Business Process-aware ERP Development and Evolution

Daisen Wei[123], Jiazhong Zhang[23], Xueqing Li[1], Longye Tang[13]

1. College of Computer Science & Technology
Shandong University
Jinan, P.R. China

2. State Key Laboratory of Novel Software Technology
Nanjing University
Nanjing, P.R. China

3. Inspur Genersoft Co.,Ltd.
Inspur Group
Jinan, P.R. China

***Abstract*—with the rapid development of the Internet and its wide application, the management of enterprises is gradually shifting from a closed environment to a dynamic and open one. So, how to improve the flexibility of ERP software under the Internet computing environment has become a key issue for ERP vendors and users. It is also the main focus of this paper. One process-aware ERP framework is proposed to support Internet-based ERP development and evolution in order to improve the flexibility of ERP software. In it, users, using visualized design tool, can reedit or change the existed business process model as required. And then the updated business process model and its changes can be automatically propagated to process engine. The latter can call the analyzer for changes to determine the complexity of these changes, which can be measured by the property of forward compatibility defined to evaluate the scale and status of the coming evolution. After that, activity instance scheduler together with change locators are called to cooperatively complete the process evolution by operating the caching sequence of activity and service instances. Comparison with current research work and our practice had shown that this framework runs well.**

***Keywords*—ERP;business process-aware; evolution; service; forward compatibility**

I. Introduction

With the rapid development of the Internet and its application, the management of enterprises is shifting from a closed environment to a dynamic and open one. And order-driven global marketing makes the implementation of enterprises a dynamic process. Under this background, different enterprises and even the same one with different products must have to frequently adjust or reengineer their business processes as required. As one of important information management tools used by modern enterprises, ERP systems are being required to dynamically adapt to these changes. That is, ERP systems should have the flexibility-an ability to provide new products or services by a high-efficiency and low-effort means to adjust or change its products or services.

Obviously, this flexibility meets the requirements from ERP users and vendors nowadays. On the one hand, users expect that ERP software could help improve the management and development of enterprises and also easily adapt to the changes of business processes, because this would make enterprises obtain more profits from the use of ERP. On the other hand, ERP vendors expect that their ERP products would always work well and suit the business management of enterprises, and also conveniently be maintained as users' requirements change.

Additionally, the rapid development and wide application of the Internet makes it a new mainstream platform for software to be used, developed, deployed and executed. In order to adapt to this environment and the fundamental change of underlying support platform, Internetware is introduced to describe the emerging software paradigm for the Internet computing environment. It is a software paradigm for Internet Computing where software is architected as the Internet, developed with the Internet, and executed on the Internet.

So, how to make use of these technologies to improve the flexibility of ERP software is attracting more and more attentions from ERP vendors. In this paper, one hierarchy models was constructed first to separately define three reused business objects of ERP software. And then a framework for business process-aware ERP software development and evolution was proposed to describe the mechanism of ERP flexible development.

The structure of this paper is organized as follows: in section 2, one hierarchy model was constructed to define different reused business objects in ERP software. In section 3, based on this model, a framework for process-aware ERP Software development was proposed to improve the flexibility of ERP Software. And a case study was given. In section 4, related work on ERP flexible development and maintenance was introduced and a comparative analysis was given. Conclusion was given in the final section.

II. ERPHM: A Hierarchy Model of ERP Software

The management of manufacturing enterprises has some specific business features, which are listed as follows.

(1)Variety of key applications. Enterprise management usually involves a variety of business fields and departments, which can be categorized into different key applications, such as *producing management, financial account, supply chain management, hired resource management, office automation system, account management, fund management, business* *intelligence, Material Management (MM), Sales and Distribution (SD)*. Each key application developed as a single product can be provided independently for users. In most cases, a customized ERP product package including several key applications can be provided to meet the personalized requirements from different users.

(2)Business process-oriented. The process defines the involved activity sequence to apply the business practices and get the work done so that a meaningful result can be achieved. A key application usually involves lots of the specific business processes with huge data, which can be changed and reengineered as enterprise's development and management strategies change. So, process-oriented business execution is one of the main features of enterprise business management.

From the above features, it can be concluded that changes in enterprise requirements can usually be reflected by two aspects, namely key application and business process. According to this fact and the idea of separation of concerns, five types of semantic layers for ERP software, namely key application, business process, activity, service and component, were defined as the following Fig. 1.

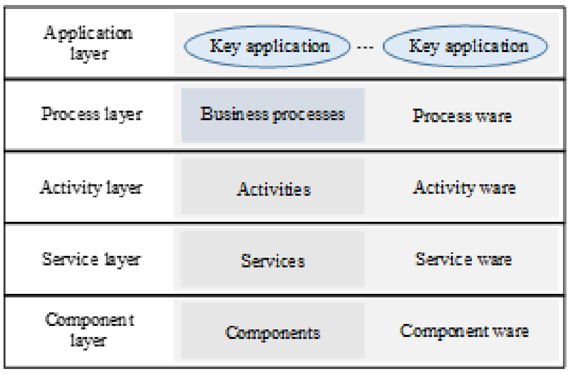

Fig. 1. Hierarchy model for ERP software (ERPHM)

ERPHM consists of five semantic layers, in which five semantic objects including key application, process, activity, service and component are defined respectively.

(1) Component layer. In this layer components are defined to implement basic computing logic. This layer is invisible for users. Here users usually refers to process designer or developers of ERP software. That is, components are deployed at locally physical nodes of the Internet. And they are developed only to support the implementation of services that are directly configured by users on a unified platform. Additionally, compared to the service defined below, their outputs after execution may be invisible to users. In our practice of ERP developments, most components are atomic and implement the basic computing logic. Comparatively, the coarse-grained business logic are encapsulated as services that can be configured and reused under Internet.

(2) Service layer. One type of reusable business objects is defined as service, which is implemented by components or the orchestration of local services. A service is a coarse-grained semantic unit with specific inputs and outputs. It usually encapsulates a specific business procedure or transaction. It may be an application service that implement a single transaction with specific computing logic, or a business service that is an orchestration of local application services and/or business services. A service *SV* is often used to implement a comparatively independent business logic with specific inputs and outputs. It may be an application or business service. And the latter is the orchestration of application ones. Correspondingly, the instance of *SV* is labeled as *SV_INST*.

(3) Activity layer. Activities are defined to depict coarse-grained semantic units to usually complete specific target (a specific business procedure or transaction). They usually have specific inputs and outputs, and are also reused in different business processes. Its definition is given as follows.

Definition 1 activity (*ACT*). An activity *ACT* is a triple (*AP*, *AT*, *AF*), in which AP represents the set of states of this ACT and $AT= \{SV_1,SV_2,\ldots SN_m\}$ (*m* is an integer) refers to the set of services binding to ACT, and $AF\subseteq AP\times AT \cup AT\times AP$ represents flow relations of ACT.

Each activity is the choreography of services together to complete a definite target. According to business logic of this activity, the involved services can be organized by some relation operators such as sequence, parallel, branching/converging and loop. Conceptually, an activity is the choreography of services, which can be done by using a global management platform tool, *BPDesigner*, to configure and bind those services to the matching activities. Additionally, from the expressional descriptive, activities can be regarded as the definitions of target nodes of business processes, yet services are the implementation of them.

(4) Process layer. A business process usually is an ordered sequence including one or many transaction nodes that complete the specific activity. A transaction node in it may be a single activity or a sub process consisting of activities. Activities are the basis of developing business processes. Different processes can also share data from the same DB table. A business process can be reengineered or updated as requirements change. In practice, a business process usually consists of clear work and data flows which describe specific business scenarios of enterprises. That is, in one ERP product, each business process is an ordered collection of form-driven computing operations (query/store/update/delete/compute) with definite outputs.

Definition 2 business process (*BP*). A *BP* is a triple (*BPL*, *BT*, *BF*), in which *BPL* represents the set of states of this BP and $BT= \{ACT_1, ACT_2, \ldots ACT_n\}$(*n* is an integer) refers to the set of activities binding to flow nodes in business process BP. And $BF\subseteq BPL\times BT \cup BT\times BPL$ represents flow relations of BP. Correspondingly, the instance of BP is labeled as *BP_INST*. In a BP, all activities are ordered by some flow operators, such as *SEQ, AND-Split/AND-Join, OR-Split/OR-Join* and *LOOP*.

(5)Application layer. Key applications are defined in this layer. For a key application (*KP*), it always consist of some processes that drive the implementation of specific work or targets. Generally, key applications can be determined according to types of products and business domains. Key applications can either stand alone or be integrated into one all-encompassing ERP system for users. Generally, the changes of requirements can correspond to reengineering or update of business processes in key applications. For the sake of describing these changes, some definitions are given as follows.

Definition 3 change for business process (CBP). For business process *BP* = (*BPL, BT, BF*) and *BP'* is the updated version of *BP*, *<SN, (SUB_ACTs, SUB_BP_R), TN>* is named as a change from *BP* to *BP'*, in which *SN* refers to the prior node followed by the changed one; *TN* represents the next node after the change ends; and *(SUB_ACTs, SUB_BP_R*) represents the change including nodes and their relations. An example is illustrated as the following Fig.2, in which the business process *BP* and *BP'* (slice) are expressed by using petri net (here, a transition represents an activity).

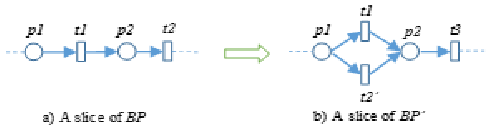

Fig. 2. Change from BP and BP'

According to Fig 2, the change from BP to BP' can be depicted as (*p1, (<t1, t2'>, <t1 PA t2' >), p2*), in which an activity *t2'* is added to run in parallel with *t1* and other nodes keep unchanged. *PA* refers to the parallel execution relation between activities. Note that, in the above example (Fig.2) the unchanged nodes keep their labels unchanged respectively in two slices of *BP* and *BP'*.

III. A Framework for Process-Aware ERP Software Evolution

The framework, including three parts, namely process editing, business process engine and activity instance scheduling, is illustrated in the following *Fig.3*

A. Description of Business Process in XML and its Changes

Users can edit or change processes by using a visualized business Process design tool, BPDesigner, as needed. After that, two documents to describe new business process, one for depicting the whole business process and the other used to record the changes that take place in this business process, will be automatically generated by BPDesigner. They are all written in XML. In two documents, three types of elements, activity node, transition edge and route node are defined respectively. For an activity, the main attributes are *Aactivity Name*, and *Services* binding to it et al. and that of a transition are *FromNode* and *ToNode*. For a service, they are *ServiceName*, and *ServiceLocation*. For a rout node, its type may be one of set {*AND-Split/AND-Join, OR-Split/OR-Join, LOOP*} that are usually flow operations to represent and branching/converging, or branching-converging and loop branching.

B. Business Process Engine(BPE)

BPE is used to create and execute the instance of business processes. Before doing it, it must call an analysis module named as analyzer for changes(AFC) to measure the status of the changes of business process, because BPE must determine whether the evolution can be done or not. The dynamic evolution of business process is a complicated and time-consuming work. So, for doing this work well, a property is given to help select the evolutional strategy.

Property: Forward Compatibility (*FC*). An evolution for a business process *BP* CAN BE DONE if and only if one of conditions is satisfied: a) the involved activity(s) in *BP* has not been executed. Or b) the involved activity(s) in *BP* has started but the changed service(s) in it has not been executed.

Note that, for the sake of description, use *BP* to represent current business process and *BP'* as the evolved version of *BP* below.

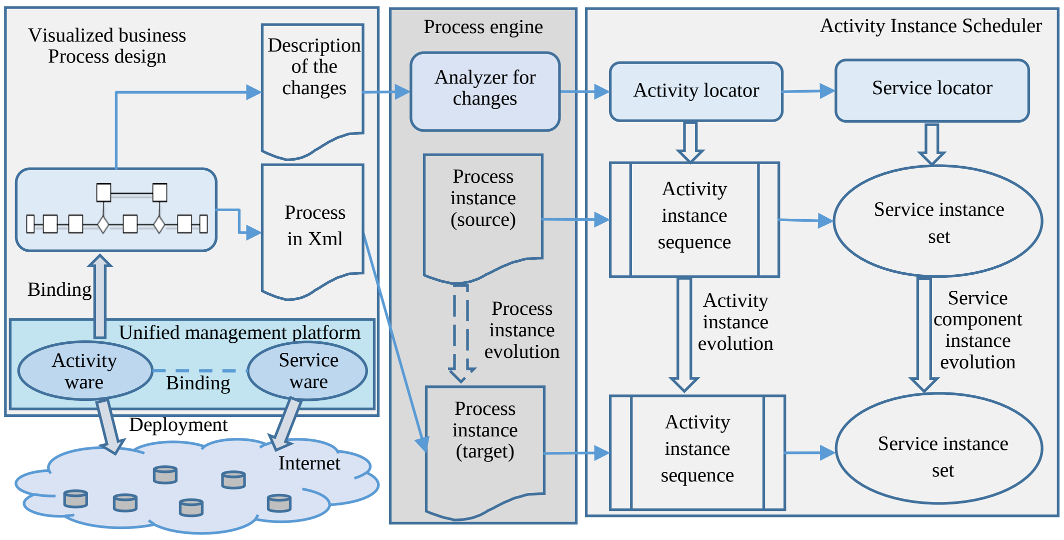
Visualized business Process design
Description of the changes
Process in Xml
Binding
Unified management platform
Activity ware
Binding
Service ware
Deployment
Internet
Process engine
Analyzer for changes
Process instance (source)
Process instance evolution
Process instance (target)
Activity Instance Scheduler
Activity locator
Service locator
Activity instance sequence
Service instance set
Activity instance evolution
Service component instance evolution
Activity instance sequence
Service instance set

Fig. 3. A framework for process-aware ERP software evolution

For a), if the involved activities in *BP* that is updated or changed has not been executed, the current instance of this *BP* can be evolved by scheduling the NEW instances of these activities. That is, a new instance of this *BP* is not created.

For b), if the involved services binding to activities in this *BP* is updated or replaced and they has not been called, the instance of *BP* can be evolved through the update or replacement of instances of these services.

According to the above property FC, if BPE makes sure that the evolution caused by changes can be done, then it will call Analyzer for changes (AFC) to format the changes and submit them to Activity Instance Scheduler (AIS). For the changes of business process, if the above property FC is not satisfied, then BPE will directly create a new instance for this business process and start it. Note that, in order to do the evolution, two lists of the instances of activities and services binding to them must be cached, named as ACT_INSTs and SV_INSTs respectively. They are the key information of the evolution of business process instances.

C. Activity Instance Scheduling

After receiving the changes from AFC, AIS is in charge of the following evolution work. And in this part, Activity Locator (AL) and Service Locator (SVL) are developed to help AIS to locate the changed activities or services respectively. The detailed evolution can be done through the following steps:

Step1: according to changes, AIS first call AL to locate the activities that has been changed or updated. If the services binding to them has also been replaced or changed, then turn to step 2, if not, then turn to step 3.

Step2: AIS calls SL to locate the place that changes occur in *SV_INSTs*. And creates the instance of new services and updates the *SV_INSTs*.

Step3: AL creates the instances of new activities and then updates the *ACT_INSTs*

Step4: after the above steps, AIS will send a message to BPE. And the latter will continue to execute the evolved instance of business process.

IV. Related work

ERP software evolution usually reflects the changes of enterprises' requirements, yet the traditional ERP development technique, such as ERP suite-based second development, do not satisfy such changes to requirements. Due to the differences among enterprises' businesses, a version of ERP product would hardly been used by different enterprise users, even which are from the same field [1][2]. So, industry-oriented ERP development had been widely implemented by ERP vendors, in which a corresponding solution contained practice content that had been developed and tested to provide a complete lifecycle management solution for a particular industry [3]. The productivity of ERP venders and ERP software quality, however, still fall short of expectations and software systems continue to suffer from symptoms of aging as they are adapted to changing requirements.

In order to support the evolution of process model at run-time stage and propagating the changes to the active process instances, Song *et al.*[4] analyzed dynamic evolution of process-aware information systems and the state of the art of it. Vasilios *et al.*[5] focused on the evolution of software services and provided a unifying theoretical framework for controlling the evolution of services that deals with structural, behavioral and QoS level-induced service changes. Additionally, software architecture, especially dynamic software architecture, was widely researched to support the implementation of software evolution. Mei et al.[6] proposed a centric software development method called ABC (architecture based component composition) to support the implementation of the whole-lifecycle software development and had obtained many achievements in component-based development and evolution for software and Internetware. In contrast to these researches, the framework proposed in this paper not only gives the method to obtain changes that occur and propagates them to the kernel of ERP process instances, but also defines the property of forward compatibility to improve the efficiency of process evolution. And according to the framework, a platform for ERP maintenance and evolution had been developed successfully and run well.

V. Conclusions

In order to improve the flexibility of ERP software under Internet, a framework for business process-aware ERP development and evolution was proposed to improve the flexibility of ERP software. For supporting the implementing of this framework, a hierarchy model was provided first according to separation of concerns and best practices from ERP development, in which five reusable units were defined respectively. And then a property is defined to help process engine to select the evolutional strategy. According to the record and analysis of changes and their propagation, BPE together with activity and service locators completed the execution of the changed business process. Our practice has showed that this method is very effective and efficient.

Acknowledgment

The research work is supported by the Open Foundation of State Key Laboratory of Novel Software Technology (Grant No. KFKT2014B01).

References

[1] Marco Panunzio, Tullio Vardanega. A component-based process with separation of concerns for the development of embedded real-time software systems. Journal of Systems and Software, Volume 96, October 2014, Pages 105–121. (DOI: 10.1016/j.jss.2014.05.076)

[2] Graham Bleakley, Keith Collyer, Joanne Scouler. Best practices for systems and software development: An introduction to IBM Rational Solutions for Systems and Software. IBM Developer Works. pp.1-12, 17 September, 2013.

[3] Shi Liang Wu, Wu He. Enterprise Systems from ERP, IERP to ERP. IEEE SMC eNewsletter issue 35, June 2011. http://www.my-smc.org/news/back/2011_06/main_article2.html.

[4] Song Wei, Ma XiaoXing, Hu Hao, Lü Jian. Dynamic evolution of processes in process-aware information systems. Journal of Software, 2011, 22(3):417−438.

[5] Vasilios Andrikopoulos, Salima Benbernou, and Michael P. Papazoglou. "On the Evolution of Services," IEEE Transactions on Software Engineering, vol. 38, no. 3, pp. 609-628, May-June, 2012.

[6] Mei H, Huang G, Zhang L, Zhang W. ABC : a method of software architecture modeling in the whole lifecycle. Sci China Ser-F Inf Sci, 2014, 44(5): 564-587