# Description of a New Feature Meta-model

Yu Song[1] , Qi Chen[2]

[1, 2]（School of Computer Science and Technology, North China Electric Power University, Bao Ding, He Bei, Email: chenqi19820417@163.com）

**Abstract.** At present, serveral feature meta-models have been come up with. However, they can't meet the requirements of dynamic Internet environment or software reuse. This paper proposes a feature meta-model based on ontology as well as its formal description. Meanwhile, FTM (Flexible Transaction Model) mechanism is considered. In particular, it is adaptable to the changes in dynamic environment and can meet the requirement of software reuse. Finally, an example is given to verify this model.

## 1 Introduction

Feature model was introduced from the Feature-Oriented Domain Analysis (FODA) methodology [Kang et al.1990] and further developed from a number of approaches[1-3]. Since its intrduction in 1990, feature modeling has attracted a great number of application domains. And it becomes the most popular method of domain analysis with the development of domain engineering and product line. In addition, a large number of tools supporting the feature modeling paradigm have been come up with. However, feature modeling still has not made its break-through into the toolbox of every software architecture or requirement engineering. What's more, in most feature-oriented methods, the construction of feature models heavily depends on the domain analysts' personal understanding, and the work of constructing feature model from the original requirements of sample applications is often tedious and ineffective. So it is necessary to build a common meta-model without misunderstanding.

According to above requirements, this paper prosposes a new feature meta-model to adapt to the dynamic network. It divides feature into *Business Action, Facet, Term,etc.* on the basis of the traditional feature modeling methods[4]. Considering FTM mechanism, it introduces ontology as a descripive method and take commonality, variablility, dependency and bindtime into account comprehensively. With the proposed meta-model, good-quality feature models can be constructed in a more effective way.

The remainder of this paper is organized as follows. Section 2 introduces FTM mechanism. In section 3, we describe the feature meta-model based on FTM and ontology in an all-around way, including the formal descriptions. Section 4 put it into practice in a real system, while conclusions and an outline for further work round up the paper are refered in Section 5.

## 2 FTM Mechanism

The goal of FTM is to make systems adapt to dynamic transactions, and its application to Supply Chain Management was given by JUN AHN and JOO PARK several years ago. To make the goal clear, we define a transaction as a "collaborative

process of exchanging information for trading goods or performing trade-related activities"[5]. This definition is different from that of traditional transaction processing literature, i.e., ACID(Atomicity, Consistency, Isolation, and Durability) which is emphasized for maintaining data integrity[6-7].

Under today's complicated and changeful network environment, FTM mechanism should be paid more attention in building software models.

## 3 Optimized Feature Meta-model

### 3.1 Ontology

A commonly accepted definition of an ontology in information science and engineering is that by Gruber, who defines an ontology as "an explicit specification of conceptualization"[8]. An ontology represents the semantics of concepts and their relationships using some description language, which is most often coupled with first-order logic or its decidable fragment. In terms of descriptive power, ontology is clearly richer and more powerful than feature, which is the reason why we combine them together to make a more powerful feature meta-model.

### 3.2 Feature Meta-Model

Feature model is a hierarchical structure with constraint relations between features and is originally developed from customers' point of view. It is also a concept description technique, but is captured logically as a propositional formula[9]. The essence of a feature model is its embodiment of hierarchy and description of variability, rather than its rendering. Each feature can be optional or mandatory for a set of systems within a domain. Figure 1 shows an example of a simple feature model.

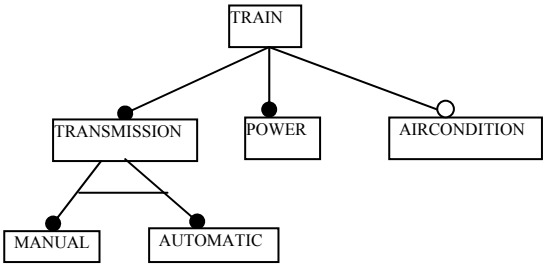


**Fig. 1. Train Feature Model**

In this figure, *AIRCONDITION* is an optional feature, while the other two features are mandatory. In addition, *MANUAL* and *AUTOMATIC* are exclusive with each other.

The ideas of modeling and expressing relations presented in FODA(Feature Oriented Domain Analysis) are further developed in FORM(Feature-Oriented Reuse Method) [Kang et al. 1998]. FORM extends FODA to the software design and

implementation phases and describes how the feature model is used to develop domain architectures and components for reuse, such as attibutes and cloning, which seem to be pushing the descriptive power of feature modeling to that of ontology[10]. However, it is difficult to use the FORM feature views because their separation is not defined precisely enough. Furthermore, reverse engineering needs a more general separation of the feature spaces. So we apply ontology to feature meta-modeling to make a more powerful description method, taking FTM mechanism into consideration.

### 3.3 Description of Meta-model Based on FTM and Ontology

Research on features has received much attention in the domain engineering community. Feature modeling plays an important role in the design and implementation of complex software systems. However, the presentation and analysis of feature models are still largely informal. There is also an increasing need for methods and tools that can support automated feature model analysis. A formal semantics for the feature modeling language is defined using first-order logic. It provides a precise and rigorous formal interpretation for the graphical notation.The proposed feature meta-model is shown in figure 2. We use OWL to describe it. It further divides feature into *Business Action, Facet, Term,etc.* on the basis of the traditional feature modeling methods. The model is denoted by ontologies and can be commonly used for applications in every domain. Considering FTM mechanism, we add several dynamic or changeful elements to this model, such as *Bind, ConfigureDepend, HasChildren, IfOptional,* etc. They are not necessary for every system, but I want to describe the meta-model as integrately as possible, so I add these elements to it for the utilization in some cases.

The meta-model consists of four ontology classes which are *BusinessAction, BusinessObject, Term and Bind Time.* Also, it includes several relations, and some relations are defined on the basis of other relations.

We divide the meta-model into two parts. The upper one which is in the dashed rectangle is commonly used. We abstract it from the complicated meta-model in order to achieve the purpose of software reuse and make a clear vision to developers and designers of software products. The nether one are dynamic and not every element in this part is necessarily be used in a certain system. Therefore, it is comprehensive and can adapt to the dynamic system environment. In this model:

***BuisnessAction*** is the semantic agent in the course of exchanging information.

***Term*** is the terminology value of *Facet*.

They bulild themselves into a hierarchical structure respectively according to the relation *Subclassof*, and show the specialized relation between *BusinessAction* and *Term*.

***Facet*** gives a precise description of *BuisnessAction* in detail. It is defined as the relation from *BusinessAction*(rdfs: domain) to *Term*(rdfs:range).

***ConfigureDepend*** stands for dependences under indirect communication situation. It signifies mutual constraint relationship when the optional features are binded.

***Subclassof*** is defined between *BusinessAction* and *Term*. It signifies direct specialized relations which can be converted into the relation *subClassof* in ontology(rdfs: subClassof), and remain direct *subClassof* relation.

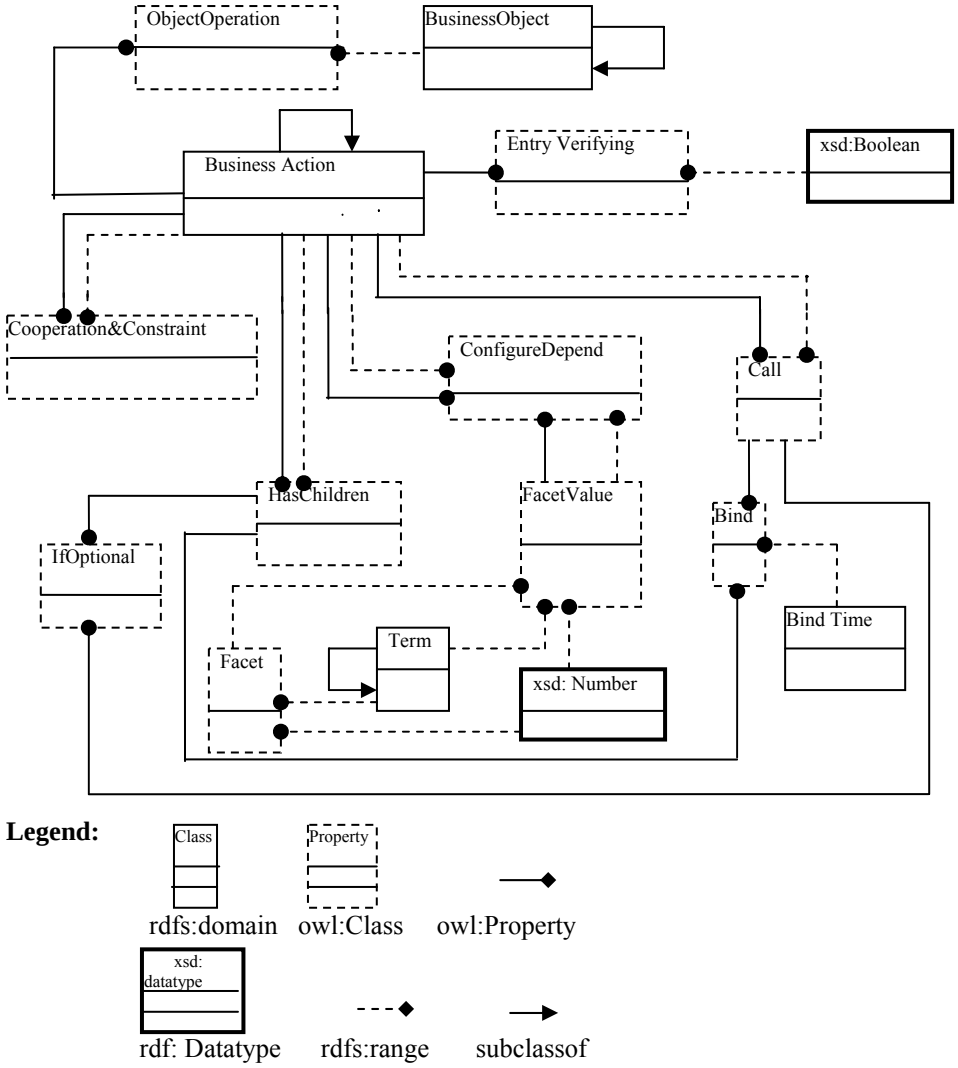


**Fig. 2. Feature Meta-model Based on FTM and Ontology**

***HasChildren*** shows the division of the parent action.

***Call*** stands for the dependency to other operations in order to achieve current function.

***IfOptional*** symbolizes whether a *BusinessAction* is optional or not.

***Cooperation&Constraint*** defines the relations among *BusinessActions*, such as notice, decision, etc.

***Bind*** describes the constraints when the optional or variable elements are related to their above *BusinessAction* or *Use*. It has three types, i.e., BuildTime, LoadTime and

RunTime which are used in system assembly, guidance and operation time respectively.

***HasChildren*** is another relationship between *BusinessActions,* the parent BusinessAction is related to a set of sub-BusinessActions according to this relationship.

### 3.4 Formal Semantics for the Feature Meta-model

In this section, we prospose a formal semantics based on the first-order logic of Z to describe the feature meta-model. The feature types can be expressed precisely through these descriptions[11].

Features represent distinguishable characteristics of a concept, while a concept consists of a set of related features with constraints. We give the definitions of Feature and Concept as follows.

[*Feature*] | *Concept*: $\mathbb{P}$ *Feature*

We define *Feature* as a given set, and *Concept* a special kind of feature, which is represented as subset of *Feature.*

*holds*: *Concept* $\leftrightarrow$ *Feature*

$\forall$ *c: Concept* • *(c, f)* $\in$ *holds*

The above defines the relationship *holds* between a concept and the feature of concept instance. This is the most basic and general relationship in a feature model. Then some of the formal descriptions of the relations in figure 2 are defined as follows:

● *IfOptional*

If the result is true, that means the feature is optional. It can be defined formally as follows:

*Optional: Concept* $\leftrightarrow$ *(Feature* $\times$ $\mathbb{P}$ *Feature)*

$\forall$ *c:Concept; pf:Feature; s:*$\mathbb{P}$*Feature* • *c Optional(pf, s)* $\Leftrightarrow$ *pf* $\notin$ *s*

$\wedge$((*c, pf* ) $\notin$ *holds* $\Rightarrow$ ( $\forall$ *f: s* • *(c, f)* $\notin$ *holds*))

The first predicate states that the parent feature *pf* should not be included in the child set *s*. The second predicate states that if the parent feature *pf* of a set of *Optional* features *s* is not included in a feature configuration, none of the set *s* can be included in the same concept instance.

On the contrary, if the result is false, the feature will be mandatory:

*Mandatory:Concept* $\leftrightarrow$ *(Feature* $\times$ $\mathbb{P}$ *Feature)*

$\forall$ *c:Concept; pf:Feature; s:*$\mathbb{P}$ *Feature* • *c Mandatory (pf, s)* $\Leftrightarrow$ *pf* $\notin$ *s*

$\wedge$((*c, pf* ) $\in$ *holds* $\Rightarrow$ ( $\forall$ *f: s* • *(c, f)* $\in$ *holds*))

$\wedge$((*c, pf* ) $\notin$ *holds* $\Rightarrow$ ( $\forall$ *f: s* • *(c, f)* $\notin$ *holds*))

The above defines *Mandatory* as a relation between a concept *c* and the parent and children of feature *f*. It states that if the parent of the *Mandatory* feature set *s* is held by a concept instance, all the feature in set *s* should be included into the description of the same concept instance; otherwise none.

● *HasChildren*

*HasChildren: Concept* $\leftrightarrow$ *(Feature* $\times$ $\mathbb{P}$ *Feature)*

$\forall$ *c:Concept; pf:Feature; s:***P** *Feature* • *c HasChildren (pf, s)* $\Leftrightarrow$ *pf* $\notin$ *s*
$\wedge((c, pf) \in holds \Rightarrow (\exists\ f: s \bullet (c, f) \in holds))$
$\wedge((c, pf) \notin holds \Rightarrow (\forall\ f: s \bullet (c, f) \notin holds))$

The above predicate suggests if the parent feature of set s is held by a concept instance, there must be at least one child which is included into the description of the same concept instance; otherwise none.

# 4 Application of This Model

## 4.1 Online Auction Management System

The advancement of Internet-based commerce has created a turbulent market environment by allowing easier introduction of new products, services, and suppliers. The time and efforts required to open new storefronts on Internet became much smaller comparing with those of traditional offline markets and various types of business models and marketing practices are newly created due to the constant development of new information technologies[12-14]. For this dynamic environment, information systems need to be designed in a flexible way to meet the changing requirements. This turbulent market environment requires strong adaptability in information systems to avoid high cost for re-implementation or re-customization. So we apply the above feature meta-model to this field to provide an improved and understanding example which can meet all the requirements refered above.

## 4.2 Description of the domain model

The description of the application of feature meta-model in Online Auction System is shown in figure 3. We can see that every kind of symbol in this graph signifies a relationship existing in the above meta-model.

*OnlineAuction* mainly includes the mandatory *Sailing*, *Purchasing* and *OrderPayment*, and also the optional *Delivery*.

*PayWhenReceive* means pay the cash to deliveryman when the consumers receive their goods. This is an optional item.

The flexibility is reflected on the dynamic dependence of *BidMateria*, *DelObtainedMaterial* and *Delivery* towards their parent BusinessActions.

Two facets are defined on Browse, i.e., *BrowseTime* and *BrowseMode*. Three types of *BrowseMode* also provide an optional operation to the customers.

*PayToBank* has three subclasses, i.e., *CCB*, *CMB* and *ICBC*. The customers can select different ways of payment at the time of system running, which shows the flexibility of design again.

# 5 Conclusions

We put forward a feature meta-model based on FTM mechanism and ontology and then apply it into Online Auction Management System and we have proved that it has the greatest discriptive power and the biggest adaptability to dynamic network

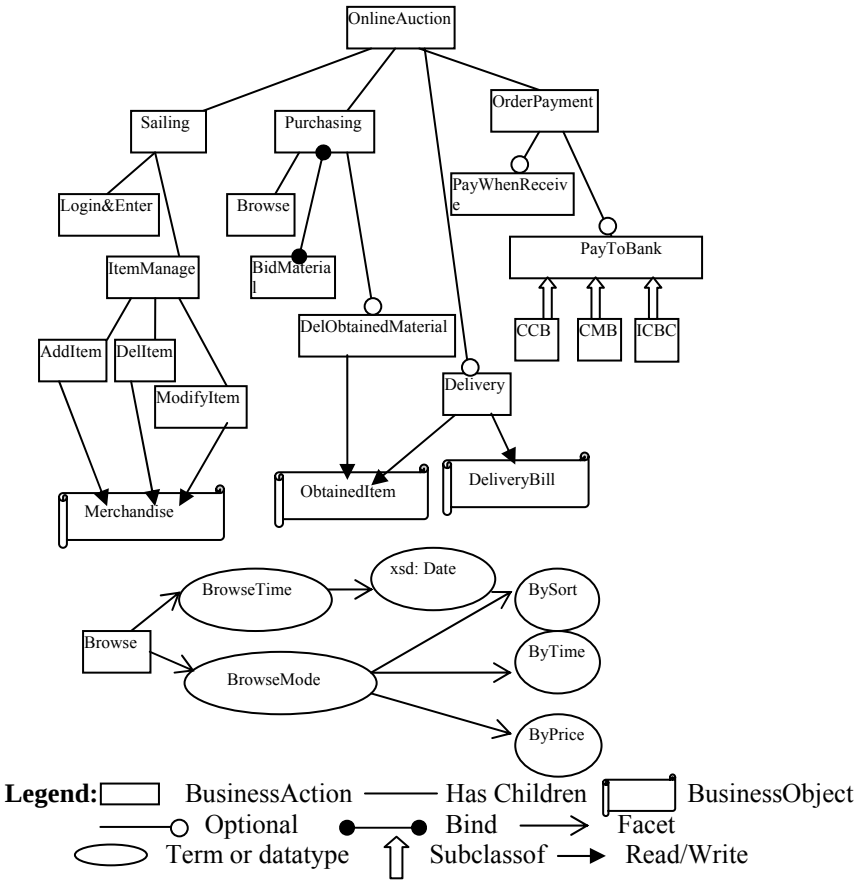


**Fig. 3. Application of Feature Meta-model**

than any traditional method. In particular, we give out formal descriptions of this model in details. However, there are still some works which need to be perfected, such as how to make the meta-model simpler and clearer and the improvement of formal description of this meta-model. Therefore, we should study further and apply the model to applications as possible as we can.

## 6 Acknowledgements

I acknowledge for the help of my tutor during my preparation for the paper.

## References

1. M. Griss, J. Favaro, and M. dAlessandro. Integrating feature modeling with the rseb. In Processdings of the Fifth International Conference on Software Reuse, pages 76–85, Victoria, Canada. IEEE Computer Society Press. see http://www.intecs.it.(1998)

2. T. Kamiya, S. Kusumoto, and K. Inoue. Cnder: A multi-linguistic token-based code clone detection system for large scale source code. IEEE Trans. Software Engineering, 28:654–670. (2002)

3. M. Riebisch, K. Boellert, D. Streitferdt, and I. Philippow. Extending feature diagrams with uml multiplicities. In Processdings of the Integrated Design and Process Technology (IDPT) 2002, pages 1–7.(2002)

4. Zhang W，Mei H．A feature—oriented domain model and its modeling process，Journal of Software, l4(8)：1345 一 l356(in Chinese with English abstract)．http：／／www．jos．org．en／1000—9825／14／1345．htm.(2003)

5. Hyung Jun Ahn*, Sung Joo Park, A flexible transaction framework for dynamic collaboration of agents一with an online travel application, International Journal of Cooperative Information Systems, Vol.13, No. 4.(2004)

6. J.Gray and A. Reuter, Transaction Processing: Concepts and Techniques (Morgan Kaufmann Publishers), pp. 5–7.(1993)

7. S. Jajodia and L. Kerschberg, Advanced Transaction Models and Architectures (Kluwer Academic Publishers), pp. 3–34.(1997)

8. T. R. Gruber. Towards principles for the design of ontologies used for knowledge sharing. Technical Report KSL93-04, Stanford University, Stanford, August.(1993)

9. D. Batory. Feature models, grammars, and propositional formulas. Technical Report TR-05-14, University of Texas at Austin, Texas, Mar.(2005)

10. Krzysztof Czarnecki, Chang Hwan Peter Kim, Karl Trygve Kalleberg, Feature Models are Views on Ontologies, IEEE(2006)

11. Jing Sun, Hongyu Zhang. Formal Semantics and Verification for Feature Modeling. IEEE.(2005)

12. T. Jelassi and S. Leenen, An e-commerce sales model for manufacturing companies: A conceptual framework and a European example, European Management Journal 21, 1,38–47.(2003)

13. B. Schlegelmilch and R. Sinkovics, Viewpoint: Marketing in the information age, Can we plan for an unpredictable future? Int. Marketing Review 15 (3) 162–170.(1998)

14. R. Yelkur and M. DaCosta, Di?erential pricing and segmentation on the Internet: The case of hotels, Management Decision 39 (4) 252–261. (2001)