TOWARDS A BUSINESS PROCESS FORMALISATION BASED ON AN ARCHITECTURE CENTRED APPROACH

Fabien Leymonerie, Lionel Blanc Dit Jolicoeur, Sorana Cîmpan, Christian Braesch, Flavio Oquendo
University of Savoy at Annecy – ESIA Engineering School - LISTIC Laboratory
B.P. 806 - 74016 Annecy Cedex - France

Keywords: enterprise modeling, business process, enterprise application integration, architectural description language, architectural styles.

Abstract: Nowadays, enterprises need to control their business processes and to manage more and more information. EAI - *Enterprise Application Integration* - solutions offer a partial response to these requirements. However, the lack of formalisation that characterises such solutions limits the reuse and verification of properties. This paper claims that business processes have to be formally defined using a formalism that presents certain features (representation of several abstraction levels, domain specific concepts, property expression and preservation, etc.) and proposes the use of an ADL - *Architecture Description Language* - as formalism. A case study illustrates our proposition.

1 INTRODUCTION

Enterprises are increasingly aware that the control of their information system is a key element in industrial performance and a differential element in competitiveness. Today, one of the main issues restraining this competitiveness is the large number of application integrations required to synchronise and optimise business processes throughout enterprises. EAI - *Enterprise Application Integration* - solutions provide a new framework able to take synchronisation of business processes into account. This paper claims that business processes have to be formally defined using a formalism that presents certain features (representation of several abstraction levels, domain specific concepts, property expression and preservation, etc.) and proposes the use of an ADL - *Architecture Description Language* - as formalism.

In the following sections we go deeper into the problem we address (management of application integration synchronised with the optimisation of business processes) and we illustrate the use of this ADL in a specific industrial case study.

2 BUSINESS PROCESSES TO OPTIMISE, APPLICATIONS TO INTEGRATE

In a free market context, enterprises have to adopt new organisation models in order to meet customer requirements. Rather than acquiring new know-how, more and more enterprises work with external partners that are already efficient and consequently cheaper. Enterprises have to be able to react quickly to changes by adopting new effective organisation instead of betting on static organisation mode. In fact, enterprises have to identify, standardise and update their business processes which become the enterprise referent. Moreover, in order to have their information system in accordance with these business processes, most of the enterprises have generally deployed an *Enterprise Resource Planning* (ERP) system.

However, even if an ERP system covers a large spectrum of functionalities, it is not modifiable enough to support all kinds of system evolution (MESA, 1997a) (MESA, 1997b). Today, the trend is clearly oriented towards integration of heterogeneous applications based on EAI concepts (Mann, 1999) (Hostachy, 2000) (Schmidt, 2000) (Beck et *al.*, 2000).

Leymonerie F., Blanc Dit Jolicoeur L., Cîmpan S., Braesch C. and Oquendo F. (2004).
TOWARDS A BUSINESS PROCESS FORMALISATION BASED ON AN ARCHITECTURE CENTRED APPROACH.
In *Proceedings of the Sixth International Conference on Enterprise Information Systems*, pages 513-518
DOI: 10.5220/0002636405130518
Copyright © SciTePress

An EAI system is often viewed as the enterprise skeleton (Hostachy, 2000). An EAI system is a set of tools and integration software which allows connecting several enterprise applications, relying on different technologies (Linthincum, 2000).

An EAI system aims at controlling interactions between heterogeneous applications existing into an enterprise (legacy software or specific applications developed for internal needs) with the purpose to (Schmidt, 2000) (Stonebraker, 1999):

– build an efficient support to enact business processes by solving consistency problems arising from application interactions,
– supply a structure able to evolve by taking environment changes into account,
– propose to the end-user aggregated information to help him take decisions; this information is built with data managed by different applications.

The architecture of the EAI system can be structured in terms four parts:

– the *connection part* interfaces the EAI system and the COTS - *Component Of the Shelf* - of the enterprise,
– the *transformation part* aims at transforming the collected data format of the COTS to a pivot format,
– the *routing part* transmits the information to the good COTS, being based on the modeling of the business processes,
– the *modeling part* aims at providing a representation of the business processes of the enterprise.

An EAI system evolves continuously in order to be adapted to the COTS and especially to the enterprise business processes. It is essential to continuously check the consistency of the EAI system to take the COTS and process evolutions into account. So, it is necessary to formalize the business processes architecture.

Moreover, due to the structure of the enterprise in several interacting and co-evolving levels, the language used for the formalisation has to exhibit some features like:

– capability to express evolvable business architectures integrating business processes at different abstraction levels,
– capability to refine the business architecture into a concrete architecture defining integration between enterprise legacy software,
– capability to express quality attributes describing extra-functional properties related to the considered domain.

Two communities of researchers tackled the problem of formalising business processes: the software engineering community and the industrial engineering one. The first community started from the software engineering perspective, and focused on on the software processes. Efforts were put on finding approapriate means for their formalization. Although the work started from the study of software processes in general, the proposed formalisms are suitable for the definition of processes in general. Recent work adopts an architecture centric approach software development process, leading to the production of a certain number of architecture description languages (ADLs). The industrial engineering community focused on formalisms for business process description, as well as on processes themselves. Several process models and formalisms are proposed in the literature, and unification efforts are noted (Braesch et *al.*, 2000) (Vallespir et *al.*, 2003) (GERAM, 1999) (Vernadat, 2001).

The first community focused more on formalisms, leading to formal description languages allowing the description of evolvable systems, while the second focused more on models, leading to the production of better organization models. In this paper we propose to make the bridge between the two communities, and use performant formalisms for better modeling the identified processes. We use thus an architecture description language for the modeling of business processes. The ADL we adopt has been proposed in the ArchWare European project presented below in this draft.

More precisely, our work aims at using a parameterized ArchWare environment able to develop solutions adapted to the industrial management of SMI/SME companies. This industrial management is based on COTS such as ERP, Quality Management Software, Maintenance Management Software, etc. In our solution, we integrate these COTS within an EAI solution built following a style-based software development process.

3 THE ARCHWARE ADL

The presented work has been partially funded by the European Commission in the framework of the IST ArchWare Project (IST–2001–32360). Hereafter, we present the main issues of the ArchWare project and the language it proposes.

The ArchWare project (www.arch-ware.org) aims to develop and validate an innovative architecture-centric software engineering framework, i.e. architecture description and analysis language, architectural styles, refinement models, architecture-centric tools, and a customisable software environment. The main concern is to guarantee required quality attributes throughout evolutionary software development (initial development and evolution), taking into account domain-specific architectural styles, reuse of existing components, support for variability on software products and product-lines, and run-time system evolution.

Evolvable software systems such as EAI systems are those that are capable of accommodating change over an extended lifetime with reduced impact for cost and schedule and controlled impact for quality. The key novelty of the ArchWare project approach is its holistic view of evolvable software systems. This starts with a high-level description of the software system expressed in a formal architecture description, possibly using domain-specific architectural styles. The required quality attributes (e.g. scalability, performance, modifiability, safety, reliability) of the software system may then be proved/checked/evaluated using analysis tools. This high-level description may then be incrementally refined into more low-level, intermediate descriptions until reaching a concrete level that may be used for application generation. Refinements are applied to architecture descriptions as well as to quality attributes.

The architecture description is based on architectural styles. An architectural style counts the whole of the functionalities and of the architectural properties of a particular field; its objective is to limit the architecture definition space (Cîmpan et *al.*, 2003). Using architectural styles allows an architect to reuse the collected wisdom of the architecture design community in much the same way that object-oriented design patterns give novice designers access to a vast array of experience collected in the object-oriented design community (Klein and Kazman, 1999).

The ArchWare ADL (Oquendo et *al.*, 2002) (Cîmpan et *al.*, 2002) is a formal language for modeling evolvable software architectures. It provides a higher level of abstraction based on formal foundation based on the concepts of the π-calculus formal algebra (Milner et *al.*, 1992), the μ-calculus formal algebra (Kozen, 1983), persistent programming and dynamic system composition and decomposition. The ArchWare ADL also defines a set of style mechanisms that allows starting from the core language, the creation of domain specific languages.

The ArchWare ADL core description language is based on the concept of formal composable components and on a set of operations for manipulating these components. The ADL supports the concepts of behaviours, abstractions of behaviours, to represent respectively running components and parametric component types. Behaviour is described using all the basic π-calculus (Milner et *al.*, 1992) operations as well as composition and decomposition. Communication between components is via channels represented by connections (representing component interfaces as well). The ArchWare ADL allows the definition of evolvable architectures, i.e. where new components and connectors can be incorporated and existing ones can be removed, governed by explicit policies.

The style layer of the language allows the definition of architectural element styles, represented by property-guarded behaviour abstractions. Thus, we can define kinds of architectural elements and express constraints on their internal structure and their behaviour. The style layer allows the definition of domain specific extensions of the core language, where the domain properties can be explicitly defined and preserved.

These ArchWare ADL features make it suitable for the modeling of enterprise processes, which due to the changes in their economic environment are constantly changing. We use this ADL to describe the evolution characteristics of an EAI system and to define the EAI domain.

4 EAI FORMALISATION

Currently, there is no EAI solution proposing a model able to represent and control the business processes of the company. We claim that the business process view of an enterprise is the key element to the definition of the EAI architecture.

Within the framework of our work, we especially were interested in the “modeling” level. In fact, we need on the one hand to verify the model at a “generic level”, and on the other hand, to verify the interactions between the different elements of the model.

4.1 EAI architecture

In order to apprehend this *modeling* part, we have decomposed it in three abstraction levels:

- The *Inter-Enterprises* level where enterprises are organised for providing a product or a service are defined.
- The *Enterprise* level where ICs - *Industrial Component* (Chaudet, 2002) (Braesch et al, 2000) -, one PS - *Performance System* - and one RMS - *Resource Management System* - are defined.
  - The *Performance System* identifies the industrial components able to satisfy a need and evaluates the goal reached by these components,

   – The *Resource Management System* manages the different enterprise resources.

– The *Workflow* level where COTS and a Workflow engine are defined. The Workflow engine synchronizes the COTS interactions.

This architectural framework aims at enhancing modifiability decoupling interactions between various architectural elements at the three abstraction levels.

In the following section we illustrate the use of the ArchWare ADL in the modeling of a framework for the *Inter-Enterprises* level, and more particularly a process related to negotiation.

4.2 Architecture formalisation

The process used to illustrate an EAI system architecture design is placed in the context where an enterprise needs to sub-contract a part of its manufacturing process and begins a negotiation process with several suppliers (Blanc dit Jolicoeur et *al.*, 2002).

The negotiation process involves a contractor and several suppliers, identified as enterprise components in the architecture (cf. Figure 1). A negotiation mediator, entailing the negotiation protocol, is added to the architecture.

The protocol is succinctly described below:

– The contractor opens an invitation to tender by sending an offer to chosen suppliers,
– The suppliers reply to this invitation to tender by sending a response (a proposal if they are interested or a signal if not),
– The contractor studies the received proposals and establishes a contract from interesting proposals.

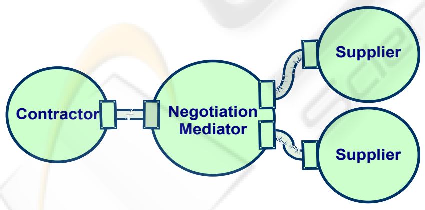

Figure 1: Architecture for negotiation process

The set of suppliers concerned by the negotiation depends on the offer. As there are several negotiations, the suppliers are not always the same. So, the architecture evolves during the production plan construction. Thus, the architecture needs to be dynamically evolutive.

We first define the communication interface of elements and the negotiation protocol between one contractor and one supplier.

The interface of an architectural element is given by a set of connections. We use the **view** type provided by the ADL in order to structure the interface (**attachment**) in several interaction points. Thus we defined the Negotiation_Attachment type.

```
type Negotiation_Attachment is view[
    offer:connection[String, String],
    answer:connection[String,String],
    noanswer:connection[String],
    contract:connection[String, String],
    close:connection[String]]
```

The ***one_one_negotiation*** protocol expresses the negotiation between one contractor and one supplier by an ordonnanced set of communication actions.

```
via ATTACHMENT_NAME ::CONNECTION_NAME send MESSAGE
```

It is formally described in the following.

```
value one_one_negotiation is abstraction(
    contractor: Negotiation_Attachment,
    supplier: Negotiation_Attachment).{
{   via contractor::offer
        receive offer:String.
    via supplier::offer send offer.
    choose
    { { via supplier::answer
            receive answer:String.
        via contractor::answer send answer.
        choose
        { { via contractor::contract
                receive contract:String.
            via supplier::contract
                send contract.
            via contractor::close receive     }
        or{ via contractor::close receive.
            via supplier::close send }
        }
    or{ via supplier::noanswer receive.
        via contractor::noanswer send.
        via contractor::close receive}
    or{ via contractor::close receive.
        via supplier::close send           }
}   }
```

Several ***one_one_negotiation*** protocols need to be dynamically generated in order to contract with several suppliers. Thus, we define a recursive abstraction (***negotiation_creation***) that instantiates a new one_one_negotiation protocol after receiving the corresponding order.

```
recursive value negotiation_creation is
abstraction().{
    value new_negotiation is free
    connection(Negotiation_Attachment).
    via new_negotiation receive
    contractor_attachment, supplier attachment.
    compose{
        one_one_negotiation(contractor_attachment,
                            supplier attachment)
    and
        negotiation_creation()
}   }
```

This particular negotiation process is thus formalized, allowing the verification of several properties. In the following section we focus on how domain know-how can be captured using architectural styles.

4.3 Domain formalisation

Using ArchWare ADL we can formalise an EAI style in order to give support for EAI architect. On the one hand the EAI style describes the various elements of EAI architectures. On the other hand it defines the whole of the constraints specific to this kind of architecture (Blanc dit Jolicoeur et *al.*, 2003a).

We use styles in order to provide specific kinds of architectural elements for the industrial domain (cf. Figure 2).

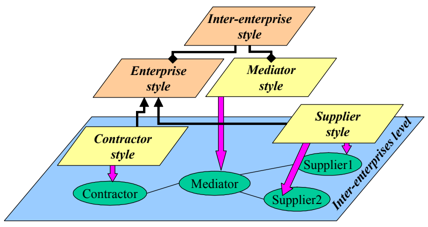

Figure 2: Architectural styles

For each main part involved in the process (enterprises, mediator) a style is introduced. Thus we define an ***InterEnterprise*** style which gives a framework for the whole system. The ***InterEnterprise*** style refers to other styles like the ***Enterprise*** style. There are two sub-styles of ***Enterprise***, namely ***Contractor*** and ***Supplier***. These styles give a framework for sub-structures of the system.

At definition, each of these styles has a parameterized computational part guarded by a set of additional properties (in the **verifying** part of the definition). The use of mixfix notations for new architectural elements (mechanism provided by the ADL) offers the possibility to introduce domain-specific notations for the new constructs (in the **as** part of the definition).

The ***InterEnterprise*** style must be parameterized with a set of ***Enterprise*** components and a ***Mediator*** component. If these components verify expressed constraints, they are composed when the style is applied. The constraints mean that components must be of ***Enterprise*** and ***Mediator*** kind, and that enterprises are not directly connected but using a ***Mediator***.

```
value InterEnterprise is
style(enterprises:sequence[Enterprise_T],
        mediator:Mediator_T).
{    compose{
        iterate enterprises by e
        from ent_comp is abstraction().{}
        accumulate compose{e and ent_comp}.
        ent_comp
     and  mediator
     }
}.
verifying(
     to enterprises apply{
        forall(e|e in style Enterprise) and
        forall(e1,e2|not(connect(e1,e2)) and
        forall(e1| connect(e1,mediator))
     }.
     mediator in style Mediator
)
as{ interEnterprise grouping
        enterprises
     which are mediated by
        mediator }
```

As a sub-style of ***Enterprise***, the ***Contractor*** style exhibits additional properties.

```
value Contractor is Enterprise{}.
verifying( CONTRACTOR SPECIFIC PROPERTIES )
```

Finaly an architecture can be instantiated from the style. In the following example we used the style specific notation for defining an EAI system with one contractor and one supplier.

```
value contractor is Contractor( PARAMETERS ).
value supplier is Supplier(PARAMETERS ).

value system is
     interEnterprise grouping
            sequence(contractor,supplier)
     which are mediated by
            negotiation_Med
```

5 CONCLUSION

As we saw, enterprises need to integrate existing applications in order to stay competitive. But these applications are heterogeneous and have to synchronise and optimise well-defined business processes. The EAI approach provides a framework enabling the synchronisation of these business processes. However, if this approach gives a broad spectrum of tools that can be used to control enterprise activities, there is a lack of formalisation leading to the incapacity to handle the co-evolution between the several enterprise processes situated in different abstraction levels. In order to face this need for co-evolution we propose the use of an architecture description language allowing the description of evolvable systems. This language has been proposed in the the ArchWare European project, who's main concern is to guarantee required quality attributes throughout evolutionary software development (initial development and evolution), taking into account domain-specific architectural styles, reuse of existing components, support for variability on software products and product-lines, and run-time system evolution.

REFERENCES

Beck, M., Finout, J., and Westlake, B., 2000. "*Change Management and EAI*", in eAI Journal, September 2000, pp. 77-79.

Blanc dit Jolicoeur, L., Braesch, C., Dindeleux, R., Gaspard, S., Le Berre, D., Leymonerie, F., Montaud, A., Chaudet, C., Haurat, A., and Théroude, F., 2002. "*Final Specification of Business Case 1, Scenario and Initial Requirements*". ARCHWARE European RTD Project IST-2001-32360. Deliverable D7.1b. December 2002.

Blanc dit Jolicoeur, L., Braesch, C., Dindeleux, R., Gaspard, S., Le Berre, D., Leymonerie, F., and Montaud, A., 2003a. "*Definition of Architectural Styles and Process Models for Business Case 1*". ARCHWARE European RTD Project IST-2001-32360. Deliverable D7.2. June 2003.

Blanc dit Jolicoeur, L., Braesch, C., Dindeleux, R., and Théroude, F., 2003b. "*An EAI system based on an Industrial Component model*". In proceedings of CESA 2003, IMACS Multiconference, July 9-11 2003, Lille, France.

Braesch, C., Théroude, F., and Haurat, A., 2000. "*OLYMPIOS : towards a component-based enterprise modeling*", CEN Workshop, Berlin.

Chaudet, C., 2002. "*Pi-Space : langage et outils pour la description d'architectures évolutives à composants dynamiques. Formalisation d'architectures logicielles et industrielles*". Ph.D. Thesis (in french) of the University of Savoy.

Cîmpan, S., Oquendo, F., Balasubramaniam, D., Kirby, G., and Morrison, R., 2002. "*The ArchWare ADL: Definition of the Textual Concrete Syntax*". ARCHWARE European RTD Project IST-2001-32360. Deliverable D1.2b. December 2002.

Cîmpan, S., Leymonerie, F., and Oquendo, F., 2003. "*ArchWare ADL Foundation Style*". ARCHWARE European RTD Project IST-2001-32360. Internal Report, R1.3-1. June 2003.

GERAM, 1999. "*Generalised Reference Architecture and Methodology*". Version 1.6.3, IFIC-IFAC Task Force, March 1999.

Hostachy, E., 2000. "*Le système d'information doit être centré sur l'EAI*", in Informatiques Magazine, May 2000, pp. 40-44.

Klein, M., and Kazman, R., 1999. "*Attribute-Based Architectural Styles*", technical report of Carnegie Mellon University.

Kozen, D., 1983. "*Results on the propositional Mu-Calculus*", in Theorotical Computer Science 27, 1983, pp 333-354.

Linthincum, D., S., 2000. "*Application Servers and EAI*", in eAI Journal, 2000.

Mann, J., E., 1999. "*Workflow and EAI*", in eAI Journal, September-October 1999, pp. 49-53.

MESA International, 1997a. "*Execution-Driven Manufacturing Management for Competitive Advantage*", White Paper number 5.

MESA International, 1997b. "*MES Functionalities & MRP to MES Data Flow Possibilities*", White Paper number 2.

Milner, R., Parrow, J., and Walker, D., 1992. "*A calculus of mobile processes*". Information and Computation, September 1992, pp. 1-40.

Oquendo, F., Alloui, I., Cimpan, S., and Verjus, H., 2002. "*The ArchWare ADL: Definition of the Abstract Syntax and Formal Semantics*". ARCHWARE European RTD Project IST-2001-32360. Deliverable D1.1b. December 2002.

Schmidt, J., 2000. "*Enabling Next-Generation Enterprises*", in eAI Journal, July-August 2000, pp. 74-80.

Stonebraker, M., 1999. "*Integrating Islands of Information*", in eAI Journal, September-October 1999, pp. 1-5.

Vallespir, B., Braesch, C., Chapurlat, V., and Crestani, D., 2003. "*L'intégration en modélisation d'entreprise: les chemins d'UEML*". MOSIM'03, French conference on modeling and simulation in enterprise, Toulouse, April 23-25 2003, pp. 140-145.

Vernadat, F., 2001. "*UEML: towards an unified enterprise modeling language*". MOSIM'01, French conference on modeling and simulation in enterprise, Troyes, April 25-27 2001, pp. 3-10.