An Approach to Dynamically Reconfiguring Service-Oriented Applications from a Business Perspective[*]

Jianwu Wang[1,2], Yanbo Han[1], Jing Wang[1], and Gang Li[1]

[1] Institute of Computing Technology, Chinese Academy of Sciences, 100080, Beijing, China
[2] Graduate School of the Chinese Academy of Sciences, 100080, Beijing, China
{wjw,yhan,wangjing,gangli}@software.ict.ac.cn

Abstract. This paper proposes an approach to dynamically reconfiguring service-oriented applications from a business perspective: CAFISE$_{adapt}$, which defines both business-level and software-level change operations to respectively express changes in the business domain and the software domain. Utilizing the convergence of these two level change operations, the approach expects application changes can be automatically coherent with business changes. Through hiding software-level technical details of applications that are necessary for traditional change operations, the business-level change operations can be used by business users to dynamically modify service-oriented application instances, which can realize the dynamic reconfiguration of service-oriented applications in a straightforward way to timely adapt to business requirement changes. This approach has been applied and validated in the project FLAME2008.

1 Introduction

Service-oriented applications are constructed by composing needed Web services to meet different business requirements. During the execution of a service-oriented application, new business requirements may be presented, which would cause the application impracticable [1]. So, dynamic reconfiguration of service-oriented applications is necessary to realize that application changes are coherent with business.

The traditional dynamic reconfiguration approaches are mainly from the software perspective: Business users need to report the business requirements changes to IT professionals, ask them to specify which application changes should be made to respond to the business requirement changes and modify application instances using pre-offered change operations. This kind of reconfiguration approaches need the communications between business users and IT professionals, which usually causes that application changes lag behind the rapid business requirement changes. Moreover, the abundance and dynamism of resources in the service environment make it difficult for the modifiers to know the exact information of the whole candidate Web services during the course of dynamic reconfiguration.

To solve the above problems, we present an approach to dynamically reconfiguring service-oriented applications from a business perspective: CAFISE$_{adapt}$, which defines software-level and business-level change operations, and these two-level change operations are correlated by convergent relations. Utilizing the approach, business users

[*] The research work is supported by the National Natural Science Foundation of China under Grant No. 60173018, the Key Scientific and Technological Program for the Tenth Five-Year Plan of China under Grant No. 2001BA904B07.

can use business-level change operations to express changes in the business domain, the relevant software-level change operations expressing corresponding changes in the software domain can be automatically got according to the convergent relations. After the execution of software-level change operations, corresponding changes in the software domain will be realized. Besides, this approach can automatically specify needed Web services through the convergent relations, which reduces the difficulty of modification comparing with traditional approaches.

The rest of the paper is organized as follows: Section 2 introduces some concepts of CAFISE approach which is the foundation of $\text{CAFISE}_{\text{adapt}}$ approach. The $\text{CAFISE}_{\text{adapt}}$ approach is detailedly illustrated in section 3 and its implement in project FLAME2008 is presented in section 4. Section 5 discusses the related work. At last, the conclusion and further work are presented.

2 CAFISE Approach

The $\text{CAFISE}_{\text{adapt}}$ approach is part of our CAFISE (Convergent Approach for Information Systems Evolution) approach [2]. The core idea of CAFISE approach is the convergence of the business domain and the software domain, expecting user's requirements can be coherent with service-oriented applications during the course of application's construction and execution through the convergence, which can support just-in-time construction and dynamic reconfiguration of service-oriented applications from the business perspective. Just-in-time construction and dynamic reconfiguration are two aspects of CAFISE approach, and respectively denoted as $\text{CAFISE}_{\text{jit}}$ and $\text{CAFISE}_{\text{adapt}}$: $\text{CAFISE}_{\text{jit}}$ represents the approach supporting just-in-time construction of service-oriented applications and enables business users to rapidly construct new applications in a business-end programming way [3], which corresponds to new business requirements at the phase of application construction; $\text{CAFISE}_{\text{adapt}}$ represents the approach supporting dynamic reconfiguration of service-oriented applications and enables business users to dynamically reconfigure applications to adapt to business requirements changes, which corresponds to business requirement changes at the phase of application execution.

In CAFISE approach, a service-oriented application consists of three parts: business-level VINCA application [3] constructed through composing needed *business services* [4], software-level Darnel application [5] constructed through composing needed Web services, and convergent relations that correlate the two level applications organically (Fig. 1). Business services are defined by business domain experts according to different industry standards, and each service can fulfill a certain business function. Comparing Web services, business services comprise semantic information which is described with DAML-S [6] and are aggregations of Web services which have the same functionality and behavior semantics. Through hiding the technical details of Web services and encapsulating semantic information, business services can be regarded as business-level representation of service resources and be understood and operated by business users. Let's take an example of a business service and its corresponding Web services: *WeatherForecast* is a business service that can forecast weather, whose information is listed in table1, and related Web services are listed in table2.

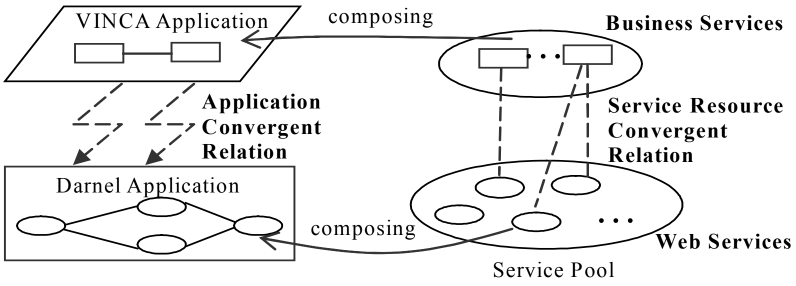


Fig. 1. Illustration of Two-level Applications in CAFISE Approach

Table 1. *WeatherForecast* Business Service

Basic Information		PublicService.WeatherForecast	
Business operation	Input	date	http://flame/KgBase/Weather.daml#Date
		city	http://flame/KgBase/Weather.daml#City
	Output	maxTemperature	http://flame/KgBase/Weather.daml#HighTemperature
		minTemperature	http://flame/KgBase/Weather.daml#LowTemperature
Non-functional constraints	Cost	[0.1￥, 1￥]	
	Provider	Beijing weather forecast bureau/ China weather forecast bureau/…	

Table 2. Web Services Related with *WeatherForecast* Business Service

Invocation Information		http://xxx.com/weatherforecast.wsdl	http://xx.com/weatherforecast.wsdl	…
Non-functional constraints	Cost	0.6￥	0.9￥	
	Provider	Beijing weather forecast bureau	China weather forecast bureau	
	…			

3 CAFISE$_{adapt}$ Approach

In this section, we will firstly introduce the principle of CAFISE$_{adapt}$ approach. Then we will illustrate the three main parts of this approach: change operations at business-level, change operations at software-level and the convergent relations between above two level change operations. At last, the usage of this approach will be demonstrated.

3.1 Principle of CAFISE$_{adapt}$ Approach

The convergence concept is firstly presented by David Taylor at 1995 to bridge the gap between the business domain and the software domain, expecting to construct software system directly through business designing and enable software to adapt to ever-changing business [7]. But the previous researches [8, 9] are mainly focused on how to utilize convergence to construct application basing on business model. Seldom researches are done to study how to take advantage of convergence to improve the application's adaptability. CAFISE$_{adapt}$ approach can realize the coherence between business-level application changes and software-level application changes through the convergence of two level change operations, which expects to realize the coherence between business requirement changes and application changes (Fig. 2).

According to CAFISE$_{adapt}$ approach, a business user firstly modifies his business-level application instance using business-level change operations; secondly the used business-level change operations are automatically transformed to corresponding software-level change operations by parsing the change operation convergent relations. At last, through the execution of software-level change operations by modification engine, the corresponding software-level application instance is modified, which eventually realize the modification of the service-oriented application. This approach can make business users instead of traditional IT professionals play the role of application modifier.

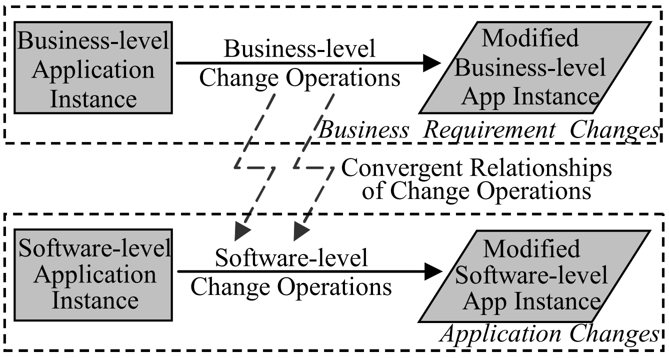

Business-level Application Instance
Business-level Change Operations
Modified Business-level App Instance
Business Requirement Changes
Convergent Relationships of Change Operations
Software-level Application Instance
Software-level Change Operations
Modified Software-level App Instance
Application Changes

Fig. 2. Principle of CAFISE$_{adapt}$ Approach

3.2 Change Operations at Business-Level

The change operations at business-level are defined basing on the survey of business users' modification requirements, and each change operation corresponds directly to a certain typical business modification request. Besides, comparing with the software-level change operations, the objects of these business-level change operations are not concrete Web services but business encapsulation of them, for concrete Web services are too complex for business users to understand and manage directly. For example, it is difficult for business users to know which Web services in the current service environment can be added in a software-level application instance to meet their request changes. In this way, business-level change operations can conform to the business users' usage pattern and be easily understood and used by business users.

The business-level change operations on VINCA application instances are defined in interface format, which include two parts: change operation command (denoted as *OP*) and change operation parameter (denoted as *para*). The format of a change operation is like *Op(para1, para2,…)*, *Op* represents the semantic of an change operation and *para* represents the concrete target object of the change operation.

The change operations on VINCA application can be classified to following four types according to the different aspects of modification:

1. **Modification of business process's control structure:** including adding and deleting a certain business service, modifying a business service's location in business process, modifying a business process's control condition.
2. **Modification of business process's message associations:** including adding and deleting a message association between two business services.

3. **Modification of business service's properties:** including modifying a business service's non-functional properties, a business service's input values and a business service's output settings.
4. **Modification of application instance's execution state:** including pausing and resuming application instance's execution.

The above change operations can modify every aspects of business process and business service, through the composition use of different change operations we can realize the transformation from a certain valid VINCA application instance to another, therefore the completeness of these change operations can be guaranteed. As for the minimality of the change operation set, because we define these operations according to user-friendliness, a certain degree of redundancy can be permitted and user can do modification in his preferable manner.

3.3 Change Operations at Software-Level

Change operations at software-level are defined based on the characters of software-level application. Software-level change operations concern the software modification details which are normally too specialized for business users to master. Research results of [10, 11] can used to ensure the minimality of change operations and the correctness of new software-level applications.

The software-level change operations on Darnel application instances are also defined in interface format, mainly including the adding and deleting operations on the constituent elements of a Darnel application instance, such as Web services, the control and data associations between Web services. The modifications of application instance's execution state and the operation on Web service resources are also included. The detail information of these change operations can be found in [12].

3.4 Convergence of Two Level Change Operations

Utilizing above business-level and software-level change operations, we can realize the modification of VINCA application instances and Darnel application instances respectively. In order to transform the changes in the business domain to changes in the software domain automatically and correctly, XML-based convergent relations between these two level change operations are designed by IT professionals in advance to depict the correlations between these two change operation sets. Then the software-level change operations corresponding to the used business-level change operations by business users can be got according to the convergent relations.

Each business-level change operation can bring some business-level application instance changes. From these business-level application instance changes, corresponding software-level application instance changes can be got according to the application convergent relations defined in CAFISE model [2], and IT professionals can specify which software-level change operations are needed to bring the corresponding software-level application instance changes. Then these specified software-level change operations have convergent relations with the business-level change operation for they can bring the coherent changes of two-level application instances. After analyzing all the pre-defined business-level change operations in this way, the whole convergent relations can be got.

The convergent relations are described in XML through a set of convergent units (Fig. 3), and each convergent unit depicts which software-level change operations are corresponded to a business-level change operation. These convergent units can be classified into two categories: simple convergent unit expressing one-to-one relation, complex convergent unit expressing one-to-many relationship between business-level and software-level change operations. A complex convergent unit is composed of a set of simple convergent units whose business-level change operations should be the same. Each simple convergent unit comprises two parts: convergent relations between change operation command and convergent relations between corresponding parameters. These convergent relations of change operation command (parameters) are minimal constituents, comprising two parts: business-level change operation command (parameters), software-level change operation command (parameters).

The above convergent relationships only denote the mapping relations of two level change operations' meta-information. When using these change operations, concrete value are assigned to the parameters (we call the change operations with concrete parameter *change operation instances*). Special programs should be designed to get software-level concrete parameters from business-level concrete parameters, for example, to get corresponding Web service set from a concrete business service ID.

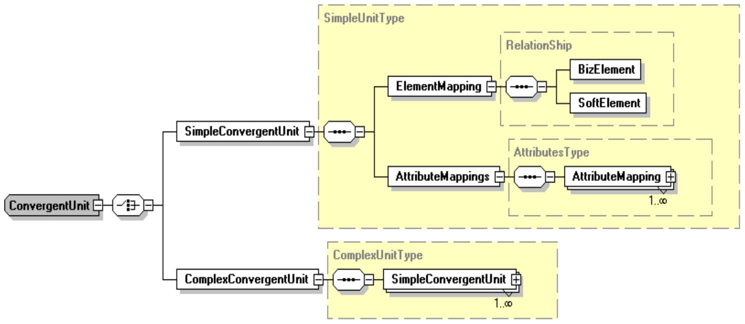


Fig. 3. XML Annotation of Convergent Relations between Two Level Change Operations

3.5 Usage of CAFISE$_{adapt}$ Approach

The usage of CAFISE$_{adapt}$ approach consists of two periods (Fig.4): preparation period and application period. In preparation period, IT professionals firstly define business-level change operations according to typical business modification requirements; secondly define software-level change operations according to the characteristics of software-level applications; lastly specify each business-level change operation's corresponding software-level change operations, which eventually result in these two level change operations' convergent relations. In application period, a business user modifies his VINCA application instance through selecting needed business-level change operations, and then the selected business-level change operation instances with concrete parameters can be recorded. By parsing change operations' convergent relations, these business-level change operation instances can be transformed into corresponding software-level change operation instances. With the modification engine executing each software-level change operation instances, corresponding Darnel application instance can be modified.

Once finishing the definition of change operations and their convergent relations in preparation period by IT professionals, these general change operations can always be used to dynamically modify different VINCA application instances. So we can see that through the prior work of IT professionals, the difficulty of subsequent modification is reduced and can be handled by common business users.

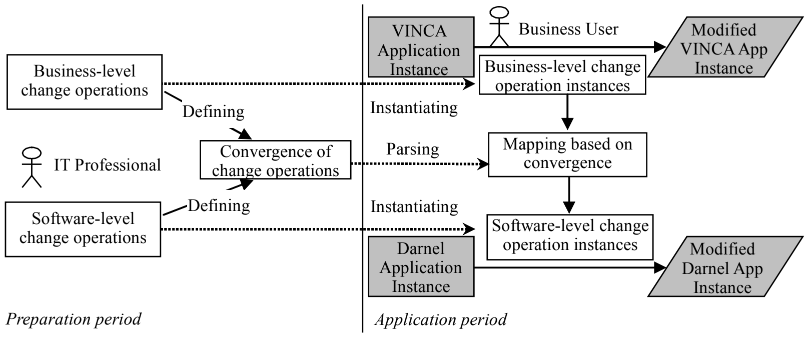

Business-level change operations
Defining
IT Professional
Convergence of change operations
Defining
Software-level change operations
Preparation period
VINCA Application Instance
Business User
Modified VINCA App Instance
Business-level change operation instances
Instantiating
Parsing
Mapping based on convergence
Instantiating
Software-level change operation instances
Darnel Application Instance
Modified Darnel App Instance
Application period

Fig. 4. Different Periods of the Usage of $\text{CAFISE}_{\text{adapt}}$ Approach

4 Implementation

The $\text{CAFISE}_{\text{adapt}}$ approach has been implemented in project FLAME2008 [13], expecting to develop service-oriented applications that provide integrated and personalized information services to the public during the Olympic Games 2008. We will validate the feasibility and features of $\text{CAFISE}_{\text{adapt}}$ approach by respectively explaining how to prepare and apply the approach in the following subsections.

4.1 Preparation of $\text{CAFISE}_{\text{adapt}}$ Approach

In this subsection, we will illustrate how to define the change operations and the convergent relations through a concrete change operation.

New business requirements may be usually brought forward during the execution of business-level application and request add proper service resources to application instance to meet the business requirement change, so adding service to application instance is a typical modification request. As a business service represents a common business functionality, different business users need set constraints to business service to express their different personalized requests, for example some business users may prefer ‘China Weather Bureau’ as the *WeatherForast* business service’s provider, yet some other business users may prefer ‘Beijing Weather Bureau’. Therefore we define the change operation *AddBS(BSID, BSCons, LocAtVINCA).* This operation’s command is *AddBS* and has three parameters: *BSID, BSCons* and *LocAtVINCA*, respectively representing the business service to be added, the constraints set by business users and location at VINCA application.

The VINCA application instance's change that *AddBS* can bring is a new business service inserted at a position that still not executed in the VINCA application instance's control flow. According to the two-level application convergent relations defined in CAFISE model, coherent change of Darnel application instance can be got, namely a corresponding new Web service inserted at corresponding position. Through analyzing the pre-defined software-level change operations, we know that two software-level change operations can implement this change of Darnel application instance: *SelWSByCons(WSSet,WSCons)* and *AddWS(WSUrl, LocAtDarenl)*. The two software-level change operations respectively fulfill the functionalities of selecting the suitable Web service and adding the Web service to Darnel application instance.

Based on the above analyses and specified software-level change operations, the convergent relations of *AddBS* can be established, whose XML fragment is shown in Fig. 5. The convergent relations depict that two software-level change operation *SelWSByCons* and *AddWS* have convergent relations with business-level change operation *AddBS*.

```
<ConvergentUnit>
  <ComplexConvergentUnit name="AddBS">
    <SimpleConvergentUnit name="AddBS1">
      <CommmandMapping>
        <BizCommmand>AddBS</BizCommmand>
        <SoftCommmand>SelWSByCons</SoftCommmand>
      </CommmandMapping>
      <ParameterMappings>
        <ParameterMapping>
          <BizParameter>BSID</BizParameter>
          <SoftParameter>WSSet</SoftParameter>
        </ParameterMapping>
        <ParameterMapping>
          <BizParameter>BSCons</BizParameter>
          <SoftParameter>WSCons</SoftParameter>
        </ParameterMapping>
      </ParameterMappings>
    </SimpleConvergentUnit>
<SimpleConvergentUnit
name="AddBS2">
  <CommmandMapping>
    <BizCommmand>AddBS
    </BizCommmand>
    <SoftCommmand>AddWS
    </SoftCommmand>
  </CommmandMapping>
    . . .
  </SimpleConvergentUnit>
</ComplexConvergentUnit>
</ConvergentUnit>
```

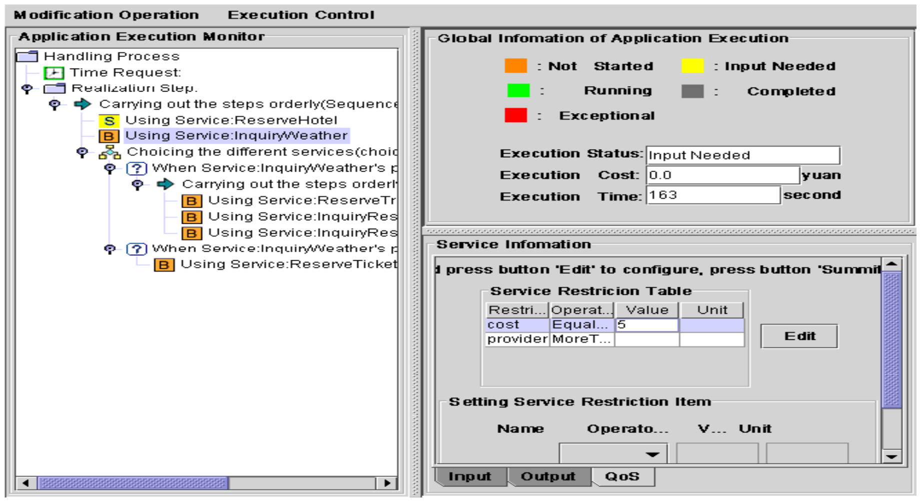
Fig. 5. XML Fragment of the Business-level Change Operation *AddBS*'s Convergent Relations

4.2 Application of CAFISE$_{adapt}$ Approach

With defining the whole set of change operations and realizing them through programs, the CAFISE$_{adapt}$ approach is implemented as a tool in the CAFISE framework (Fig. 6). Business-level change operations can be used through selecting proper menu items in graphical user interfaces. Now, we will illustrate how business users can achieve the dynamic reconfiguration of service-oriented application through a simplified scenario in the project FLAME2008 according to this approach.

Mr. George wants to travel Beijing during the Olympic Games 2008. Based on his travel schedule, he can construct his VINCA application by composing need business services in graphical interfaces. But during his traveling at Beijing, he hears that the recent weather in Beijing is often rainy, so he wants to add weather forecast functionality to his running traveling application in order to know the weather information timely. But as a business user, he doesn't have much knowledge about which Web services can provide the weather forecast functionality and how to invoke them, so he can't dynamically reconfigure his application by himself according to traditional approaches.

Utilizing CAFISE$_{adapt}$ approach, he needn't know the concrete information about Web services. After pausing his VINCA application instance, Mr. George can look up proper business service at Service Community in a graphical user interface. Supposing he finds business service *WeatherForecast* at the public service category, then he can add the business service to proper location in his VINCA application, and set some constraints to express his personalized requirements, for example, the provider should be 'Beijing Weather Bureau'. After the above operations, the work of Mr. George has finished and a business-level change operation instance is created: *AddBS(WeatherForecast, BSCons, LocAtVINCA).*



Handling Process
Time Request:
Realization Step.
Carrying out the steps orderly(Sequence
S Using Service:ReserveHotel
B Using Service:InquiryWeather
Choicing the different services(choic
When Service:InquiryWeather's p
Carrying out the steps orderl
B Using Service:ReserveTr
B Using Service:InquiryRes
B Using Service:InquiryRes
When Service:InquiryWeather's p
B Using Service:ReserveTicket

Execution Status: Input Needed
Execution Cost: 0.0 yuan
Execution Time: 163 second

d press button 'Edit' to configure, press button 'Summit

Restri...	Operat...	Value	Unit
cost	Equal...	5	
provider	MoreT...		

Name	Operato...	V...	Unit

Fig. 6. Dynamic Reconfiguration Tool in the CAFISE Framework

The business-level change operation instance should be mapped to corresponding software-level change operation instances so that proper Web service can be selected and added to Darnel application instance. Fig. 5 depicts the convergent relations of the *AddBS* change operation which is designed by IT professionals in advance. The convergent relations are represented through a complex convergent unit which includes two simple convergent units, meaning that the *AddBS* operation is related with two software-level change operations. By parsing the convergent relations, corresponding software-level change operation instances can be gotten which are shown as follows:

- *SelWSByCons (WeatherForcastServiceSet, WSCons)*
  Select most suitable Web service according to user's constraints: the *WeatherForcastServiceSet* parameter depicts the set of Web services that can achieve weather forecast functionality; the *WSCons* parameter depicts the corresponding Web service constraints that the provider should be 'Beijing Weather Bureau'. The result is denoted as *WSUrl.*
- *AddWS(WSUrl, LocAtDarenl)*
  Add the selected Web service to the running Darnel application at the corresponding location.

After software-level modification engine executes the above software-level change operations respectively, the Web service is added to the running Darnel Application. Then the reconfiguration course is finished and Mr. Georges can resume his application and use the weather forecasting service in his application.

This modification scenario depicts that how business users can modify his VINCA application instance according to $\text{CAFISE}_{\text{adapt}}$ approach. The menu items in Fig. 6 depict the possible business-level change operations at the execution moment and business users can implement modification through selecting proper menu items that express his modification requirements, without seeking help from IT professionals.

5 Related Work

Research on dynamic modification of traditional process-based applications, such as workflow applications, emphasizes particularly on defining the minimal change operation set that ensures the correctness of modified application [10, 11, 14]. Correctness means that the information of new application including the control dependences, data dependences, time dependences and instance execution states should be consistent with the information of the old one. This is necessary for the new application's execution but not in direct association with requirement changes at business-level, and the technical details of software-level resources should be known for modifiers to use these change operations, which makes these operations' usage a complex labor. So it is difficult for business users to use them and we can classify these change operations into software-level change operations. Moreover, because the available resources in traditional process-based applications are usually restricted, these change operations do not consider the abundance and dynamism of resources which are the main characteristics in the service environment.

The related research on dynamic modification of service-oriented applications is still not much. The following paragraphs will discuss two representative works in this research field:

DY_{flow} [15] supports the dynamic modification of service-oriented applications through dynamically modifying business rules according to the changes of business environment in order to invoke most suitable services at runtime. But this is only a part of modifiable contents and many other modifiable aspects are not supported.

In eFlow [16], the ad-hoc modification approach need the modifier firstly defines target process and verifies whether the target process is consistent with the source process. If it is consistent, the instant execution state of source process will be transferred to target process. This approach considers the dynamism characteristic of services and supports many aspects of modification. When users want to add e-services in eflow which are managed by e-services platforms to his application instance, he need to define service node through specifying needed service functionality and service selection rules by himself. Although having the similar dynamic service discovery functionality with the business services in $\text{CAFISE}_{\text{adapt}}$ approach, the service nodes in eflow should be defined by user himself from scratch during the course of dynamic modification, yet business services can be gotten directly from service community. So the request for modifier's ability of $\text{CAFISE}_{\text{adapt}}$ approach is reduced comparing the approach in eflow. Besides, the description of service in eflow is not based

based on a united semantic infrastructure which is used in CAFISE$_{adapt}$ approach, so service providers have to describe their services in their own ways, which will result in the difficulty of service selection for modifiers.

6 Conclusion and Future Work

As a part of CAFISE approach, CAFISE$_{adapt}$ approach defines user-friendly business-level change operations on the basis of correctness-ensuring software-level change operations, and these two-level change operations are correlated by convergent relations. Compared with change operations defined in traditional dynamic modification approaches, the objects of business-level change operations are not software-level resources but the business encapsulation of them. So the business-level change operations hide software technical details. This approach reduces the difficulties of modification and business users can modify his business application instances using the business-level change operations in a straightforward way to timely adapt to business requirement changes. This approach reflects a trend in software field: IT professionals concern the infrastructure and enabling techniques of dynamic and open Internet environment; while it is business professionals that concern the rapid (re)configuration of resources at business-level.

The approach is still an ongoing research whose goal is really making dynamic reconfiguration of service-oriented applications usable for business users. To achieve this goal, many aspects are to be improved in future, such as change operations that can modify the application's global constraints, the pre-condition and post-condition of each change operation and the change impact analysis at business-level.

References

1. L. Zeng, D. Flaxer, et al. PLM $_{flow}$-Dynamic Business Process Composition and Execution by Rule Inference. Technologies for E-Services: Third International Workshop (TES2002). Hong Kong, China, August 2002, pages 141-150.
2. Y. Han, Z. Zhao, et al. CAFISE: An Approach Enabling On-Demand Configuration of Service Grid Applications. Journal of Computer Science and Technology. Vol.18, No.4, 2003, pages 484-494.
3. Y. Han, H. Geng, et al. VINCA - A Visual and Personalized Business-level Composition Language for Chaining Web-based Services. The First International Conference on Service-Oriented Computing (ICSOC2003). Trento, Italy, 2003, pages 165-177.
4. Z. Zhao, Y. Han, et al. A Service Virtualization Mechanism supporting Business User Programming. Accepted by Journal of Computer Research and Development (in Chinese), 2004.
5. CAFISE group. Darnel Language Specification. Technical Report. Software Division, ICT, CAS, 2002.
6. DAML-S Coalition. DAML-S versions 0.9. Available at http://www.daml.org/services/daml-s/0.9/, 2003.
7. D. Taylor. Business Engineering with Object Technology. John Wiley & Sons. 1995.
8. R. Hubert. Convergent Architecture: Building Model-Driven J2EE Systems with UML. John Wiley & Sons. 2002.
9. J. Koehler, G. Tirenni, S. Kumaran. From Business Process Model to Consistent Implementation. The 6th IEEE International Enterprise Distributed Object Computing Conference. Lausanne, Switzerland, 2002, pages 96-106.

10. M. Reichert and P. Dadam. ADEPT$_{flex}$ – Supporting Dynamic Changes of Workflows without Losing Control. Journal of Intelligent Information Systems - Special Issue on Workflow Management. Vol.10, No.2, 1998, pages 93-129.
11. S. Sadiq and M. Orlowska. Architectural Considerations for Systems Supporting Dynamic Workflow Modification. Workshop of Software Architectures for Business Process management at the CaiSE'99, Heidelberg, June 1999.
12. G. Li, J. Wang, et al. MASON: A Model for Adapting Service-oriented Grid Applications. The Second International Workshop on Grid and Cooperative Computing (GCC2003). Shanghai, China, 2003, pages 99-107.
13. B. Holtkamp, R. Gartmann, and Y. Han. FLAME2008-Personalized Web Services for the Olympic Games 2008 in Beijing. Conference of eChallenges 2003. Bologna, Italy, Oct, 2003.
14. F. Casati. Models, Semantics, and Formal Methods for the design of Workflows and their Exceptions. Ph.D. Thesis. Dipartimento di Elettronicae Informazione, Politecnico di Milano, Milano, Italy, 1996-1998.
15. L. Zeng, B. Benatallah, et al. Flexible Composition of Enterprise Web Services. International Journal of Electronic Commerce and Business Media. Vol.13, No.2, 2003.
16. F. Casati and M. Shan. Dynamic and Adaptive Composition of E-services. Information Systems. Vol.26, No.3, 2001, pages 143-163.