2011 IEEE Ninth International Conference on Dependable, Autonomic and Secure Computing

Migrating Complex Business Process to Cloud based on M_{spoa} and $CBPM$

Hai WAN*, Yang YU†
Software School
Sun Yat-Sen University
Guangzhou, People's Republic of China
**Corresponding author: wanhai@mail.sysu.edu.cn*
†yuy@mail.sysu.edu.cn

***Abstract*—Eliciting and describing complex business process consistently and unambiguously is important and critical for migrating complex business processes to cloud platform. M_{spoa} (*Subject Predicate Object Adverbial complex business process description Meta-model*) is proposed in this paper, which can represent the static relationship in business process description problem space. Based on M_{spoa}, clewed and cored by form business process, $CBPM$ (*Complex Business Process Model*) is presented to describe dynamic behavior relationship. By the means of the migration steps described in this paper, incoherence and bounce in the course of business processes analysis can be overcome, assuring the correctness of the artifacts.**

***Keywords*-Business process migration; Cloud computing; Static model M_{spoa}; Dynamic behavior model $CBPM$**

I. Introduction

Because of its apparently cheap, simple and scalable nature, cloud computing becomes a disruptive technology and fundamental change set to change how IT systems are deployed [1]. As more organizations planning to migrate automating business process system or enterprise services into hybrid cloud-based deployments, the migration of data, services and processes to the cloud platform must be achieved based on well defined models [2] [3].

In this paper, we focus on the migration of business processes to the cloud platform, in which individual application components or legacy systems can be deployed and applied accurately in the cloud. It is increasingly common to describe organizations as sets of business processes that can be analyzed and improved by approaches such as business process modeling. Successful business process modeling relies on an adequate view of the nature of business processes, however, there is a surprising divergence of opinion about the nature of these processes. Typically, Mansar proposes a strategy for the implementation of business process redesign, whose backbone is formed by the analytic hierarchy process multi-criteria method [4]. Nuno proposes a conceptual framework to organize different views of business processes under four headings [5]. As the domain knowledge are quite different for different software applications, different affecting dimensions provide guidance for task definition, independent variables and controls [6]. Wand described three models they have developed of information systems' deep-structure properties, such as entities, relationships, characteristics, processes, and roles. [7].

Semiformal, formal, and executable (or informal) specification languages [8] have been developed for describing business process. Informal specification: uses natural language, which is easy to describe and understand, but ambiguity and inconsistency; Semi-formal specification: means that the specification techniques have formal syntax without formal semantics, including JSD, object-oriented visual modeling language (such as: UML [9]); Formal specification: is composed of syntax, semantics. Currently, there are more than 80 formal specification languages.

The contribution of this paper is proposing M_{spoa} (*Subject Predicate Object Adverbial complex business process description Meta-model*) and $CBPM$ (*Complex Business Process Model*). As M_{spoa} describing problem space with hierarchical and multi-view points, $CBPM$ can be represented as the logical combination of three layers with formalism semantics.

- Business process layer describes the effects of form businesses on form states cored with form life cycle.
- Multi-process interactive layer expresses global processes based on interactive messages between form businesses processes.
- Business structure layer consists of user interfaces as form business ingredient.

It is demonstrated M_{spoa} and $CBPM$ can describe requirements naturally, incarnate global business process, and embody functional and non-functional requirements.

The paper is organized as follows. In the next section, we depict the problem and related factors in business process migration. Section 3 considers business process from meta-model perspective and defines M_{spoa} (*Subject Predicate Object Adverbial complex business process description Meta-model*), as well as $CBPM$ (*Complex Business Process Model*) used in describing dynamic behavior relationship. Section 4 proposes complex business process description language $CBPM_o$ based on M_{spoa} and $CBPM$. Section 5 describes complex business process migration steps. Finally, we summarize our work and discuss future research.

978-0-7695-4612-4/11 $26.00 © 2011 IEEE
DOI 10.1109/DASC.2011.90

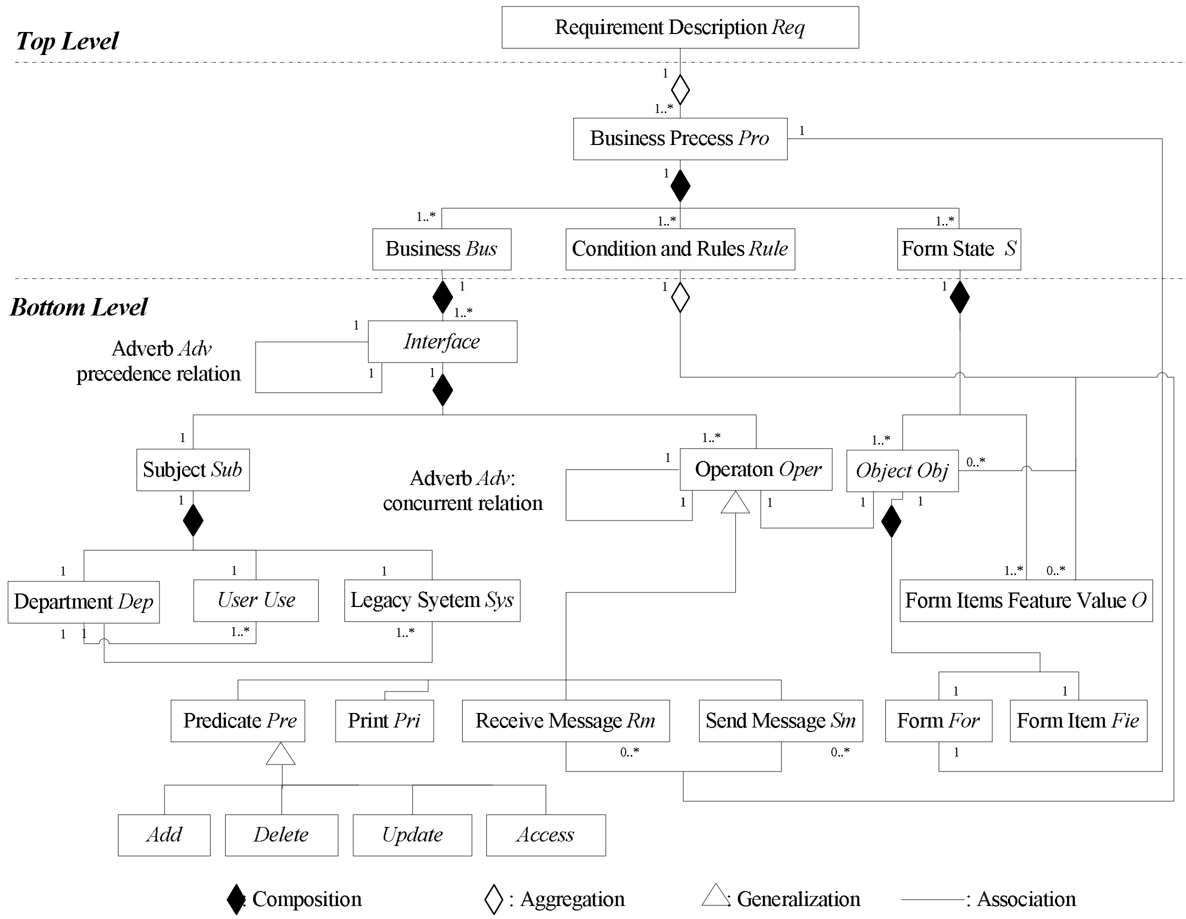

Figure 1. *Subject Predicate Object Adverbial complex business process description Meta-model* M_{spoa}

II. Business Process Analysis

In order to analyze related factors in business process migration formally, assuming that:

- Problem space ε: Department Dep, User Use, Legacy Sys, Form For, Form item Fie, Operation $Oper$, Business process Flo, Condition and rules Rul, Requirement description Req, etc.;
- Solution space ρ: Data relation Da, Role Rol, Operation process Pro, System configuration Con, Interface Int, etc.;
- Business process specification S.

Subsequently, we can describe the task of business process analysis is:

$$\rho \leftarrow \varepsilon, S \tag{1}$$

and S is described on the basis of business process problem space:

$$S \leftarrow Dep, Use, Sys, For, Fie, Oper, Flo, Rul, \ldots, Req \tag{2}$$

Business process designers develop system based on S:

$$Da, Rol, Pro, Con, \ldots, Int \leftarrow S \tag{3}$$

The features of business process analysis can be characterized as following:

- The complex of business process analysis is that there so many factors in problem space ε which should be described clearly. For example, Operation $Oper$ is processed by User Use, Business process Flo is related with the operation on Form For and Form items Fie, Condition and rules Rul can be regarded as the constraint of Flo, User Requirement description Req is related with the Department Dep, Business process Flo or legacy Sys;
- We can describe the mapping relationship about the factors between problem space ε and solution space ρ, such as: $Use \times Oper \times For \rightarrow Rol$, $Dep \times Req \rightarrow Con$, $For \times Fie \rightarrow Da$, $Flo \rightarrow Pro$, however, it is difficult for us to describe this relationship totally.

III. M_{spoa} and $CBPM$ Framework

A. M_{spoa} framework

In order to describe the static relationship in business process problem space clearly, M_{spoa} (*Subject Predicate Ob-*

ject Adverbial complex business process description Metamodel) is proposed in this section, shown in Figure 1.

Without regarding the dynamic behavior relationship, M_{spoa} can describe problem space with hierarchical and multi-view points.

(1) Sub, Pre, and Obj are presented in the bottom Level of M_{spoa}:

Sub describes the affiliation relationship between Dep, Use, and Sys:

$$\begin{aligned} &Sub = (Dep \times Use) \cup (Dep \times Sys), \\ &Use \rightarrow Sub, \\ &Sys \rightarrow Sub. \end{aligned} \tag{4}$$

Obj describes the affiliation relationship between For and Fie:

$$\begin{aligned} &Obj = (For \times Fie), \\ &Fie \rightarrow Obj. \end{aligned} \tag{5}$$

Operations $Oper$ are generalized as four types, including predicate Pre, receive message Rm, send message Sm, and print Pri. Pre is the atom action $Action$, including: add, $delete$, $update$, and $access$:

$$\begin{aligned} &Pre = (Action \times Obj), \\ &Action = \{add, delete, update, access\} \end{aligned} \tag{6}$$

$Interface$ is the basic unit in the bottom of M_{spoa}, which means there is a Subject Sub performs many operations $Oper$. One predicate Pre just performs only one atom action on one Obj. Adv means the precedence relationship between interfaces and concurrent relationship between many actions in one interface. Form Flo is the basic unit of business process Pro, including: business Bus, business state S, and condition or rule $Rule$. Form business Bus is composed of some interfaces $Interface$ with precedence relationship. Form state is described based on object Obj and its characteristic value of the form item. Receive message Rm is regarded as triggering event. The operation condition and rules $Rule$ of business process is described by object's form item characteristic value used as servo conditions.

The top level of M_{spoa} describes user Requirement description Req by presenting business process Flo and their communication, which is composed of receive message Rm, send message Sm and message data.

B. $CBPM$ Framework

As M_{spoa} describing the static model of problem space, in order to analyze dynamical behavior (such as the form business in business process, the triggering and conversion relationship between form state, as well as multi-process communication relationship, $CBPM$ (*Complex Business Process Model*) is presented in this section, which is composed of business process layer, business layer, and Multi-process Interactive Layer (shown in Figure 2).

(1) Business process layer

Business process layer describes the effects of form businesses on form states cored with form life cycle based on extended finite state machine. (Obj) is used as business process basic unit and defined as transition system, in which nodes is corresponding to form states, arcs is labeled by form businesses, triggered event and servo condition.

(2) Business layer

Business layer consists of user interfaces as form business ingredient, in which independent interface indicates which user (sub) processes what operations, (including: basic operation (pre) with atom action on field (obj), receiving and sending message, and printing operation). adv is used to describe the relationship between interfaces, denoting not only sequence relationship between interfaces but also synchronous operations in single interface.

(3) Multi-process interactive layer

Multi-process interactive layer describes global processes based on interactive messages between form businesses processes, which are classified as chained processes, synchronous processes, asynchronous processes, main process with subprocess, and nested processes, etc.

IV. $CBPM$ Description Language $CBPM_o$

Based on action language L_o [10], $CBPM$ Description Language $CBPM_o$ is proposed in this section, which can describe business process based on M_{spoa} and $CBPM$.

A. $CBPM_o$ Grammar

- DEPT: department type, DEPT={Dept | Dept $\in$ department set};
- ROLE: role type, ROLE ={Role | Role $\in$ business or process related role set};
- ACTION: atom action type, ACTION={Action | Action $\in$ interface related action set }, including: {add, delete, update, and access};
- FORM: form type, FORM={Form | Form $\in$ business related form set};
- FIELD: form item type, FIELD={Field | Field $\in$ form item in form};
- MT: process communication message type, MT = {mt | mt $\in$ process communication message type}, including: {syn(synchronous), asyn(asynchronous), oneway, and query}.

Definition 1. *Predicate set.*

(1) *Subject predicate set* $SUB = \{role(Role), dept(Dept), sub(Role, Dept), lev(Dept_i, Dept_j), Dept, Dept_i, Dept_j \in DEPT;\ Role \in ROLE;\ i \neq j, i, j \in N\}$,
- $sub(Role, Dept)$*: role Role belonging to department Dept,*
- $lev(Dept_i, Dept_j)$*: department* $Dept_i$ *is the lower level than department* $Dept_j$*;*

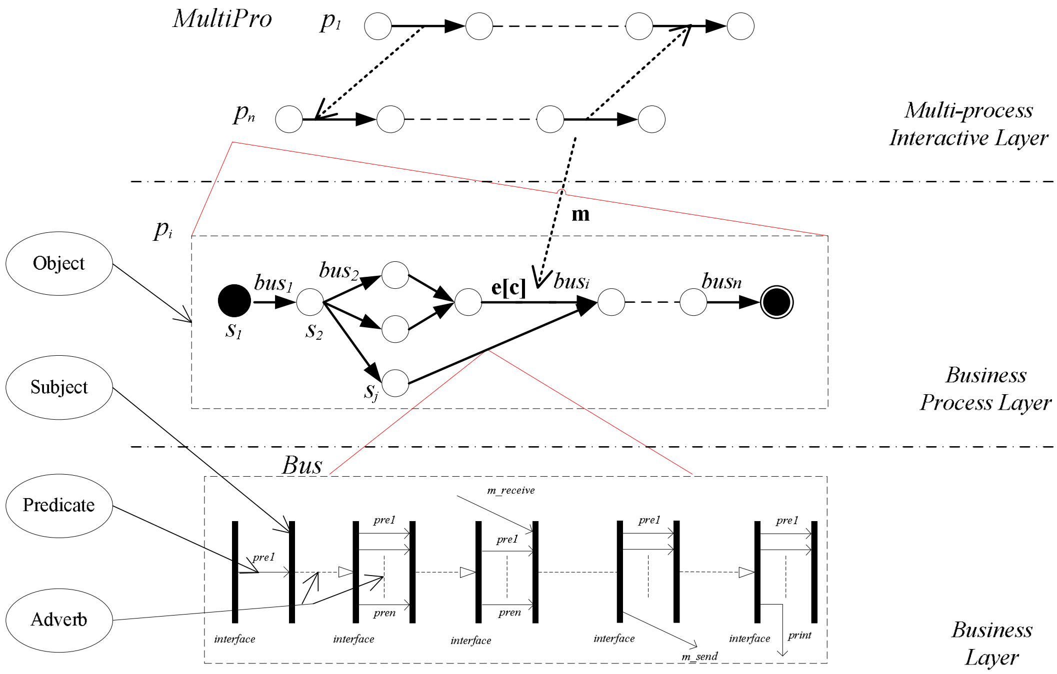

Figure 2. *Complex Business Process Model $CBPM$ framwork*

(2) *Object predicate set* $OBJ = \{form(Form), field(Field), obj(Field, Form), Field \in FIELD;\ Form \in FORM\}$,
  - $form(Form)$: *form* $Form$,
  - $field(Field)$: *form item* $Field$,
  - $obj(Field, Form)$: *form item* $Field$ *belonging to form* $Form$;

(3) *Form predicate set* $FOR = \{bus(Form, I), s(Form, J),\ Form \in FORM;\ I, J \in N\}$,
  - $bus(Form, I)$: *belonging to form* $Form$ *the* I*-th form business* ,
  - $s(Form, J)$: *belonging to form* $Form$ *the* J*-th form state.*

Definition 2. ***Subject sub.***
Action's operator, finite set , regarding sub set as Sub;

(1) *sub is a pair :* $< Role, Dept >$, *and* $Role \in ROLE$, $Dept \in DEPT$, $Sub = ROLE \cup DEPT$, $ROLE \cap DEPT = \phi$;
(2) *define hierarchy relationship* $\preccurlyeq$ *in* $DEPT$, *which is partially ordering relation, If* $Dept_1 \preccurlyeq Dept_2$, *then* $Dept_1$ *is the lower department of* $Dept_2$;
(3) *any role Role must belong to a department Dept, which is reflexive, antisymmetric, and non-transitive relation, regarding as* $Role \preccurlyeq_b Dept$;
(4) *role* $Role$ *is belonging to one department, i.e.* $\neg((Role_i \preccurlyeq_b Dept_j) \wedge (Role_i \preccurlyeq_b Dept_k)), Role_i \in ROLE; Dept_j, Dept_k \in DEPT; j \neq k$.

Definition 3. ***Object obj.***
The object of action, finite set , regarding obj set as Obj;

(1) *obj is a pair :* $< Field, Form >$, $Field \in FIELD$, $Form \in FORM$, $Obj = FIELD \cup FORM$, $FIELD \cap FORM = \phi$;
(2) *any form item* $Field$ *must belong to a form* $Form$, *which is reflexive, antisymmetric, and non-transitive relation, regarding as* $Field \preccurlyeq_b Form$;
(3) *form item* $Field$ *belongs to one form* $Form$, *i.e.* $\neg((Field_i \preccurlyeq_b Form_j) \wedge (Field_i \preccurlyeq_b Form_k))$, $Field_i \in FIELD; Form_j, Form_k \in FORM; j \neq k$.

Definition 4. ***Form business bus.***
The basic unit of form business process, composed of some interfaces; regarding the form business set of the same form $form$ *as* Bus_{form};

(1) *bus is a pair :* $< Form, I >$;
(2) *any form business bus must belong to a form* $Form$, *which is reflexive, antisymmetric, and non-transitive relation , regarded as* $bus \preccurlyeq_b Form$;
(3) *define precedence relation* $\preceq_s$ *in bus, if* $bus_i \preceq_s bus_j$, $i \neq j$, bus_i *is processed before* bus_j.

Definition 5. ***Interface***.
The basic unit of form business, finite set , regarding $interface$ *set as Int;*

(1) $interface$ *is a triple :* $< Role, Bus_{form}, K >, K \in N$;
(2) *any interface* $interface$ *must belong to one form business* bus_{form}, *which is reflexive, antisymmetric,*

and non-transitive relation , regarded as $interface \preccurlyeq_b bus_{form}$;

(3) one interface $interface$ belongs to one form business bus_{form}, i.e. $\neg((interface \preccurlyeq_b bus_{form}) \wedge (interface \preccurlyeq_b bus_{form'})), form \neq form'$;

(4) interfaces belonging to the same form business bus_{form} is precedence relation $\preceq_s$, if $interface_i \preceq_s interface_j, i \neq j$, then $interface_i$ is processed before $interface_j$.

Definition 6. ***Message m.***
Communication type between processes, finite set;

(1) $< Interface_{k1}, Interface_{k2}, Mt, Mdata_{k1} >$;

(2) means interface $interface_{k1}$ send message mt with $mdata_{k1}$ to interface $interface_{k2}$;

(3) Mt message m type, $Mt \in MT$;

(4) if message has no data, then $mdata_{k1}$=0; if message transfers data, then its message data can be regarded as $mdata_{k1}$, the data of message is composed of the form item of the send message interface;

(5) any message m must belong to one interface $interface$, which is reflexive, antisymmetric, and non-transitive relation , regarded as $m \preccurlyeq_b interface$.

Definition 7. ***Predicate pre.***
Atom action of object form item, finite set, regarding the predicate set belonging to the same interface $interface$ as $Pre_{interface}$:

(1) pre is a triple : $< Interface, Action, Obj >$;

(2) $Action$ is composed of 4 atom actions, i.e. $Action = \{add, delete, update, access\}$, assuming form item is F_i, the data before action is $F_i(\alpha)$, the data after action is $F_i(\beta)$:
   - add: $iff\ (F_i(\alpha) = \phi) \wedge (F_i(\beta) \neq \phi)$;
   - delete: $iff\ (F_i(\alpha) \neq \phi) \wedge (F_i(\beta) = \phi)$;
   - update: $iff\ (F_i(\alpha) \neq \phi) \wedge (F_i(\beta) \neq \phi) \wedge (F_i(\alpha) \neq F_i(\beta))$;
   - access: $iff\ (F_i(\alpha) \neq \phi) \wedge (F_i(\beta) \neq \phi) \wedge (F_i(\alpha) = F_i(\beta))$;

(3) any predicate pre belongs to one interface $interface$, which is reflexive, antisymmetric, and non-transitive relation , regarded as $pre \preccurlyeq_b interface$;

(4) the predicates in the same interface is concurrency relationship, assuming $pre_1, \ldots, pre_n \in Pre_{interface}$, i.e. $pre_1 \wedge, \ldots, \wedge pre_n$.

Definition 8. ***Print.***
Print action in one interface, finite set, regarding the $print$ set belonging to the same interface $interface$ as $Print_{interface}$:

(1) $print$ is a pair : $< Interface, Pdata >$;

(2) $print$ is described as the action print for data $pdata$ in interface $interface$;

(3) any print $print$ must belong to one interface $interface$, which is reflexive, antisymmetric, and non-transitive relation , regarded as $print \preccurlyeq_b interface$;

(4) $pdata$ is the data of print, composed of the form item.

Definition 9. ***Form state s.***
The basic unit of form business process;

(1) s is pair : $< Form, J >$;

(2) $s \in S_{Form}, S_{Form}$ is finite set, related with the life cycle of form $Form$;

(3) any form state s must belong to one form $Form$, which is reflexive, antisymmetric, and non-transitive relation , regarded as $s \preccurlyeq_b Form$;

(4) define precedence relation $\preceq_s$ in S, $s_i \preceq_s s_j$, then s_i is the s_j last form state , s_j is s_i next form state ; If $\nexists s_k, s_k \preceq_s s_\nabla$, then regard s_∇ as initial form state ; If $\nexists s_l, s_\triangle \preceq_s s_l$, then regard $s_\triangle$ as ending form state .

Definition 10. ***Triggering event e.***
Trigger form business bus happening condition, regarding e set as $Event$;

(1) e is $< Bus >$;

(2) the happening of triggering event e can not lead to business bus directly;

(3) any triggering event e must belong to one form business bus, which is reflexive, antisymmetric, and non-transitive relation , regarded as $e \preccurlyeq_b bus$;

(4) one form business bus only has one triggering event e, i.e. $\neg((e_i \preccurlyeq_b bus) \wedge (e_j \preccurlyeq_b bus)), e_i, e_j \in Event; bus \in Bus; i \neq j$.

Definition 11. ***Servo conditions c.***
The condition of form business bus happening, regarding c set as $Cond$;

(1) c is $< Bus >$;

(2) composed of form business bus receive message type, message data and related form item characteristic value;

(3) any servo conditions c must belong to one form business bus, which is reflexive, antisymmetric, and non-transitive relation , regarded as$c \preccurlyeq_b bus$;

(4) one form business bus has only one servo conditions c, i.e. $\neg((c_i \preccurlyeq_b bus) \wedge (c_j \preccurlyeq_b bus)), c_i, c_j \in Cond; bus \in Bus; i \neq j$.

B. $CBPM_o$ semantics

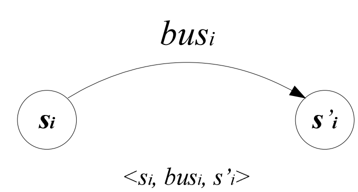

Figure 3. *Conversion relationship of Buss and States*

Based on the definitions above, action on form business will lead to the conversion of form state; regarding form state

$s_i, s'_i \in S$ as node, form business $bus_i \in Bus$ as directed arc, we can build directed graph; the conversion of form state s_i, s'_i can be described as $t \in T$, i.e. $t =< s_i, bus_i, s'_i >$. This is the basic conversion relationship (shown in Figure 3).

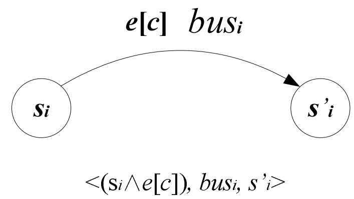

Figure 4. *Extended conversion relationship of Buss and States*

Furthermore, extended conversion relationship of Buss and States (shown in Figure 4) describe form state conversion $s_i, s'_i \in S$ which should satisfy triggering event e, as well as servo conditions c, form business $bus_i \in Bus$ activate; $e[c]$ and bus_i build directed arc, the conversion of form state s_i, s'_i can be presented as $t \in T$, i.e. $t =< (s_i \wedge e[c]), bus_i, s'_i >$, regarded as $\hat{s}_i = s_i \wedge e[c]$, then $t =< \hat{s}_i, bus_i, s'_i >$.

Because form can only stay in only one form state, we can get the theorem shown as following:

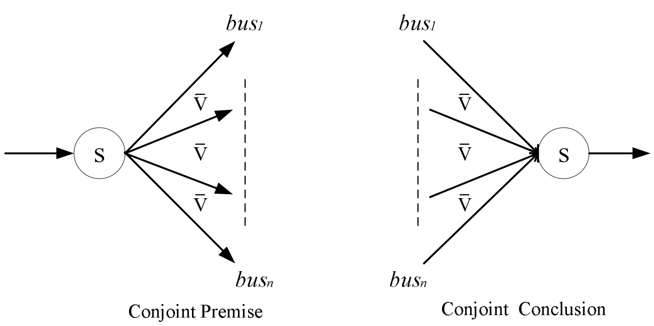

Figure 5. *Conjoint premise or conclusion relationship*

Theorem 1. *Assuming* $t_{f1}, \ldots, t_{fn} \in T_{form}$*, if* $t_{f1}, \ldots, t_{fn}$ *have conjoint premise or conclusion relationship, then* $t_{f1}, \ldots, t_{fn}$ *have exclusive or relationship.*

Typical interfaces with predicate, receive message, send message and print are shown in Figure 6:

Based on definition 5, there is just one subject activate action; assuming the predicate set of this interface as $Pre = \{pre_1, \ldots, pre_i\}$, receive message set $M\_receive = \{m_1, \ldots, m_j\}$, send set message $M\_send = \{m'_1, \ldots, m'_k\}$, print set $Print = \{print_1, \ldots, print_l\}$, then for the same interface, the relationship between Pre, M_send, $M\_receive$, and $Print$ is concurrent relationship, which is shown as following.

$$interface \leftarrow Pre \wedge M\_receive \wedge M\_send \wedge Print \quad (7)$$

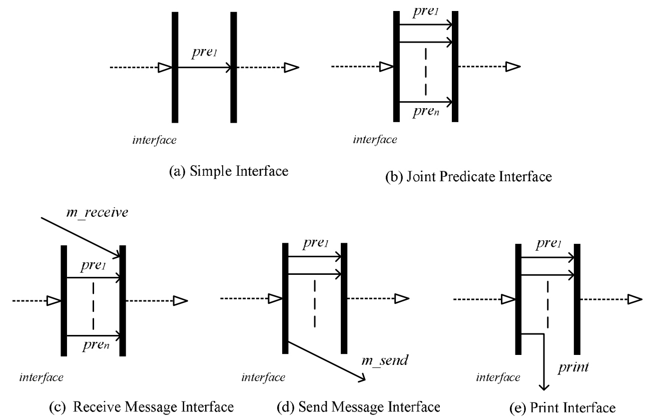

Figure 6. Typical interfaces

By the means of M_{spoa} and $CBPM$, we can describe typical business processes, as shown in Figure 7:

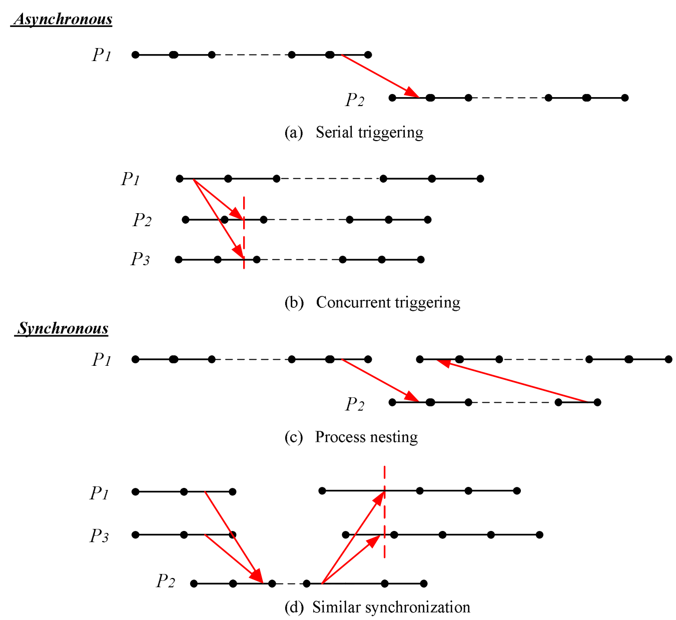

Figure 7. Typical business processes

(1) Asynchronous relationship:

- Serial triggering : after finishing process p_1, trigger process p_2(7(a));
- Concurrent triggering : process p_1 and p_2 activate concurrently, but do not stop simultaneously(7(b));

(2) Synchronous relationship:

- Process nesting: process p_1 triggers p_2, after p_2 finishing, p_1 continues process(7(c));
- Similar synchronization: after process p_1 and p_3 synchronously activating at the business of p_2, they continue their own process individually (7(d)).

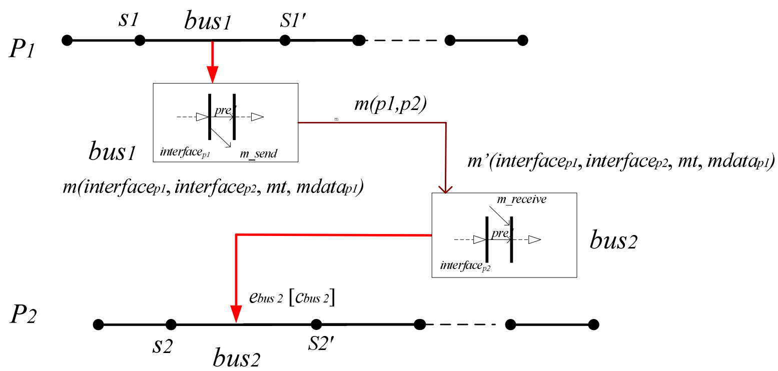

Figure 8. Asynchronous business processes

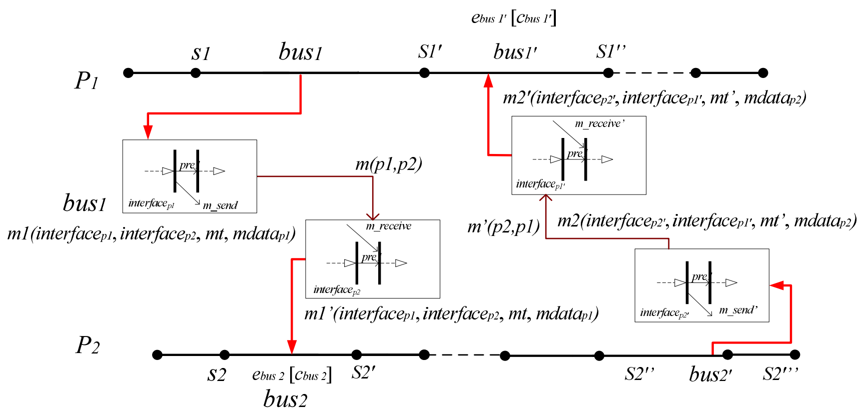

Figure 9. Synchronous business processes

Theorem 2. *Assuming process p_1 and process p_2, satisfying:*

- *as for process p_1, $\exists s_1, s_1', bus_1$, satisfying $do(s_1, bus_1) \to s_1'$;*
- *as for process p_2, $\exists s_2, s_2', bus_2, e_{bus_2}[c_{bus_2}]$, satisfying $do((s_2 \wedge e_{bus_2}[c_{bus_2}]), bus_2) \to s_2'$;*
- *as for business bus_1, $\exists$ $(interface_{p1} \wedge m(interface_{p1}, interface_{p2}, mt, mdata_{p1}))$, in business bus_2, $\exists$ $(interface_{p2} \wedge m'(interface_{p1}, interface_{p2}, mt, mdata_{p1}))$, and $mt = \{syn, asyn, oneway, query\}$, because message m and m' are sent simultaneously with same message data, regarded as $m(p1, p2)$;*
- *as for the triggering event e_{bus_2} of business bus_2, satisfying $oms(m(p1, p2)) \to e_{bus_2}$; the servo conditions of business bus_2 c_{bus_2}, satisfying $oms(m(p1, p2)) \to c_{bus_2}$;*

then process p_1 and process p_2 can be processed in bus_1, bus_2 by the means of $m(p1, p2)$.

Figure 8 shows that we can describe asynchronous business processes with messages sent from one interface in one process $p1$ to another process $p2$. Message $m(p1, p2)$ is sent by business bus_1, satisfying $bus_1 \to m(p1, p2)$, which means if business process bus_1 is finished, message $m(p1, p2)$ can be processed successfully. On the other hand, $m(p1, p2)$ is the triggering event of business bus_2, satisfying $oms(m(p1, p2)) \to c_{bus_2}$ which means if business bus_2 receives message $m(p1, p2)$, then the servo condition c_{bus_2} is valid. Subsequently, when form business bus_1 is processed successfully, because of $m(p1, p2) \wedge bus_2 \wedge e_{bus_2}[c_{bus_2}]$, and $do((s_2 \wedge e_{bus_2}[c_{bus_2}]), bus_2) \to s_2'$, we can come to an conclusion $bus_1 \wedge bus_2 \wedge m(p1, p2)$

Similarly, we can describe synchronous business processes (shown in Figure 9) with two messages communicated between process $p1$ and another process $p2$. The business and message in figure 9 satisfies: $bus_1 \wedge bus_2 \wedge m(p1, p2)$, and $bus_2' \wedge bus_1' \wedge m'(p2, p1)$. Based on $CBPM_o$ semantics, i.e. $do(s_1, bus_1) \to s_1'$, $do((s_1' \wedge e_{bus_1'}[c_{bus_1'}]), bus_1') \to s_1''$, and $bus_1 \preceq_s bus_1'$; therefore, we can conclude that $(bus_1 \wedge bus_2 \wedge m(p1, p2)) \wedge (bus_2' \wedge bus_1' \wedge m'(p2, p1)) \wedge (bus_1 \preceq_s bus_1')$.

V. Migration Steps

Main steps of migrating complex business process to the cloud are the observation of current system and analysis of tasks, with the intention of developing, eliciting and revising user demands. In logic, this procedure is the process of eliciting and describing $Subject$, $Predicate$, $Object$ and $Adverbial$, where $Subject$ stands for the users of a software system, $Predicate$ stands for the operations available for the users to activate, and $Object$ is the object manipulated by the $Subject$. In this way, the main jobs in migration steps are: make it clear what the users ($Subject$) are and their relationships, what functions ($Predicate$) they have, what data ($Object$) these functions produce or what influence is put on data after executing, mainly consisting of the following ten steps:

(1) Obtain the user information, interpret the $Subjects$ and describe their relationships in $Organization\ Structure\ Diagram$ (OSD);

(2) Start from $Forms$, use $Form\ Flow\ Diagram$ (FFD) to depict the flow direction of each concerned Form and the key operations on each, then illustrate the relation among the Forms in a table named $Form\ Rules\ Interpreter$(FRI);

(3) Based on FFD, we use layered $Role\ Function\ Diagram$ (RFD) to describe the functional requirement;

(4) By integrating and analyzing FFD and RFD, we get $Business\ Flow\ Diagram$ (BFD) with swim lanes from different point of view to show the business procedure of every key function;

(5) According to FFD and BFD, fill in the $Subject\ Predicate\ Object\ Table$ (SPOT), and design the $Data\ Origin\ Diagram$ (DOD);

(6) Recursively interpret the domain vocabulary, namely the $Object$;

(7) Draw $Activity\ Flow\ Diagram$ (AFD) to decompose those non-atomic and key activities in the BFD of step 4;

(8) Recursively interprets the $Predicate$ gathered by step 4 and 7;

(9) Mark the dynamic changing states of the $Objects$ by $Statechart\ Diagram$ (SD);

(10) Indicate the functional and data constraints of different users by $Operational\ Right\ Diagram$ (ORD) and $Data\ Right\ Diagram$ (DRD).

Migration steps really integrate as a whole by smooth and small steps from the beginning through to the end. Note that these ten steps are not fixed but extensible instead. Nor are they executed straight forth, on the contrary, they form an iterative, refined step-by-step procedure.

VI. Conclusion

In order to migrate complex business processes to cloud platform, with which individual application components or legacy systems can be deployed and applied accurately in the cloud, M_{spoa} (*Subject Predicate Object Adverbial complex business process description Meta-model*), and $CBPM$ (*Complex Business Process Model*) are proposed in this paper, which can describe static and dynamic behavior relationship in problem space individually. Based on action language L_o, $CBPM$ Description Language $CBPM_o$ is presented in detailed, which can describe synchronous or asynchronous business processes accurately based on M_{spoa} and $CBPM$. With the intention of developing, eliciting and revising user demands during migrating complex business process to the cloud, this paper describes migration steps with the help of some kinds of forms or tables. There are several issues to be addressed: M_{spoa} and $CBPM$ are to be formalized and followed by the reasoning and validation of business processes to some extent.

Acknowledgment

This work is Supported by the National Natural Science Foundation of China under Grant Nos.60573160,60873162; the Natural Science Foundation of Guangdong Province under Serial No.10151027501000028; the Research Foundation of Science and Technology Plan Project in Guangzhou City under Grant No.2010Z1-D00041; the Fundamental Research Funds for the Central Universities under Grant No.10000-3165002.

Many thanks to our colleagues and the same thanks to the anonymous reviewers for their helpful suggestions that improved the quality of this paper.

References

[1] M.Creeger. CTO roundtable: cloud computing. In *Communications of the ACM*, volume 52, 50–56, 2009.

[2] M.Hajjat, X.Sun, and Y.Wei et.al. Cloudward bound: Planning for beneficial migration of enterprise applications to the cloud. In *ACM SIGCOMM 2010*, 243–254, 2010.

[3] M.Armbrust, A.Fox, and R.Griffith et.al. Above the clouds: A berkeley view of cloud computing. Technical Report UCB/EECS-2009-28, EECS Department University of California, Berkeley, Feb 2009.

[4] L.Mansar, S.Reijers, and Hajo A. et.al. Development of a decision-making strategy to improve the efficiency of BPR. *Expert Syst. Appl.*, 36:3248–3262, March 2009.

[5] M.Nuno and P.Michael. A conceptual framework for understanding business processes and business process modelling. *Information Systems Journal*, 10(2):105–129, 2000.

[6] A.Gemino and Y.Wand. A framework for empirical evaluation of conceptual modeling techniques. *Requirements Engineering,Springer-Verlag*, 248–260, Sep. 2004.

[7] Wand Y. and Weber R. On the deep structure of information systems. *Information System*, 203–223, May 1995.

[8] R.Studer, V.R.Benjamins, and D.Fensel. Knowledge engineering: Principles and methods. *Data & Knowledge Engineering*, 25(1-2):161–197, 1998.

[9] S.W. Clyde, D. E. Embley, and S.N. Woodfield. Turnable formalism in object-oriented systems analysis: meeting the needs of both theoreticians and practitioners. *SIGPLAN Not.*, 27:452–465, October 1992.

[10] E.Giunchiglia and V.Lifschitz. An action language based on causal explanation: preliminary report. In *Proceedings of the 15th National Conference on Artificial Intelligence (AAAI–98), AAAI Press, Menlo Park, CA.*, 1998.