Decision Support for Dynamic Adaptation of Business Systems Based on Feature Binding Analysis

Liwei Shen, Xin Peng and Wenyun Zhao
Computer Science and Engineering Department, Fudan University,Shanghai,China
{061021062, pengxin, wyzhao}@fudan.edu.cn

Abstract

Dynamic evolution has been an essential requirement for more and more business systems which attempt to provide 7(days) x 24(hours) availability and flexible adaptability on the changing business environment. Therefore, these systems are expected to be self-adaptable at run-time with little user intervention. CBSD provides an architectural way for self-adaptation, in which adaptation can be performed on the macro level of architecture and easier to control. However, the big gap between the problem space (business goal and environment) and the solution space (software architecture and components), and the runtime decision-making for adaptation are two difficulties for the implementation of business-oriented self-adaptation. In this paper, we propose an approach of decision support for dynamic adaptation of business systems based on feature binding analysis. In the method, feature model is introduced to represent the business policy and bridge the gap. So, dynamic adaptation can first be performed on feature binding analysis. The other characteristic of the method is CBR (Case Based Reasoning) based adaptation decision on environment factors captured by all kinds of sensors.

1. Introduction

Increasingly, dynamic evolution has been an essential requirement for more and more business systems. Since many of the systems attempt to provide 7(days) x 24(hours) availability and flexible adaptability on the changing business environment, they are desired to be self-adaptable at run-time according to specific business goals. CBSD (Component Based Software Development) provides an architectural way for dynamic reconfiguration [1]. In such systems, the 'computation' (implemented by components) which implies the functions of the system and the 'coordination' (embodied by connectors) that organizes the individual function in a reasonable way are separated on the architecture level [2]. Developers just need to consider how to compose the components in order to form a new system rather than how to implement or modify each of them. Therefore, adaptations can be achieved by reconfigurations on the macro architecture and be non-intrusive to the internal implementation of components [1].

Dynamic context is the main driver for self-adaptation. The context can be physical environment which is often concerned in ubiquitous computing (e.g. location, collaborating devices, etc), and business environment including service and business status (e.g. current performance, storage, etc). In this paper we focus on business oriented adaptation which has the goal of gaining the flexibility of business. However, two difficulties should be considered. One lies in how to reduce the big gap between the problem space (business goals and environment) and the solution space (software architecture and components) which is caused by the difficulty in mapping between the requirements and the artifacts. The other lies in how to make decision for self-adaptation at runtime.

In this paper, we propose an approach of decision support for self-adaptations of business systems based on feature binding analysis. Feature model proposed in feature-oriented domain engineering [3,4,5,6] is introduced to bridge the gap between business policies and software architecture identified above. In feature oriented approach, features are regarded as the first-class entities in the requirements level [6], and the feature model is a kind of domain model which is structured from customers' point of view [7]. In a domain specific feature model, commonalities as well as differences among applications of the specific domain are captured [5]. The model can be customized for specific application by deciding whether or not to include the design-time feature variations in the new system. Runtime variations can also be embodied in feature model to be decided according to the runtime context. In our method, most dynamic business

31st Annual International Computer Software and Applications Conference(COMPSAC 2007)
0-7695-2870-8/07 $25.00

policies can be adequately represented by feature model and their runtime adaptations are directly performed on runtime variations in the model. After that, adaptations upon feature model can be translated into evolution of the architecture.

For the second difficulty of runtime decision-making identified above, we adopt case-based reasoning (CBR), which is performed in feature binding analysis to achieve the business-oriented adaptation. In CBR, a case is an aggregation of attributes describing the domain problems and the solutions to solve them [8]. In our method, a case is an adaptation operation about feature binding for a specific instance. It has two parts which must be organized in pair. The former refers to the environmental factors such as data, system status, and others which can be captured by all kinds of sensors, while each attribute in the case denotes a factor classified in a different angle of view. The latter refers to the corresponding feature binding strategy which can be retrieved from the experience stored in case base.

The remainder of this paper is structured as follows. Section 2 shows the overview on the framework of our approach. Section 3 describes the feature model and the feature-oriented adaptation. Section 4 represents the decision-making process for dynamic adaptation based on CBR in detail. Some issues in practice when adopting the method are discussed in Section 5. A hypothetical example of online bookstore system using the decision support process is given in Section 6 to illustrate the conceptual feasibility of our approach. Finally we conclude the paper with some discussions and perspectives in Section 7.

2. Framework overview

The framework of business-oriented self-adaptation is shown in Figure 1. The adaptation is driven by the business environment which is affected by business systems and captured by sensors. The environment includes system's runtime status (e.g. CPU load, memory, bandwidth available, etc) and business data produced by the business system (e.g. current storage, sale, etc) which are often stored in database. At runtime, environment changes trigger a process of adaptation. First, an optimal solution for feature binding can be acquired through the decision support process, which is based on CBR to retrieve and merge the solutions of the best similar cases from the case base aiming at current situation. The feature dynamic binding is then implemented according to this solution and the feature model is adapted as a result. After that, the adapted model is mapped into the reconfiguration of the architecture, and the running system is evolved following the SA to achieve the business goal and ensure the stability and correctness. The holistic process will not stop as long as the system is running and the environment is changing. The feature model, the architecture and the running system are regarded to be self-adapted with little user intervention.

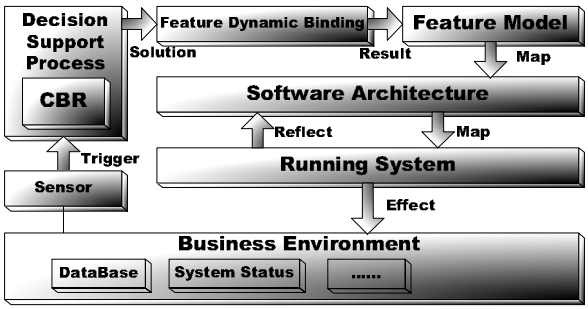

Figure 1. Framework of business-oriented self-adaptation

3. Feature model and feature-oriented adaptation

3.1. Feature model

Feature model is the target of the solution acquired from the decision support process and it plays an important role in the business-oriented self-adaptation. Many researches have focused on its construction [3,4,5,6,10]. In particular, [5,10] propose an ontology-based feature modeling approach which is used to describe the business requirement in a more natural and comprehensible way. The approach adopts the W3C recommended ontology language 'OWL' as a formal foundation of the feature meta-model. Features and their relations are divided into several categories (Action, Facet and Term as feature, subClassOf, HasElement, IfOptional and Use as relation). The most important ones are listed as following:

Action represents business operations of various granularities with domain specific semantics.

Facet is defined as perspective or dimension of precise description for Action.

Term is used to restrict value space for Facet.

HasElement is a type of specialization relationship and it divides the function of a father feature into the functions of its son features.

Use indicates a dependency relationship upon other action to fulfill its whole function.

The detailed introduction of ontology-based feature model can be referred to [5,10].

An ontology-based feature model for a hypothetical online bookstore system is illustrated in the upper part

of Figure 2. The system consists of two main actions: *Purchase* and *Distribution,* which represent business operations in the book sale domain. *Purchase* is a customer-oriented process which includes actions of *browsing* products, *booking* them and *paying* at last. When a customer browses, a *ranked list* indicating the popular products will be embedded in the first page. The list includes the ranked items sorted by the total sales amount or just that of the day. Before paying, he may accept a *discount* as a way of sales promotion if today's sales amount is great. Furthermore, *security,* which is defined as a perspective of action 'purchase' and is called a 'facet', can be ranked in four levels (each one is a value for facet and is called a 'term'), but the higher level (more advanced and time-consuming security technology) may lead to lower system performance (longer response time). In "*Distribution*" part, operator *reviews* the booking lists and the goods are going to be *transported*. The *review* process can be *manual* or *automatic* depending on the number of non-reviewed booking lists. In this figure, rectangles represent actions; ellipses represent the facet as well as the terms belonging to it; and grey-filled ones will be bound at runtime.

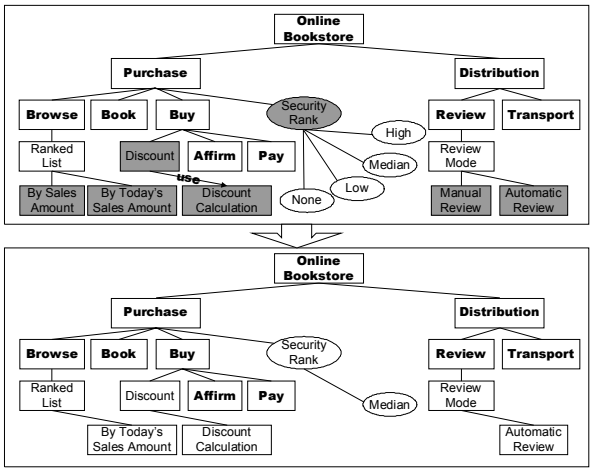

Figure 2. An adaptation example of feature model for online bookstore

3.2. Feature binding and feature-oriented adaptation

We are informed from figure 2 that both the action and the facet need to be bound. Binding an action denotes to bring the action's function into new system and vice versa, while binding a facet means to choose the specific term as a value to describe the perspective of the action. However, it's not necessary to decide all the actions because some of them can be decided indirectly by the others. In the bookstore example, action 'Discount' uses action 'Discount Calculation' to fulfill its own function. Therefore, the latter must be bound as soon as the former is bound at runtime.

Let's return to the previous example to see how the adaptation is performed on the feature model. Supposing at a special time, environment changes occur: the sales amount became huge in that day, several kinds of products had been sold, the security was in low level in order to provide high performance, and many booking lists were left to be reviewed. The changes captured by sensors triggered the process for adaptation. After the solution gained from the decision support process was implemented, an adapted feature model can be seen in the subjacent part of Figure 2. The action "By Today's Sales Amount" was chosen to decide the items in the ranked list. The action "Discount" took effect and the "Discount Calculation" had to be bound accordingly. The action "Automatic Review" was chosen and the facet "Security Rank" was reified to a term of median level. The detailed process can be seen in section 6.

The adaptation through the example described above looks different with the sophisticated meaning of 'adaptation'. Since it is acknowledged that powerful adaptation will be able to add new modules and actions in new context, the adaptation in our approach is more like a customization process upon feature model according to the changing environment. The overall actions, facets and terms are identified at the beginning and will not be modified through the automatic adaptation, so we just need to decide the binding status for each kind of the feature variant at runtime.

Reconfiguration of the system architecture was achieved according to the adapted feature model. Components providing the functions of the bound features as well as the median security technology were to be embedded in software architecture. However, since we focus on the adaptation of feature model in this paper, the process of evolution upon architecture will not be discussed.

4. Decision support process for software evolution based on CBR

CBR is adopted in our method to solve the difficulty of runtime decision-making. By contrast with rule-based deductive reasoning, CBR has some evident advantages [8]. The most significant advantages in business field include solving problems with partially understood domains and allowing more convenient knowledge acquisition. However, its main disadvantage resides in the fact that the applicability of the solution retrieved is not guaranteed, due to the complexity and fairly wide range of contexts.

The process of the decision support is illustrated in Figure 3. It consists of several steps. At the beginning a

new case is organized from the environment changes which are collected by sensors and then one or more solutions belonging to historical cases can be acquired through the module of case retrieval. Solutions will be merged in the case adaptation module to compose the optimal decision if there is more than one solution available. Finally the decision is implemented upon feature model and the new case-solution pair is learnt to enrich the case base.

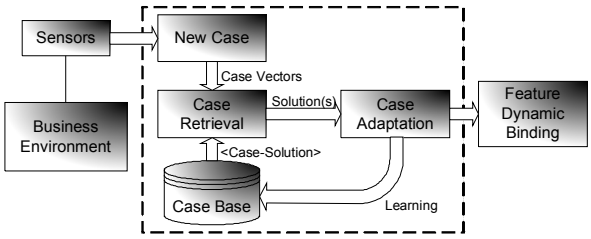

Figure 3. Decision support process based on CBR

4.1. Process prerequisite

Feature model is one prerequisite because the solution is implemented upon it. Researches on the construction of feature model have been described in section 2.

Another requisite is the metrics which can be regarded as reference attributes for decision-making. In our method, we use Goal-Question-Metric (GQM) approach to identify them. GQM is a mechanism for quantitative evaluation of software and process [9]. It has three levels: conceptual level (Goal), operational level (Question) and quantitative level (Metric). The goal here is business-oriented self-adaptation. Thus several questions can be enumerated to indicate the dimensions that have influence on the goal. Finally a set of data is associated with each question in order to answer it in a quantitative way. Furthermore, we need to assign a weight value for each question and metric in order to define its degree of importance in achieving the goal. Particularly, each weight value must be in the range of 0 and 1, and the sum of them should be 1. However, the definition of weight is acknowledged difficult in practice, especially when multiple people are included. We suggest that developers and domain experts can negotiate together to find out an appropriate weight definition.

4.2. Case representation

In order to represent case in a proper way, we propose a method of case attribute classification upon GQM model. Each case can be regarded as an integration of several dimensions of attributes while each attribute is corresponding to a metric having influence upon the goal of adaptation. A new case is described as the following:

$C = \{A_1, A_2, \ldots, A_i, \ldots, A_m\}$, where C denotes a case. i = 1,2,…,m, m denotes the number of attribute dimensions, which is corresponding to the number of Question in GQM model.

$A_i = \{a_{i1}, a_{i2}, \ldots, a_{ij}, \ldots, a_{in}\}$. Each a_{ij} means the jth attribute in the ith attribute dimension. It corresponds to the Metric belonging to a specific Question. In addition, each attribute has a weight value according to the GQM model, which is needed in computing the similarity.

Furthermore, environment changes usually reside in some specific dimensions, so the adaptation is to meet the demand in these fields. We think it necessary to tag these dimensions for later use during the case adaptation stage if two or more solutions get conflict.

4.3. Case base

Case base is the database storing cases and corresponding solutions, which surely appear in pair <C , S>, where C denotes the case and S denotes the solution. $S = \{b1, b2, \ldots, bk, \ldots, bp\}$, where p means the number of tactics and b_k represents a specific binding tactic to a specific feature variant in the feature model. The solution may only include the tactics referring to some of the features rather than all of them so that *p* is not a fixed value.

4.4. Case retrieval

Case retrieval is the core of the decision support process while the goal of this module is to find the historical cases and solutions with highest similarity aiming at the input case. In our process, we employ the Nearest-Neighbor algorithm, which gets the similarity value between two cases through calculating the distance with weight:

$$Sim_r = \sum_{i=1}^{m}\sum_{j=1}^{n} w_{ij} Sim_{ijr} \qquad (1)$$

Sim_r denotes the similarity value of the input case and one case C_r in the case base. w_{ij} denotes the weight of the jth attribute in the ith dimension and $\sum_{i=1}^{m}\sum_{j=1}^{n} w_{ij} = 1$. $Sim_{ijr} \in [0,1]$, represents the similarity between the attributes in two cases, which is defined as: $Sim_{ijr} = 1 - \frac{|c_{ij} - c_{ijr}|}{k_{ij}}$, where k_{ij} represents the range of the jth attribute in ith dimension of all the cases.

In order to reach a reasonable and considerate decision, we require users to set a threshold value

between 0 and 1 at the beginning. It means the cases with similarity higher than the threshold can be obtained and the corresponding solutions are taken into consideration to form a composite and optimal decision.

4.5. Case adaptation

If there is just one solution retrieved, we get the only adaptation decision to be implemented. Contrariwise, solutions belonging to the retrieved cases should be merged in a proper way, even though there are conflicts among them.

A solutions-merging algorithm is proposed. The input of the algorithm includes the solutions, the tagged dimensions denoting what kind of environment data have changed sharply and the input case. Its output is the solution acquired finally. The main idea of the algorithm is to maintain non-conflictive binding tactics upon some of the features. As to the features with conflictive tactics, the similarity should be recomputed referring to the tagged dimensions (mentioned in the case representation stage) between the retrieved cases and the input case, because the tagged dimensions have great impact on adaptation. The recomputing process still follows the formula (1), but the i value is restricted by the sequence number of the tagged dimensions. Then upon these features, the tactics of the case that has the new highest similarity are chosen. Finally, a decision can be composed of the two kinds of tactics. The pseudo code is shown in Figure 4.

```
Algorithm:solution-merging(case-solution[ ],
                    taggedDimensions[ ],inuptcase):solution
  if taggedDimensions[ ]=all dimensions
     {return solution with highest similarity value }
  else {
     for each solution S in case-solution[ ] {
        for each <feature-tactic> pair in S {
           if feature exists in decisonList[ ] {
             if tactic<>feature's tactic* in decisonList[ ]
               feature→candidateList[ ];
             remove  <feature,tactic*> from
                    decisonList[ ];}
          else { <feature-tactic>→decisonList[ ]; }}}
     recompute similarity value (sv) between inputcase
           and each case in case-solutions[ ];
     for each feature f in candidateList[ ] {
        choose the tactic** of case
              with highest sv→<f,tactic**>;}
   return
     tactics in decisonList[ ] + tactics in candidateList[ ] ;
  }
```

Figure 4. Solutions-merging algorithm

5. Issues in practice

When we put our decision-making process into practice, some aspects of issues become evident.

5.1 Can user intervene the process

The decision-making process runs in an automatic way. However, sometimes the solution is not satisfying particularly in the early stage. It's a good idea to provide manual evolution (user intervention) in practice. It allows the users to decide the final solution in the 'Case Adaptation' step, determine the feature binding strategy (the feature model provides a business view for users to realize), and they could modify the case-solution pairs in case base as well.

5.2. How to initialize the case base

CBR is based on the abundant historical cases. Provided that few cases exit in the base at the beginning, the result is affirmatively inaccurate. We suggest that users make decisions manually during the first several months until the test results of automatic adaptation can be accepted by users.

6. Example

The description as well as the feature model of online bookstore system can be seen in section 3.

By analyzing the system using GQM approach, we get to the case representation in Table 1.

Table 1. Case representation of the system

Dimension	Attribute	Weight
Sale	Today's sale amount	0.25
	Today's booking variety number	0.15
Booking list	Today's list number	0.05
	Number of non-reviewed booking lists	0.15
System current status	Performance	0.15
	Security	0.15
	Number of online customers	0.1

At runtime, the environment changes acutely. The decision support process is implemented in the following stages.

1. A new case is organized from the sensors, which is represented as {[95300,97], [320,270], [7,3,115]}. The number in the expression corresponds to the current value of the attribute listed in table 1.

2. After computing the similarity value between the new case and each case in the case base, two historical cases are selected with similarity higher than the

threshold value set beforehand. The corresponding solutions are shown in the second and third columns in Table 2.

Table 2. Selected historical solutions

	1	2	final
List by today's sales amount	Bound		bound
List by total's sales amount			
Discount	bound		bound
Security rank		median	
Manual review			
Automatic review	bound	median	bound

3. Merge the two solutions through solutions-merging algorithm mentioned in 4.5 and then the final and optimal one is in the last column of Table 2. The corresponding adapted feature model can be seen in the subjacent part of Figure 2.

4. The new case along with the final solution are packaged and inserted into the case base in order to enrich the historical experiences.

7. Conclusion and perspective

In our approach of decision support for dynamic adaptation of business systems based on feature binding analysis, feature model is introduced to represent the business policy and bridge the gap between the problem space and the solution space. Thus the binding decision upon the feature model is aligned with the changes in environment and can be mapped into the reconfiguration of architecture. In addition, we employ the CBR approach at runtime to obtain the optimal solution towards feature binding in an experiential way aiming at the changing environment.

The example in this paper only illustrates the conceptual feasibility of our approach. So, one work remained for us is to make our idea into realized tool which supports the automatic adaptation based on feature model.

In addition, our work is based on the reasonable mapping between feature model and overall software architecture, which is acknowledged difficult in practice. Lots of work has concentrated on this subject and we are also going to resolve it in the future or else our work in this paper will be in vain.

Our final goal is to provide whole technique support for the software self-adaptation framework, make the process feasible and flexible.

Acknowledgments. This work is supported by the National High Technology Development 863 Program of China under Grant No.2005AA113120, the National Natural Science Foundation of China under Grant No.60473061 and 60473062.

8. References

[1] Xin Peng, Wenyun Zhao, Liang Zhang and Yijian Wu, "An intelligent connector based framework for dynamic architecture", Proceedings of the 5th International Conference on Computer and Information Technology (CIT2005).

[2] Bass L, Clement P and Kazman R, "Soft Architecture in Practice", Addison-wesley Longman,inc.,1998.

[3] Kyo C. Kang, Sajoong Kim, Jaejoon Lee, Kijoo Kim, Gerard Jounghyun Kim and Euiseob Shin, "FORM: A feature-oriented reuse method with domain-specific architecture", Annals of Software Engineering, 1998, Vol 5, pp.143-168.

[4] Kyo C. Kang et al, "Feature-Oriented Domain Analysis Feasibility Study", SEI Technical Report CMU/SEI-90-TR-21, November 1990.

[5] Xin Peng, Wenyun Zhao, Yunjiao Xue and Yijian Wu, "Ontology-Based Feature Modeling and Application-Oriented Tailoring", Proceedings of the 9th International Conference on Software Reuse (ICSR2006), Springer LNCS, Vol.4039.

[6] Wei Zhang, Hong Mei and Haiyan Zhao, "A Feature-Oriented Approach to Modeling Requirements Dependencies", Proceedings of the 2005 13th IEEE International Conference on Requirements Engineering (RE'05).

[7] Ilian Pashov, Matthias Riebiscih and Ilka Philippow, "Supporting Architectural Restructuring by Analyzing Feature Models", Proceedings of the 8th European Conference on Software Maintenance and Reengineering (CSMR'04).

[8] Xu, L.D. "Case based reasoning", Potentials, IEEE, Vol 13, Issue 5, Dec 1994-Jan 1995, pp.10 – 13

[9]V.R. Basili, G. Caldiera and H.D. Rombach, "Goal Question Metric Paradigm". In J.J. Marciniak, ed., Encycl. of SW Eng.,vol. 1, pp. 528–532. John Wiley & Sons, 1994.

[10]Xin Peng, Yijian Wu and Wenyun Zhao, "A Feature-Oriented Adaptive Component Model for Dynamic Evolution", 11th European Conference on Software Maintenance and Reengineering (CSMR20007), to appear.