A Reflective Approach to Keeping Business Characteristics in Business-End Service Composition

Zhuofeng Zhao[1, 2], Yanbo Han[1], Jianwu Wang[1, 2], and Kui Huang[1, 2]

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

Abstract. Business-end service composition can be best characterized as a user-centric approach to web application construction and promises to better cope with spontaneous and volatile business requirements and the dynamism of computing environments. The mapping of business-end service composites into software composites is a key issue in realizing business-end service composition, and a corresponding reflective approach is proposed in the paper. It is also investigated how to keep the business characteristics of business-end service composites during the mapping and how to adapt to changes of the correlation information for mapping. Basic components and patterns for keeping business characteristics are designed first. Then, using the principle of reflection, the contents for keeping business characteristics and the correlation information for mapping are combined and maintained on a meta-level. The result following the approach is a system implemented for mapping business-end service composites to software composites.

1 Introduction

Service composition implies a new way of application construction. A number of approaches to service composition have been proposed, for example BPEL4WS [2], BPML [3], SELF-SERV [5], SWORD [15], SAHARA [16] and so on, covering different aspects of service composition, such as modeling, language, system implementation, etc. However, these works are mainly from software perspectives, and require professional knowledge to develop business applications [14]. Most of them are still weak in dealing with a spectrum of application scenarios that require Web services be quickly composed and reconfigured by non-IT professionals in order to cope with the spontaneity and volatility of business requirements. Examples of such application scenarios include dynamic supply chain, handling of city emergency, and so on. Business-end service composition, as a novel way for service composition, promises to better cope with spontaneous and volatile business requirements. It can be best characterized as a user-centric approach to application construction and an effective way to achieve service composition from business perspectives.

Motivated by the above-stated considerations, we have designed a user-centric, business-end service composition language – VINCA [7]. VINCA specifies the following four aspects of business requirements in a process-centric manner: VINCA process, VINCA business services, user contexts and interaction patterns. However,

X. Zhou et al. (Eds.): WISE 2004, LNCS 3306, pp. 479–490, 2004.

VINCA only provides a way for modeling service composition from business perspective. In order to realize business-end service composition completely, business-end service composites in VINCA have to be mapped to IT-end composites supported by standard service composition technologies, such as BPEL4WS that is used in fact as an IT-end service composition language in our paper. To solve this problem, we propose a reflective approach which result in a system for the mapping.

The rest of the paper is organized as follows: Section 2 highlights the main issues for solving the mapping problem through a reference example. In section 3, the conceptual model of the reflective approach is presented. System implementation for the reflective approach is given in section 4. In section 5, we briefly discuss the related work. Section 6 concludes the paper and discusses some future works.

2 Problem Statement with a Reference Example

In this section, we discuss the issues of the mapping problem through a simplified example excerpted from the FLAME2008 project[1]. The example is shown in Fig. 1, it is a simple business process: *Mr. Bull is going to visit Beijing during the Olympic Games. He schedules two activities for his first day in Beijing before the Olympic Games: ordering a restaurant for dinner after booking a sightseeing tour from a travel agency.*

Fig. 1. Business Process of the Reference Example

Assume that Mr. Bull builds a business-end service composite for the above-stated example using VINCA. A VINCA process is defined, which contained *Book Excursion* activity and *Order Restaurant* activity. *Travel Agency* business service[2] and *Restaurant* business service are arranged to fulfill the two activities respectively. Furthermore, Mr. Bull would like to book the cheapest travel agency by means of configuring *Travel Agency* business service flexibly. In addition, he plans to order a restaurant near his location in the evening, so he configures the *Order Restaurant* activity as a context-awareness activity. The VINCA specification of this example is shown in Fig. 2.

From this example, we can first sense the importance of business-end service composition. Furthermore, we can note that the resulted business-end service composite in VINCA shows the following characteristics:

- **Context-awareness**. User contexts can be used to serve as a sort of implicit inputs in defining and executing a VINCA process, such as the *Order Restaurant* activity.

[1] FLAME2008 is a project for developing service-oriented applications that provide integrated, personalized information services to the public during the Olympic Games 2008.

[2] Business service is the core mechanism in VINCA. It supports service virtualization through abstracting Web services into business services so that the technical details of Web services can be hidden.

- **Dynamic composition of services.** Services composed in a VINCA process can be determined dynamically at runtime, such as the *Travel Agency* business service used in *Book Excursion* activity.
- **Flexible support of user interaction.** Users are allowed to appoint special places in a VINCA process for interaction during process execution, e.g. Mr. Bull can decide to interact after execution of the *Book Excursion* activity.

```
<VINCAProcess name="Bull's Travel">
  ...
  <SequentialActivity>
    <BizActivity name="Book Excursion">
      <Inputs>...</Inputs>
      <Outputs>...</Outputs>
      <QoSConstraint>
        <Cost>cheapest</Cost>
      </QoSConstraint>
      <ReferredBizService>.../TravelAgency</ReferredBizService>
    </BizActivity>
    <BizActivity name="OrderRestaurant"
      <Inputs>
        <BizEntity name="location" source="User Context">
        ...
      </Inputs>
      <Outputs>...</Outputs>
      <QoSConstraint/>
      <ReferredBizService>.../Restaurant</ReferredBizService>
    </BizActivity>
  </SequentialActivity >
</VINCAProcess>
```

Fig. 2. Snippet of the VINCA Specification of the Reference Example

The main issues in mapping such a business-end service composite to a corresponding IT-end composite in BPEL4WS are:

- **Maintenance of business characteristics**. BPEL4WS does not provide support for above-stated three business characteristics of VINCA directly. So, how to keep the business characteristics of business-end service composites during the mapping is an important issue to be solved.
- **Adaptability of correlation information**. The mapping from VINCA to BPEL4WS needs to follow a set of correlation information between them, specifying for example how an element in VINCA is mapped to elements in BPEL4WS. However, BPEL4WS is in progress and evolving. Furthermore, the business characteristics of VINCA may be augmented to adapt to a new business domain. These mean that the correlation information will change over time. How to adapt to changes of the correlation information is another important issue.

3 Conceptual Model of the Reflective Approach

To deal with the above-stated two issues, we propose a reflective approach. In this section, we will first introduce the conceptual model of the approach.

The principle of reflection [6, 11] has been one of the most useful techniques for developing adaptable systems, and it opens up in the possibility of system inspecting and adapting itself using appropriate metadata. To solve the problems introduced in section 1, we consider designing some special components and patterns for keeping

business characteristics first. Then using the reflection principle to maintain the correlation information in an adaptive way.

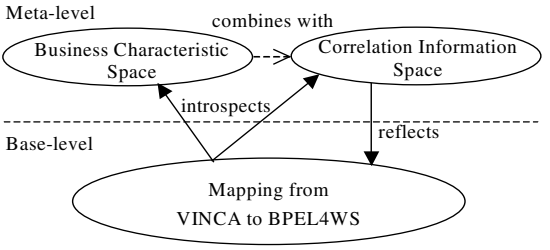

Fig. 3. Conceptual Model of the Reflective Approach

As shown in Fig. 3, the key elements of the reflective approach are *Business Characteristic Space* and *Correlation Information Space* on the meta-level, which are put forward to deal with the issues of maintaining business characteristics and adaptability of correlation information respectively. They can be used to introspect and reflect the concrete mapping from VINCA to BPEL4WS on the base level. Since BPEL4WS of current version requires static binding of Web services to flow, some special services for keeping business characteristics and its' usage patterns have to be provided. The metadata about these service components and patterns are maintained in *Business Characteristic Space*. Furthermore, the goal of Correlation Information Space is to reify and control the mapping behavior from VINCA to BPEL4WS through maintaining the correlation information. At the same time, through combining the contents in Business Characteristic Space with the correlation information in Correlation Information Space, the business characteristics can be kept during the mapping controlled by Correlation Information Space. In the following subsection, we detail these two key elements on the meta-level respectively.

3.1 Business Characteristic Space

Business Characteristic Space contains metadata about contents of keeping business characteristics. The metadata can be divided into two categories. One is about the special service components for keeping business characteristics; another is about the patterns for using the components to keep business characteristics. The metadata about components includes component name, textual description, operation interface, access address, etc; the metadata about patterns includes pattern name, textual description, condition for using, address of pattern file, etc.

Components for Keeping Business Characteristics

With respect to the three business characteristics of VINCA, we design the following components to keep business characteristics:

- Context Collection Component

User contexts are represented in terms of key-value pairs in VINCA. Keys are called dimensions and may take static values (such as name and birth date) or dynamic values (such as the location of user at a certain time point). *Context collection component* can be used to get both kinds of contextual values according to the designed context dimensions.

- User Interaction Component

User interaction in VINCA is defined to allow a user to input parameters during the execution period of a VINCA process. *User interaction component* can be used to trigger the interaction with users for certain input parameters in a VINCA process, and return the values the users input. Considering the participation of human beings, this component should be used in an asynchronous mode.

- Service Selection Component

Service selection component can be used to select the most appropriate service according to the users' demand. In VINCA, users can define their demands in terms of QoS constraints for services, and the best services for users' demands can be chosen according to a corresponding service selection policy. Service selection component implements the service selection policy.

- Service Invocation Component

Service invocation component acts as a proxy for invoking services dynamically. It can be used to invoke a service according to the access information of the service and returns the results after the invocation.

Patterns for Keeping Business Characteristics

Patterns for keeping business characteristics define the default pattern of using the above-listed components to keep business characteristics of VINCA. Each pattern that may include one or more components is designed specially for one category of business characteristics. Accordingly, there are three patterns for VINCA:

- Context-Awareness Pattern

Context-awareness pattern defines the usage pattern of context collection component to keep the business characteristic of context-awareness. This pattern can be used when context-aware business activities are defined in a VINCA process.

- User Interaction Pattern

User interaction pattern defines the usage pattern of user interaction component to keep the business characteristic of user interaction. This pattern can be used when there are business activities that need user interaction in a VINCA process.

- Dynamic Composition Pattern

Dynamic composition pattern defines the usage pattern of service selection component and service invocation component to keep the business characteristic of dynamic composition. This pattern contains two components. It can be used when there are business services that are used in the dynamic mode in a VINCA process.

The detailed representation of the three patterns in BPEL4WS will be given in section 4.1.

3.2 Correlation Information Space

In order to adapt to changes of the correlation information, including information about keeping business characteristics given above, *Correlation Information Space* is set to maintain the correlation information between VINCA and BPEL4WS. Table 1 shows an overview of the correlation information in Correlation Information Space.

The correlation information is represented as the mapping relationship between VINCA and BPEL4WS in terms of metadata of VINCA and BPEL4WS with constraints. Note that the correlation information has three parts as shown in the table: metadata of VINCA, metadata of BPEL4WS, and constraints on the mapping relationship. The metadata of VINCA and BPEL4WS can be gotten from the basic

constructs of VINCA and BPEL4WS languages. Some constructs of VINCA and BPEL4WS can be mapped directly, such as control logic. However, the mapping relationships from VINCA to BPEL4WS are not always one to one indeed, e.g. when the business characteristics need to be kept, constraints are provided to express the complicated mapping relationships. In this way, the patterns for keeping business characteristics can be defined as constraints on the correlation information. As such the contents in *Business Characteristic Space* can be combined with the correlation information. Besides, some basic constraints can be defined, such as constraints for transformation of elements' name from VINCA to BPEL4WS. Due to the space limitation, the complete correlation information is not listed here.

Table 1. Overview of Correlation Information between VINCA and BPEL4WS

Metadata of VINCA	Metadata of BPEL4WS	Constraints
VINCA Process	BPEL4WS Process	Basic constraints
Business Service	PartnerLink	Basic constraints, Dynamic composition keeping pattern
Business Activity	Invoke, Receive, Reply	Basic constraints, Context-awareness keeping pattern, User interaction keeping pattern
Business Entity	Variables	Basic constraints
Control Logic (same as BPEL4WS)	Sequence, Flow, While, Switch	Null

4 Implementation and Application

In this section, we discuss the implementation of the reflective approach. Fig. 4 shows the system architecture of implementation. There are two layers in the architecture:

- *Reflective Platform*, it maintains all the contents on meta-level introduced in the previous section and affects the function of Transformation Engine in Transformation Platform.
- *Transformation Platform*, it is in charge of transforming the VINCA specifications of business-end service composites into BPEL4WS specifications of IT-end service composites.

4.1 Reflective Platform

As shown in Fig.4, the Reflective Platform has two main parts: auxiliary service set and Metaobjects.

Auxiliary Service Set

Auxiliary service set is composed of a set of services corresponding to the components for keeping business characteristics introduced in section 3.1. For our purposes, four kinds of services are defined, namely context collection service (*ContextCollectionServ*), user interaction service (*UserInteractionServ*), service

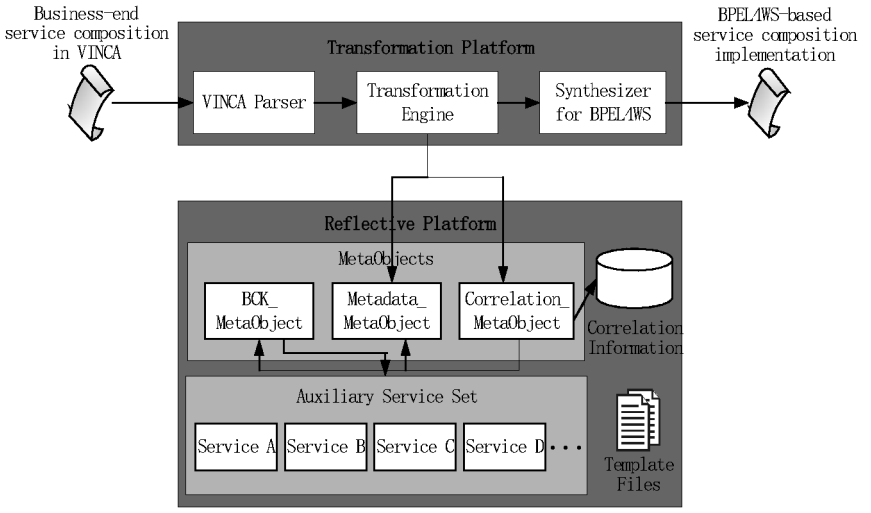

Fig. 4. System Architecture of the Reflective Approach

selection service (*ServiceSelectionServ*) and service invocation service (*ServiceInvocationServ*). These services are implemented as Web services so that they can be used in BPEL4WS directly.

```
<invoke partnerLink="ContextCollectionServ"
            portType=" getUserContextPT"
            operation=" getUserContext"
            inputVariable="varContextNames"
            outputVariable="varContextValues">
</invoke>
<invoke>
      ……   // Invoke activity corresponding to
              business activity
</invoke>
```

Fig. 5. Context-Awareness Pattern in BPEL4WS

```
<invoke partnerLink="UserInteractionServ"
            portType=" getUserInputPT"
            operation=" getUserInputs"
            inputVariable="varInputNames">
</invoke>
<receive partnerLink="UserInteractionServ"
            portType=" getUserInputCBPT"
            operation="returnUserInputs"
            Variable="varInputValues">
</receive>
<invoke>
      ……   // Invoke activity corresponding to
            business activity
</invoke>
```

Fig. 6. User Interaction Pattern in BPEL4WS

```
……
<invoke partnerLink="ServiceSelectionServ"
            portType="dynamicSelectionPT"
            operation="dynamicSelection"
            inputVariable="varQoSDemands"
            outputVariable="varServAddress">
</invoke>
<invoke partnerLink="ServiceInvocationServ"
            portType="dynamicInvocationPT"
            operation="dynamicInvocation"
            inputVariable="varServInfo"
            outputVariable="varResult">
</invoke>
……
```

Fig. 7. Dynamic Composition Pattern in BPEL4WS

Table 2. Fuctional Operations of BC_Metaobject

Operation	Function
addBCKS(ServiceDesc service)	Add a service for keeping business characteristic into Business Characteristic Space
addBCKP(Pattern pattern)	Add a pattern for keeping business characteristic into Business Characteristic Space
getBCKS(String serviceName)	Get a service for keeping business characteristic
getBCKP(String patternName)	Get a pattern for keeping business characteristic

Table 3. Fuctional Operations of MetaData_Metaobject

Operation	Function
buildMetaData(String dataName, String dataType, Vector dataAttrs)	Build a metadata of VINCA or BPEL4WS
buildAttribute(String attrName, String attrValue, String attrType)	Build an attribute of a metadata
getMetaData (String dataName)	Get a metadata according to its name
getAttribute (MetaData data, String attrName)	Get an attribute of a metadata according to its name

With these services, patterns for keeping business characteristics introduced in section 3.1 can be defined in the BPEL4WS format. Context-awareness pattern is defined as shown in Fig. 5, where an additional *invoke* activity for invoking *ContextCollectionServ* is inserted before the *invoke* activity corresponding to the business activity of context-awareness. User interaction pattern is defined as shown in Fig. 6. User interaction component should be used in an asynchronous mode as we have mentioned before, so the pattern defines the usage of *UserInteractionServ* by inserting an *invoke* activity and a *receive* activity before the *invoke* activity corresponding to the business activity that needs user interaction. Fig. 7 shows the dynamic composition pattern defined in BPEL4WS, which contains two *invoke* activities, invoking *ServiceSelectionServ* and *ServiceInvocationServ* respectively. These patterns are stored as XML files and will be used as target templates by transformation platform when producing service composites in BPEL4WS.

Metaobjects

Metaobject is viewed as an effective means to implement reflection [8]. In this paper, three Metaobjects are designed to maintain the contents on the meta-level of our reflective approach. They form the kernel of the reflective platform.

(1) BC_Metaobject (Business Characteristic Metaobject)

BC_Metaobject is responsible for maintaining the metadata in *Business Characteristic Space*. As shown in Table 2, it provides functional operations for getting and setting metadata about the services and patterns for keeping business characteristic.

(2) MetaData_Metaobject

MetaData_Metaobject is responsible for maintaining the metadata of VINCA and BPEL4WS in Correlation Information Space. The functional operations provided by MetaData_Metaobject are given in Table 3.

(3) Correlation_Metaobject

Correlation_Metaobject is responsible for maintaining the correlation information between metadata of VINCA and BPEL4WS in Correlation Information Space. It is

implemented based on above two Metaobjects. Correlation_Metaobject provides functional operations for defining and acquiring correlation information, which are shown in Table 4.

Table 4. Fuctional Operations of Correlation_Metaobject

Operation	Function
buildMetaDataCorrelation(String corrName, Vector vecLeftMetaData, Vector vecRightMetaData, String strConstraint)	Build correlation between metadata of VINCA and BPEL4WS
buildAttributeCorrelation(String corrName, String strLeftAttr, String strRightAttr, String strConstraint)	Build correlation between attributes of metadata of VINCA and BPEL4WS
getMetaDataCorrelation(String metadataName)	Get correlation between metadata according to metadata name
getAttributeCorrelation (String metadataName, String attrCorrName)	Get correlation between attributes of metadata according to its name

The correlation information between VINCA and BPEL4WS maintained by these three Metaobjects is saved persistently in our implementation. These Metaobjects can be used to support the implementation of Transformation Engine in the Transformation Platform and affect the function of Transformation Engine by changing the correlation information.

4.2 Transformation Platform

The Transformation Platform collects the VINCA specifications of business-end service composites from users, and transforms them into BPEL4WS specifications of IT-end service composites. The output from the Transformation Platform can be exported for direct execution by a BPEL4WS engine. The components - VINCA Parser, Synthesizer for BPEL and Transformation Engine are the core components of the Transformation Platform. VINCA parser is in charge of parsing VINCA file and making preparations for the transformation; VINCA synthesizer for BPEL4WS produces the BPEL4WS file in the XML format. Transformation Engine is the central element of the Transformation Platform, which is realized with the help of MetaData_Metaobject and Correlation_Metaobject. It does not use BC_Metaobject directly. Instead, Correlation_Metaobject uses BC_Metaobject implicitly. The algorithm of the transformation engine is given as follows:

Transformation Algorithm

```
INPUT: bizApp  //The parsing result object of business-end service composite in VINCA
OUTPUT: bpelApp  //The transformation result object for synthesizer for BPEL4WS
{
   bpelApp = new Vector();  //Produce the space for storing bpelApp
   for every element in bizApp
   {
      sourceElement = bizApp(i); //Get an element in bizApp
      /*Get the metadata of VINCA that the element belongs to*/
      Metadata metadata = MetaData_Metaobject.getMetaData(sourceElement.type);
      /*Get the correlation information about the obtained metadata*/
      correlations = Correlation_Metaobject.getMetaDataCorrelation(metadata);
      /*Do transformation according to the obtained correlation information*/
      for every correlation in correlations
```

```
        {
            Correlation correlation = correlations(j);
            /*Produce bpelApp element according to correlation information*/
            targetElement = transformation(correlation, sourceElement);
            bpelApp.add(targetElement);
        }
    }
    return bpelApp;
}
```

According to this algorithm, Transformation Engine can be implemented in an adaptive way through utilizing MetaData_Metaobject and Correlation_Metaobject.

4.3 Application

With the mechanisms and implementation discussed so far, the business-end service composite of the reference example in VINCA given in section 2 can be transformed to the IT-end composite in BPEL4WS now. *Book Excursion* activity in VINCA process, which refers to *Travel Agency* business service in the dynamic mode, is mapped to two *invoke* activities for invoking *ServiceSelectionServ* and *ServiceInvocationServ* services according to the dynamic composition pattern in BPEL4WS. The *Order Restaurant* activity, which is context-aware, is mapped to a corresponding *invoke* activity, but an additional *invoke* activity for invoking *ContextCollectionServ* service is inserted according to the context-awareness pattern in BPEL4WS. In this way, the resulted BPEL4WS specification of the example keeps the business characteristics defined in the business-end service composite in VINCA. Due to space limitation, the resulted BPEL4WS specification of the example is omitted here.

The above example shows that business characteristics of business-end composites in VINCA can be kept through the presented patterns. The following advantages can be gained: the shortage of BPEL4WS for supporting business characteristics can be complemented by special services and the information for keeping business characteristics can be maintained in an explicit way. Furthermore, when VINCA and BPEL4WS evolve, e.g. augmenting new business characteristics in VINCA, it is easy to adapt to changes through reconfiguring the reflective meta-space. We can reconfigure the correlation information including the information about business characteristics through the Metaobjects to achieve the adaptability. In this way, it needs not recode the Transformation Engine any more.

The approach and implementation proposed in this paper are experimenting in a real world project called FLAME2008 to facilitate business-end service composition.

5 Related Work

Though concepts of business-end service composition are relatively new and unexplored, there are a number of outstanding research efforts in this area. The research work in [17], with the goals like ours, also aims at supporting service composition from business perspectives. It provides a Business Process Outsourcing Language to capture business requirements. Then, the business-level elements can be

mapped into Web services flow to achieve on-demand Web services composition. The correlation between business requirements described in BPOL and Web services flow composition is not specified explicitly and is wired in codes directly.

In [4, 12], the authors raised the level of abstraction for developing service composition implementations in BPEL4WS, following an MDA approach. The correlation between a specific UML profile for BPEL4WS and constructs in BPEL4WS is defined and managed. However, because the UML profile is defined for BPEL4WS specially and do not support more business characteristics.

In [9, 10], a model-driven approach is proposed to transform platform-independent business models into platform-specific IT architectural models. The core mechanism for transformation is that special business patterns of business-level model are abstracted and assigned with corresponding IT implementation. However, the contents for keeping business characteristics and mapping are not maintained explicitly and need to be hard-coded when realizing the approach. An important contribution of this approach is the consistency checking of transformation, which will be considered in our ongoing work.

The work on model transformation shares some methodological similarity with our work. In [13], a framework for model transformation is given. A transformation scheme is defined in UML through extending the CWM Transformation metamodel. Through the transformation scheme, transformation can be represented explicitly. In [1], authors emphasized the importance of a high-level model transformation language. With this language, the behavior of model transformers can be formally specified. As such, the representation and the implementation of the behavior of transformers can be separated, and the development or maintenance of these transformers gets easier.

6 Conclusion

In this paper, we present a reflective approach to mapping business-end service composites in VINCA to IT-end service composites in BPEL4WS. The focuses are on: how to keep the business characteristics during the mapping; how to adapt to changes of the correlation information for the mapping. The approach provides the following valuable contributions:

- It allows representing the correlation information between VINCA and BPEL4WS on the meta-level explicitly. As such the correlation information can be maintained in an adaptive way and it is easy to adapt to new versions of VINCA and BPEL4WS.
- It can achieve the goal of keeping business characteristics through special components and patterns. By combining the contents about the components and the patterns with the correlation information on the meta-level, the business characteristics can be maintained in an adaptive way.
- It results in a lightweight and extendable Transformation Engine. It is lightweight because it “interprets” the Metaobjects to transform VINCA into BPEL4WS; it is extendable in the sense that it is easy to extend the transformation function by modifying the correlation information through the Metaobjects.

In our current work, the IT-end composites in BPEL4WS mapped from business-end composites in VINCA are still proprietary. How to support more sophisticated function of BPEL4WS, such as compensation, except handling, etc, is one of the important goals of our undergoing research work. How to express these contents in the correlation information and how to maintain them through Metaobjects are still open questions. Furthermore, we will also address the deployment and load-balancing of the services for keeping business characteristics to ensure effective usages.

References

1. A. Agrawal, Metamodel Based Model Transformation Language to Facilitate Domain Specific Model Driven Architecture, In OOPSLA'03, Anaheim, California, USA, 2003.
2. T. Andrews, F. Curbera et al, Business Process Execution Language for Web Services, http://www-106.ibm.com/developerworks/webservices/library/ws-bpel/, 2003.
3. A. Arkin, Business Process Modeling Language - BPML1.0, http://www.bpmi.org, 2002.
4. J. Amsden, T. Gardner et al, Draft UML 1.4 Profile for Automated Business Processes with a mapping to BPEL 1.0, IBM, 2003.
5. B. Benatallah, Q. Z. Sheng, and M. Dumas, The Self-Serv Environment for Web Services Composition, IEEE Internet Computing, 7(1): 40-48.
6. D. Edmond and A.H.M. Hofstede, Achieving Workflow Adaptability by means of Reflection, In The ACM Conf. Computer Supported Cooperative Work (Workshop on Adaptive Workflow Systems), Seattle, USA, 1998.
7. Y. Han, H. Geng et al, VINCA - A Visual and Personalized Business-level Composition Language for Chaining Web-based Services, In First International Conference on Service-Oriented Computing, Trento, Italy, 2003, LNCS 2910, 165 - 177.
8. G. Kickzales, J. Rivieres, and D. G. Bobrow, The Art of the Metaobject Protocol, MIT Press, Cambridge, Massachusetts, 1991.
9. J. Koehler, G. Tirenni, and S. Kumaran, From Business Process Model to Consistent Implementation, In Proceedings of EDOC'02, Switzerland, 2002, IEEE, 96-108.
10. J. Koehler, R. Hauser et al, A Model-Driven Transformation Method, In Proceedings of EDOC'03, Brisbane, Australia, 2003, IEEE, 186-197.
11. P. Maes. Concepts and Experiments in Computation Reflection, ACM SIGPLAN Notices, 1987, 147-155.
12. K. Mantell. Model Driven Architecture in a Web services world: From UML to BPEL, http://www-106.ibm.com/developerworks/webservices/library/ws-uml2bpel/, 2003.
13. J. Oldevik, A. Solberg et al, Framework for model transformation and code generation, In Proceedings of EDOC'02, Lausanne, Switzerland, 2002, IEEE, 181-189.
14. B. Orriens, J. Yang, and M. P. Papazoglou, Model Driven Service Composition, In First International Conference on Service-Oriented Computing, Trento, Italy, 2003, LNCS 2910, 75 - 90.
15. S. R. Ponnekanti and A. Fox, SWORD: A Developer Toolkit for Building Composite Web Services, In The Eleventh International World Wide Web Conference, Honolulu, Hawaii, USA, 2002.
16. B. Raman, Z. M. Mao et al, The SAHARA Model for Service Composition across Multiple Providers, In Proceedings of the First International Conference on Pervasive Computing, Zurich, Switzerland, 2002, LNCS 2414, 1-14.
17. L. J. Zhang, B. Li et al, On Demand Web Services-Based Business Process Composition, In International Conference on Systems, Man & Cybernetics, Washington, USA, 2003.