RESEARCH ON COMPONENT COMPOSITION BASED ON FEATURE MODEL

Lirong Xiong[1,2], Zibin Niu[2]
[1]*College of Computer Science and Technology, Zhejiang University, HangZhou, China*
[2]*College of Computer Science and Technology, Zhejiang University of Technology, Hangzhou, China*

Jing Fan
College of Computer Science and Technology, Zhejiang University of Technology, Hangzhou, China

Keywords: Component semantics, Trustworthiness, Component composition, Feature meta-model, Credit rating.

Abstract: Traditional component description methods lack sufficient semantic information. It is difficult for users to find suitable components to match their requirements. And it is also difficult in doing automatic composition and verification of components. Furthermore, the component trustworthiness is an important factor that must be considered during component composition process. Feature-Oriented Domain Analysis (FODA) that uses features and relations between features to describe the problem domain can provide necessary supports to component composition. This paper focuses on component functional semantics and component trustworthiness. A feature meta-model is proposed to provide sufficient semantic information, and trustworthiness is taken into account in the feature meta-model. Based on feature meta-model, it presents the component composition algorithm. Finally, an application of the approach in the credit evaluation domain is presented.

1 INTRODUCTION

The technology of software reuse is one of hotspots in the field of computer software development. Software Product Line (SPL) is one of the successful approaches to achieve large scale software reuse (Yuqin Lee, 2006). SPL is first introduced by SEI of Carnegie Mellon University (CMU/SEI) that gets the idea from product line of traditional manufacturing. The procedure of SPL is divided into two stages: domain engineering and application engineering. CMU/SEI has established FODA (Kyo C Kang, 1990) method which introduces feature and feature model in the stage of domain engineering in 1990. Now, the concept of feature model has been widely adopted in domain requirements capturing and specifying (Xin Peng, Wenyun Zhao, Yunjiao Xue and Yijian Wu, 2006). Feature model provides necessary supports for component composition by using features and relations between features to describe the problem space in specific domain.

The main challenge of component composition is to judge the usefulness of components. Functional semantics of a component is an important factor for users to understand and judge the usefulness (Xin Peng, Wenyun Zhao and Leqiu Qian, 2006). So it is necessary to give the definition of component functional semantics. Traditional component description methods lack of sufficient semantic information. For this reason, it is difficult for users to find suitable components to meet their requirements. And automatic composition and validation of components still are challenges, while the research of component semantics based on FODA has attracted lots of attentions. The structure of component semantics is described from three aspects: domain field, definition field and context field in domain analysis (JIA Yu and GU Yu-qing, 2002.). However, this approach is abstract and it is hard to go into operation. Peng Xin et al (Xin Peng, Wenyun Zhao and Leqiu Qian, 2006) present component semantics and they construct semantic composition algorithm based on a feature meta-model, while the feature meta-model ignores dependency relations between features. Haining Yao and Letha Etzkorn (Haining Yao, 2004) focus on

Xiong L., Niu Z. and Fan J..
RESEARCH ON COMPONENT COMPOSITION BASED ON FEATURE MODEL.
DOI: 10.5220/0003484302140222
In *Proceedings of the 13th International Conference on Enterprise Information Systems* (ICEIS-2011), pages 214-222
ISBN: 978-989-8425-55-3
Copyright © 2011 SCITEPRESS (Science and Technology Publications, Lda.)

how to classify and retrieve components based on component semantics, but they ignore information for component composition.

In the stage of application engineering, the component trustworthiness has a very strong effect on component composition. Because the trustworthiness of Component-Based System (CBS) is depended on the component trustworthiness. Recent years, trustworthiness has attracted increasing attentions among researchers. But most researchers focus their study on measurement and guarantee of component trustworthiness in application engineering. Domain analysis is the first step in software product line and an essential activity for successful reuse. And features are viewed as first-class objects throughout the software life cycle and across the problem and solution domain (Carlton, 1999). It is necessary to consider trustworthiness in feature modelling of domain analysis.

This paper presents feature meta-model based on functional semantics and trustworthiness. This meta-model provides sufficient semantics information and takes trustworthiness into account. On the basis of feature meta-model, this paper describes the relevant definition of component semantics and the algorithm of component composition. Finally, we apply our approach in credit rating domain.

The remainder of the paper is organized as follows: Section 2 presents a feature meta-model which analyzes feature dependence based on functional semantics and discusses component trustworthiness from many aspects. Section 3 defines component semantics, provides a component composition algorithm based on functional semantics, and introduces the component composition process. Section 4 illustrates and analyses our approach by an example of credit rating domain. Section 5 draws a conclusion and some suggestions for future work.

2 FEATURE META-MODEL BASED ON FUNCTIONAL SEMANTICS AND COMPONENT TRUSTWORTHINESS

Domain analysis is the basis of component composition. Feature modelling as the mainstream method in domain analysis provides good supports for the component composition.

2.1 Non-Functional Attribute in Feature Meta-Model

A feature is a prominent or distinctive and user-visible aspect, quality, or distinctive characteristic of a software system or systems (Michael, 2010). The attributes of a feature are divided into non-functional ones and functional ones. Non-functional attributes describe the characteristics of a feature, such as the feature name and feature description. Functional attributes reflect the functional dependencies of a feature with other features. For example, one feature requests some data provided by another feature. So there is a dependency named HasDataDep between the two features. In Figure 1, the non-functional attributes of a feature consist of Name, Description, Facet, BindingTime, BindingState, etc.

Name: This attribute is the name of a feature. It is the unique identifier of a feature. The type of Name is a string.

Description: This attribute is the description of a feature. It is a brief description of a feature, including function and application domain of the feature. The type of Description is a string.

Mandatory: This attribute denotes whether a feature is mandatory or not. The range of Mandatory is Boolean type. The value“True”means this feature is mandatory, while the value “False”means this feature is optional.

BindingTime: This attribute presents the time when a feature is bound. The range of BindingTime is Time. Several common types of Time include CompileTime, InstallTime, LoadTime, RunTime. Take CompileTime for example, CompileTime means the feature is bound during the program compiling phase.

BindingState: This attribute presents the binding status of a feature. The range of BindingState is State. Three kinds of State are distinguished as follows: Undecided, Bound, and Removed.

Facet: This attribute describes a feature from different perspectives, viewpoints and dimensions. It provides precise and detailed description of a feature. The range of Facet is Term. A facet maps a number of terms which make up a term space.

Map: This attribute defines the mapping relation between features and components. Usually, there may be several components contributing to a feature.

2.2 Functional Semantics in Feature Meta-Model

A feature is often considered as a set of tight-related

requirements. Therefore a feature can be decomposed into small unit features and also have relations with other features. The relation between features is reflected by feature dependencies. Kwanwoo Lee and K.C.Kang (Kwanwoo Lee and Kyo C. Kang, 2004) analyze feature dependencies that are useful in the design of reusable and adaptable product line components and extend the feature model. They present design guidelines based on the extended model. Although several common types of feature dependencies are given, they are not sufficient for development of reusable and adaptable product line assets.

Yuqin Lee et al (Yuqin Lee,2006) classify feature dependencies in both static and dynamic methods, and use directed graph to analyze domain requirement dependencies. They emphasize on how to obtain domain requirements easily and get effective mapping between requirements dependencies and feature dependencies.

This paper presents a component-composition oriented approach. The approach analyzes feature dependencies in feature meta-model. Several types of feature dependencies are given as follows: HasDecomposeDep,HasDataDep, HasCommunicate Dep,HasSequenceDep,HasParallelDep, HasControl Dep and HasConfigDep.

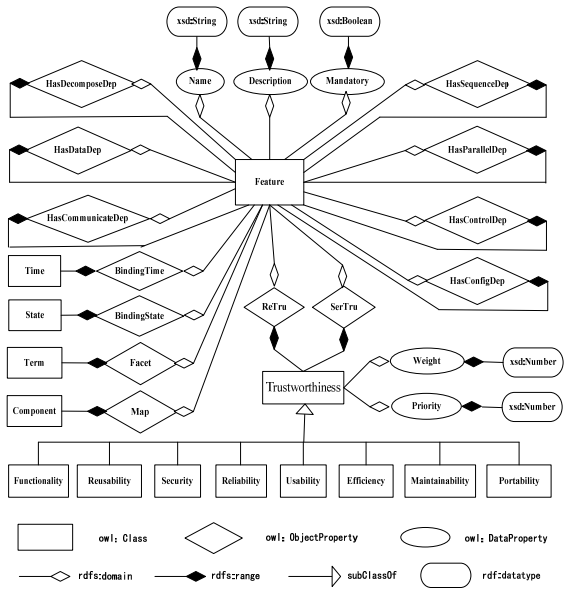

Figure 1: Feature meta-model based on functional semantics and component trustworthiness.

HasDecomposeDep: If a parent feature is decomposed into a number of child features, the dependency relation of the parent feature and child feature is named HasDecomposeDep.

HasDataDep: When one feature changes data which is used by another feature, the dependency relation of the two features is named HasDataDep.

HasCommunicateDep: If one feature sends a message to another feature and notifies it to update itself according to the message, the dependency relation of the two features is named HasCommunicateDep.

HasSequenceDep: If one feature must be active after another feature, the dependency relation of the two features is named HasSequenceDep. The feature being active first is called precondition feature, while the other is called postcondition feature.

HasParallelDep: If two or more features work together to finish a task and must be synchronized during their active period, the dependency relation of the two features is named HasParallelDep.

HasControlDep: If one feature determines the status or behavior of another feature, the dependency relation of the two features is named HasControlDep.

HasConfigDep: When one feature is bound according to another feature, the dependency relation of the two features is named HasConfigDep.

Among these features dependencies, HasParallelDep is bidirectional and the others are unidirectional. HasParallelDep describes two or more features that depend on each other and work together to finish a task.

2.3 Trustworthiness in Feature Meta-Model

A trusted component is a reusable software element possessing specified and guaranteed property qualities. The component trustworthiness includes all the quality attributes (GUO Shu-Hang, 2007). In 2003, Bertrand Meyer (Bertrand Meyer, 2003) brought forward a framework for a component quality model which is called “ABCDE”. “A” stands for acceptance, “B” for behaviour, “C” for constraints, “D” for design, and “E” for extension.

The ISO/IEC 9126 standard (GB / T 16260. ISO / IEC 9126, 2003) is a general model to specify and evaluate the component quality from different perspectives. This model includes two parts. The first part defines both internal and external characteristics. The second part defines quality in use characteristics. Based on “ABCDE” and ISO/IEC component quality models, we adopt component trustworthiness in our component-composition oriented feature meta-model. Component trustworthiness includes 8 aspects:

Functionality, Reusability, Security, Reliability, Usability, Efficiency, Maintainability, and Portability.

In the feature meta-model, two new non-functional attributes are adopted named Retru and SerTru.

Retru: This attribute defines trustworthiness a feature requires.

SerTru: This attribute defines trustworthiness a feature provides.

The range of Retru and SerTru is Trustworthiness. Trustworthiness has two attributes which is called Weight and Priority. These attributes are used to guide the development and composition of reusable components.

As to component development, trusted components are very important for trusted CBS. Component developers try to develop trusted components according to requirements of trustworthiness weight and trustworthiness priority. For example, in Large-scale real-time 3D game system, the painting and rendering component must have high efficiency to meet the players' needs of high speed. So efficiency of trustworthiness has a heavy weight and a high priority. From this way, component developers' focus is to solve the efficiency of painting and rendering components.

As to component composition, users try to find available trusted components using trustworthiness weight and priority as selection criteria. For instance, when developing E-commerce systems, for those same functional semantic components, security is the most important indicator. Components with high security are preferred.

Because ontology has good presentation skill and reasoning ability, it has been widely adopted in domain knowledge modelling. In this paper, we adopt ontology language (OWL) as the definition foundation of the feature meta-model. Figure 1 shows the feature meta-model we proposed.

In Figure 1, Feature, Time, State, Term and Trustworthiness are defined as owl Class(owl:Class). BindingTime, BindingState, Facet, ReqTru and SerTru are defined as owl ObjectProperty(owl:ObjectProperty), range (rdfs:range) of which is owl Class(owl:Class). Name, Description and Mandatory are defined as owl DataProperty (owl: DataProperty), range(rdfs:range) of which is datatype(rdf:datatype).

3 COMPONENT COMPOSITION BASED ON FEATURE META-MODEL

3.1 Component Semantics

In order to narrow the gap between the problem domain and the solution domain, an explicit and formal mapping must be used for consistency checking and automatic composition (Tijs, 2007.). In the above feature meta-model， non-functional property information of feature is presented, including Name, Description, Facet, BindingTime, BindingState, etc. The functional dependencies between features consist of HasDecomposeDep, HasDataDep, HasCommunicateDep, HasSequenceDep, HasParallelDep, HasControlDep and HasConfigDep. Base on the feature meta-model, our approach is applied for component composition on two accounts: first, features are mapped to components (artifacts), second, we regard feature dependencies as the functional dependencies between components(or artifacts) and component trustworthiness is an important factor in the process of component composition.

Definition 1: Set of feature dependencies.

FeatureDepSet={HasDecomposeDep,HasDataDep,HasCommunicateDep,HasSequenceDep,HasParallelDep,HasControlDep,HasConfigDep }

If FeatureA is dependent on FeatureB, it is recorded as (FeatureA) HasDep (FeatureB) where HasDep belongs to FeatureDepSet,. Here, FeatureA is subject, and FeatureB is object.

Definition 2: Mapping between feature and component.

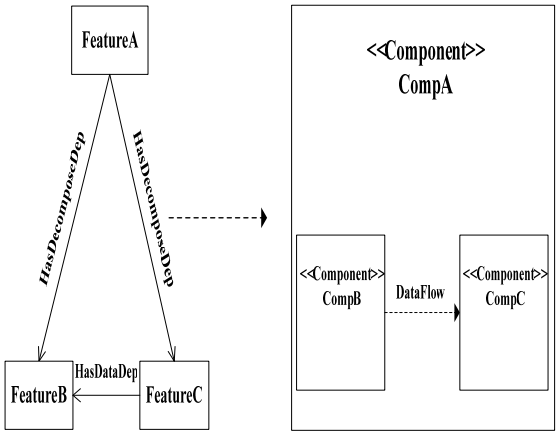

Figure 2: The mapping between features and components.

The expression “CompA.Feature” maps a component “CompA” to a feature “Feature” in feature meta-model.

The equation “CompA.Feature=CompB.Feature” means that components “CompA and CompB” are mapped to the same feature in the feature meta-model.

As shown in Figure 2, if (CompA.Feature) HasDep (CompB.Feature), we can get (CompA) HasDep (CompB).

Considering trustworthiness, if CompA requires function provided by CompB, the trustworthiness CompA requires is denoted as (CompA)ReqTru(CompB). Meanwhile, the trustworthiness which is provided by CompB, is denoted as (CompB)SerTru(CompA).

Definition 3: Component semantic.

Component Semantic::=<ZDDepSet,BDDepSet, ReqTruSet, SerTruSet >

Take a component named CompA for example, The details of each set are given below.

ZDDepSet: This set defines a relationship that a component initiatively requires other components. This relationship is reflected by the features that components map to.

CompA.ZDDepSet={(CompX.Feature)HasDep(CompY. Feature)|CompX=CompA, HasDep FeatureDepSet };

BDDepSet: This set defines a relationship that a component is required passively by other components. This relationship is also reflected by the features that components map to.

CompA.BDDepSet={(CompX.Feature)HasDep(CompY. Feature)|CompY=CompA, HasDep FeatureDepSet};

ReqTruSet: When a component requires other components, this set defines the value set of trustworthiness that a component requires other components.

CompA.ReqTruSet={(CompX)ReqTru(CompY) |(CompX.Feature)HasDep(CompY.Feature) Comp A.ZDDepSet};

SerTruSet: When a component is required by other components, this set defines the value set of trustworthiness that a component is required by other components,

CompA.SerTruSet={(CompX)SerTru(CompY)|(CompX. Feature)HasDep(CompY.Feature) CompA.BDDepSet }

The inferences from the above definitions are as follows:

1. The sufficient and necessary condition for CompA to be dependent on CompB is CompA. ZDDepSet∩CompB. BDDepSet≠∅ ;

2. The sufficient and necessary condition of meeting requirement of trustworthiness is (CompA)ReqTru(CompB)≤(CompB)SerTru(Comp A).

(CompA) ReqTru(CompB) ≤ (CompB) SerTru(CompA) is an abstract expression, which means the trustworthiness CompB provides is higher than that CompA requires.

3.2 Component Composition based on Component Semantics

In product line engineering, automatic configuration of product line instances still remains a challenge (Don, 2006). The whole process of component composition can be decomposed into a series of chain processes. Two small granular components are combined to be a big granular component. Then this big granular component and another component are combined to be a bigger granular component. In this iterative process, the software system is developed.

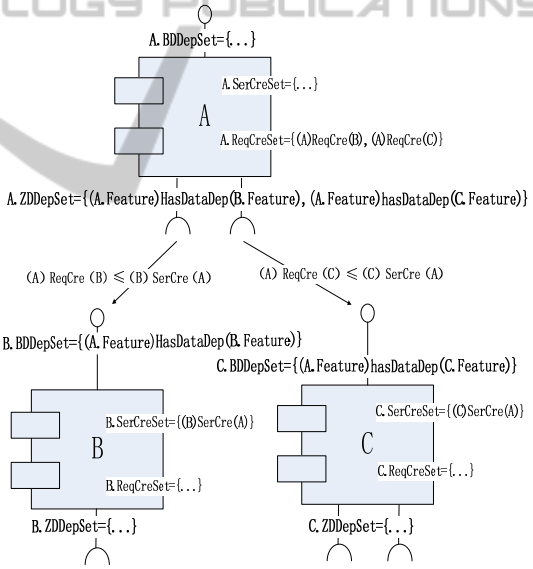

Figure 3: Composition algorithm based on component semantics.

In figure 3, the process of component composition goes as follows:

1. For two components named A and B, if A.ZDDepSet∩B.BDDepSet≠∅ , we can draw a conclusion that A is dependent on B. Then go to next step. Else, go to step 6.

2. The functions a component provides is reflected by its interfaces, including parameters, return types and exceptions etc. Start to match component interfaces. If the matching succeeds, go

to next step. Else, go to step 6.

3. Component runtime environment consists of operating platform and application domain etc. Start to match component runtime environment. If the matching succeeds, go to next step. Else, go to step 6.

4. If (A)ReqTru(B)≤(B)SerTru(A),go to next step. Else, go to step 6.

5. If the matching succeeds in other aspects, A and B are combined to form a new component named AB, and component AB's semantic is as below.

AB.ZDDepSet=A.ZDDepSet∪ B.ZDDepSet-(A.ZDDepSet ∩B.BDDepSet);

AB.BDDepSet=A.BDDepSet∪ B.BDDepSet-A.ZDDepSet∩B.BDDepSet);

AB.ReqTruSet=A.ReqTruSet∪ B.ReqTruSet-((A)ReqTru (B));

AB.SerTruSet=A.SerTruSet∪ B.SerTruSet-((B)SerTru (A)).

The composition succeeds, go to step 7.

6. The matching fails.

7. End.

3.3 Component Composition based on Component Semantics

Figure 4 shows the component composition process based on feature model. Firstly, based on domain knowledge, the feature model of the specific domain is obtained by refining features and analyzing functional dependencies of features. Then based on feature model, the ontology knowledge depository is constructed and a serial of components are developed by component developers during domain application stage. Components are notated, managed and organized in component library. Then suitable components can be easily retrieved to composite compound component based on component semantic information and trustworthiness calculation.

As to the calculation of trustworthiness, the indicators of trustworthiness have great ambiguity and fuzziness, so there are no uniform and standard methods or models to calculate component trustworthiness. It is impossible to give an absolute grade or value to indicate the trustworthiness of a component. Based on the evaluation principles, several evaluation models and approaches are proposed to measure the trustworthiness of component, such as neural network, fuzzy analytic hierarchy process, fuzzy multiple criteria decision making.

For those components that can complete the same task, the most suitable component is selected according to component's trustworthiness. Finally, the users composite a system based on component.

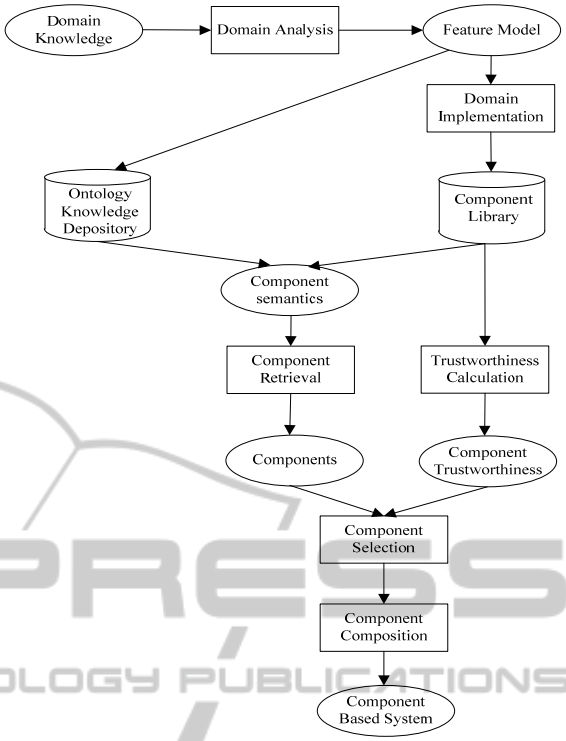

Figure 4: The component composition process based on feature model.

4 COMPONENT COMPOSITION IN CREDIT RATING DOMAIN

The above approach of feature modeling and component compositing method is applied in the design and realization of credit rating system. With rapid economic development, credit rating has become more and more important in our life. And nowadays, software systems should be much more flexible to accommodate the frequent changes of user requirements and operating environments (Sheng-Xiang Zou, 2005). In order to promote software reusability, improve software efficiency and quality, it is necessary to manage and organize the product line assets of credit rating domain.

First, a feature model of credit rating domain is obtained after domain analysis. In Figure 5, the feature dependencies in the credit rating domain are presented. There may be two or more types of feature dependencies between two features, so two or more feature dependencies are needed to describe feature relationships. In order to simplify the

SCITEPRESS SCIENCE AND TECHNOLOGY PUBLICATIONS

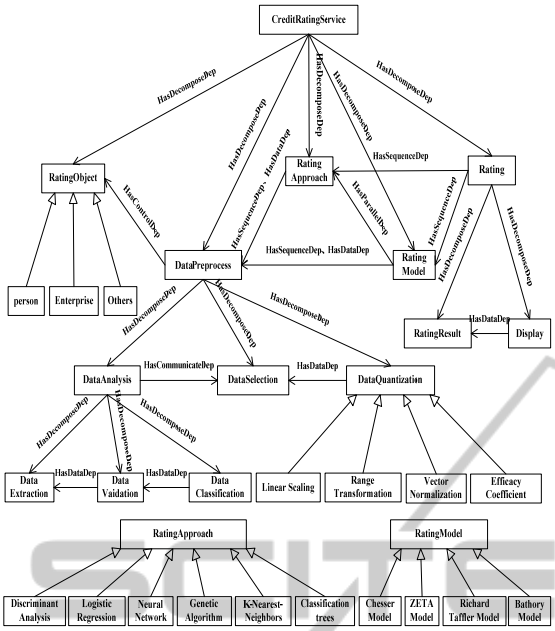

Figure 5: Feature model of credit rating domain.

relationship between features, we use only one key feature dependency.

Feature-oriented domain analysis is a time-consuming process. It is usually difficult to obtain complete and accurate feature model because of the complexity of feature dependencies. So, we simplify the feature models to get high efficiency.

Figure 6: Component Library of credit rating domain.

After developing a serial of components based on domain feature model, we have developed a component library to manage and organize these components, as shown in figure 6. The process of component composition is described briefly. Take Comp_DataPreprocess for example, according to ontology knowledge repository and composition algorithm, we easily draw this conclusion, as shown in Table 1.

Table 1: The functional dependencies of Comp_DataPreprocess.

Subject	HasDep Feature-DepSet	Object
Comp_Data-Preprocess	HasControlDep	Comp_Rating-Object
Comp_Data-Preprocess	HasDecomposeDep	Comp_Data-Analysis
Comp_Data-Preprocess	HasDecomposeDep	Comp_Data-Selection
Comp_Data-Preprocess	HasDecomposeDep	Comp_Data-Quantization
Comp_Credit-RatingService	HasDecomposeDep	Comp_Data-Preprocess
Comp_Rating-Approach	HasDataDep, HasSequenceDep	Comp_Data-Preprocess
Comp_Rating-Model	HasDataDep, HasSequenceDep	Comp_Data-Preprocess

When the interface and runtime environment between components match successfully, we decide to choose suitable components in a set of components which have the same functions. One example shows as following.

Assuming component A and B are similar components which implement the same feature DataQuantization. The expression is A.Feature= B.Feature.

Table 2: The trustworthiness information of A and B.

	A	B
Functionality	0.9	0.8
Reusability	0.7	0.8
Security	0.6	0.6
Reliability	0.8	0.9
Usability	0.7	0.6
Efficiency	0.7	0.9
Maintainability	0.8	0.6
Portability	0.5	0.7

Assuming the maximum value of trustworthiness is 1.0 and the minimum value of trustworthiness is 0. If the trustworthiness of a component is measured to be 1.0, it is absolutely trusted. If the trustworthiness of a component is measured to be 0.0, it is not trusted at all. Table 2 shows trustworthiness information of component A and B.

For components that quantize data, their Functionality and Efficiency are more important. So we give Functionality and efficiency heavy weight. Weight coefficient of the factors, in the order shown

as above table, is 30%, 10%, 5%, 10%, 10%, 20%, 10%, 5%.

The trustworthiness of A is

0.9*30%+0.7*10%+0.6*5%+0.8*10%+0.7*10%+0.7*20%+0.8*10%+0.5*5%=0.765

The trustworthiness of B is

0.8*30%+0.8*10%+0.6*5%+0.9*10%+0.6*10%+0.9*20%+0.6*10%+0.7*5%=0.775

Obviously, the trustworthiness of component B is higher than that of component A (0.775>0.765), we tend to choose B for composition.

In another way, the priority of the trustworthiness factors is set as: Functionality, Efficiency, Reusability, Reliability, Usability, Maintainability, Security, and Portability. Functionality is first priority. Functionality of A is higher than that of B (0.9>0.8). In this condition, we think A is more suitable component.

In order to get the result of credit rating, we need to composite a serial of components. Figure 7 shows the example that three component are combined to a component that completes the task of credit rating. These three components are named Comp_RatingObject, Comp_DataPreprocess, and Comp_RatigModel. Comp_RatingObject operates the object of credit rating. Comp_DataPreprocess preprocesses the data of credit rating. Comp_RatigModel is a ratig model that calculates the result of credit rating according to the credit data.

Figure 7: The example of component composition.

In figure 7, we input the credit rating object, the credit rating data and the credit rating model. Figure 8 shows the result of credit rating

Figure 8: The result of credit rating.

5 CONCLUSIONS

This paper builds feature meta-model based on functional semantics and credibility. This meta-model provides sufficient semantics information and take credibility into account. We introduce ontology as a description basis of domain feature modeling so as to achieve the formal description and automatic reasoning. On the basis of feature element model, the relevant definitions of component semantics and component assembly algorithms are presented. Finally, we adopt applications of credit rating domain to conduct the research on feature modeling, component composition, etc. The main research works of this paper are as follows:

1. This paper analyzes the dependencies between features and summarizes some functional semantics dependencies such as decomposition dependency, data dependency, communication dependency from the perspective of functional dependence. These functional semantics can be transited to component functional semantics.

2. This paper introduces trustworthiness into feature meta- model. We refine trustworthiness from Functionality, Reusability, Security, Reliability, Usability, Efficiency, Maintainability, and Portability. And then we can obtain trustworthiness parameter and set weight and priority for these trustworthiness factors. Thus, trustworthiness is regarded as an important factor to guide component development and composition.

3. Based on feature element model, this paper defines component semantics. Then, component composition algorithm based on component semantics is presented. This algorithm emphasizes on how to judge component reusability and consider credibility.

4. This paper takes credit rating domain as application example, and conduct corresponding feature modeling and component composition in this domain. And it proves that our method is feasible and effective.

Component composition is a complex process of multi-dimensional matching. When the matching fails, it's impossible to composite components. On this condition, we need to modify the component on some aspects. Based on the above work, we will research on how to adapt component when the matching fails.

ACKNOWLEDGEMENTS

This paper is supported by the subproject (No. 2-1)

of the key Science and Technology innovation team project (2009R50009), Zhejiang province.

REFERENCES

Yuqin Lee, Chuanyao Yang, Chongxiang Zhu and Wenyun Zhao, 2006. "An Approach to Managing Feature Dependencies for Product Releasing in Software Product Lines," *Lecture Notes in Computer Science, vol.4039, pp.127 - 141.*

Kyo C Kang, Sholom G Cohen, and James A Hess, 1990. "Feature-oriented domain analysis feasibility study," *SEI Technical Report CMU/SEI-90-TR- 21.*

Xin Peng, Wenyun Zhao, Yunjiao Xue and Yijian Wu, 2006. "Ontology-Based Feature Modeling and Application-Oriented Tailoring," *Lecture Notes in Computer Science, vol.4039, pp.87 - 100.*

Xin Peng, Wenyun Zhao and Leqiu Qian, 2006. "Semantic Representation and Composition of Business Components Based on Domain Feature Ontology," *ACTA ELECTRONICA SINICA. vol.34, pp.2473 - 2477.*

JIA Yu and GU Yu-qing, 2002. "Domain Feature Space Based Semantic Representation of Component," Ruan Jian Xue Bao (J. Softw.), vol.13, pp. 37 - 49. Jan.

Haining Yao and Letha Etzkorn, 2004. "Towards a semantic-based approach for software reusable component classification and retrieval," *ACM-SE 42 Proceedings of the 42nd annual Southeast regional conference, pp.110 - 115.*

Carlton Reid Turner, Alfonso Fuggetta and Luigi Lavazza, 1999. "*A conceptual basis for feature engineering," Journal of Systems and Software, vol. 49, pp.3-15.*

Michael Eichberg, Karl Klose, Ralf Mitschke and Mira Mezini, 2010. "Component Composition Using Feature Models,"*Lecture Notes in Computer Science, vol. 6092, pp.200 - 215.*

Kwanwoo Lee and Kyo C. Kang, 2004. "Feature Dependency Analysis for Product Line Component Design," *Lecture Notes in Computer Science, vol.3107, pp. 69-85.*

GUO Shu-Hang, LAN Yu-Qing and JIN Mao-Zhong, 2007. "*Some Issues about Trusted Components Research," Computer Science, vol.34, pp.243 - 246.*

Bertrand Meyer, 2003. "The grand challenge of trusted components," *Software Engineering Proceedings 25th-Interuational Conference, vol.3, pp. 660 - 667.*

GB / T 16260. ISO / IEC 9126, 2003. *"Software engineering-Product quality,".*

Tijs van der Storm, 2007. "Generic Feature-Based Software Composition," *Computer Science, vol.4829, pp.66 - 80.*

Don Batory, David Benavides and Antonio Ruiz-Cortes, 2006. "Automated analyses of feature models: Challenges ahead," *Communications of the ACM - Software product line,vol.49, pp.45 - 47.*

Sheng-Xiang Zou, Wei Zhang, Hai-Yan Zhao and Hong Mei, 2005. *"Modeling variability in software product family," Ruan Jian Xue Bao (J. Softw.). vol.16, pp. 311 - 316. Jan.*

SCITEPRESS SCIENCE AND TECHNOLOGY PUBLICATIONS