MODELING THE VARIABILITY WITH UML FOR TRIZ BASED CAI SYSTEM

Jianhong Ma[1], Runhua Tan[2]
[1]*School of Computer Science and Software Engineering, Hebei University of Technology, TianJin, 300130, China, Eamil:jh_ma@eyou.com.;* [2]*School of Mechanical Engineering, Hebei University of Technology, TianJin, 300130, China*

Abstract: Going with the inventive knowledge applied broadly in different science fields, special Computer Aided Invention(CAI) software product customized for different science fields should be developed with low cost and short time. In order to minimize the number of unexpected adaptations and features within application engineering projects, variability is analyzed not only for current requirements but also for future requirements. Based on analyzing CAI software domain, the development trend of CAI family products is predicted, and the variability of this domain is modeled with extended UML in this paper. The feature-based object-oriented domain analysis approach is adopted to model the variability of CAI family products.

Key words: Computer Aided Invention (CAI); TRIZ; modeling variability; UML; software product line

1. INTRODUCTION

TRIZ, the theory of solving inventive problem, is proved to be useful in invention in practice. Many CAI products have been completed and applied in industries. Going with the inventive theory developed thoroughly and generalized broadly and the requirements of market changefully, the power of related products should be enhanced rapidly. The characteristics of CAI family tools are:

1. Evolving
   The CAI family tools are evolving and becoming powerful with new modules gradually being added to the previous version. For example,

This project is supported by the Project of Natural Science Foundation of China under Grant No.50375045, the Project of Natural Science Foundation of Tianjin under Grant No.043802211 and the Project of Educational Foundation of HeiBei Province under Grant No.2004412

Please use the following format when citing this chapter:
Ma, Jianhong, Tan, Runhua, 2006, in International Federation for Information Processing (IFIP), Volume 207, Knowledge Enterprise: Intelligent Strategies In Product Design, Manufacturing, and Management, eds. K. Wang, Kovacs G., Wozny M., Fang M., (Boston: Springer), pp. 406-411.

CAI tool 3.0 version is developed by integrated TMMS module and 76 standard solutions module into 2.0 version.

2. Integrative
   In inventive design domain, it is trend that CAI tool will be a synthetical tool with many different inventive theories, such as QFD, TRIZ and AD etc[1].
3. Flexibility and Customizability
   The CAI tool has been applied in different domains. So, the special CAI tool for special domain must be developed with flexibility and customizability.
4. Extensibility
   The knowledge of enterprises is playing a more and more important role in activities of enterprises culture and inventive design. It is the customer's requirements to mix the databases of the CAI tool with the knowledge of inner enterprises to form the repository with own intellectual property rights.

So, efficient method to model and manage the variability of family products for reuse turns out to be necessary, both from single product process and economic viewpoint. Lots of researches on exploiting an efficient method to model the commonality and variability have been undertaken.

2. SOFTWARE PRODUCT LINE

2.1 Commonality and Variability

A product family consists of a group of related software sharing some common features.

2.1.1 Commonality

Commonalities of a product family are characteristics that all the family products own. They serve to characterize the domain. The determination of whether a characteristic is a commonality or variability is often a strategic decision rather than an inherent property of the product family.

2.1.2 Variability

The common definition of variability is given as: Software variability is the ability of a software system or artifact to be changed, customized or configured for use in a particular context.

Variability in software systems can be identified as the functional and non-functional variability. Functional variability means that the system can provide different functionalities in different contexts.

2.2 Software product line

The approach of software product line aims in decreasing the costs and lifecycle required to produce a customer specific product. It includes two processes: Application engineering and Domain engineering. In domain engineering the communality and the variability of the product family are defined and developed as core asserts for reusability. The variability in such a domain (or several domains) is explicitly modeled and separated from the common parts. During application engineering a customer specific application will be defined and ideally developed by selecting and configuring core assets resulted form the domain engineering. Domain engineering core assets are evolved based on the feedback from the application engineering.

3. MODELING VARIABILITY WITH UML FOR CAI FAMILY PRODUCTS

An effective representation of the variability not only identifies and models alternatives among the products in a product line, but also defines what characteristics are associated with what products, as well as what dependencies and interrelationships exist among variability.

The representative approaches are classified into three catalogues: feature-oriented method2, object-oriented method3 and integrative method4. In the paper, the feature-based object-oriented method, a typical integrative method, is applied to modeling the variability for CAI product family, and variability is described at feature model, object model and subsystem model with extended UML.

3.1 Modeling feature with extension UML

Feature means the attribute and character belong to the systems. Feature model captures the end user's understanding of the general capabilities of applications in a domain. It represents the common and the variable features of concept instances and the interdependencies between the variable features.

Features in feature model are classified into following categories:

1. Mandatory feature: means features must be selected, it represents the core attributes and characters of the domain.

2. Optional feature: means that there are 1 to n feature to be selected. In this case, some of them may not belong to several family products.
3. Alternative feature: at least one feature should be selected. In this case, the different products will take different methods at the same attributes and characters.
4. External feature: a feature realized by the underlying platform, not by the system itself. It is useful for describing relations to external (e.g. platform) requirements.
5. Extension feature: features that will be extended in future.

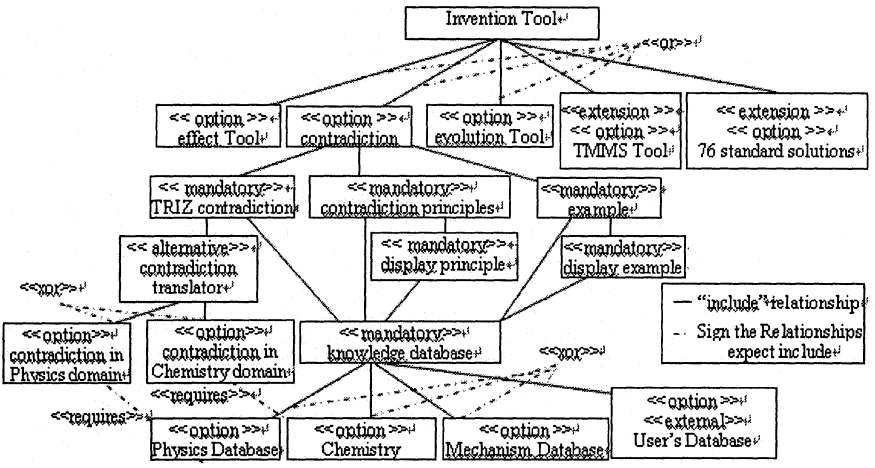

Figure 1. Feature model for CAI family products with UML

The features are organized in a tree with the modeled concept as the root constructed from composition or generalization relationships. The relationships between features include: (the relationships are modeled with their UML notation – filled diamond and generalization-arrow)

1. composition or generalization relationships: expressed as <<include>> or omitted .
2. dependency relationship: expressed as << requires>>.
3. exclusive and non-exclusive alternatives: expressed as << xor>> and <<or >>.

The feature model of CAI product family is shown as figure 1 above. It supplies some modules to be selected according to the customer' requirements, some have been finished, such as effect tool, contradiction tool, evolution tool, and some will be developed or finished in future, such as TMMS tool and 76 standard solution tool. So, the relationship of these features is expressed as <<or>>. Take the contradiction tool as the example, contradiction tool must have the following features: TRIZ general

contradiction, contradiction principles and examples, and these features are the result of querying the TRIZ contradiction database. If the being developed product is applied in special domain, such as physics, etc, the "contradiction translator "feature must be selected, which the special domain contradictions will be translated into TRIZ general contradictions. Of course, the feature of special contradiction databases depends on the selection of the special application domain, so the relationships between "contradiction special domain" and "special databases"expressed as <<requires>>.

3.2 Object model

Object model represents the domain structure in terms of objects and their relationships. There are three steps to model objects: identify object from feature model and their relationships; handle the variability through analyzing the relationships between objects, such as aggregation, generalization and association, and assigning the variant to object model.

3.2.1 Identifying the object from feature model

The guideline presented by K. Lee[4] is:

1. Modeling capability features as service-state hiding objects or user role objects.
2. Modeling operating environment features as interface objects.
3. Modeling domain technology features as objects that encapsulate requirement decisions.
4. Modeling implementation technique features as objects that encapsulate communication methods, design decisions and implementation methods.

The relationships between objects fall into generalization and aggregation relationships in object-oriented programming. The aggregation and generalization relationships can be respectively derived from the same types of relationships in the feature model and interactions between the identified objects.

3.2.2 Handling the variability in object model

To describe the variability in object model, the notion of variation points is applied into the object model. The extension for variation points consists of two parts: explicitly marking the location of a variation and the distinguishing the different ways of binding this variation.

Variant points exist in the objects which identified from extended features, optional features, alternative features and extension features, also including the feature relationships signed as {xor} {or}. The variation

point determines when (binding time) and how many variants (multiplicity) can be bound (i.e. selected for the desired product). In addition, constraints for the variants can be modeled.

4. CONCLUSION

The approach of feature-based object oriented domain analysis has been applied to the CAI domain. With this method, the feature model and object model explicitly distinguish the commonality and variability. A new special application developed based on the CAI software product line will be completed quickly through tailoring or binding the core asserts.

5. REFERENCES

1. Tan Rnnhua, Kraft Dieter. A conceptual design methodology for variety using TRIZ and QFD [A]. *Proceedings of the ASME Design Engineering Technical Conference* [C]. 2002. V4. 377-384
2. Kang KC, Kim S, Lee J, Kim K, Shin E, Huh M. FORM: A feature-oriented reuse method with domain-specific reference architectures. *Annals of Software Engineering*, 1998, 5:143~168.
3. Griss ML, Favaro J, d'Alessandro M. Integrating feature modeling with the RSEB. In: Devanbu P, Poulin J, eds. *Proceedings of the 15th International Conference on Software Reuse. Victoria: IEEE Computer Society,* 1998. 76~85.
4. Kwanwoo Lee, Kyo C. Kang, Wonsuk Chael and ByoungWook Choi Feature-based approach to object-oriented engineering of applications for reuse *Software—Practice And Experience*. 2000; 30:1025–1046
5. Hassan Gomaa, *An Object-Oriented domain ananlysis and modeling method for software reuse*. 0073-1129-1/92 1992 IEEE.