Supporting Commonality and Variability Analysis of Requirements and Structural Models

Kentaro Kumaki
Ryosuke Tsuchiya
Dept. Computer Science
Waseda University
Tokyo, Japan
ken-kumaki@fuka.info.waseda.ac.jp
ryousuke_t@asagi.waseda.jp

Hironori Washizaki
Dept. Computer Science,
Waseda University
GRACE Center, National Insitute of Informatics
Tokyo, Japan
washizaki@waseda.jp

Yoshiaki Fukazawa
Dept. Computer Science
Waseda University
Tokyo, Japan
fukazawa@waseda.jp

ABSTRACT

The commonality and variability analysis of legacy software assets requires high costs in terms of personnel and time in extractive core asset development. We propose a technique for supporting the commonality and variability analysis, targeting the requirements and structural models of legacy software assets for the development of a feature diagram and a product line architecture (PLA). We analyze the commonality and variability of the sentences as requirements and classes as structural models by calculating similarities based on a vector space model. By using our technique, the costs in terms of personnel and time required for the analysis of legacy software assets can be reduced.

Categories and Subject Descriptors

D.2.13 [**Reusable Software**]: Reuse models

General Terms

Design, Documentation

Keywords

Feature Diagram, Product Line Architecture, Traceability

1. INTRODUCTION

There are three approaches to developing core assets in software product line development: proactive, reactive, and extractive[1]. In the proactive approach, core assets are developed prior to software development. In the reactive approach, common and variable components are repeatedly developed during software development. In the extractive approach, existing legacy software assets are analyzed and organized to extract core assets.

In the information industry marked by rapid changes in technology, market, and environment, the risk associated with the development of core assets by the proactive approach is high because the prediction of software assets that will be necessary for future development is difficult. Therefore, rather than the proactive approach, it is more reasonable to develop products in specific domains to a certain degree, then extract them as core assets once their quality is confirmed[2]. Therefore, the development of core assets by the extractive approach is considered to be effective.

In the extractive core asset development, we need to analyze the commonality and variability of requirements and structural models to extract the commonality and variability characteristic of a product line, and to develop a product-line architecture (PLA). However the work associated with the manual commonality and variability analysis of the requirements of many legacy software assets is complicated and has high cost.

Moreover, to reuse the PLA as core assets effectively, it is necessary to determine traceability links between features and structures. However, it is usually difficult to determine traceability links because structural models are developed to realize various functional and non-functional requirements.

2. SUPPORTING FEATURE DIAGRAM AND PLA DEVELOPMENT

To address the above-mentioned problems, we propose a technique for analyzing the commonality and variability of requirements and structural models automatically to support developing a feature diagram and a PLA.

2.1 Overview

We use a set of sentences as requirements and a set of classes represented in design-level UML class diagrams as structural models[1]. Targeting these materials, we analyze commonality and variability using a vector space model, and traceability links between sentences and classes focusing on requirements and structural models from which sentences and classes are extracted.

By automating the analysis process, the costs in terms of personnel and time required for the commonality and variability analysis can be reduced. In addition, our tech-

[1]We do not consider the multiplicity, association name, role, or package of class diagrams.

nique recommends traceability links between requirements and structural models. On the basis of the results of the commonality and variability analysis and the recommendation of traceability links, we aim to support the development of a feature diagram and a PLA with clear traceability.

The target of our technique is a product line developed by a single development group. The purpose of restricting the domain range is to suppress the variety of inputs. Inputs are multiple combinations between product requirements and structural models. The output is the results of the commonality and variability analysis of sentences and classes, as well as the recommended traceability links between sentences and classes. We support developing a feature diagram and a PLA by showing these outputs.

In our technique, it is assumed that the requirements and class diagrams, which are the input, are the correspondence. The overview of our technique is shown in Figure 1. The automatic support by our technique entails step 1 to 3.

1. Analyze the commonality and variability of sentences as requirements.
2. Analyze the commonality and variability of classes as structural models.
3. Recommend traceability links between requirements and structures.
4. SPL experts develop a feature diagram and a PLA.

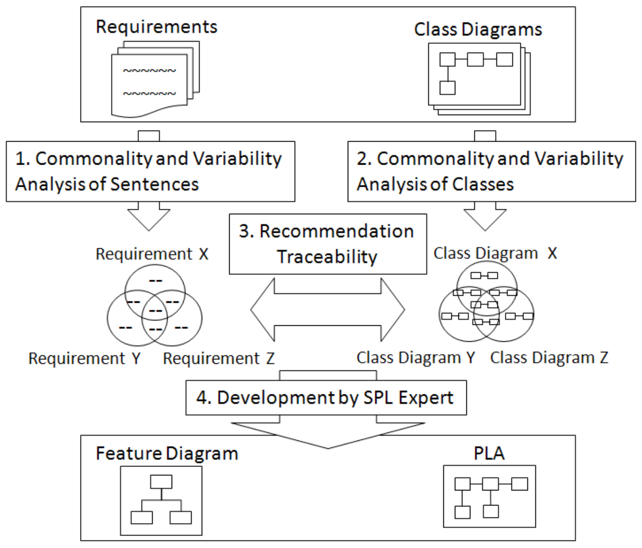
Requirements
Class Diagrams
1. Commonality and Variability Analysis of Sentences
2. Commonality and Variability Analysis of Classes
3. Recommendation Traceability
Requirement X
Requirement Y
Requirement Z
Class Diagram X
Class Diagram Y
Class Diagram Z
4. Development by SPL Expert
Feature Diagram
PLA

Figure 1: Overview of our technique

2.2 Commonality and Variability Analysis of Sentences

We deal with sets of sentences so that we can use the vector space model proposed by Salton et al [3] which can measure the similarity between sentences. By using the vector space model, we measure the similarity between sentences to extract similar sentences from all the combinations of product requirements as the commonality and variability analysis of sentences. With such a vector space model, a sentence is represented by one vector depending on valid words appearing in the sentence and that are nouns, verbs, and adjectives in our technique. When valid words, i.e., v_1, v_2, $\cdots$, v_M, are extracted from a sentence S_x in the vector space model, the M-dimensional vector $\overrightarrow{d_x}$ is expressed as

$$\overrightarrow{d_x} = (\mathrm{w}(v_1,\ S_x),\ \mathrm{w}(v_2,\ S_x),\ \cdots,\ \mathrm{w}(v_M,\ S_x)).$$

Where $\mathrm{w}(v_p,\ S_x)$ $(1 \leq \mathrm{p} \leq \mathrm{M})$ is the number of appearances of v_p in the S_x. Here, the similarity between the two sentences S_i and S_j is obtained as the cosine of the angle between the two sentence vectors $\overrightarrow{d}_i$ and $\overrightarrow{d}_j$(cosine similarity), which can be obtained by dividing the inner products of $\overrightarrow{d}_i$ and $\overrightarrow{d}_j$ by the magnitudes of $\overrightarrow{d}_i$ and $\overrightarrow{d}_j$. We define $\mathrm{SSim}(S_i,\ S_j)$ (Sentence Similarity, $0 \leq SSim \leq 1$) using the cosine similarity as follows:

$$SSim(S_i, S_j) = \frac{\overrightarrow{d}_i \overrightarrow{d}_j}{|\overrightarrow{d}_i||\overrightarrow{d}_j|}.$$

As an example of the cosine similarity calculation, let us consider the following two sentences: "Trace a line" and "Trace a running line". The sentence vectors of the two sentences can be disassembled as in Table 1.

Table 1: Sentence vectors of two sentences

Sentence	trace	running	line
Trace a line	0	1	1
Trace a running line	1	1	1

By using the sentence vectors in Table 1, the cosine similarity is calculated as

$$SSim(\text{"Trace a line", "Trace a running line"})$$
$$= \frac{1*0+1*1+1*1}{\sqrt{0^2+1^2+1^2}\sqrt{1^2+1^2+1^2}} \fallingdotseq 0.819$$

2.3 Commonality and Variability Analysis of Classes

We measure the similarity between classes using the vector space model to extract similar classes from all the combinations of class diagrams as the commonality and variability analysis of classes. With the vector space model, a class C_y is represented by one N-dimensional vector $\overrightarrow{d_y}$ depending on valid elements, i.e., e_{y1}, e_{y2}, $\cdots$, e_{yN}, which are class names, operations, and attributes appearing in the class. Here, we distinguish class names, operations, and attributes even if the strings of elements are the same. For example, operations are compared with operations and not with class names nor attributes. We define the similarity between two classes C_i and C_j as $\mathrm{CSim}(C_i,\ C_j)$ (Class Similarity, $0 \leq CSim \leq 1$) using the cosine similarity as follows:

$$CSim(C_i, C_j) = \frac{\overrightarrow{d}_i \overrightarrow{d}_j}{|\overrightarrow{d}_i||\overrightarrow{d}_j|}.$$

$$\overrightarrow{d}_i \overrightarrow{d}_j = max \sum SSim(e_{is}, e_{jt})$$

$sim(e_{is}, e_{jt})$ is the cosine similarity between e_{is} and e_{jt}. When we calculate $\overrightarrow{d}_i \overrightarrow{d}_j$, all e_{is} and e_{jt} are used only once and we do not calculate similarities between different types of elements, like attribute and operation.

As an example of the class similarity calculation, let us consider the two classes in Figure 2. First, class names, attributes, and operations are assumed to be sentences for calculating the cosine similarity and determining the combinations of high similarity. The cosine similarity is calculated using the sum of the above similarities as the inner product and is used as the class similarity.

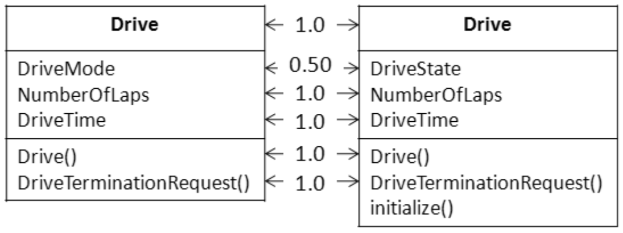

Figure 2: Example of measureing class similarity

$$CSim(\text{Drive, Drive}) = \frac{1.0 + 0.5 + 1.0 + 1.0 + 1.0 + 1.0}{\sqrt{6}\sqrt{7}}$$
$$\fallingdotseq 0.85$$

2.4 Recommendation of Traceability Links between Sentences and Classes

We focus on the combinations of requirements and that of class diagrams from which sentences and classes are extracted to recommend traceability links between sentences and classes. Figure 3 shows the core idea of determining traceability links between sentences and classes. The venn diagram on the left is classification of sentences based on from which requirements sentences are extracted, and that on the right is classification of classes based on from which class diagrams classes are extracted.

A set of classes extracted from class diagrams in a set of legacy software assets is considered to be designed to realize a set of sentences extracted from requirements in the same set of legacy software assets. Therefore, we assume that there are correspondences between sentences and classes extracted from requirements and class diagrams in the same set of legacy software assets.

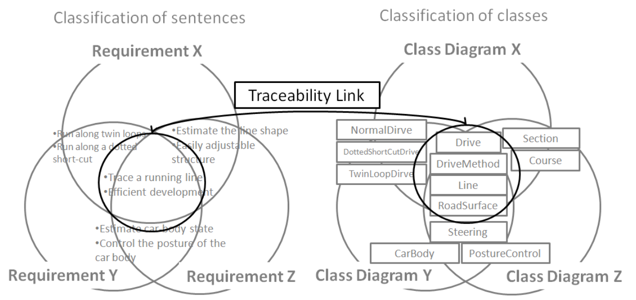

Figure 3: Traceability links between sentences and classes

For example, the two sentences "Trace a running line" and "Efficient development" and the four classes "Drive", "DriveMethod", "Line", and "RoadSurface" are both extracted from requirements and class diagrams in a set of legacy software assets, {x,y,z} in Figure 3. Therefore, we consider that there are traceability links between these two sentences and four classes.

3. CASE STUDY

We conducted experiments to evaluate the usefulness and validity of our technique by using data obtained from a team that participated in the experiment (participating team).

3.1 Setting

The target domain is the ET Software Design Robot Contest (ERC)[4]. In the ERC, participants compete in terms of the running performance of robots designed to autonomously run on a course by sensing the black line using optical sensors. The legacy software assets of a group that participated in the ERC for four consecutive years were used in the experiment. Each asset is composed of 31–69 (46 on average) sentences in requirements and 16–29 classes (21 on average) in structural models. Some of those sentences and classes are shown in Figure 3.

A participating team was a team who participated in the ERC 2011 championship, consisted of six Master's students experienced in software development, and developed combinations of a feature diagram and a PLA with the output of our technique. The results of the experiment were evaluated by comparing the followings.

- SOM: a set of a feature diagram and a PLA with definite traceability links developed and reviewed by the participating team using our technique.
- STA: a set of a feature diagram and a PLA developed and reviewed by the participating team without using our technique as the answer.

By this comparison, we evaluated the level of the reduction in the costs required for the commonality and variability analysis of legacy software assets and the validity of the recommended traceability links between sentences and classes. Figures 4, 5 show excerpts of a feature diagram and a PLA of SOM. There are traceability links between features and classes having the same shade and frame.

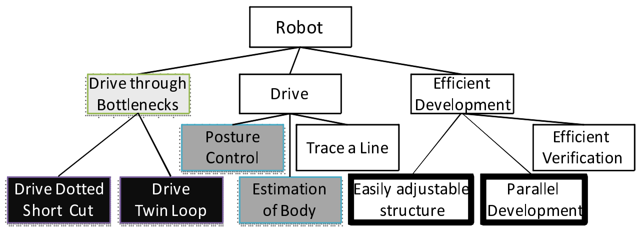

Figure 4: Feature diagram of SOM

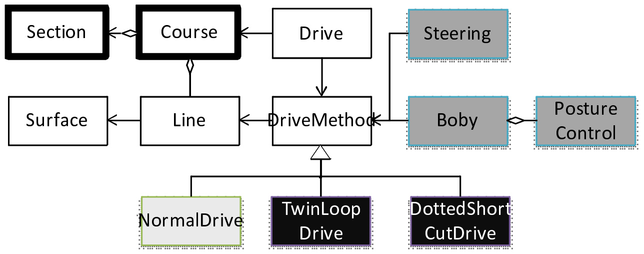

Figure 5: PLA of SOM

3.2 Cost

Table 2 shows the numbers of features and classes of SOM and STA as the indication of the scales of the feature diagram and the PLA, and the man-hours required for the development of combinations between the feature diagram and the PLA for SOM and STA. As shown in Table 2, the numbers of features and classes are almost identical among SOM and STA. We can say that SOM and STA have feature diagram and PLA of similar scales. And, the man-hours of SOM were less than half that of STA.

We observed that most of the efforts in the development of SOM were spent for organizing the feature diagram and adjusting the resulting PLA to be appropriate according to the feature diagram, since the set of feature candidates with commonality and variability and the PLA candidate were automatically extracted by our technique. From this result, our technique is expected to reduce the cost required for developing combinations of the feature diagram and the PLA.

Table 2: Scales and man-hours

	Features	Classes	Man-hours
SOM	109	50	11.6
STA	100	48	24.6

3.3 Validity and usefulness of analysis

To confirm the validity of SOM, we calculated the percentage of correct features and classes in SOM that correspond to features and classes in STA, and also calculated the graph distance[5] between feature diagrams in SOM and STA and between PLAs in SOM and STA. The graph distance can measure the distance between two graph structures like feature diagrams or PLAs.

We obtained the percentage of correct features was 73.4% and that of correct classes is 92.0%. The graph distance between the feature diagrams in SOM and STA was 0.88, and that between the PLAs was 0.1. From these results, our commonality and variability analysis approach seems to be effective for extracting classes and features, and organizing a structure of classes as PLA in this experiment. However the graph distance between the feature diagrams is quite high so that it is necessary to introduce additional techniques for supporting feature diagram modeling.

Moreover we observed that many of traceability links of SOM were valid and useful for supporting future product-line development since these links include not only functional requirements but also non-functional ones.

For example the variable structure regarding the course setting presented by "Course" aggregating a set of "Section" in Figure 5 can be traced from the features "Easily adjustable structure" and "Parallel Development" in Figure 4. Usually such non-functional requirements with those realization parts are hard to specify due to the nature of non-functional requirements cross-cutting over realization structures; our technique is useful to recommend such complex traceability link.

4. RELATED WORK

There are several linguistic and heuristic-based approaches for constructing feature models from mainly requirements documents[6] together with some code base[7]. These conventional approaches could be utilized for supporting feature modeling after applying our technique to a set of different project requirements documents in the same domain.

Moreover there are several linguistic and information retrieval-based techniques for recovering traceability links between requirements and other software materials, such as code[8]. Although the ideas of utilizing linguistic engineering for recovering traceability are close, our technique is different from the viewpoint of context and target; our technique is beneficial to support developing PLAs by analyzing commonality and variability in a set of different project results.

Nomoto et al. proposed a technique for extracting analysis patterns as common parts from the combination of class diagrams and requirements documents[9]. In contrast, Our technique deals with variability in addition to such commonality in structural models and requirements documents.

5. CONCLUSION

We propose a technique for supporting commonality and variability analysis of requirements and structural models of legacy software assets with the aim of supporting the development of feature diagrams and PLAs. By the experimental evaluation, we confirmed that our technique is useful to obtain feature diagrams and PLAs that are similar to those developed manually, with reduced cost. Our future work includes experimental validity evaluations using large-scale complex legacy software, and further studies to support feature modeling and the development of PLAs.

Acknowledgments

We would like to thank Tomoji Kishi of Waseda University, and Hideyuki Kanuka, Kentaro Yoshimura, and Ryota Mibe of Hitachi, Ltd. for interesting discussions on SPL.

6. REFERENCES

[1] C. Krueger. Eliminating the adoption barrier. *Software, IEEE*, 19(4):29–31, 2002.

[2] K. Yoshimura, D. Ganesan, and D. Muthig. Assessing merge potential of existing engine control systems into a product line. *International Workshop on Software Engineering for Automotive Systems (SEAS'06)*, 2006.

[3] G. Salton and M. J. McGill. *Introduction to Modern Information Retrieval.* McGraw-Hill, New York, 1983.

[4] http://www.etrobo.jp/

[5] H. Bunke and K. Shearer. A graph distance metric based on the maximal common subgraph. *Pattern Recogn. Lett.*, 19:255–259, March 1998.

[6] N. Weston, et al. A framework for constructing semantically composable feature models from natural language requirements. *13th International Software Product Line Conference (SPLC'09)*, 2009.

[7] S. She, et al.: Reverse Engineering Feature Models. *33rd International Conference on Software Engineering (ICSE'11)*, 2011.

[8] G. Antoniol, et al. Recovering Traceability Links between Code and Documentation. *IEEE Transactions on Software Engineering*, 28(10):970–983, 2002.

[9] Y. Nomoto, et al. Automatic extracting of analysis pattern based on similarity of structure and word. *2nd Asian Conference on Pattern Languages of Programs (AsianPLoP 2011)*, (III):106–115, 2011.