using letter **'G'**. For example, if a *CM* has six goals, then we number these goals from one to six (i.e., g1, g2, ... , g6) and they can be represented as $1P2P3P4P5P6$.

During the process, GCIRs are transferred between Goal Mappers and HBase Temp Tables, in which Goal

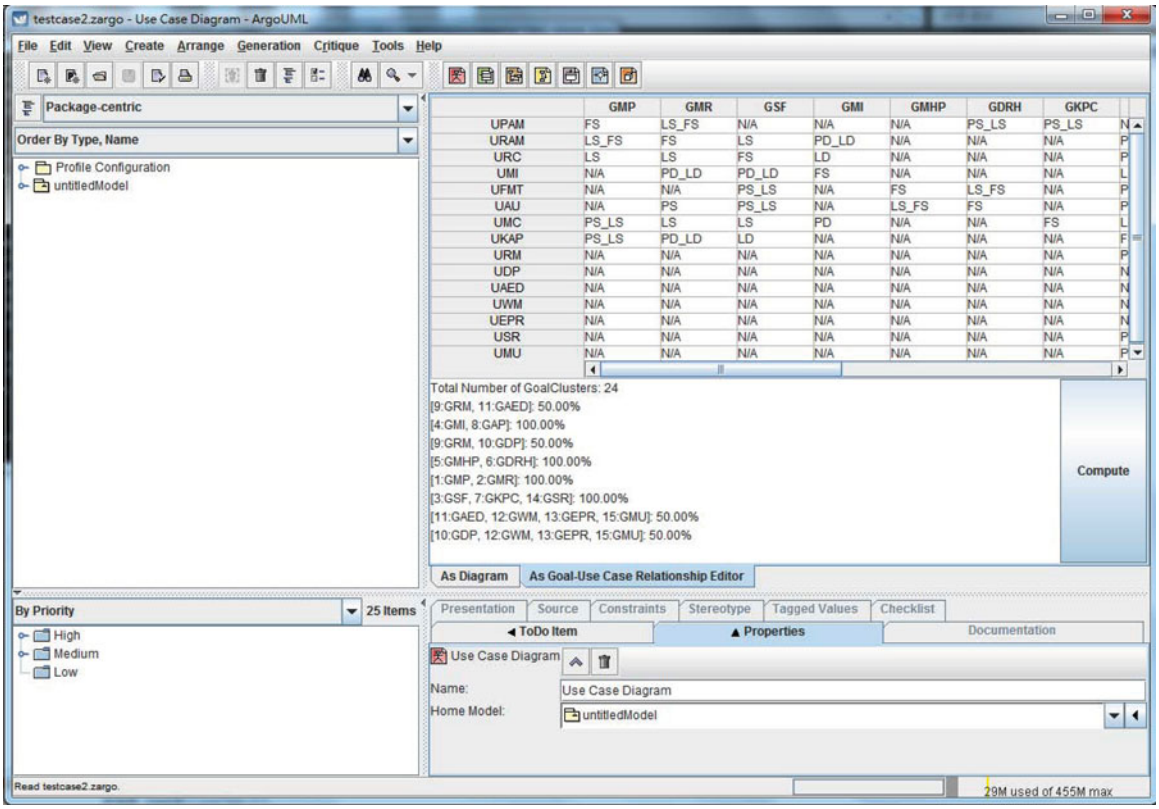


Fig. 10. User interface of editing the relationships among goals and use cases of the meeting scheduler system.

Mappers group goals into goal clusters represented in GCIR from HBase Temp Table and store results back to HBase Temp Tables using GCIR format. Goal Mapper takes CMs as its input and is denoted as $<X, GCIR>$, where X means don't care and outputs $<K2, V2>$ as $<Tablename, GCIR>$. If the process of a CM is finished, the output goes to the HBase Final Result Table, otherwise, it goes back to the HBase Temp Table and waits for further processing. For example, a clustering matrix of size 6 * 6 is to be processed, then it can be represented as $1P2P3P4P5P6$. If the goal mapper groups $G2$ and $G3$, then a new GCIR $1P2G3P4P5P6$ will be constructed as $V2$ and put back to HBase Temp Table.

Furthermore, if there is another possible grouping of goals, said $G2$ and $G4$, is to be grouped, then another GCIR $1P2P3G4P5P6$ will also be constructed as $V2$ and put back to HBase Temp Table, which causes a CM to generate multiple GCIRs (i.e., CMs) for further deviation. When a CM has been marked finished, it will be put into another table, the HBase Final result Table. Result Mapper reads data from the HBase Final Result Table and transforms it into $<K2, V2>$: $<GoalClustersinGCIR, 1>$ which is then further processed by Result Reducers to accumulate the total number of the occurrence of each goal clusters to obtain the final results, the occurrence frequency of each goal cluster grouped.

By applying MapReduce and HBase, CMs in the queue (see Fig. 7) can be processed in parallel by different Mappers, which improves the performance of the clustering algorithm. In other words, if the number of Mappers increases, the time needed to complete the clustering algorithm decreases.

Features of the extended ArgoUML tool include: draw use case diagrams and establish a relationship matrix. As shown in Fig. 10, the relationship of each pair of goal and use case determined by the developers is entered with a pull down menu containing all the possible relationships, such as fully satisfied, largely to fully satisfied, largely satisfied, partially to largely satisfied, partially satisfied, irrelevant, partially denied, partially to largely denied, largely denied, largely to fully denied, and fully denied. The evaluation of the effect from a use case to a goal is determined by developers and can be subjective. In the end of the computation in the clustering algorithm, a final result is shown directly under the relationship matrix along with the frequency of the occurrence of each goal cluster appearing in all derived system states. For example, a goal cluster followed by 50.00 percent on the screen in Fig. 10 indicates that goals with a total number of 100 final system states that are derived, goals in the goal cluster are grouped together for 50 times.

3.6 Identification: Identify Early Aspectual Candidates

Refer to the meeting scheduler system (see Fig. 10), by applying the clustering algorithm, eight goal clusters can be identified along with percentage frequency of occurrence, namely, (G_{RM}, G_{AED}): 50 percent, (G_{MI}, G_{AP}): 100 percent, (G_{RM}, G_{DP}): 50 percent, (G_{MHP}, G_{DRH}): 100 percent, (G_{MP}, G_{MR}): 100 percent, (G_{SF}, G_{KPC}, G_{SR}): 100 percent, (G_{AED}, G_{WM}, G_{EPR}, G_{MU}): 50 percent, and (G_{DP},G_{WM}, G_{EPR}, G_{MU}): 50 percent. Among these goal clusters, it is noted that some of the possibility values of the frequency of occurrence are

not equal to 100 percent, which suggests that developers should set a threshold to determine how strong the likelihood is of a goal cluster to become an early aspectual candidate.

If the threshold is set to 60 percent, then four goal clusters: (G_{RM}, G_{AED}), (G_{RM}, G_{DP}), (G_{AED}, G_{WM}, G_{EPR}, G_{MU}) and (G_{DP},G_{WM}, G_{EPR}, G_{MU}), will be filtered out. This will leave us only four early aspectual candidates in the meeting scheduler system, namely, (G_{MI}, G_{AP}), (G_{MHP}, G_{DRH}), (G_{MP}, G_{MR}), and (G_{SF}, G_{KPC}, G_{SR}).

To determine early aspects from early aspectual candidates, the following 4-step guideline is suggested:

1) Examine the description of goals in an early aspectual candidate for common objectives shared by these goals.
2) Check the description of use cases associated with goals in an early aspectual candidate (that is, a goal cluster) for similar behaviors shared by these use cases.
3) Inspect base courses and extension courses of use cases associated with goals in an early aspectual candidate for similar effects that achieve, maintain or optimize goals either directly or through side effects in the early aspectual candidate.
4) Consider the findings obtained in the above three steps, which can be served as a basis for common properties or important stakeholders' concerns and can be used to derive early aspects.

For example, G_{MHP} (Meeting Handle in Parallel) is achieved by performing use case "Fork multiple threads" and G_{DRH} (Decentralized Requests Handled) is achieved by performing use case "Authorize Users". These two goals are derived from the problem statements "The system must in general handle several meeting requests in parallel" and "The system accommodates decentralized requests from initiators that have been authorized". By reviewing the scenarios of the two use cases associated to G_{MHP} and G_{DRH}, it is found that they both address the issue of maximize the number of meeting planned. Therefore, an early aspect named "Max. Number of Meeting" is discovered to crosscut the two goals.

In the proposed approach, early aspects are denoted as

EarlyAspect: (UC_a, [$Goal_1$, $Goal_2$, . . . , $Goal_n$]),

where *EarlyAspect* denotes the name of an early aspect; UC_a denotes the use case that realizes the early aspect with a recurring property or important stakeholders' concerns that are derived from goals formulation; and [$Goal_1$, $Goal_2$, . . ., $Goal_n$] denotes the goals that have been crosscut by the early aspect.

The four early aspectual candidates in the meeting scheduler system, namely, $EarlyAspect_{AP}$ that crosscuts G_{MI} and G_{AP} through performing the use case "Keep Performance", $EarlyAspect_{MNMP}$ that crosscuts G_{MHP} and G_{DRH} through executing the use case "Max. Number of Meeting", $EarlyAspect_{FP}$ that crosscuts G_{MP}, and G_{MR} through performing the use case "Flexible Planning", and $EarlyAspect_{VPC}$ that crosscuts G_{SF}, G_{KPC} and G_{SR} through executing the use case "Verify Program Correctness" are denoted as

$EarlyAspect_{AP}$: (Keep Performance, [G_{MI}, G_{AP}])

$EarlyAspect_{MNMP}$: (Max. Number of Meeting, [G_{MHP}, G_{DRH}])

$EarlyAspect_{FP}$: (Flexible Planning, [G_{MP}, G_{MR}])

$EarlyAspect_{VPC}$: (Verify Program Correctness, [G_{SF}, G_{KPC}, G_{SR}]).

The benefit of the proposed approach is that it makes easy for developers to identify early aspects by focusing only on the relationships between goals and use cases in a pairwise fashion. Furthermore, the results delivered to developers are represented in a frequency-based way, which allows developers to cut a threshold to determine how strong the likelihood is of an early aspectual candidate containing an early aspect.

If a target system to be modeled contains a fairly large number of goals, it may overwhelm developers in entering all the relationships between goals and use cases, which is of $O(n^2)$ complexity (n is the total number of goals). It is also likely that users could have neglected or missed some of the interactions in conducting the pairwise comparison. To reduce the impact, all the possible early aspectual candidates will be explored based on the interactions captured in the goals and use cases.

3.7 Comparing with Our Previous Work

Comparing with our previous work [33], the enhancement of this work is focused on the improvement of the clustering algorithm by reducing the time complexity to a computationally manageable level. In our previous work, the clustering of goals (i.e., the bidding process) begins with the bidding of goals to form a goal cluster, followed by the checking of total interaction degrees of all goal clusters, and finally through the use of scattering and tangling degrees to validate each bid.

That is, goals are bid by goal clusters one at a time. Each time a goal is to be bid by goal clusters, the similarity of the goal and goal clusters are computed. For n goals, it costs $O(n^2)$ time to compute the similarity of all goals and all goal clusters. The highest score of similarity is picked and the associated goal cluster is the winner of the bid. The goal is grouped with the winning goal cluster. A stability function is performed to serve as a check-and-balance mechanism in the clustering, which engages total interaction degree for checking the validity of the bidding and scattering degrees together with tangling degrees for balancing the total number of goal clusters. If there are multiple winning goal clusters, the derivation will be processed concurrently. If there is only one highest score of similarity in each bidding, it gives the time complexity of $O(n^3)$ for the best case scenario since there are n goals to be bid. However, if the scores of similarity of goals to goal clusters in each round of bidding are all equal, it generates $n * n$ concurrent derivations at the first round of bidding and $n * n * (n-1) * (n-1)$ concurrent derivations at the second round of bidding, and so on, which results in a total of $O(n^n)$ time complexity for the worst case scenario. As a result, when the number of goals increases, the number of bids will also increase. This will cause a cascading effect on the performance of the bidding algorithm.

In this work, a similarity matrix and interaction relation matrix of all goals are constructed prior to each round of

goal clustering. Although the dimensions of the two matrices may increase while the number of goals increases, the effort in computing the two matrices can be minimized comparing to the effort spent on each round of the bid in our previous work. Moreover, after each round of goal clustering, the dimensions of the two matrices decrease due to the grouping of goals and goal clusters, which further reduces the effort in computing the two matrices for the next round of goal clustering. For example, if there are n goals and each time there is only one highest similarity score in the clustering matrix, searching for the highest score in the matrix takes $O(n^2)$ and processing n goals requires $O(n)$, which results in a total of $O(n^3)$ time complexity by multiplying these two factors together. For the worst case scenario, there are $(n-1)*(n-1)/2$ highest similarity scores in the first clustering matrix and the dimensions of the clustering matrix will be reduced by one after each round of clustering, which gives us a total of $O(n!)$ time complexity for the worst case scenario.

Comparing the proposed approach with our previous work, the time complexity for the worst case has been improved from $O(n^n)$ to $O(n!)$. No computation result can be obtained by applying our previous algorithm to the course enrollment system due to a large number of parallel nodes expansion.

The supporting tool is also redesigned to integrate with an open source software modeling tool — ArgoUML. MapReduce, and HBase are also incorporated into the ArgoUML-based supporting tool as a means to improve the performance of the proposed clustering process. Adopting MapReduce enhances the clustering algorithm by performing grouping procedures concurrently, and using HBase helps manage the storing of intermediate data and final results generated by Mappers during the grouping of goals.

4 Experimental Evaluation

To illustrate the benefits of the proposed approach, a course enrollment system developed at National Central University in Taiwan is adopted in this section as a means to demonstrate how the proposed approach could advantage developers in the discovery of early aspects. By following the goal-driven early aspect process, goals are formulated for constructing GDUC model and goal clusters are classified by using the ArgoUML-based supporting tool to evaluate the relationships between goals and use cases. Early aspects are identified by following the 4-step guideline to identify early aspectual candidates obtained from goal clusters. In the process, developers can reduce the efforts in looking for crosscutting concerns or common properties across the whole system, and focus on relationships between goals and use cases for discovering early aspects. Furthermore, it better provides modularity insights in the analysis and design phases of software development by using the GDUC model.

An experiment is conducted to evaluate the benefits of the proposed approach by inviting three groups of people, including 15 college professors (P group), 15 graduate students (S group) and 15 software engineers from a software R&D institute in Taiwan (E group), to a survey: P and S groups apply the GEA to the course enrollment system to discover early aspects, while the E group discovers early aspects without using the GEA. Additionally, Mann-Whitney U-test [29] is performed to validate the results of the experiment to show that the difference between with GEA and without GEA is statistically significant. The reasons why we choose Mann-Whitney U-test are two-fold: one is that the data are not randomly drawn from a normally distributed population, and the other is that samples are independent.

4.1 A Course Enrollment System

In the experiment, a course enrollment system for the university is considered and a set of requirements and use cases are collected from the development team in the university computer center. The old version of the course enrollment system was built prior to the course information system, which made it cumbersome for the course enrollment system to retrieve or update course information correctly from or to the course information system. Therefore, modifications have to be made to the course enrollment system to make it robust to retrieve and update course information correctly. Furthermore, the old version of the course enrollment system did not support functionality, such as summer courses enrollment, inter-school enrollment, enrollment for adult education, enrollment for foreigner students, and query for parents to get enrollment information of their children. These are all new requirements to be included in the new version of the course enrollment system.

Main requirements are summarized as follows: (1) to integrate the databases of course enrollment system and course information system; (2) to redesign the user interface of the system; (3) to provide enrollment functionality for summer courses, inter-university courses, adult education courses, and etc.; (4) to provide an access control mechanism according to user identity; and (5) to provide English user interface for the system.

By applying the goal-driven early aspect process, 27 goals in the systems are identified and formulated by examining the description of the requirements. Based upon these goals, a goal-driven use case model of the course enrollment system is derived and constructed using the ArgoUML-based supporting tool (see Fig. 11).

As a result, in the course enrollment system, nine goal clusters are obtained below:

- $(G_{SystemManagement\&Maintain}, G_{ViewableRecordLogs})$
- $(G_{Q\&A}, G_{SystemStateDisplay})$
- $(G_{IncreaseNetworkSecurity}, G_{UserFunctionClassification})$
- $(G_{QueryStudentEnrollmentData}, G_{StudentInfo.SelfMaintain}, G_{VariousCourseInfo.Query})$
- $(G_{EnrollmentforOutsiders}, G_{SummerEnrollment})$
- $(G_{SystemMainPage}, G_{LatestNewsDisplay})$
- $(G_{RegularEnrollment}, G_{SummerCourseEnrollment}, G_{InterschoolEnrollment}, G_{ApplicationFormPrinting})$
- $(G_{CourseDataManagement}, G_{ChangeCourseData}, G_{DSManageDepart.CourseInfo.})$
- $(G_{EnrollmentAvailableforStudents}, G_{SystemLoginFunction})$

Among these goal clusters, it is noted that all the occurrences are counted as 100 percent indicating that these goal clusters can be treated directly as the early aspectual candidates. For example, one of the goal clusters has two goals, $G_{IncreaseNetworkSecurity}$ and $G_{UserFunctionClassification}$. According

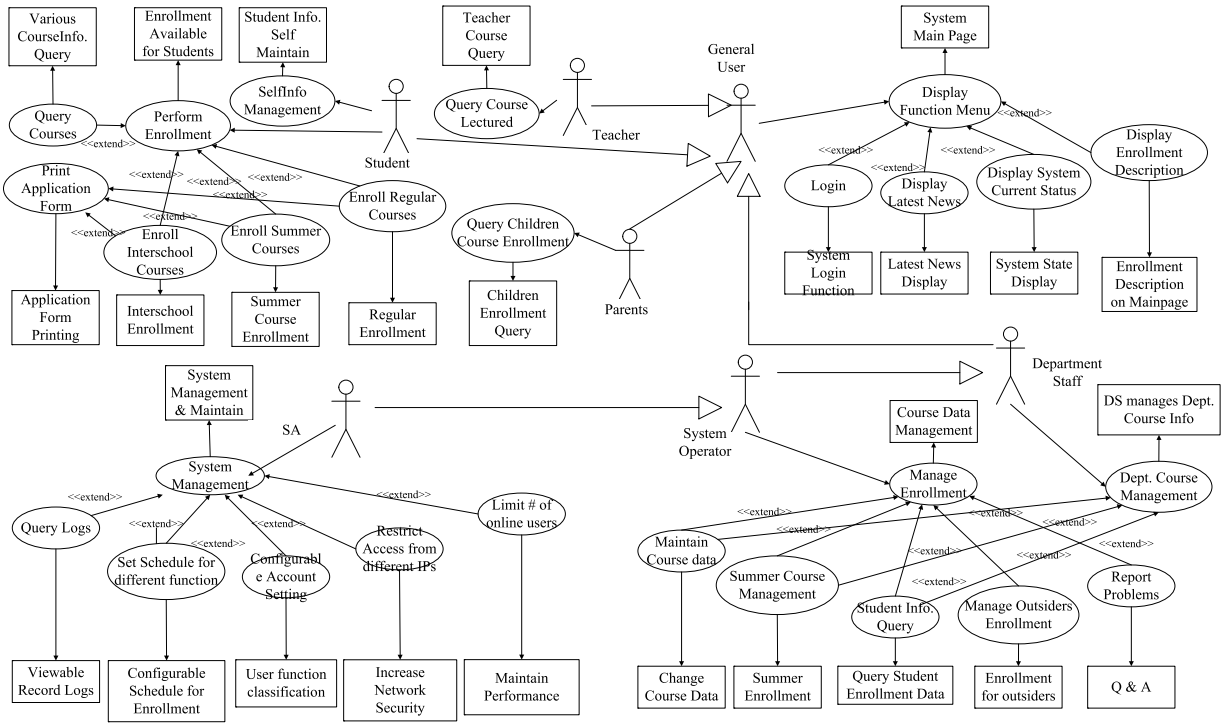


Fig. 11. Goal-driven use case model of the course enrollment system.

to the guideline provided in Section 3.6, developers are encouraged to find the common property or crosscutting concerns in these two goals. They may find that an early aspect named "Construct Authentication Matrix" can be considered to crosscut these two goals to offer multiple levels of security level controls and various kinds of user roles. Consequently, a use case "Construct Authentication Matrix" can be added to realize the early aspect and denoted as $EarlyAspect_{UAM}$: (Construct Authentication Matrix, [$G_{IncreaseNetworkSecurity}$ and $G_{UserFunctionClassification}$]).

The rest of the early aspects discovered are listed as follows:

- $EarlyAspect_{MSHD}$: (Maintain System Historical Data, [$G_{SystemManagemenr\&Maintain}$, $G_{ViewableRecordLogs}$])
- $EarlyAspect_{SII}$: (Support Interactive Inquiry, [$G_{Q\&A}$, $G_{SystemStateDisplay}$])
- $EarlyAspect_{UAM}$: (Construct Authentication Matrix, [$G_{IncreaseNetworkSecurity}$, $G_{UserFunctionClassification}$])
- $EarlyAspect_{MEDI}$: (Maintain Enrollment Data Integrity, [$G_{QueryStudentEnrollmentData}$, $G_{StudentInfo.SelfMaintain}$, $G_{VariousCourseInfo.Query}$])
- $EarlyAspect_{SMTC}$: (Support Multiple Types of Courses, [$G_{EnrollmentforOutsiders}$, $G_{SummerEnrollment}$])
- $EarlyAspect_{SUFI}$: (Support User Friendly Interface, [$G_{SystemMainPage}$, $G_{LatestNewsDisplay}$])
- $EarlyAspect_{SFE}$: (Support Flexible Enrollment, [$G_{RegularEnrollment}$, $G_{SummerCourseEnrollment}$, $G_{InterschoolEnrollment}$, $G_{ApplicationFormPrinting}$])
- $EarlyAspect_{MCDI}$: (Maintain Course Data Integrity, [$G_{CourseDataManagement}$, $G_{ChangeCourseData}$, $G_{DSManageDepart.CourseInfo.}$])
- $EarlyAspect_{MSS}$: (Maintain System Security, [$G_{EnrollmentAvailableforStudents}$, $G_{SystemLoginFunction}$])

4.2 Experiment

At the beginning of the experiment, a 30-minute lecture is given to the participants prior to conducting the experiment. The P and S groups are briefed with the concept of evaluating the relationships between goals and use cases, the usage of the ArgoUML-based supporting tool, and a summary of requirements statements of the NCU course enrollment system. The E group is briefed with the concept of early aspects along with guidelines for finding early aspects and a summary of requirements statements of the NCU course enrollment system along with the GDUC model of the system.

The number of goal clusters identified by the P group and S group using GEA in the survey ranges from 2 to 20 with an average of 8.067 goal clusters, and the frequency of occurrence of goal clusters also varies from 11, 20, 33.3, 40, 66.7, 80, to 100 percent.

Two indicators are established: goal-grouping-strength and aspect-crosscutting-modularity, to serve as a basis for further evaluating the proposal approach and for illustrating the benefits of our proposed approach.

The goal-grouping-strength indicator is defined as the grouping strength between goals under different thresholds. By setting different thresholds from 0, 15, 30, 50, 75, to 100 percent (see Fig. 12), the grouping of goals still bears a similar pattern, that is, goals being grouped together under different thresholds usually will be grouped into clusters even though by different developers.

For example, goals 3 and 4 are grouped for 17 to 21 times under different thresholds setting from 100 to 0 percent, which means that the proposed approach can provide a similar grouping results performed by various types of developers—college professors, programmers, or graduate students. Furthermore, the peaks in the graph

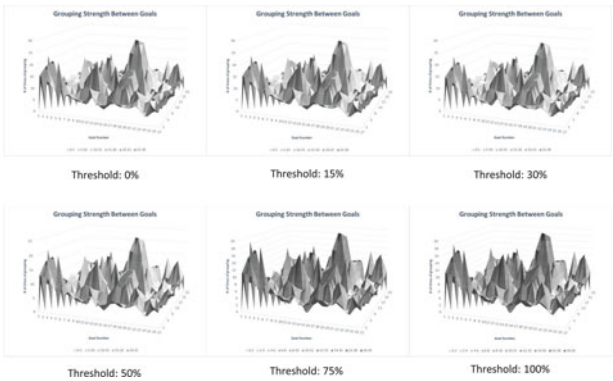


Fig. 12. Goal-grouping-strength indicator.

also indicate that potential early aspectual candidates offer developers numerous opportunities to discover early aspects in a target system in order to overcome one of the deficiencies of traditional aspect-oriented approaches that few concerns can be separated in the discovery process as mentioned in [9].

The aspect-crosscutting-modularity indicator is defined as the relationship between the number of goals in a goal cluster and the number of goal clusters. Figure [32] shows the number of goals in a goal cluster across all groupings of goals during the experiment conducted by 30 software developers. For example, the peak at 130 shown in Fig. 13 stands for the total number of goal clusters with two goals conducted by all developers. Similarly, the 0 located at 27 in the x-axis means that there does not exist a goal cluster with 27 goals conducted by all developers. It is noted that peaks at 130 (with two goals) or 19 (with three goals) or 29 (with four goals) imply that the goal clusters containing those goals can be treated as candidates of modules since goals in a goal cluster either behave similarly or have higher degrees of being achieved by associated use cases. This indicator illustrates that our proposed approach provides a better modularity insight in the analysis and design phases of software development which is considered as an important criteria for evaluating an aspect-oriented approach as mentioned in [29].

Table 6 shows the number of early aspectual candidates discovered by the P, S, and E groups, meanwhile P and S groups are further attached with different percentage thresholds for determining early aspectual candidates. The key question is that: "Is the difference between with GEA and without GEA statistically significant?". To

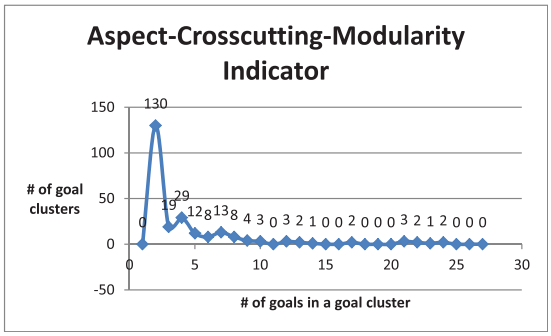


Fig. 13. Aspect-crosscutting-modularity indicator.

TABLE 6
Number of Early Aspectual Candidates Found by Participants

P(100%)	P(60%)	P(55%)	P(0%)	S(100%)	S(70%)	S(60%)	S(0%)	E
9	9	9	9	5	5	5	11	4
5	9	9	18	7	7	7	7	5
7	10	10	12	7	7	7	7	9
1	3	5	20	3	3	5	7	12
1	6	6	11	9	9	9	9	9
7	7	7	7	8	8	8	8	4
3	3	3	19	7	7	7	7	6
1	1	1	7	2	2	2	2	12
7	7	7	7	2	2	2	2	4
4	4	4	4	3	3	3	3	8
4	4	4	4	0	0	0	4	14
3	3	3	5	1	1	1	18	10
3	3	3	3	5	5	5	10	3
2	2	2	2	6	6	6	11	5
3	3	3	3	4	4	4	4	7

answer this question, three pairs of groups are established for the purpose of comparison by performing Mann-Whitney U-test.

In (P versus E) and (S versus E), a null hypothesis H_0 is stated as "There is no statistical difference between the number of early aspectual candidates found with GEA and without GEA". The research hypothesis H_A is "There is a statistical difference between the number of early aspectual candidates found with GEA and without GEA".

We use the commonly accepted value of $\alpha = 0.05$ to obtain the critical value 64 for the two pairs of groups with 15 data samples in each group. In other words, there is a 95 percent chance that the statistical findings are real and not due to chance.

In (P versus E), two types of results are obtained. When the threshold used for determining the early aspectual candidates is set to 60 percent or above, the U value is less than 60.5, which is smaller than the critical value 64 and therefore the null hypothesis H_0 is rejected. When the threshold is set to 55 percent or below, the U value 65 is larger than the critical value and therefore the null hypothesis H_0 is accepted. Similar results are obtained in (S versus E). It rejects the null hypothesis H_0 when the threshold is set to 70 percent (U value is 61) or above and accepts the null hypothesis H_0 when the threshold is set to 60 percent (U value is 65.5) or below.

In (P versus S), we would like to find out: "Is the difference between the results performed by different people with GEA statistically significant?". The null hypothesis H_0 is stated as "There is no statistical difference between the numbers of early aspectual candidates found by people with the help of GEA". The research hypothesis H_A is "There is a statistical difference between the numbers of early aspectual candidates found by people with the help of GEA". We use $\alpha = 0.05$ to obtain the critical value 64 for (P versus S) group. The U value for pair (P versus S) ranges from 97.5 at threshold 100 percent to 122 at threshold 0 percent, which accepts the null hypothesis H_0 and indicates that there is no statistical difference between the numbers of early aspectual candidates found by P and S groups with GEA.

Based on the results of performing Mann-Whitney U-test, we have the following findings: It is statistical significance between the numbers of discovered early aspectual

candidates with GEA and without GEA when the threshold for determining early aspectual candidates in GEA is set to 60 percent or higher. However, there is no statistical difference between the numbers of early aspectual candidates found by people with the help of GEA.

As a result of the experiment, we show that the difference between with GEA and without GEA is statistically significant.

5 Related Work

In what follows, we outline three dimensions of related work on aspect-oriented software engineering that have shed some light on this work: the first is researches on early aspects [1], [2], [32], [38], [39], [40], [41], the second is work on the relationships between early aspect and use cases [33], [42], [43], [44], and the final one is work that focuses on relationships between crosscutting concerns [9], [16], [45], [46].

5.1 Early Aspects

Aspects are behaviors that are tangled and scattered across a system [38]. In requirements documents, aspects may reveal themselves as interleaving and interdependent behaviors. Some aspects may be easily identified, as specifications of typical crosscutting behavior, while others may be more subtle and difficult to discover. Many recent studies have attempted to identify and apply the concept of aspects to the early stage of software development, called early aspects, in the hope of better addressing important stakeholders' concerns in the requirements analysis and design phases.

Elisa Baniassad et al. [1] emphasize the importance of identifying and managing requirements-level and architecture-level aspects instead of merely focusing on the implementation phase in the software life cycle. In their work, early aspects are identified and captured explicitly in requirements and architecture activities, and carried over the entire software development life cycle.

Theme-based approaches [38] assume that two behaviors are related if they occur in the same requirement. Themes are classified as base themes or crosscutting themes. Base themes may share some structures and behaviors with other base themes. Crosscutting themes have behaviors that overlay the functionality of the base themes and are treated as aspects. The disadvantage of theme-based approaches is the excessive effort required in grouping the actions into larger themes and identifying aspects.

AORE [2] identifies candidate aspects by representing the relationships between concerns and stakeholder requirements in a contribution matrix based on the negative or positive effects of each aspect on others. Conflicts with stakeholders are resolved by prioritizing concerns. The requirements specification is then revised based on the new priorities.

Early-AIM [39] adopts corpus-based natural language processing techniques to help automate the identification and modeling of early aspects in a requirements document. The main aim of Early-AIM is to discern the candidate aspects in a document, irrespective of the document structure. EA-Miner [47] is the realization of this concept and offers automated support for identifying and visualizing early aspects from various requirements-related documents.

Aspect-oriented Multi-View Modeling [48] proposes to model a software from multi-views by utilizing various notations, such as class diagrams, sequence diagrams, and state diagrams. By using the reusable aspect models, it can support aspect dependency chains, which allows an aspect for providing complex functionality to reuse the functionality provided by other aspects. However, the focus of the work is on reusing existing aspects not on discovering new aspects.

ACE [32] seeks to identify crosscutting concerns through the application of automated clustering techniques. It utilizes a probabilistic model to compute the similarities between different requirements and uses a hierarchical algorithm to cluster similar requirements. Concerns represented by dominant terms were detected during an initial clustering phase, while those represented by less dominant terms were detected by removing away the dominant terms from requirements in subsequent phases. Generated clusters, candidate early aspects, where evaluated using metrics to measure their physical dispersion across the requirements specification and their level of interaction with other dominant concerns. Although ACE addresses the early aspect identification by means of clustering, it is still impeded by the following two problems: one is that the early aspect candidates found in the final clusters have a coherent and imprecise problem, namely, the result clusters may still contain a few unwanted or unrelated requirements; and the other is that a same concern addressed by various requirements may be overlooked due to the expression of the concern by different terms which may impact the similarities of requirements, which is addressed in [49] by Kit et al. with latent semantics analysis.

Table 7 summaries the works on early aspects. The main limitation of the above-mentioned related work of early aspects is that early aspects are mainly identified based on keywords across the whole system requirements or artifacts, which could hinder developers from focusing on major system functionalities since the identification of early aspects is accomplished by finding crosscutting concerns across the whole target system, and could possibly change the main course of the system construction. The theory we proposed makes easy the identification of early aspects through a numerical manner by computing the similarity of goals and by checking the validity of the formation of a goal cluster with the total interaction degree.

5.2 Use Cases with Early Aspects

Many researchers have adopted use cases in requirements specification, analysis and design, and have attempted to adopt them as test beds for introducing early aspects into the requirements phase.

Sousa et al. [42] propose modeling crosscutting concerns as use cases, and presented a new relationship between use cases, called $\ll crosscut \gg$. Information about the composition between a crosscutting use case and the use cases that it affects is described in a composition table that enables the join points to be defined, instead of in the base use case. The composition between an extension and a base use case can be fully non-invasive. A heuristic rule is provided to determine when to connect two use cases via a crosscutting relationship.

TABLE 7
Comparison of Works on Early Aspects

Researches \ Features	Phases of EAs discovered	Constructs	Ways to discover EAs
Elisa et al. [1]	Requirements and architecture level	Requirement and architecture activities	Requirements and architecture analysis
Theme-based approaches [38]	Requirements phase	Themes	Theme analysis
AORE [2]	Requirements phase	Relationships matrix	By developers modeling
Early-AIM [39] & EA-Miner [47]	Requirements documents	Corpus-based natural language	Automatic identification
Aspect-oriented Multi-View Modeling [48]	Requirements to design phases	model from multi-views	By developers modeling and reuse
ACE [32]	Requirements phase	probabilistic model	Automated clustering

AspectU [43] is an aspect language for modularizing crosscutting concerns within a use-case model, and extends the use case model with support for modularizing crosscutting behavior. It introduces an aspect entity structured similarly to an aspect in AspectJ that comprises of pointcuts and advice.

Moreira et al. [44] present a model to identify and specify quality attributes that crosscut requirements, including the integration of quality attributes, into the functional description at an early stage of the software development process. A model of the crosscutting behavior is devised to consider a quality attribute in a new stereotype use case, and made the base use cases include the stereotype use case.

Araujo et al. [50] adopt the concepts of overlapping, overriding and wrapping operators to compose functional requirements with aspects that crosscut non-functional requirements. A non-functional requirement crosscuts if it affects more than one use case. They model these aspects by defining new stereotype use cases, and adopted the stereotype relationships $\ll wrappedBy \gg$ to connect those wrapped use cases to the crosscutting use case.

Table 8 summaries the works contribute to the relationships between use cases and early aspects along with the proposed approach. Differ from the above methods, use cases in our proposed approach are used as a means to derive relationships among goals, such as similarity and interaction degrees, and to group goals into goal clusters based on the two degrees. Similar attempt of identifying early aspects has been proposed in our previous work [33], where early aspects are discovered in a possibility based approach. However, the effort on the computation of early aspectual candidates could cost considerable resources and time, which diminishes the usability of that approach.

TABLE 8
Relationships between Use Cases and Early Aspects

Researches	Relationships
Sousa *et al.* [42]	Modeling crosscutting concerns as use cases
AspectU [43]	Aspect language for modeling crosscutting concerns with in a use case model
Moreira *et al.* [44]	Crosscutting behavior as a quality attribute in a new stereotype use case
Araujo *et al.* [50]	Requirements that crosscuts more than one use case
Our Approach	Use cases are used as a means to derive relationships for discovering early aspectual candidates

5.3 Relationships between Crosscutting Concerns

There are other researches that aim to identify separation of concerns during the requirement analysis phase, in which the analysis of functional or non-functional requirements is used to lead the discovery of aspects.

In [9], Cosmos provides a general-purpose concern-space modeling schema for modeling logical and physical concerns, in which logical concerns can be further distinguished into classifications, classes, instances, properties, and topics while physical concerns includes instances, collections, and attributes. Four categories of relationship are also identified in Cosmos, including categorical, interpretive, mapping, and physical. By providing concern-space modeling schema, Cosmos can be used to support many software development tasks, such as rationale capture, impact analysis, change propagation, and software composition and decomposition.

In [46], Yu et al. propose to use a goal model to discover aspects from relationships among goals, in which functional and non-functional requirements are represented through goals and softgoals along with their tasks that contribute to their satisfaction. The model is then further analyzed to identify aspects by detecting the tasks that contribute to some soft goals while also satisfying some functional goals. There is a limitation that it only identifies soft goals as aspects without considering the rigid goals, which could also be the source of aspects.

In [45], an approach that supports the establishment of early trade-off among crosscutting and overlapping requirements is proposed to facilitate negotiation and decision-making among stakeholders. In the approach, it treats all concerns in a uniform fashion, that is, concerns in the model imply any coherent collection of requirements and can support multi-dimensional separation of concerns at requirements level.

In [16], a uniform treatment of concerns is proposed at requirement engineering level, which is based on the observation that concerns in a system are a subset, and concrete realization, of abstract concerns in a meta concern space. The notion of a compositional intersection that allows choosing appropriates sets of concerns in multi-dimensional separation as a basis to observe trade-offs among other concerns is introduced in their work, which provides a rigorous analysis of requirements-level trade-offs to satisfy a particular functional or non-functional concerns.

TABLE 9
Comparison of Our Approach with Other Approaches

Features	Our Approach	Cosmos [9]	Yu et al. [46]	Rashid et al. [45]	Moreira et al. [16]
Requirements Representation	goals formulation	concern-space modeling	V-graphs	multi-dimensional	multi-dimensional
Mechanism for Discovering Aspects	GEA process	Logical/physical concerns identification	Based on goal model	Projection and composition rules	Compositional intersection
Conflict Resolution	Interaction degrees in GDUC	-	Goal analysis tool	Prioritization approach for conflict resolution	Attribute weights and prioritize

Table 9 shows the comparison of our approach to the aforementioned approaches. In our approach, the requirements are represented using goals formulation in a tuple-form while others use concern-space modeling, V-graph or multi-dimensional way to represents the source where to discovery aspects or concerns. The mechanism used for discovering aspects in our approach is the clustering algorithm which is based on numerical computation of relationships among goals. Others use logical/physical concerns identification, goal model based identification, projection and composition rules, or compositional intersection to discover aspects in their approaches. While discovering aspects, conflicts could occur in different stages during the process. In our approach, conflicts are handled at the grouping of goals using the total interaction degrees. In [46], it uses goal analysis tool to detect conflicts and deteriorations. In [16], [45], a prioritization approach is proposed to resolve conflicts based on attribute weights.

In dealing with conflicts while discovering aspects, [35], [51] also propose different ways to handle conflicts. In [35], a technique to support conflict management at the AORE level is proposed based on Analytic Hierarchy Process, AHP. It could be used to support architectural choices during software architecture design. In [51], it aims to assess the suitability of multi-criteria decision making (MCDM) methods to support software engineers' decisions. A HAM (hybrid assessment method) is proposed to give user the ability to perceive the influence different decisions may have on the final result.

Comparing with the related work, the benefits of the proposed approach can be summarized as follows:

1) It makes easy for developers to identify early aspects by focusing only on the relationships between goals and use cases in a pairwise fashion.
2) By setting a different threshold, the proposed approach gives an experienced requirements engineer a prism into all the possible potential early aspectual candidates to prevent any neglect or overlook that may occur based purely on intuition.
3) The numerical representation of relationships among goals is more informative and can be more easily processed computationally, which can be used as a basis for discovering early aspects by exploring the existence of common properties shard by goals.

6 Concluding Remarks

In this work, we propose a goal-driven approach (called GEA) to the discovery of early aspects through goal clustering by means of a clustering algorithm as an attempt towards the analysis of software systems, in which two main features are devised: (1) evaluating the relationships among goals and use cases to obtain the degrees of similarity and interaction relationships among goals; and (2) discovering early aspects through the exploration of interactions among goals and use cases, which engages similarity degree among goals for clustering goals, total interaction degree for checking the validity of the clustering.

Introducing early aspects not only helps further enhance the goal-driven requirements modeling to manage crosscutting properties, but also better addresses important stakeholders' concerns in the analysis and design phases of software development.

There are also two additional benefits of the proposed approach: one is from the viewpoint of using numerical representation of relationships among goals, and the other is from the developers' viewpoint. Representing relationships among goals numerically is more informative in facilitating the discovery of early aspects as compared with using qualitative terms and is easier to be processed computationally.

From the developers' viewpoint, it makes easy for developers to identify early aspects by focusing only on the relationships between goals and use cases in a pairwise fashion, which, we believe, can reduce the effort in looking for crosscutting concerns or common properties across the whole system. The results delivered to developers are represented in a frequency-based manner, which allows developers to set a threshold to determine how strong the likelihood is of an early aspectual candidate containing an early aspect. Moreover, by setting a different threshold, the proposed approach gives an experienced requirements engineer a prism into all the possible potential aspectual candidates.

An experiment is conducted to evaluate the benefits of the proposed approach by inviting three groups of people to a survey. The result of the experiment is further validated through Mann-Whitney U-test to show that the difference between with GEA and without GEA is statistically significant.

Our future research plan includes: (1) to transform early aspects identified in the requirements phase to aspect-oriented programming language; (2) to discover aspects through exploring relationships among classes and methods in source code; (3) to conduct software refactoring by removing bad aspect smells detected in the identified aspects; and (4) to establish an evaluation process for assessing the refactoring method.

ACKNOWLEDGMENTS

This research was supported by Ministry of Science and Technology (Taiwan) under Grants NSC 100-2221-E-002-001-MY3.

REFERENCES

[1] E. Baniassad, P. C. Clements, J. Araújo, A. Moreira, A. Rashid, and B. Tekinerdoğan, "Discovering early aspects," *IEEE Softw.*, vol. 23, no. 1, pp. 61–70, Jan.-Feb. 2006.

[2] A. Rashid, A. Moreira, and J. Araújo, "Modularisation and composition of aspectual requirements," in *Proc. 2nd Aspect-Oriented Softw. Develop. Conf.*, 2003, pp. 11–21.

[3] M. Mortensen, S. Ghosh, and J. M. Bieman, "Aspect-oriented refactoring of legacy applications: An evaluation," *IEEE Trans. Softw. Eng.*, vol. 38, no. 1, pp. 118–140, Jan./Feb. 2012.

[4] S. Miller, "Aspect-oriented programming takes aim at software complexity," *Comput.*, vol. 34, no. 4, pp. 18–21, Apr. 2001.

[5] N. Noda and T. Kishi, "On aspect-oriented design-an approach to designing quality attributes," in *Proc. 6th Asia Pac. Softw. Eng. Conf.*, 1999, pp. 230–237.

[6] M. Shomrat and A. Yehudai, "Obvious or not? regulating architectural decisions using aspect-oriented programming," in *Proc. 1st Int. Conf. Aspect-Oriented Softw. Develop.*, Apr. 2002, pp. 3–9.

[7] J. Viega and J. Voas, "Can aspect-oriented programming lead to more reliable software?" *IEEE Softw.*, vol. 17, no. 6, pp. 19–21, Nov./Dec. 2000.

[8] A. Rashid, A. Moreira, and B. Tekinerdogan, "Early aspects—Aspect-oriented requirements engineering and architecture design," *IEEE Proc. Softw.*, vol. 151, no. 4, pp. 153–155, Aug. 2004.

[9] S. M. Sutton Jr. and I. Rouvellou, "Modeling of software concerns in cosmos," in *Proc. 1st Int. Conf. Aspect-Oriented Softw. Develop.*, 2002, pp. 127–133.

[10] A. Dardenne, A. van Lamsweerde, and S. Fickas, "Goal-directed requirements acquisition," *Sci. Comput. Programm.*, vol. 20, no. 1–2, pp. 3–50, 1993.

[11] J. Mylopoulos, L. Chung, and B. Nixon, "Representing and using nonfunctional requirements: A process-oriented approach," *IEEE Trans. Softw. Eng.*, vol. 18, no. 6, pp. 483–497, Jun. 1992.

[12] A. van Lamsweerde, R. Darimont, and E. Leitier, "Managing conflicts in goal-driven requirements engineering," *IEEE Trans. Softw. Eng.*, vol. 24, no. 11, pp. 908–926, Nov. 1998.

[13] J. Lee and Y. Fanjiang, "Modeling imprecise requirements with xml," *Inform. Softw. Technol.*, vol. 45, no. 7, pp. 445–460, May 2003.

[14] W. Lee, W. Deng, J. Lee, and S. Lee, "Change impact analysis with a goal-driven traceability-based approach," *Int. J. Intell. Syst.*, vol. 25, pp. 878–908, Aug. 2010.

[15] F. Steimann, "Domain models are aspect free," in *Model Driven Engineering Languages and Systems*, New York, NY, USA: Springer-Verlag, 2005, pp. 171–185.

[16] A. Moreira, A. Rashid, and J. Araújo, "Multi-dimensional separation of concerns in requirements engineering," in *Proc. 13th IEEE Int. Conf. Requirements Eng.*, 2005, pp. 285–296.

[17] A. Rashid and A. Moreira, "Domain models are not aspect free," in *Proc. 9th Int. Conf. Model Driven Eng. Lang. Syst.*, Springer, 2006, pp. 155–169.

[18] L. Constantine and L. Lockwood, *Software for Use*. Reading, MA, USA: Addison-Wesley, 1999.

[19] J. Rumbaugh, "Getting started: Using use cases to capture requirements," *J. Object-Oriented Programm.*, vol. 7, no. 5, pp. 8–12, Sep. 1994.

[20] K. Pohl, "The three dimensions of requirements engineering: A framework and its applications," *Inform. Syst.*, vol. 19, no. 3, pp. 243–258, 1994.

[21] J. Araújo, E. Baniassad, P. Clements, A. Moreira, A. Rashid, and B. Tekinerdoğan, "Early aspects: The current landscape," Softw. Eng. Inst., Carnegie Mellon Univ., Tech. Rep. COMP-001-2005, May 2005.

[22] J. Lee and K. Hsu, "Modeling software architectures with goals in virtual university environment," *Inform. Softw. Technol.*, vol. 44, pp. 361–380, Apr. 2002.

[23] J. Lee, N. Xue, and J. Kuo, "Structuring requirement specifications with goals," *Inform. Softw. Technol.*, vol. 43, pp. 121–135, Feb. 2001.

[24] J. Lee and N. Xue, "Analyzing user requirements by use cases: A goal-driven approach," *IEEE Softw.*, vol. 16, no. 4, pp. 92–101, Jul./Aug. 1999.

[25] J. Lee and J. Kuo, "New approach to requirements trade-off analysis for complex systems," *IEEE Trans. Knowl. Data Eng.*, vol. 10, no. 4, pp. 551–562, Jul./Aug. 1998.

[26] S. E. Sim, S. Easterbrook, and R. C. Holt, "Using benchmarking to advance research: A challenge to software engineering," in *Proc. 25th Int. Conf. Softw. Eng.*, 2003, pp. 74–83.

[27] M. S. Feather, S. Fickas, A. Finkelstein, and A. van Lamsweerde, "Requirements and specifications exemplars," *Autom. Softw. Eng.*, vol. 4, no. 4, pp. 419–438, Oct. 1997.

[28] A. van Lamsweerde, *Requirements Engineering: From System Goals to UML Models to Software Specifications*. Hoboken, NJ, USA: Wiley, 2009.

[29] H. B. Mann and D. R. Whitney, "On a test of whether one of two random variables is stochastically larger than the other," *Ann. Math. Statist.*, vol. 18, no. 1, pp. 50–60, 1947.

[30] C. Rolland, C. Souveyet, and C. Achour, "Guiding goal modeling using scenarios," *IEEE Trans. Softw. Eng.*, vol. 24, no. 12, pp. 1055–1071, Dec. 1998.

[31] K.-H. Hsu, "Model architecture with goals," Ph.D. dissertation, Dept. Comput. Sci. Info. Eng., Nat. Central Univ., Taoyuan County, Taiwan, 2002.

[32] C. Duan and J. Cleland-Huang, "A clustering technique for early detection of dominant and recessive cross-cutting concerns," in *Proc. Early Aspects ICSE: Workshops Aspect-Oriented Requirements Eng. Archit. Des.*, May 2007, p. 1.

[33] J. Lee, K. Hsu, S. Lee, and W. Lee, "Discovering early aspects through goals interactions," in *Proc. 19th Asia-Pac. Softw. Eng. Conf.*, Dec. 2012, pp. 97–106.

[34] T. L. Satty, *Analytic Hierarchy Process*. New York, NY, USA: McGraw-Hill, 1980.

[35] I. S. Brito, F. Vieira, A. Moreira, and R. A. Ribeiro, "Handling conflicts in aspectual requirements compositions," in *Trans. Aspect-Oriented Software Development III*, Berlin, Heidelberg, Springer-Verlag, 2007, pp. 144–166.

[36] J. Dean and S. Ghemawat, "Mapreduce: Simplified data processing on large clusters," *Commun. ACM*, vol. 51, no. 1, pp. 107–113, 2008.

[37] D. Carstoiu, E. Lepadatu, and M. Gaspar, "Hbase-non SQL database, performances evaluation," *Int. J. Adv. Comput. Technol.*, vol. 2, no. 5, pp. 42–52, 2010.

[38] E. Baniassad and S. Clarke, "Finding aspects in requirements with theme/doc," in *Proc. Workshop Early Aspects: Aspect-Oriented Requirements Eng. Archit. Des.*, 2004, pp. 15–22.

[39] A. Sampaio, A. Rashid, and P. Rayson, "Early-aim: An approach for identifying aspects in requirements," in *Proc. 13th IEEE Int. Conf. Requirements Eng.*, 2005, pp. 487–488.

[40] J. Araújo, J. Whittle, and D.-K. Kim, "Modeling and composing scenario-based requirements with aspects," in *Proc. 12th IEEE Int. Conf. Requirements Eng.*, 2004, pp. 58–67.

[41] J. Whittle and J. Araújo, "Scenario modelling with aspects," in *Softw., IEE Proc.*, vol. 151, no. 4, IET 2004, pp. 157-171.

[42] G. Sousa, S. Soares, P. Borba, and J. Castro, "Separation of crosscutting concerns from requirements to design: Adapting an use case driven approach," in *Proc. Early Aspects Workshop at AOSD*, 2004, pp. 93–102.

[43] J. Sillito, C. Dutchyn, A. D. Eisenberg, and K. D. Volder, "Use case level pointcuts," in *Proc. Eur. Conf. Object-Oriented Programm.*, 2004, pp. 244–266.

[44] A. Moreira, J. Araújo, and I. Brito, "Crosscutting quality attributes for requirements engineering," in *Proc. 14th Int. Conf. Softw. Eng. Knowl. Eng.*, 2002, pp. 167–174.

[45] A. Moreira, J. Araújo, and A. Rashid, "A concern-oriented requirements engineering model," in *Proc. 17th Int. Conf. Adv. Inform. Syst. Eng.*, 2005, pp. 293–308.

[46] Y. Yu, J. C. S. D. P. Leite, and J. Mylopoulos, "From goals to aspects: Discovering aspects from requirements goal models," in *Proc. 12th IEEE Int. Requirements Eng. Conf.*, 2004, pp. 38–47.
[47] A. Sampaio, R. Chitchyan, A. Rashid, and P. Rayson, "Ea-miner: A tool for automating aspect-oriented requirements identification," in *Proc. 20th IEEE/ACM Int. Conf. Autom. Softw. Eng.*, 2005, pp. 352–355.
[48] J. Kienzle, W. A. Abed, and J. Klein, "Aspect-oriented multi-view modeling," in *Proc. 8th ACM Int. Conf. Aspect-Oriented Softw. Develop.*, 2009, pp. 87–98.
[49] L. K. Kit, C. K. Man, and E. Baniassad, "Isolating and relating concerns in requirements using latent semantic analysis," in *Proc. 21st Annu. ACM SIGPLAN Conf. Object-Oriented Programm. Syst., Lang., Appl.*, 2006, pp. 383–396.
[50] J. Araújo, A. Moreira, I. Brito, and A. Rashid, "Aspect-oriented requirements with UML," in *Proc. Workshop Aspect-Oriented Model. UML*, vol. 7, 2002.
[51] R. A. Ribeiro, A. M. Moreira, P. Van den Broek, and A. Pimentel, "Hybrid assessment method for software engineering decisions," *Decis. Support Syst.*, vol. 51, no. 1, pp. 208–219, 2011.

Jonathan Lee received the PhD degree in computer science from Texas A&M University in 1993. He is a professor in the Department of Computer Science and Information Engineering at National Taiwan University (NTU) in Taiwan. He was the department chairman from 1999 to 2002 and was the director of Computer Center at National Central University from 2006 to 2012. His research interests include software engineering, service-oriented computing, and software engineering with computational intelligence. He received IBM Shared University Research Award (2010), CIEE Electrical Engineering Outstanding Professor Award, NCU Distinguished Professor Award, (2006-2013), and NCU Distinguished Research Award (2004). He also served as the program chairs of the 12th Asia-Pacific Software Engineering Conference (APSEC 2005) and the 8th International Fuzzy Systems Association World Congress (IFSA 1999). He is a senior member of the IEEE Computer Society.

Kuo-Hsun Hsu received the BS degree in computer and information science from the National Chiao-Tung University, Taiwan, in 1996 and the PhD degree in computer science and information engineering from National Central University, Taiwan, in 2003. He is an assistant professor in the Department of Computer Science at National Taichung University of Education in Taiwan. His research interests include software engineering, requirement engineering, software architecture, service-oriented architecture, and CMMI. He is a member of the IEEE.