Goal and scenario based domain requirements analysis environment

Jintae Kim *, Minseong Kim, Sooyong Park

Department of Computer Science, Sogang University, Shinsu-Dong, Mapo-Gu, Seoul, 121-742, Republic of Korea

Received 6 December 2004; accepted 20 June 2005
Available online 23 January 2006

Abstract

Identifying and representing domain requirements among products in a product family are crucial activities for a successful software reuse. The domain requirements should be not only identified based on the business goal, which drives marketing plan, product plan, and differences among products, but also represented as familiar notations in order to support developing a particular product in the product family. Thus, our proposal is to identify the domain requirements through goals and scenarios, and represent them as variable use cases for a product family. Especially, for identification of the domain requirements, we propose four abstraction levels of requirements in a product family, and goal and scenario modeling. For representation of them, variable use case model is suggested, and also the use case transfer rules are proposed so as to bridge the gap between the identification and representation activity. The paper illustrates the application of the approach within a supporting tool using the HIS (Home Integration System) example.

Keywords: Goal; Scenario; Use case; Domain requirements analysis

1. Introduction

Managing differences between products in a product family is one of the essential matters that must be considered when building a product line. Recently, the focus is being changed from mere commonality management to variability management (Geyer and Becker, 2002; Jaring and Bosch, 2002; Myllymäki, 2001).

Feature based approaches such as FODA (Kang et al., 1990) and FeatuRSEB (Griss et al., 1998) have been widely used in domain analysis to model mandatory and variant (optional and alternative) requirements as a feature graph. However, it has been recognized that the feature based approaches do not adequately support some issues (refer to Section 6). In brief, they do not support a well defined process and the source of variability. Without a well defined process, they are less applicable. Handling the reason of variability makes it easy to predict when the variation happens and where the variation comes from.

Therefore, in this paper, we focus on a well defined process for identification and representation of domain requirements considering the source of variability. The characteristics of our approach are (1) to provide a process for domain requirements analysis, (2) to provide how to represent domain requirements, and a tool supporting both (1) and (2). Three characteristics of our approach (see Fig. 1) contribute to the achievement of these objectives.

First, there is the identification of domain requirements using goals and scenarios: goal and scenario modeling is an effective way to identify requirements in requirements engineering (Kim et al., 2004; Potts, 1997; Rolland et al., 1998b; Sutcliffe, 1998). A goal provides the rationale for requirements—a requirement exists because of some underlying goal which provides a base for it (Dardenne et al., 1991; Kim et al., 2004; Sommerville and Sawyer, 1997). Since scenarios describe real situations, they capture real requirements (Rolland et al., 1998b). In the context of a product line, an organization’s high-level business goals drive marketing plans and product plans, and force
products in a product family to have common and variant requirements. On the basis of the business goals, the characteristics of products are determined, and the goal provides the rationale for the domain requirements. Consequently, goal and scenario modeling makes it possible to manage adequately the variations among products in the product family. In addition, the domain requirements level is proposed to help separation of concerns in product line requirements elicitation as follows: business, service, interaction, and internal level. These levels are abstraction hierarchies. Each of them has its own perspective on separation of concerns in the product line requirements. Goal and scenario modeling is thus to identify and analyze the domain requirements corresponding to each of the domain requirements level via the goal and scenario template proposed.

The second characteristic of the approach is to represent the domain requirements identified through goal and scenario modeling as a use case diagram through the transfer guiding rules. The guiding rules help to represent the domain requirements including common and variant requirements as variable use cases. Thus, the domain requirements are expressed by the use case diagram extended for representation of variability.

Finally, the tool supports our proposal through the whole process and their outputs. Especially it generates XML (extensible markup language) based use case specification, which can be shown as a graphical diagram in CASE tools such as Rational Rose (http://www-306.ibm.com/software/rational/) and Together (http://www.borland.com/together/). The tool consists of 4 tabs corresponding to the four levels of abstraction for goal and scenario. The tool supports goal and scenario authoring and the use case transfer rules.

The paper is organized as follows. The next section briefly discusses HIS (Home Integration System) used in this paper to illustrate our approach. How the domain requirements are identified through goal and scenario modeling is described in Section 3. A use case diagram representing domain requirements is presented in Section 4. The transfer rules helping to represent the domain requirements as a use case diagram are discussed in Section 5. Section 6 describes a supporting environment and related work is discussed in Section 7. The last section sums up the essential properties of our approach and discusses future works.

2. Case description: Home Integration System

We apply our approach to a HIS. To demonstrate the feasibility of our approach, we have chosen the example of a HIS (Kang et al., 2002; Chastek et al., 2001, 2002). HIS enhances comfort, safety, and security of a home. The HIS enables homeowners to access, control, and integrate equipment in their homes such as the ones listed below:

- climate control systems—heating and cooling;
- security systems—intrusion, fire, and flood detection and response; and
- entertainment systems—televisions, radios, and music-playing devices.

HIS represents a multi-billion dollar market and offers an opportunity for an organization with strong experience and expertise in the HIS arena to expand into a new market (e.g., integration, networks, devices, home or office security systems). The HIS market is relatively new and immature, and is changing rapidly as new devices and manufacturers continually appear. Most homeowners have no experience with an HIS and moreover, the consumer market for HIS is also quite varied.

3. Domain requirements analysis using goal and scenario

This section discusses goal and scenario modeling that helps to identify and analyze the domain requirements based on the rationale for variations, i.e., a goal. A goal drives marketing plan, product plan, and the differentiations among most of the products because an organization makes a plan to develop products based on its business goal. The goal means, therefore, the direction, purpose, and objective of the organization. Otherwise scenarios show the behavior of system. They represent how the goal can be achieved by a set of purposeful actions. However, in requirements engineering, it has been recognized that analyzing requirements through goal and scenario is not very easy because it is not defined at which level goal and scenario modeling should be stopped. In this paper, goal and scenario modeling is implemented corresponding to the domain requirements level, such as business, service, interaction, and internal level. These levels explain where goal and scenario modeling ends. Details of the domain requirements level, the concept of goal and scenario, and goal and scenario modeling are discussed in the following sections.

3.1. The concept of goal and scenario

A goal is defined as “something that some stakeholder hopes to achieve in the future” (Plihon et al., 1998) or
“high level objectives of the business, organization or system” (Antón, 1996) in requirements engineering. In this paper, a goal is defined in the context of a product line, as an objective of the business, organization or system that some stakeholder hopes to achieve with that product line. We propose four types of a goal corresponding to the domain requirements level (refer to Section 3.2), which are a business goal, an interaction goal, and an internal goal. Clearly, a goal (Prat, 1997) is associated to a verb and to one or more parameters, where each parameter plays a different role with respect to the verb. In this paper, a goal is described as a combination \((V + Target + Direction)\), where \(V\) is an active verb, Target is a conceptual or a physical object, and Direction is either source or destination. For example, the goal, ‘Provide Entertainment System to customers’ is described as follows:
\[
\text{Provide}_i \text{Verb}_i \text{(EntertainmentSystem)}_i \text{Target}_i \text{(ToCustomers)}_i \text{Dir}_i
\]

A scenario is “a possible behavior limited to a set of purposeful interactions taking place among several agents” (Plhôn et al., 1998). Potts et al. (1994) defines a scenario as “particular cases of how the system is to be used” and additionally states that, “in a broad sense, a scenario is simply a proposed specific use of the system”. In the context of a product line, we define a scenario as a possible behavior limited to a set of purposeful actions in order to achieve some goals. A scenario also has three types such as a service scenario, an interaction scenario, and an internal scenario. In this paper, a scenario is authored as a combination \((S + V + Target + Direction + Manner)\), where \(S\) is either agents except for the designed system or the designed system itself, \(V\) is an active verb, Target is a conceptual or a physical object, Direction is either source or destination, and Manner is the way in which the scenario is implemented. For example, the scenario, ‘the HIS checks the current temperature with a temperature sensor’ is described as follows:
\[
\text{(the HIS)}_i \text{Subject}_i \text{(checks)}_i \text{Verb}_i \text{(the current temperature)}_i \text{Target}_i \text{(with a temperature sensor)}_i \text{Manner}_i
\]

### 3.2. The relationship between goals and scenarios in domain requirements analysis

The scenarios capture real requirements since they describe real situations or concrete behaviors, and goals can be achieved through the execution of scenarios. Thus, scenarios have their goals, and typically, the goals are achieved by the scenarios. In other words, just as goals can help in scenario discovery, so also scenarios can help in goal discovery. As each individual goal is discovered, a scenario can be authored for it. Once a scenario has been authored, it is explored to yield goals (Rolland et al., 1998b). In a product line, since a goal provides the rationale for variations in domain requirements, it is used as a discriminator that enables us to identify common and variant requirements. Variant requirements (or variation point Jacobson et al., 1997) are identified when scenarios achieve a goal in two ways; first, alternative variation: from a set of alternative scenarios, only one scenario can be chosen to achieve a goal—defining an exclusive relationship, which means a “1 from n choice”. Second is optional variation: optional scenarios for a goal can be integrated or not. That means from a set of optional scenarios, any quantity of these scenarios can be chosen, at the rank to zero to all. Hence, the optional scenarios can be modeled by means of an optional relationship.

Fig. 2 depicts the variation point from the relationship between goals and scenarios. For example, a goal, “Provide entertainment service to customers” can be achieved by lots of concrete ways, which are represented as scenarios. Scenarios such as “the HIS provides a movie service on TV”, “the HIS provides electronic games on TV”, or “the HIS provide home shopping on TV” can be possible ways. In these cases, domain experts and analysts should decide which scenario should be common, alternative, or optional. This is determined based on the goal providing the rationale for the variations. For example, if the scenario, “the HIS provides a movie service on TV” has a common type, it should be involved in all products in a product family. If the other scenarios are either alternative or optional, they can be involved in any products or not in other products. Therefore, variant point comes from the relationship between goals and scenarios.

### 3.3. Domain requirements level

In this paper, we reorganize domain requirements collection in a four levels abstraction hierarchy, namely, business, service, interaction, and internal level based on Rolland et al. (1998a). This can be useful in clarifying the concerns of requirements and help separating concerns in requirements elicitation. Domain requirements level also helps to stop at which level goal and scenario modeling is processed. This was proved useful when applying the approach to the ELEKTRA real case (Nurcan et al.,...
Each level reflects its own aspect of requirements. That is, the business level shows what the business goal of an organization is. The service level shows which services can achieve the business goal, and how the scope in a product line is determined. The requirements in the product line are identified at the perspective of interaction between the system itself and external entities, namely agents at the interaction level. The internal level shows the functionality among objects of system. This is a convenient way to elicit domain requirements through goal and scenario modeling because these levels make it possible to refine the goals (Rolland et al., 1998b) and show at which level goal and scenario modeling should end.

- **Business level.** The aim of the business level is to identify the ultimate purpose of a product family. At this level, the goal is given by any kind of organization or persons. The business goal is represented as a business strategy or as a business objective. For HIS example, the business goal “Become the best provider in HIS market within 10 years” is a final purpose of the organization.

- **Service level.** The service level addresses identifying the product plan or marketing plan, recognizing what kinds of products in a product line will be developed, and deciding their characteristics. At this level, the domain requirements at this level are represented as a pair \((G, Sc)\) where \(G\) is a service goal and \(Sc\) is a service scenario. All of service goals correspond to a given business goal. For example, the service goal “Provide high-end products to customers” is one possible way of satisfying the business goal (The high-end product contains more functions). For another example, the service goal “Provide low-end products to customers” is another possible way of corresponding the business goal (The low-end product contains less functions than the high-end product). Service scenarios describe the flow of services, which are necessary to fulfill the service goal. For example, the service scenario “customers get entertainment services through High-end HIS” achieves the service goal “Provide high-end products to customers.”

- **Interaction level.** At the interaction level, the focus is on the interactions between the system and its external entities. The domain requirements at this level are represented as a pair \((G, Sc)\) where \(G\) is an interaction goal and \(Sc\) is an interaction scenario. These interactions are required to achieve the services assigned to the system at the service level. The interaction goal “Provide entertainment services to our High-end product customers” expresses a way of providing a service. The interaction scenario describes a flow of interactions between the system and agents. For example, the interaction scenario “the High-end HIS helps customers to watch TV” and “customers can play a video game through High-end HIS” can achieve the interaction goal.

- **Internal level.** The internal level focuses on what the system should provide to satisfy the interactions selected at the interaction level. At this level, the domain requirements are represented as a pair \((G, Sc)\) where \(G\) is an internal goal and \(Sc\) is an internal scenario. For example, “operate a video game” is an internal goal. The associated internal scenario describes the flow of interactions among the system objects to fulfill the internal goal. For example, “the entertainment manager connects a main game server on Internet” is the internal scenario.

### 3.4. Goal and scenario modeling process for a product line

The prior proposals (Griss et al., 1998; Kang et al., 1990) take into account only what are to be considered in order to identify domain requirements in products of a product family, not how the domain requirements are to be identified and analyzed. This section discusses how goal and scenario modeling for a product line is implemented when identifying the domain requirements corresponding to the domain requirements level. The process of goal and scenario modeling is shown by using HIS example.

#### 3.4.1. The structure of goal and scenario modeling process

Goal and scenario modeling is conducted in each domain requirements level. Through this modeling, the domain requirements including common and variant requirements are identified. Fig. 3 shows the structure of goal and scenario modeling process.

As shown in Fig. 3, at first, the business goal is given by the organization. In general, the business goal represents the directional value of the organization. The business goal is achieved by one or more service goals. The service goal explains what the scope of the product line is, what kinds of the characteristics are included in each product, and what the production plan or marketing plan of the future is. From the service level, goals have the corresponding scenarios showing the functions of products with the variant types such as common, alternative, and optional. The goals are yielded from the scenarios at the previous level and they author the corresponding scenarios. This process is done at from the service level to the internal level. During this process, common and variant requirements are identified by goals and scenarios. The following sections explain how our process can be implemented by using HIS example.

#### 3.4.2. HIS example of goal and scenario modeling process

- **At the business and service level**
  
  In the HIS example, let us assume that a company “ABC” has projected a multibillion-dollar market for HISs. The company intends to become a major vendor with two initial HIS products: a low-end product (LE-HIS)—a small system with a few services, and a high-end product (HE-HIS) with additional services. The key marketing strategy of this company is to build scalable products that allow budget-conscious customers to start with a small system and then grow to a
bigger one by adding additional services instead of buying new products.

The intention of the company as mentioned above drives a business goal “Become the best provider in HIS market within 10 years (BG: Business Goal)”. This business goal becomes concrete by means of the service goals. Service goals are influenced by the marketing strategy of the organization. The marketing strategy of “ABC” is to divide HIS market into a low-end product HIS and a high-end product HIS responding to budget-conscious customers. Due to this marketing strategy, service goals corresponding to their business goals are generated as follows: “Provide high-end products to customers (Sg1: Service goal 1)” and “Provide low-end products to customers (Sg2: Service goal 2)”. The goal Sg1 is related to the high-end products and Sg2 is related to the low-end products, which means that the organization can choose either Sg1 or Sg2. In the instant developing time, if Sg1 is chosen for the marketing plan, Sg2 is exclusive. Otherwise if Sg2 is chosen, Sg1 is exclusive. Only one of two service goals must be developed. Thus two service goals have an alternative relationship with each other.

As shown in Figs. 2 and 3, service scenarios are generated in the same way as a goal is implemented by scenarios. Because Sg1 fulfills the marketing strategy for a high-end product, Sg1 has the following scenarios (Ss) corresponding to high-end services (see Table 1).

Otherwise because Sg2 fulfills the marketing strategy for a low-end product, Sg2 has fewer scenarios than Sg1. A high-end product fulfilled by Sg1 includes Ss1 “Customers control climate through HIS remotely”, Ss2 “Customers secure their home through remotely”, and Ss3 “Customer enjoy an entertainment service on TV”. A low-end product fulfilled by Sg2 includes Ss1 “Customers control climate through HIS remotely” and Ss2 “Customers secure their home through remotely”. Because both Ss1 and Ss2 are involved in Sg1 and Sg2, all of Ss1 and Ss2 have a common relation type. However Ss3 is just involved in Sg1, not Sg2. Thus Ss3 has an optional relationship. Fig. 4 depicts goals and scenarios at business and service level so far in the HIS example.

- At the interaction level

Goals at the interaction level are derived from scenarios at the service level as illustrated in Fig. 3. The scenarios yield thus possible interaction goals at the interaction level. For example, Ss1, Ss2, and Ss3 yield goals at interaction level: “Control climate remotely (IAG1: Interaction goal 1)”, “Secure home remotely (IAG2: Interaction goal 2)”, and “Provide an entertainment service to customers (IAG3: Interaction goal 3)”. These goals inherit all characteristics (e.g., variant type) of the scenarios at the previous level. Then, the domain experts and analysts choose the interaction goals among the possible interaction goals.

<table>
<thead>
<tr>
<th>Sg1</th>
<th>Ss1</th>
<th>Ss2</th>
<th>Ss3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Provide high-end products to customers</td>
<td>Customers control climate through HIS remotely</td>
<td>Customers secure their home through HIS remotely</td>
<td>Customers enjoy an entertainment service on TV</td>
</tr>
<tr>
<td>Sg2</td>
<td>Ss1</td>
<td>Ss2</td>
<td>Ss3</td>
</tr>
<tr>
<td>Provide low-end products to customers</td>
<td>Customers control climate through HIS remotely</td>
<td>Customers secure their home through HIS remotely</td>
<td></td>
</tr>
</tbody>
</table>

Table 1
Goals and scenarios at service level in HIS example (Ss: service scenario)
At business and service level

At the interaction level

Interaction level

At the internal level

At business and service level

Next, the interaction scenarios are generated to achieve the interaction goals, which should reflect the characteristics of their own level. Fig. 5 shows the goals and the associated scenarios at the interaction level. When an interaction goal is achieved by the scenarios, these scenarios are also authored with common, optional, or alternative type. The variant type of the interaction scenarios is determined according to the way that each interaction scenario achieves its own interaction goal. In Fig. 5, there are five scenarios corresponding to IAg1. All of IAs1 (Interaction scenario 1), IAs2, and IAs4 are essential for IAg1. However, IAs3 and IAs3’ have a different variant type according to whether manually or not. On the one hand, IAs3 supports a control climate service automatically. On the other hand, IAs3’ supports a control climate service manually, which means that HIS operates an inner climate both automatically and manually. The way that HIS operates a climate service depends on customers’ favor. If IAg1 is assigned to “Control climate remotely economically”, IAs3’ could not be generated because an automatic climate control is generally rather expensive than a manual climate control. Therefore, a goal decides which variant type a scenario has.

• At the internal level

The goal and scenario modeling process is done at the internal level in the same way that is done at the interaction level. Of course internal goals and internal scenarios are generated to correspond to the characteristics at the internal level. At this level, internal goals are yielded from the scenarios at the previous level in Fig. 5. The internal goals and scenarios supplement goals and scenarios at the interaction and service level in more detail from the interaction among the internal objects of view. Fig. 6 depicts a partial example at the internal level. For a goal INg2, it has five scenarios representing the interaction flows of objects within the inner system. The objects interacting with each other are HIS engine, Communicator, and Discriminator. HIS has a role that coordinates all objects in HIS. Communicator is responsible for communicating with external objects. Discriminator checks whether connection information is valid or not.
3.5. How do goals provide the rationale of variation?

In this paper, goals are used to analyze domain requirements. The output from goal and scenario modeling process is a goal tree with four levels. Since domain requirements are analyzed in terms of goals and a goal tree, all goals have their own relationship among themselves such as a vertical relationship and a horizontal relationship. The combination with horizontal relationship and vertical relationship describes how goals provide the rationale of variation.

The parent goals contain customers’ needs, marketing strategy, business plan, and so on. When a parent goal is refined to one or more child goals, they have different variation types. Their various variation come from different ways to achieve their parent goal, various objects used to achieve their parent goal, and different directions for objects to move (object, direction, and ways are already mentioned in Section 3.1). For example, in Fig. 5, it is preceded to control a climate in two ways. One is an automatic control. The other is a manual control. These variations come from different ways. These different ways come from a goal “Control climate remotely in various ways”. Thus rationale of variation of child goals depends on their parent goal. The fact that goals provide the rationale of variation makes it possible to manage variations when requirements are modified, deleted, and added.

4. Variable use case diagram

The success of use cases to capture and communicate functional requirements for single systems has generated ideas to utilize use case driven approaches for product lines. However, classical use cases are not sufficient to especially represent variability among products in a product family (Griss et al., 1998; John and Muthig, 2002; Maßen and Lichter, 2002). Recently, there have been some approaches in order to extend use cases for product lines and especially for expressing variability. If use case diagrams support the modeling of functional variability, they can be used to describe common and different characteristics among products in a product line. Thus, in the paper, the variable use case diagram for a product line based on Bertolino et al. (2002), John and Muthig (2002) and Maßen and Lichter (2002) are used to express explicitly the domain requirements. However, they do not support how the domain requirements are identified, and where use cases come from. Therefore, we suggested an approach for identifying the domain requirements and providing the rationale for them with namely, goal and scenario modeling. After goal and scenario modeling for a product line, the domain requirements identified are thus represented in the variable use case diagram.

The variable use case model should be extended by two new relationships: option and alternative. These new relationship is represented as stereotype. All use cases have variant type marked by stereotype. A particular product in a product line is produced by the composite of these use cases.

5. Transfer guiding rules

In this section, we propose how the domain requirements can be transferred to variable use case diagram. In order to do it, the transfer rules should be proposed. The following subsections describe how the domain requirements are transferred into the variable use cases, the transfer rules, and the example of them.

5.1. The core idea of the transfer rules

The core idea of this approach is that the goals and scenarios at the interaction level of the domain requirements level are used to help to identify and construct use cases. A goal at the interaction level is achieved by scenarios and a use case contains the set of possible scenarios to achieve the goal. This is due to the fact that, in our approach, the interaction level focuses on the interactions between the system and its agents, and the purpose of the use case specification is to describe how the agent can interact with the system to get the service and achieve his/her goal. Fig. 8 shows the relationships among agents, goals, scenarios and use cases.

In Fig. 8, an actor is derived from all agents who interact with the system. An actor has one or more goals that he/she wants to achieve by interacting with the system. In the HIS example, the customer (one of the actors) desire HIS to control a climate remote in various ways. Thus, the customer as an actor has his/her own goal, “control a...
When real systems are developed, we need to determine how the goal can be achieved and implemented. The scenarios can explain how to achieve the goal by the flow of events or actions. In our approach, a goal is mapped to a use case. This means that the goal is transformed into a use case for a chunk of the functional requirements of the system (as stated above, the interaction goal is used to identify and create a use case). The use case contains the scenarios for achieving that goal.

After analyzing each use case, it is augmented with internal activity elements of software-to-be. The goals and scenarios at the internal level represent the internal behavior of the system under consideration to achieve the goals of the interaction level. Accordingly, use cases can be performed by achieving the goals at the internal level, and completed by integration of the scenarios at that level.

5.2. Transfer guiding rules

We have developed the common transfer patterns in order to guide the transfer of the domain requirements to the variable use cases. The formalization of these patterns results in the current set of guiding rules. These rules are generic in the sense that they can be applied to the transfer of many domain requirements. In this section, each rule is introduced via the following template (Definition, Comment). The definition describes the point of each rule. The comment explains the things to be considered when applying the rules.

Transfer guiding rule 1 (T1)
Definition: Goals at the interaction level become use cases.
Comment: The key idea of this rule is that each goal at the interaction level is mapped into a use case because at the interaction level the focus is on the interactions between the system and its agents. This definition is similar to the definition of the use case. A goal names a use case as shown in Fig. 8, which means a goal becomes a use case. For example, there are three interaction goals in Fig. 5. All of three goals can become three use cases. All use cases inherit the characteristics of the corresponding goals.

Transfer guiding rule 2 (T2)
Definition: An agent who wants the interaction goal to be achieved can become a primary actor.
Comment: This actor is called as a primary actor, who interacts with a use case in the beginning. It could be a human, a machine or the external systems.

Transfer guiding rule 3 (T3)
Definition: Scenarios become the inner specification of a textural use case and an agent in either direction or subject of the scenario authoring template defined in Section 3.1 can become an actor.
Comment: In rule 3, an agent not becoming a primary actor becomes a secondary actor. In general, actors are found in the interaction scenarios. For example, in Fig. 5, five scenarios corresponding to IAg1 have ‘Customers’ agents in a slot of subject and direction. Thus an actor associated with IAg1 is ‘Customers’.

Transfer guiding rule 4 (T4)
Definition: Goals with variant requirements, at the interaction level, become variant use cases with variant type. If a goal has the ‘Alternative’ relationship, a use case with $\text{Alternative}$ stereotype is represented in a use case diagram. If a goal has ‘Optional’ relationship, a use case with $\text{Optional}$ stereotype is represented in a use case diagram.
Comment: The notation is based on the prior proposals such as Bertolino et al. (2002), John and Muthig (2002) and Maßen and Lichter (2002).

Transfer guiding rule 5 (T5)
Definition: Goals and scenarios at the internal level are described in each textual use case specification.
Comment: When the interaction goal becomes the use case, it has its own specification specified by the associated internal goals and scenarios. For example, as shown in Fig. 6, INg2 and the associated scenarios are textually involved in “control a climate remote” use case derived from IAg1.

5.3. The HIS example for transfer guiding rules

This section shows how the transfer guiding rules can be used in the HIS example. After goal and scenario modeling process, there are three interaction goals and the corresponding scenarios.

Fig. 9 partially shows a variable use case diagram showing the domain requirements in the HIS example. By rule...
1. Usecase Name: Remote control for a climate

2. Brief description:
   1. Customers connect HIS remotely
   2. HIS checks the authentication of customers
      2.1 The HIS engine asks the communicator to receive the id and password of customers
      2.2 The Communicator asks the discriminator to identify whether id and password are reserved
      2.3 The Discriminator returns the result of identification to the communicator
      2.4 The communicator returns the result of identification and the state of customers' contact to customers
      2.5 The HIS engine returns the result and the state of customers' contact to customers
   3. Customers ask HIS to control a climate manually
   4. Customers ask HIS to control a climate automatically
   5. HIS returns the result of control climate service

3. Variant Type: Common

4. Candidate Actors: 1) Customer

Fig. 10. The textual specification of a use case: remote control for a climate.

T1, all interaction goals can become use cases: Remote control for a climate (from IAg1), Remote Security (from IAg2), and Entertainment service (from IAg3). In the case of actors, customer (from IAs1), police (among scenarios corresponding to IAg2), and TV station (among scenarios corresponding to IAg3) become actors by rules T2 and T3. By rule T4, each use case had its own variant type. For example, a use case “Remote control for a climate” has common variant type. By rule T5, the internal goals and scenarios are described the textual as shown in Fig. 10.

6. Supporting environment

This section introduces a tool supporting fully goal and scenario modeling process and transfer guiding rules. The input to the tool is domain requirements. The outputs from the tool are goal lists, textual specifications of all the use cases, and XML based use case specification, which can be shown as a graphical diagram in CASE tools such as Rational Rose (http://www-306.ibm.com/software/rational/) and Together (http://www.borland.com/together/). This tool is available at a URL (http://selab.sogang.ac.kr/~canon/dream.zip). How to use the tool and the characteristics of this tool are explained below.

The tool consists of 4 tabs corresponding to the four levels of abstraction for requirements. The first tab shows a process at business level and service level (see Fig. 11). The second tab shows a process at interaction level (see Fig. 12). The third tab shows a process at internal level. The last tab shows use cases derived from goals and their textual specification (see Fig. 13). The first tab and the second tab have a possible goal list at upper right side, not business goal description.

At the business and service level, we firstly describe a business goal at 2. A service goal is described according to goal template at 3. After ‘Goal Generate’ button is clicked, a service goal appear at 4. A service scenario achieving its service goal is authored according to scenario template at 3. After ‘Scenario Generate’ button is clicked, a service scenario appear at 5. After the whole process is finished at business and service level, it is required to click ‘next’ button for continuous goal and scenario modeling process at 6.

Fig. 11. The first tab supporting a process at business and service level.
At the interaction level, most of the second tab is similar to the first tab. However, the second tab has a candidate goal list instead of business goal description (see Fig. 12). A candidate goal list has six columns: candidate goal id, number, scenario, C&V, relationship, and select. Candidate goal id is a sequential number of a candidate goal. Number is an id of scenario at previous level. Scenario is a scenario name combined with each element of scenario template. C&V is a variant type which a candidate goal has. Relationship is a special relation which is worth referring. Select plays a role in choosing whether a candidate goal is used at lower level or not. The third tab is similar to the second tab at the point of functions of the tool.
In the third tab, candidate use cases are generated after ‘next’ button is clicked. The last tab shows candidate use cases and their textual specification.

Candidate use cases are listed at (5). A candidate use case list consists of use case name, description, variant type, actors, and select. Use case name is the name derived from an internal goal. Description is a textual specification of actors, and select. Use case name is the name derived from an internal goal. Description is a textual specification of actors, and select. Actors are feasible actors related to their own use cases. Select plays a role in choosing whether a candidate use case is used in a real project or not.

After the whole process, the XML based use case specification is generated as shown in Fig. 14.

XML based use case specification can be easily imported and used in other CASE tools for supporting specific product development in a product line.

7. Related work

This section surveys related work in domain analysis especially commonality and variability (C&V) analysis from the perspective of software product lines. In order to analyze commonality and variability among products in a product line, feature base approaches like Griss et al. (1998), Kang et al. (2002) and Vici and Argentieri (1998) have been used widely in industry and academia since the FODA method (Kang et al., 1990) was introduced in 1990 by the Software Engineering Institute. They attempt to analyze commonality and variability in terms of features. Feature models that illustrate a structural view on a system are concentrated on the representation of common and variant requirements. However, the feature based approaches do not provide a well defined process to identify features and C&V. Therefore the practical use of FODA can be constrained in industry (Griss et al., 1998). On the other hand, both FeatureRSEB (Griss et al., 1998) and FODAcom (Vici and Argentieri, 1998) have used used-case models with feature models for C&V analysis. Product Line Analysis (PLA) (Chastek et al., 2001) combines traditional object-based analysis with FODA for a product line analysis. However, unlike our approach, the prior methods have not provided a systematic and concrete mechanism for identifying common and variant requirements as well as the rationale for them.

There are some approaches in order to extend use cases for product lines. Reuse-Driven Software Engineering Business (RSEB) (Jacobson et al., 1997) introduces “variation points” into use case diagrams and also uses variation points in textual use cases. It does not say anything about how variant or generic use cases can be identified and
instantiated. FODAcom, FeatuRSEB and PLA mentioned previously do not provide how C&V should be integrated and described in use case diagrams and textual use case descriptions. In Maßen and Lichter (2002), to model explicitly the types of variability, the use case meta-model extended by two new relationships, in other words, optional and alternative are introduced. And, in John and Muthig (2002), to utilize use cases for product line modeling, they are extended with a variability mechanism. However, in comparison with our approach, the two approaches (John and Muthig, 2002; Maßen and Lichter, 2002) focus on only modeling use cases for product lines and representing variability. Consequently, the approaches given above have no proposal to systematically identify commonality and variability as well as use cases for product lines, and provide the rationale for them. Moreover, as has been pointed out, the results of C&V analysis should satisfy an organization’s high-level business goals. Therefore, C&V and use cases must be identified to satisfy these goals and the rationale for those must be provided. Yet, the approaches do not clearly take into account both of them.

8. Conclusion

The domain requirements (C&V requirements) should be identified and represented based on the business goals of an organization, which drive all differences among products in a product family. This paper proposed the systematic approach to identify the variant requirements founded on the business goal, and represent them by the variable use cases. In particular, first, goal and scenario modeling for a product line was proposed, which enables one to identify the variant requirements with the rationale for them by the requirements traceability links. It is very meaningful to provide the rationale for the variations among products since the rationale makes it possible to handle the variant requirements on the basis of the goals. Second, the variable use cases with the transfer guiding rules, which represent the domain requirements for the product family, were suggested. The transfer rules help to represent unambiguously the domain requirements. Finally, we introduced the supporting tool. It helps to analyze domain requirements and to generate variable use case diagram semi-automatically. The final output from the tool is a variable use case diagram in a format of XML. Thus this tool can support use case driven approach in the aspect of the representation and analysis of the domain requirements, and also be extended for developing a particular product within a product line.

Due to the three characteristics mentioned above, this proposal is original in comparison with other approaches which did not provide the systematic process identifying the domain requirements. Now we are studying more complicated goal and scenario modeling and transfer guiding rules. An empirical validation of the approach is planned for the near future.

Acknowledgements

We would like to thank the guest editor, Professor Bae for considering our paper to be included as part of the Journal of Systems and Software special issue. This work was supported by University IT Research Center Project (ITRC) in Korea.

References

Griss, M.L. et al., 1998. Integrating feature modeling with the RSEB. In: Proc. 5th Int. Conf. on Software Reuse (ICSR’98), Victoria, BC, Canada, pp. 76–85.

938


Jintae Kim is a Ph.D. candidate in the Department of Computer Science and Engineering at Sogang University. He is going to receive the Ph.D. degree in Computer science in 2005. He received his B.S. degree from Sogang University in 2000. From 2000 to 2001, he worked for Daicom System Technologies as a software engineer. In 2003, he received the Best Paper Award in Korea Conference on Software Engineering. His current research interests are Software variability management, Adaptive software, and Requirements Engineering. He is a member of the Korea Information Science Society and IEEE Society.

Minsseong Kim received his B.S. degree in computer science from Sogang University, Korea, in 2001. From January 2001, he was a teaching assistant at Department of Computer Science and Engineering at Sogang University, until he received the M.S. degree in 2003. In 2002, he received the Paper Award from Korea Information Science Society. He is currently a Ph.D. candidate and his research interests include software product lines, requirements engineering, and self-managed systems.

Sooyong Park is an Associate Professor of Computer Science Department at Sogang University. He received his Bachelor of Engineering degree in Computer Science from Sogang University, Seoul, in 1986, the Master of Science degree in Computer Science from the Florida State University, in 1988, and the Ph.D. in Information Technology with major in Software Engineering from George Mason University, in 1995. In 1994, he received an American Institute of Aeronautics and Astronautics (AIAA) Software Engineering Award. He served as a Senior Software Engineer at TRW ISC during 1996–1998. His research interests include requirements engineering, embedded and adaptive software development, software architecture.