A Feature-Based Service Identification Method to Improve Productivity of Service-Oriented System

Dongsu KANG†, Member, CheeYang SONG††, and Doo-Kwon BAIK†∗, Nonmembers

SUMMARY This paper proposes a feature-based service identification method to improve productivity using a feature relationship; where a feature can express service properties. We define the distance measured between features by considering their selective (node) and relational (edge) attributes and present the service boundary concept. The result of an evaluation of the proposed method shows that it has higher productivity than existing methods.

key words: service identification, feature model, productivity

1. Introduction

Software product lines (SPLs) and service-oriented architectures (SOAs) are two concepts that are currently receiving much attention both in research and for practical application [1]–[3]. The feature model is an SPL tool that can be used for modeling the common variable requirements of a product in a software product line. However, current methods of feature modeling are not based explicitly on service orientation with feature organization. Various extensions of feature models have been introduced to compensate for their purported ambiguity and lack of precision and expressiveness. Therefore, we propose a feature-based service identification method to make services that support the reuse of service-oriented principles. The proposed method reflects the business requirement in binding features. In addition, it provides services with high cohesion, which enables improved modularity. Better modularity results in improved productivity in developing a service as a software product.

2. Feature-Based Service Identification Method

2.1 Binding Features

Binding features refer to clustering features that contain similar concepts in the feature model. Features with similar functional and non-functional characteristics are grouped according to the top-level requirement. The final products of this activity are service candidates. The primary objective is to produce service candidates that are based on the business requirements.

(1) Setting a key feature. A key feature is a top-level requirement of a service-oriented system for grouping features. Any feature that satisfies a business requirement can be a key feature of the feature model, as well as a top-level requirement feature of a service-oriented system. Thus, we recommend that a feature with business relevance should be a key feature in the top layer.

(2) Grouping features. Feature binding involves finding similar semantics between a key feature and other features. A feature group that has commonality is a highly reusable fundamental building block of the product line. Thus, the features of a mandatory selective attribute or a composed of relational attribute have priority as features of a service candidate for reusability during feature binding. Most feature types of a service candidate are mandatory attributes, whereas the boundary features are optional, alternative, and or attributes.

2.2 Relational and Selective Counting between Features

The distance between a feature and an adjacent feature can be measured by counting relational (edge) and selective (node) attributes because these attributes configure the feature model for a service. The distance between features can be expressed in terms of their relational and selective attributes on the basis of the feature attributes. This concept simplifies the counting of edges and nodes because the counting weight for the Is-a relation is set to zero as a base line, and the weights of the Part-of and Has-part relations are set to one considering strong cohesion. In addition, the Part-of and Has-part relations represent composition, whereas the Is-a relation represents generalization. The following formula justifies the proposed distance counting with respect to the relation in Eq. (1) and selective attributes in Eq. (2).

Relational (edge) Attributes:

\[ C(r) = \text{Weight}_{\text{type}}(A \rightarrow_r, B) \]
\[ = 0, \quad \text{for Generalized-by and Implemented-by} \]
\[ \quad \text{edge types} \]
\[ = 1, \quad \text{for the Composed-of edge type} \]

Here, \( \rightarrow_r \) is the semantic distance of relation attributes, and A and B are features.

Selective (node) Attributes:

\[ C(s) = \text{Weight}_{\text{type}}(A \rightarrow_s, B) \]
Example 1 (Calculation of Distance)  The measured distance between features F1 and F2 is two, which implies that the respective scores of the relational and selective attributes are one and one, as shown in Fig. 1.

2.3 Refining Service Candidates

Refining service candidates generally involves modifying, adding, and eliminating features from service candidates according to the service boundary concept. When a new service candidate is constructed, refinement involves eliminating or avoiding conflicts between the features of the service candidate to achieve a high-quality service. Refinement of service candidates involves the service boundary and uses the formulae for counting relational and selective attributes. The service candidates can be classified as tightly or loosely coupled according to their boundary (i.e., edge-based or node-based). The service boundary provides a proper distance from a key feature \( f_R \) to a boundary feature \( f_i \). Thus, a key feature relates to the number of features within a radius \( R \). Therefore, the service boundary can be defined as in Eq. (3).

\[
\text{Service boundary } [f_R, R] = [f_i] \text{ such that } \\
\forall i \ D(f_R, f_i) \leq R \text{ for } 1 \leq i \leq n
\]  \hspace{1cm} (3)

Here, \( D() \) is the distance between a key feature \( f_R \) and other features \( f_i \) within radius \( R \).

\[
R = \sum_{i=1}^{m} D(f_R, f_i), \quad D(f_R, f_i) = \sum_{j=1}^{n} \frac{P_j}{e + n},
\]

Here \( P_j \) is the weighted sum of relational and selective attributes, \( R_k \) is the weight of relational attributes, \( S_k \) is the weight of selective attributes, \( e \) is the number of edges, \( n \) is the number of nodes, and \( m, i, j, k \) are the numbers of features.

3. Evaluation

3.1 Exploratory Case Study

To ensure the reliability of the proposed feature-based service identification, an exploratory case study is considered to illustrate the application of the Threat Evaluation System (TES) of the Command and Control System (CCS) for Network Centric Warfare (NCW). The feature-based service identification involves binding features and refining service candidates by counting edges and nodes. There are 108 top-level requirements of threat evaluation for NCW. The method identifies 35 services for reuse in the same kinds of TES product lines.

The following describes the components of service identification in the top-level requirements of TES. The components of the top-level requirements are Auto-evaluation, Weapon Recommendation, and Response Priority to hostiles. The features of the TES feature model are bound according to the binding rule. First, the Auto-evaluation, Weapon Recommendation, and Response Priority features are selected as key features according to the business requirements. Second, features of mandatory type are grouped with the key feature. Each candidate is then named and identified as a service candidate. Finally, three service candidates from the TES are captured; Auto-evaluation \( \text{AE} \), Weapon Recommendation \( \text{WR} \), and Response Priority \( \text{RP} \). Figure 2 illustrates service identification derived using the TES feature model.

3.2 Evaluation of Performance

Cohesion is an indicator of how strongly the various features of a service are focused and related. A service with high cohesion tends to be preferable because of improved modularity. A service comprises features with functional and logical cohesion. Functional cohesion means constancy of the set of features, and logical cohesion requires a set of similar features. This type of cohesion ranges from zero (low cohesion) to one (high cohesion). Service cohesion is based on the service boundary representing the distance between features in a service. That is, it is measured using the semantic boundary given below.

\[
\text{Service Cohesion} = \sum_{i=1}^{n} \frac{\text{Relationship}(i)}{\text{Feature}(i)}
\]  \hspace{1cm} (4)

In this formula, \( \text{Feature}(i) \) represents the number of
feature nodes and edges, and Relationship(i) is the distance between features. The cohesion range for the weights is between 0 and 1.

The values of service cohesion of [AE], [WR], and [RP] are 0.50, 0.44, and 1.00, respectively. The service cohesion of [RP] means that the service has a reflexivity property. [RP] has 50% and 56% greater cohesion than [AE] and [WR], respectively. Figure 3 illustrates the service cohesion of the identified services, and Fig. 4 shows the cohesion rates of the identified services.

Finally, we use models for software reuse developed by Gaffney and Durek [4] to measure productivity because a service is a unit of a module such as a component. This model is described by Eq. (5).

\[
\text{Productivity} = \frac{1}{(b-1)R + 1}
\]  

(5)

\(R\) is the proportion of reused features in the service \((R \leq 1)\), \(b\) is the relative cost (for all new features) of incorporating the reused features into the new service \((b = 1\) for all new features). The value of \(b\) depends on the life-cycle phase of the reusable service. Figure 5 shows the productivity fluctuation according to the number of services, determined using the productivity formula. For the proposed method, productivity increases steadily as the number of services increases, whereas there is no productivity for other methods.

### 4. Related Works

Medeiros et al. [5] proposed an approach for service-oriented product line architectures that combine the SPL and SOA concepts and techniques to achieve high customization for systematic planned reuse. Salvador et al. [6] focused on a scenario in which different products from different product lines were combined to form a third product line. Helferich et al. [7] compared SPL and SOA, and showed that both the methods are actually complementary. In any case, a further step is still needed to improve the benefits of their combined use. Our method can be used to achieve desirable benefits such as improved reuse and the production of flexible applications that are adaptable to the requirements of a business by means of feature binding, with key features of business requirement, feature relationship, and the service boundary concept.

Fareghzadeh [9], S. Kim et al. [10], and V. Dwivedi et al. [11] proposed business-driven service identification methods using business process, business goal, and scenario. In contrast, F. Chen et al. (2005) [12], Z. Zhang et al. [13], and F. Chen et al. (2009) [14] proposed reengineering methods for a legacy system that considered features, architecture, and ontology. However, our approach supports both the business aspect, by feature binding using
key features related to top level requirements, and the reengineering aspect by means of the service boundary concept, which calculates the distance between features. This leads to increased traceability back to the business and provides services with high cohesion, which makes improved modularity possible.

Another approach to using a feature-oriented analysis to identify services is described in J. Lee et al. [8]. This method is based on a feature analysis technique that enables us to identify the services of a service oriented system as molecular services. The feature binding analysis consists of feature binding unit identification and binding time determination. It also orchestrates supports for reliable and mobile molecular services. In contrast, our feature binding is used to produce service candidates using a set of features for business requirements to reflect business aspects. Although J. Lee et al. used the same binding time concept for feature binding, our feature binding uses the selective (node) and relational (edge) attributes of features. In addition, the method of J. Lee et al. requires time information and a domain expert’s professional judgment to orchestrate services. In contrast, we can refine the service candidates mechanically by using the service boundary concept, which is calculated using the feature attributes for service cohesion. This makes it possible to avoid conflicts between features with appropriate granularity for reuse. Thus, it maximizes the modularity and reusability of services by providing a service boundary in accordance with the refinement. The service boundary provides services with high cohesion and enables improved modularity, which in turn improves the productivity and allows the development of a service-oriented system.

5. Conclusion

We aimed to improve service modularity and the productivity of a service-oriented system. We proposed a feature-based service identification method for a software product. The evaluation results showed that service modularity enables service reuse and improves the productivity of service-oriented product lines.

Acknowledgments

This paper was supported by the Second Brain Korea 21 Project, Korea University, 2010 and the National IT Industry Promotion Agency (NIPA) under the program of Software Engineering Technologies Development.

References