Goal and Scenario-based Feature Identification Techniques from Legacy System

Jeong Ah Kim¹, HakJoon Lee², Rhan Jung³,* and Suntae Kim³

¹Dept. of Computer Education, Kwandong Univ
²Dept. of Management Information System, Kwandong Univ
³Department of Computer Engineering, Kangwon National University
clara@kd.ac.kr, dlgkrwns@gmail.com, jungran@kangwon.ac.kr,
Jipson08@gmail.com

Abstract

Since many existing systems have been still working even these system have several problems in extending or maintaining. This is big obstacle for company to adopt the product line engineering. Extractive approach can be alternative approach when the company has their own system and adopt product line engineering. In extractive approach, commonality and variability modeling is critical also. In this paper, we suggested feature identification techniques using user interface and scenario-based analysis based on goals. With user interface scenario we can identify the features and we can group the features and make the hierarchy among the features with goal-based analysis.

Keywords: goal & scenario, feature identification, legacy system, extractive approach, software product line

1. Introduction

Software product line engineering enables the development of reusable architecture and core assets by analyzing various applications present in a single domain and identifying similarities and dissimilarities, and hence similar future software products can be effectively produced. The software development framework focuses on upgrading reusability rather than a systematic designed approach for developing a software product line. With this objective, as an alternative to developing a single system using newer technologies, the framework instead seeks to develop different products through reusable software assets [1–3].

Various introduction strategies are adopted for developing a software product line; among them, the prominent ones are the proactive, reactive, and extractive approaches. According to a survey entitled “A Survey of Variability Modeling in Industrial Practice” [4] carried out by Variability Modeling of Software-intensive Systems (VaMoS) in 2013, the extractive approach was most commonly used in companies. In the extractive approach, a software product line is developed from established and operational legacy systems.

The feature model is the key technology adopted for developing a software product line. As with requirement analysis, a great number of studies have been carried out on methods for developing a feature model in the early phase of development. Studies on techniques

* Corresponding author

Department of Computer Engineering, Kangwon National University
jungran@kangwon.ac.kr
for extracting features from a system, which are used in practice in operations, however, are few and far between.

This paper proposes a technique based on goals and scenarios to identify features from the user interface of a legacy system. This technique includes the following three methods: 1) initially, creating a user interface navigation scenario model through user interface navigation analysis by the operating system, 2) creating a target and scenario model through a target and scenario modeling step, and 3) finally creating a feature model by identifying features through the feature identification step.

2. Related Research

2.1. Software Product Line Engineering

Software product line engineering is a technique for developing a new system by ensuring sufficient core assets through analysis of similarities and dissimilarities by considering functional aspects and quality of various systems belonging to the same domain, and then combining these assets to develop a new system.

The Feature-Oriented Reuse Method (FORM) from the Pohang University of Science and Technology Software Engineering Laboratory [5], Product Line Practice (PLP) from CMU/SEI, United States [6], Algebraic Hierarchical Equation for Application Design (AHEAD) from the University of Texas [7], and Product Line Software Engineering (PuLSE) from Fraunhofer IESE, Germany, are the prominent software product line development methodologies.

The prominent approach methods of software product line development are proactive, reactive, and extractive. Most software product line methodologies adopt the proactive approach, which involves domain analysis, design development, and so on, performed in order.

In the reactive approach, the software product line evolves reactively when the initially applied software product line is no longer relevant owing to changes in client requirements or software upgrade requirements.

The extractive approach is similar to re-engineering that is adopted in software engineering, which focuses on developing the core assets of a product line, based on existing products. The extractive approach facilitates effective software reusability, because it consumes less time and cost compared to the proactive approach. The refactoring process adopts a strategy different from other approaches develops core assets by reverse engineering the product line, which enable it to identify similarities and dissimilarities from previously developed systems. This approach could prove effective if the existing system or product can be reused, or in particular, if the similarities between the systems are great and the dissimilarities are consistent.

2.2. Goal and Scenario Model

Goal-based analysis using scenarios provides scenario generation methods for accomplishing goals, and it further provides methods for setting relationships between goals [8]. Requirement analysis and methods based on goals and scenarios are widely used in requirement deduction and traceability. The abstraction level of goal- and scenario-based requirements are divided as business level, service level, interaction level, and internal level [9, 10].

2.3. Feature Model

The feature model has been recognized as an efficient tool for facilitating effective communication between domain experts and developers by systematically expressing the existing domain assets in terms of similarities and dissimilarities between products.
Existing feature-centric methods make it possible to identify features by providing feature categories in a feature identification framework. In other words, in the case of a fully developed stable domain, domain features provided by a feature category are identified by analyzing domain terminology [11]. The feature model comprises four layers in order to express the various perspectives: capability, domain technology, operation environment, and implementation technique.

3. Feature Identification Technique Based on Goal-based Scenarios

This study proposes a feature model and consistency validation rule for identifying features from a legacy system by defining user interface navigation analysis, goal and scenario modeling, and feature identification.

3.1. Feature Identification Process Based on Goals and Scenarios

The process of identifying features from a legacy system consists of three steps. Output is created through operations based on each step, and then features are finally identified. Table 3-1 provides the definitions of activities and output of each step.

<table>
<thead>
<tr>
<th>Step</th>
<th>Activity</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Writing output through a user interface navigation analysis step from a legacy system</td>
<td>- User interface navigation - User interface navigation scenario</td>
</tr>
<tr>
<td>2</td>
<td>Writing the output of the first step by applying a goal and scenario modeling step</td>
<td>- Goal and scenario model</td>
</tr>
<tr>
<td>3</td>
<td>Applying the output of the second step to the feature identification step and writing the outputs by combination</td>
<td>- Feature model</td>
</tr>
</tbody>
</table>

Through these steps, the user interface navigation scenario, goal and scenario model, and features can be identified, and correlation relationships can be defined between them for constructing a feature model.

3.1.1. User Interface Navigation Analysis

The user interface navigation analysis step involves writing scenarios by analyzing the actual drive screen of a legacy system. Two outputs are written during this step. First and foremost, the user interface navigation of a legacy system is written by analyzing it, and then the user interface navigation scenario between users and system is written.

- **User interface navigation** captures the progression of events in order, from the main screen by driving the legacy system, and further captures all screens till the final screen of the event progression.
- **The user interface navigation scenario** is written based on the created user interface navigation. Grouped single functions are selected from the user interface navigation. Scenarios between a user and system for each screen are written in the order of event progression in the grouped screens.
3.1.2. Goals and Scenario Modeling

The goal and scenario modeling step is written based on the written user interface navigation scenario, corresponding to the four levels of the goal and scenario model. Table 3-2 shows the writing order of the goal and scenario model.

Table 3-2. Goal and Scenario Model Writing Order

<table>
<thead>
<tr>
<th>Order</th>
<th>Activity</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Selecting one written user interface navigation scenario, and writing from the mutual interaction level.</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Sequentially writing system scenarios starting from writing user scenarios.</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>After completion of writing scenarios, writing a command function for a user interface navigation scenario over a relevant scenario.</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>After writing the mutual interaction level, grouping the similarities between goals.</td>
<td>- Goal and scenario model</td>
</tr>
<tr>
<td>5</td>
<td>Writing the goals of the mutual interaction level in service-level scenarios through grouping.</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>After writing scenarios for the service level, writing the goals of relevant scenarios in the function command</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>After completion of writing goals and scenarios for the service level, writing goals and scenarios for the business level by a similar method.</td>
<td></td>
</tr>
</tbody>
</table>

Written user interface navigation scenarios are listed in order from the start of the event until the termination of the event, and then the goals of the listed scenarios are written. Common goals at the mutual interaction level are grouped and written by service-level scenarios. Likewise, the goals of service-level scenarios are written.

3.1.3. Feature Identification

In the feature identification step, the goals and scenario model written through user interface navigation scenarios are applied to the capability layer of the feature model. From the user’s perspective, the capability layer of the feature model is made up of the important features of a system.

Goals and scenario models are applied to the capability layer of the feature model by distinctly representing the correlation between low- and high-level features by grouping similar goals. Goals for the service level are written by first-level features; furthermore, the goals of the service level are written by top-level features of the capability layer to help users understand service functions.

First-level features are written by first writing the goals for the mutual-interaction level by low-level features. Top-level features, i.e., the goals for the service level, possess relevant scenarios by low-level features, and the relevant scenarios become the goals of the mutual interaction level. Third-level features are written from the perspective of users by extracting the important features of a system from among the scenarios of the mutual-interaction level.

The content among the user scenarios, which include events (input, selection, etc.), is written by the lowest-level feature as it is input to the capability layer. In the same way as above, top-level features and low-level features of the capability layer can be identified by mapping goals and the scenario model to relevant feature levels through the feature identification step. Only, create, read, update, and delete (CRUD) are not included in the features.
3.2. Case Study

The defined feature identification process was applied to an already developed system, and the output of each step was written. Then the output results were written by feature model and further validated by comparing with the feature models developed by domain experts.

In the present study, the feature identification process was applied to a teaching behavior analysis system. Teaching behavior analysis systems are used by teaching behavior analysts for calculating operational efficiency of presently used teaching methods by monitoring and analyzing teaching and teaching participants.

Second- and third-level features were identified in the user interface navigation analysis, and in the next step, first-level features were identified through the goals and scenario model and finally mapped with low-level features. Thus, the feature model was written through mapping of the features.

The existing proactive approach of identifying features consumes more effort, time, and cost. However, maximum time and effort in identifying features was saved by using the proposed process. The proposed process is very useful for product linearization in the initial development stage by identifying features of a legacy system that was developed without considering the product line. Compared to the feature model using the existing development paradigm, the feature model with a level similar to that of a feature model of a teaching behavior analysis system developed using the proactive approach, was written.

4. Conclusions

Companies are hesitant to introduce software product lines due to the initial cost, time, and other factors. The extractive approach is receiving attention as a solution to these limitations, and it can enable the companies to introduce software product lines more confidently. Companies developing products by the existing software development methods cannot afford such an adventure of changing the development paradigm by invalidated theoretical methods.

According to a survey by VaMoS [5], the extractive approach is the most widely used method around the world. Many companies have attempted to incorporate the extractive approach by changing the existing development paradigm. In the future, this approach will be widely used in companies. The methods for constructing a feature model for a software product line is, however, implicit based on the company, and the model is constructed by identifying features using random methods that are based only on assets.

The feature identification process was proposed in this study to overcome this disadvantage and provide a favorable extractive approach for developing a software product line. A process was defined for application to the capability layer of a feature model from the existing system using a goal and scenario model. The defined process can be used to identify features based on outputs written by each step using the actual system.

References


