Formalizing Interactive Institution with RBA Calculus

Guoyong Cai[1] Ji Gao[1] Junyan Qian[2] Lingzhong Zhao[2]

[1]College of Computer Science and Technology, Zhejiang University, Hangzhou 310027，P. R. China

[2]College of Computer Science and Technology, Guilin University of Technology, Guilin 541004，P.R. China

Abstract

Although organization oriented paradigm is a promising way to design dynamic interactive computational systems in open and heterogeneous networked environment, there is still lack of a suitable formal approach to support this paradigm. RBA, a boxed ambient calculus extended with separated regulating mechanism, is proposed to design such systems. In RBA, an ambient is an encapsulating mobile unit parameterized with role-governors that regulating actions of the corresponding role-players participating into this ambient. Firstly the formal syntax and semantics of RBA are given. Secondly an electronic institution model is applied to illustrate the RBA approach. Based on the bisimulation theory of process calculi, verification and validation of design specifications are supported with RBA approach.

Keywords: Interactive computation, Ambient calculus, Electronic institution, Behavior regulation

1. Introduction

Globally interactive and collaborative computational systems are presently one important type of information-based systems, which forms different kind of virtual organizations penetrating to people's everyday lives, such as electronic community, electronic commerce systems. To develop such kind of systems efficiently and correctly, different methodologies and frameworks have been proposed to cope with the difficulties while designing such systems; among them agent-oriented methodology and organization-based frameworks are becoming the most promising ways. Agent-oriented paradigm [1] provides good structural methods to identify the main building blocks of such systems exploiting agents, roles and their relations in organizational settings, such as electronic institution or some policy-based frameworks, which provide good starting points toward understanding development of such systems. However currently these techniques haven't provided adequately support for building this type of systems in a formal way that helps the verification and validation at suitable abstraction level. Especially when concerning the design phase of such systems, the main complexity arises from the complex interactive and regulating patterns among agents and their regulators [2]-[3]. Presently these agent-oriented or organization-oriented paradigms do not provide effective means to deal with these aspects adequately [4]-[6]. Therefore there is a great need to enhance these paradigms by providing an appropriate design level specification language to encapsulate regulative and interactive patterns in suitable abstraction and compositional modes.

On the other hand, due to its compositional and formal features, process calculi has been widely adopted for the design specification of concurrent systems, especially those with communicating, concurrently executing software components. However, most of the efforts are oriented to study process algebras suitable for homologous and closed environment, such as CCS [7] or the π-calculus [8], although another type of process calculi, such as ambient calculus [9]-[10], provide some encapsulation mechanism to process movement in networked environment, it is still not adequate to deal with the complex interactive and regulative patterns appear in virtual organization in open and heterogeneous networked environment. Further more traditional process calculi study does not incorporate itself with well formed concepts originating from organizational theory, such as role, agent, and policy.

To achieve better practice in organization oriented collaborative system development, we argue that both organization oriented and process oriented structural mechanisms are necessary, process calculi should be extended with organization-based regulating mechanisms. In this paper, RBA (Regulating Boxed Ambient), an extended boxed ambient calculus with regulating mechanism under organizational framework is proposed. Electronic institution model are applied to show the usage of the proposed calculus language.

The paper is organized as follows. In Section 2 the feature of ambient calculus is briefly introduced, and then the syntax and semantics of RBA is given. In section 3, an electronic institution model is introduced

which will serve as the meta regulating framework towards open and heterogeneous virtual environment for interactions and regulations. In section 4, RBA is applied to design regulated interactive computational system under electronic institutional framework. Some conclusion is drawn in the final section.

2. Ambient calculus with regulating mechanism

2.1. Background and motivation

Ambient Calculus (AC) [10] is a process algebra that focuses on the notions of locations, mobility and authorizations. The underlying model of the AC is based on the notion of ambient. An ambient is a place limited by a boundary where computations or interactions take place. They are hierarchically structured and the evolving path towards a destination is not abstracted away. Processes are confined to ambients and ambients move under the control of processes. Terms in Ambient based calculi describe configurations of locations and sub-locations, and interactive computation happens as a consequence of movement of locations. The three primitives for movement allow: an ambient to enter another ambient (In-movement), an ambient to exit another ambient (Out-movement), a process to dissolve an ambient boundary thus obtaining access to its content (Open-movement). Generally an ambient is denoted by *n[P]*, which means process *P* runs in ambient *n*. The basic capabilities in AC are defined as *M::= n| x | in M| out M| open M| ε| M.M'*. The process is defined as *P::=0|(x).P|<M>| (νn)P| n[P] | !P | P|Q | M.P | P+Q | X(v)*

Different variants of AC have been proposed [10]. Boxed ambient is a variant of the AC that drops the open capability and introduce fine-grain mechanisms for ambient interactions, such as non-local communications between one ambient and its parent/child ambients. These changes provide better support for the specification of management policies. Type systems are designed to enforce secure ambient interactions[11]. However type systems are static and inflexible to deal with complex regulative patterns appearing in open organizations.

Instead of depending on type systems, in this paper we propose to enhance the boxed ambient with parameterized regulating capabilities. That is every ambient is equipped with a number of governors that supervise the activities of the corresponding components (sub-ambients) running inside the ambient. Hence the ambient notation *n[P]*(resp. *n[a[P]]* etc.) is extended to *n(G)[P]*(resp. *n(G)[a[P]]*) which denotes that process *P* (resp. ambient *a[P]*) runs in ambient *n* and is regulated by *G*. If there is no need to have such governors for an ambient, it is also denoted by original notation *n[P]*. Therefore *n[P]* is taken as a special form of *n(G)[P]*. We call *n(G)[P]* supervised ambient and *n[P]* non-supervised ambient. Based on this idea originated from organizational supervising theory, in the following section, we propose RBA calculus--an enhanced ambient calculus.

2.2. An extension of boxed ambient

We assume a countable set of names, N, ranged over by *m, n,…*; *u, v,w,…*; *x, y, z,…* and their decorated versions (m',...) and vector versions ($\overline{v}, \overline{w}$,…). To simplify reading, we shall use *m, n,…* to denote ambient names, *x, y, z,…*to denote input variables, *u, v,w,...* to denote generic names, and W, X, ….to denote the identifier of defined processes or governors. RBA is presented from five aspects: syntax, structural equivalence, process reductions, governor reductions and configuration reductions.

Definition 2.1. (RBA processes, actions and conditions)

$P::=0 \mid (x)^{\eta}.P \mid \langle M\rangle^{\eta}.P \mid M.P \mid u[P] \mid P|P \mid (\nu n)P \mid !P \mid X(\overline{v})$

$M::=u \mid in\_u \mid out\_u \mid M.M \mid new(n,X) \mid spawn(P) \mid box(n,P)$

$\eta::= \mid u$

This definition is an enhance version of classical ambient calculus. It adopts a shared channel way to neighboring communications among parent ambient and child ambient via input $(x)^{\eta}.P$ and output $\langle M\rangle^{\eta}.P$. It also adds new capabilities (*new*, *spawn* and *box*) for dynamic generation of name mapping, process and ambient. *new(n,X)* is used to create a fresh association between *n* and *X*. *spawn*$(P_1).P_2$ activate P_1 in parallel with P_2, thus it corresponds to $P_1|P_2$. *box(n,P)* is used to create a new ambient, i.e. *n[P]*.The other notations are the same as in original ambient calculus.

Definition 2.2. (RBA structural equivalence) The structural equivalence relation, ≡, is the least equivalence closed by parallel composition, restriction and ambient encapsulation, including alpha-conversion and satisfying the following axioms.

$P|0\equiv P \quad P_1|P_2\equiv P_2|P_1$

$P_1|(P_2|P_3) \equiv (P_1|P_2)|P_3 \quad !P\equiv P|!P$

$(\nu n)0\equiv 0 \quad (\nu n)\nu m)P\equiv(\nu m)(\nu n)P$

$P_1|(\nu n)P_2\equiv(\nu n)(P_1|P_2)$ if $n\notin fn(P_1)$

$m[(\nu n)P] \equiv(\nu n)m[P]$ *if* $n\neq m \quad (M.M').P\equiv M.(M'.P)$

Definition 2.3. (RBA process reductions) The non-supervised process reduction relation, →, is the least relation satisfying the following axiom and rules:

(1) $new\ \dfrac{}{new(?x).P \xrightarrow{new(n)} P[x := n]}$

(2) $spawn\ \dfrac{}{spawn(P_1).P_2 \xrightarrow{spawn(P_1)} P_2}$

(3) $box\ \dfrac{}{box(n,P_1).P_2 \xrightarrow{box(n_1,P_1)} P_2}$

(4) $rec\ \dfrac{P[\bar{x} := \bar{v}] \xrightarrow{a(\bar{v})} P'}{X(\bar{v}) \xrightarrow{a(\bar{v})} P'}$ if $X(\bar{x}) \triangleq P$

(5) $in\ \dfrac{}{in(n).P \xrightarrow{in(n)} P}$ (6) $out\ \dfrac{}{out(n).P \xrightarrow{out(n)} P}$

(7) $wr\ \dfrac{}{<\bar{v}>.P \xrightarrow{wr(\bar{v})} P}$

(8) $rd\ \dfrac{}{(\bar{x}).P \xrightarrow{rd(\bar{v})} P[\bar{x} := \bar{v}]}$

(9) $rwr\ \dfrac{}{<\bar{v}>^n.P \xrightarrow{<\bar{v}>^n} P}$

(10) $rrd\ \dfrac{}{(\bar{x})^n.P \xrightarrow{(\bar{v})^n} P[\bar{x} := \bar{v}]}$ (11) $\dfrac{P \to P'}{n[P] \to n[P']}$

(12) $\dfrac{P_1 \to P_1'}{P_1 \mid P_2 \to P_1' \mid P_2}$ (13) $\dfrac{P \to P'}{(vn)P \to (vn)P'}$

For regulating the process actions or interactions, we propose a separated special process, called governor process, which regulate (permit or reject) the actions of the regulated processes. A governor G perform reduction $G \xrightarrow{p(v)} G'$, which denotes G currently permit $p(v)$ occur in the regulated process, and if $p(v)$ requests actually in the regulated process, then G transfers to G'. The syntax of governor processes is defined as follows.

Definition 2.4. (RBA governors and permissions)

$$G \in \mathsf{Gov} ::= \text{when } b\ \ p(\bar{x})\ \text{ then } G \mid \text{when } b\ \ G \setminus p(\bar{x}) \mid \bot \mid \top \mid G_1 \wedge G_2 \mid G_1 \vee G_2 \mid W(\bar{v})$$

$$p \in \mathsf{Perm} ::= \overline{new} \mid \overline{spawn} \mid \overline{in} \mid \overline{out} \mid \overline{lcom} \mid \overline{wrdn} \mid \overline{rddn}$$

$$b \in \mathsf{Bool} ::= \bot \mid \top \mid v_1 = v_2 \mid b_1 \vee b_2 \mid b_1 \wedge b_2 \mid \neg b$$

The basic construct for governors is *when* b $p(x)$ *then* G, where p is the permitted action, x is a sequence of variables bound by the construct, b is a boolean expression, and G is a governor process. It means that if b is evaluated to true, then $p(x)$ is permitted and the governor evolves to G. The basic permission set *Perm* is inspired by the co-actions in safe ambient calculus [11]. But here we use co-actions as regulated authorizations, not used as those in synchronous calculi. $G \setminus p(x)$ denotes authorization allowed by G except $p(\bar{x})$. The other constructs are similar to those used logic expressions.

Definition 2.5. (RBA governor process reductions) The governor reduction relation, →, is the least relation satisfying the following axiom and rules:

$permit_1\ \dfrac{b[\bar{x} := \bar{v}]\ \text{true}}{\text{when } b\ p(\bar{x})\ \text{ then } G \xrightarrow{p(\bar{v})} G[\bar{x} := \bar{v}]}\ \bar{v}$ closed

$permit_2\ \dfrac{\neg b[\bar{x} := \bar{v}]\ \text{true}\ \ G \xrightarrow{p(\bar{v})} G'}{\text{when } b\ G \setminus p(\bar{x}) \xrightarrow{p(\bar{v})} \text{when } b\ G' \setminus p(\bar{x})}$

$permit_3\ \dfrac{G \xrightarrow{p'(\bar{v})} G'}{\text{when } b\ G \setminus p(\bar{x}) \xrightarrow{p'(\bar{v})} \text{when } b\ G' \setminus p(\bar{x})}\ p' \neq p$

$ref\ \dfrac{G[\bar{x} := \bar{v}'] \xrightarrow{p'(\bar{v})} G'}{W(\bar{v}') \xrightarrow{p(\bar{v})} G}\ W(\bar{x}) \triangleq G$

$or\ \dfrac{G_i \xrightarrow{p(\bar{v})} G'}{G_1 \vee G_2 \xrightarrow{p(\bar{v})} G'}\ i \in \{1,2\}$

$and\ \dfrac{G_1 \xrightarrow{p(\bar{v})} G_1'\ \ G_2 \xrightarrow{p(\bar{v})} G_2'}{G_1 \wedge G_2 \xrightarrow{p(\bar{v})} G_1' \wedge G_2'}$

$\top\ \dfrac{}{\top \xrightarrow{p(\bar{v})} \top}\ \bar{v}$ closed $\qquad \bot\ \dfrac{}{\bot \longrightarrow \bot}$

Generally a RBA system consists of multiple processes and governors. For simplicity we denote the form of system configuration as *n(G)[P]*, where *P* is a process or an ambient. If processes are not assigned governors, then they will evolve according to reduction rules defined in definition 2.3. If a process is assigned a governor, system configuration transitions will depend on both the process and the governor. The system configuration transition rules are defined in the following.

Definition 2.6. (RBA system configuration transition rules)

$new\ \dfrac{P \xrightarrow{new(n)} P'\ \ G \xrightarrow{\overline{new}(n)} G'}{n(G)[P] \mapsto n(G')[P']}$

$spawn\ \dfrac{P \xrightarrow{spawn(P_1)} P'\ G \xrightarrow{\overline{spawn}(P_1)} G'}{n(G)[P] \mapsto n(G')[P_1, P']}$

$box\ \dfrac{P \xrightarrow{box(n_1,P_1)} P'\ \ G \xrightarrow{\overline{box}(n_1,P_1)} G'}{n(G)[P] \mapsto n(G')[n_1[P_1], P']}$

$in\ \dfrac{\begin{array}{c} P \xrightarrow{in(n_2)} P'\ \ G_1 \xrightarrow{\overline{in}(n,n_2)} G_1' \\ G_2' \xrightarrow{\overline{in}(n,n_1)} G_2'\ \ G \xrightarrow{\overline{in}(n_1,n_2)} G' \end{array}}{\begin{array}{l} n(G)[M, n_1(G_1)[P, M_1], n_2(G_2)[M_2]] \mapsto \\ n(G')[M, n_2(G_2')[n_1(G_1')[P', M_1], M_2]] \end{array}}$

$out\ \dfrac{\begin{array}{c} P \xrightarrow{out(n_2)} P'\ \ G_1 \xrightarrow{\overline{out}(n,n_2)} G_1' \\ G_2 \xrightarrow{\overline{out}(n,n_1)} G_2'\ \ G \xrightarrow{\overline{out}(n_1,n_2)} G' \end{array}}{\begin{array}{l} n(G)[M, n_2(G_2)[n_1(G_1)[P, M_1], M_2]] \mapsto \\ n(G')[M, n_1(G_1')[P', M_1], n_2(G_2')[M_2]] \end{array}}$

$$lcom\ \frac{P_1 \xrightarrow{wr(v)} P_1'\ \ P_2 \xrightarrow{rd(v)} P_2' \quad G \xrightarrow{\overline{lcom}(v)} G'}{n(G)[M,P_1,P_2] \mapsto n(G')[M,P_1',P_2']}$$

$$rcom_1\ \frac{P_1 \xrightarrow{\langle v \rangle^{n_2}} P_1'\ \ P_2 \xrightarrow{(v)^{n_1}} P_2' \quad G_1 \wedge G_2 \xrightarrow{\overline{wrdn}} G_1' \wedge G_2'}{n_1(G_1)[P_1, n_2(G_2)[P_2]] \mapsto n_1(G_1')[P_1', n_2(G_2')[P_2']]}$$

$$rcom_2\ \frac{P_1 \xrightarrow{(v)^{n_2}} P_1'\ \ P_2 \xrightarrow{\langle v \rangle^{n_1}} P_2' \quad G_1 \wedge G_2 \xrightarrow{\overline{rddn}} G_1' \wedge G_2'}{n_1(G_1)[P_1, n_2(G_2)[P_2]] \mapsto n_1(G_1')[P_1', n_2(G_2')[P_2']]}$$

3. Electronic institutional models

Electronic institution (EI) is a promising framework to develop regulated interactive computation system in open networked environment. To illustrate the RBA capability, we use RBA to formally represent the EI model. But before presenting the procedure, we first briefly introduce the EI model. Based on the work [12], EI is formulated as follows.

Definition 3.1. An electronic institution is defined as a 5-tuple $EI = < PS, IR, \succeq, ssd, N_{PS} >$, where:

(1) *PS* stands for a performative structure; (2) *IR* is a subset of roles representing the institutional roles; (3) $\succeq$ stands for the hierarchy partial order over the roles; (4) *ssd* is the set of static separation of duties between roles; and (5) N_{PS} stands for a set of normative rules.

Definition 3.2. The performative structure *PS* is $PS = < S, T, s_0, s_\Omega, E; f_L, f_T; f_E^O, C, ML, \mu >$ where: (1) S is a set of scenes; (2) T is a set of transitions; (3) $s_0 \subseteq S$ is the initial scene; (4) $s_\Omega \subseteq S$ is the final scene; (5) $E = E^I \cup E^O$ is a set of arc identifiers where $E^I \subseteq S \times T$ is a set of edges from scenes to transitions and $E^O \subseteq T \times S$ is a set of edges from transitions to scenes; (6) $f_L : E \to DNF_{2^{V_A \times R}}$ maps each arc to a disjunctive normal form of pairs of agent variable and role identifier representing the arc label; (7) $f_T : T \to \Im$ maps each transition to its type; (8) $f_E^O : E^O \to \exists$ maps each arc to its type (one, some, all or new); (9) $C : E \to ML$ maps each arc to a meta- language expression of type Boolean, i.e. a formula representing the arc's constraints that agents must satisfy to traverse the arc; (10) ML is a meta-language; (11) $\mu : S \to \{0,1\}$ states whether a scene can be multiply instantiated at run time or not.

Definition 3.3. A scene of EI is a tuple $S = < R, DF_S, W, w_0, W_f, (WA_r)_{r \in R}, (WE_r)_{r \in R}, \theta, \lambda, min, max >$ where : (1) *R* is the set of scene roles involved in that scene; (2) DF_S is the restriction to the scene *s* of the EI dialogical framework defined below; (3) *W* is the set of scene states; (4) $w_0 \in W$ is the initial state; (5) $W_f \in W$ is the set of final states; (6) $(WA_r)_{r \in R} \subseteq W$ is a family of sets such that WA_r stands for the set of access states for role $r \in R$; (7) $(WE_r)_{r \in R} \subseteq W$ is a family of non-empty sets such that WE_r stands for the set of exit states for role $r \in R$; (8) $\theta \subseteq W \times W$ is a set of directed edges; (9) $\lambda : \theta \to L$ is a labelling function, where L can be a timeout, or an illocution schemata and a list of constraints; (10) $min, max : R \to \mathbb{N}$ *min(r)* and *max(r)* return the minimum and maximum number of agents that must and can play role $r \in R$.

Definition 3.4. A dialogue framework DF is a tuple *DF=<O, L, I, R_I,R_E,R_S>*, where: (1) *O* stands for the EI domain ontology; (2) *L* stands for a content language to express the information exchanged between agents; (3) *I* is the set of illocutionary particles, usually it takes the form of *i(a, r, a',r',m,t)*, meaning that agent *a* playing role *r* sends illocution *i* with content *m* to agent *a'* playing role *r'* at time t. (4) R_I is the set of internal roles; (5) R_E is the set of external roles; (6) R_S is the set of relationships over roles

Dialogue framework, scene, performative structure provides a basic interactive space of agents. Based on these structural notions, some predicates or functions can be defined. These definitions will further applied to specify normative rules for an EI. For example, if we define the following predicates, one type of normative rules takes the form of definition 2.5.

(1) *uttered(s, w, i)* denoting that a grounded illocution unifying with the illocution scheme *i* has been uttered at state *w* of scene *s*.

(2) *uttered(s, i)* denoting that a grounded illocution unifying with the illocution scheme *i* has been uttered at some (unspecified) state of scene *s*.

Definition 3.5. Normative rules are first-order formulae of the form $(\bigwedge_{j=1}^{n} uttered(s_j, [w_{k_j}], i_{l_j}) \wedge \bigwedge_{k=0}^{m} e_k) \to (\bigwedge_{j=1}^{n'} uttered(s_j', [w_{k_j}'], i_{l_j}') \wedge \bigwedge_{k=0}^{m'} e_k')$ where: s_j, s_j' are scene identifiers, w_{k_j}, w_{k_j}' are states of s_j and s_j' respectively; i_{l_j}, i_{l_j}' are illocution schemata i_l of scenes s_j and s_j' respectively, and e_k, e_k' are Boolean

expressions over variables from the illocution schemata i_{l_j} and i'_{l_j} respectively.

The intuitive meaning of normative rules is that if grounded illocutions matching $i_{l_1}, \ldots, i_{l_n}$ are uttered in the corresponding scene states and the expressions $e_1, \ldots, e_m$ are satisfied, then grounded illocutions matching $i'_{l_1}, \ldots, i'_{l_n}$ satisfying the expressions $e'_1, \ldots, e'_m$ must be uttered in the corresponding scene states.

Although EI model has been studied by several researchers and this type of EI representation provides a quite clear image to electronic institutions, however it is rather informal and concurrency haven't been considered in appropriate way [3][6][12]-[13]. As lack of concrete syntax and semantics definition for EI, we can not execute the EI specification and prove some properties that an EI should maintained with structural techniques. Thus in the following, we propose to represent EI with RBA, and then EI verification or validation can be coped with process bisimulation theory.

4. Design electronic institutions with RBA

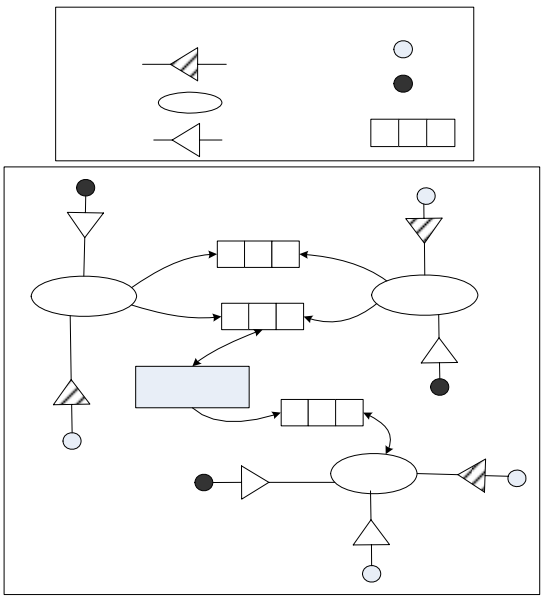
From the above presentation of RBA and EI, there is an attractive mapping of scenes to immobile ambients with governors, where the conversation can take place via speech acts, and of agents to mobile ambients without governors, which move from scene to scene under its internal decisions given the appropriate role-playing capability. That is for each scene, a corresponding ambient with role governors will be built to regulate the joint task activities performed by the joining role playing agents, which itself will be modelled by an mobile ambient with role playing capabilities. For instance, we use ambient $s_1(\underline{r_1}, \underline{r_2})[PS]$ to denote a scene named s_1 with two role governors r_1 and r_2, and PS denotes the process own by the scene itself; and we use $ag[r_1]$ to denote an agent named ag which has capability of playing role r_1. Combining these two parts, we can generate a scene instance, such as $s_1(\underline{r_1}, \underline{r_2})[PS, ag_1[r_1], ag_2[r_1], ag_3[r_2]]$. In this instance, there are two agents (ag_1 and ag_2) who move into the scene and play the role r_1, while only one agent (ag_3) moves into and plays the role r_2.

Generally an EI is mapped to RBA constructs according to the criteria listed in table 1.

EI	RBA	notation
Scene name	Immobile ambient name	n,s,…∈Name
Role-player(agent)	Mobile ambient with role playing capability	ag[role]
Role-governor	Ambient governor	r,g,…∈G
Scene with roles	Ambient with governors and processes	$Scene_i(r_{ij})[S_i P]$
Performative Structure — Scenes	Set of ambients	$\{Scene_i(r_{ij})[S_i P]\}$
Performative Structure — transtions	capbilities	p∈Perm
illocutions	domain task messages	x
norms	Capability constraints	b∈Boolean

Table 1: Transfer elements from EI constructs of RBA.

To further illustrate the application of RBA to EI design, a classical example designed with RBA is presented in the following. Suppose that there is a need to construct an academic workshop institution which supports to deal with the working scenes of organizing a workshop. We suppose there are three main scenes involving in this institution. That is submitting papers, reviewing papers and paying publication fees. For each scene, there exist a scene manager and participating agents. For example, in the submitting scene, there are one manager role (manager1) and one author role. The three scenes and attached roles are shown in figure 1. The illocutions related to domain activities are labelled as tags along the lines in figure 1. To understand easily the data flow, institutional data files are explicitly separated from institutional scenes in this figure.



```
EI::=Scene0 Scene1 Scene2 Scene3
Scene0::=top(register)[S0P],a[register,Pa]
    register=in top. (query+register+ out top)
    S0P=S0P1 S0P2
    S0P1=(?registerInfo). t . <certificate>. S0P1
    S0P2=(?qury). t . <queryRes>. S0P2
    register=<registerInfo>. (?certificate). (register+ out top)
    query=<qury>. (?queryRes). t . (query+ out top)
    Pa=?    //other processes of the agent
Scene1::=submitPaper(author,manager1)[S1P],a[author],a[manager1]
    author=in submitPaper. sub-que. out submitPaper. author
    sub-que=(submit+ query). (nil+sub-que)
    submit=<subPaper>. (?result). t
    query=<queryInfo>. (?result). t
    manager1=in submitPaper. start. stop. out submitPaper
    S1P=S1P1 S1P2 S1P3
    S1P1=(?subPaper). t . <result>. S1P1
    S1P2=(?qury). t . <result>. S1P2
    S1P3=(?start). t . (nil+(?stop). t )
Scene2::=assignPaper(reviewer, manager2)[S2P],a[reviewer],a[manager2]
    manager2=in assignPaper. start. assign. stop. out assignpaper.
    assign=<getPaper>. (?Paper). assignPaper.(assign+nil)
    S2P=S2P1 S2P2
    S2P1=(?getPaper).t . <asnPaper>. S2P1
    S2P2=(?getAssignPaper). <ashPaper>.S2P2
    reviewer=<getAssignPaper>.(?asnPaper). t . <return judgement>.
        (reviewer+ out assignPaper)
Scene3::=payment(payer, payee,manager3)[S3P],a[payer],
        a[payee],a[manager3]
    ...
```

Table2: Process specification for workshop institution

According to the criteria, the RBA for this EI is specified in table 2. Line (1) indicates that this institution is composed of four scenes (scene$_{0\text{-}3}$). Scene$_0$ is the top scene that represents the outer scene in which agent can register to play roles of the institution. If one agent successfully registers into the institution, a certificate is issued to the agent with which authorized agents can join other corresponding scenes if they decide to do so. For example if an agent registers successfully as an author player in scene$_0$, then it can join submitting scene to submit a paper to the workshop or query the information related to its paper submitted. During the scene interaction, author player agent can only do the authorized actions according to the author governor norms, others will not be permitted.

With RBA formalism of EI, EI design can be verified with process theory. This aspect will not be presented here for beyond the scope of this paper.

5. Conclusions

Developing trustable interactive system is a difficult work in open and heterogeneous environment. Social and organizational theories provide suitable starting point to guide the development ofsuch type of systems, but there is still lack of systemic engineering methods to make them much practicable usage. For instance, EI has been discussed by some papers [8]-[9], but many of them focus on modeling EI with deontic logic or state-based formalism, which is difficult to use when

focusing on design and implementing EI. The work in this paper proposes to design globally interactive collaboration system starting from electronic institution model and specified with RBA. RBA is designed to present the complex interactive and regulative patterns in system design specification. Through formal defined operational semantic rules, system requirement norms can be verified at design level. A proof of concept example has been working out and the experience shows some great advantages over other approaches when designing such type of interaction complex systems. As far as future work is concerned, we are studying on verifying EI in RBA formalism.

Acknowledgement

This work is partially supported by National Nature Science Foundation of Guangxi Province, China (Grant No. 0728089).

References

[1] F. Zambonelli, N. Jennings, M. Wooldridge, Developing multiagent systems: the gaia methodology. ACM Transactions on Software Engineering and Methodology, 12 (3): 317-370, 2003.

[2] Nalini Venkatasubramanian and Carolyn L. Talcott, Reasoning about meta level activities in open distributed systems. In Symposium on Principles of Distributed Computing, pp.144-152, 1995.

[3] F. Dignum, Norms and Electronic Institutions. L. Goble and J.-J.C. Meyer (Eds.): DEON 2006, LNAI 4048: 2-5, 2006.

[4] Egon L. van den Broek, Catholijn M. Jonker, Alexei Sharpanskykh and Jan Treur, et.al., Formal Modeling and Analysis of Organizations. O. Boissier, et al. (Eds.): ANIREM and OOOP 2005, LNAI 3913: 18-34, 2006.

[5] C. Sibertin-Blanc, P. Amblard and M. Mailliard, A Coordination Framework Based on the Sociology of Organized Action. O. Boissier et al. (eds.): ANIREM and OOOP 2005, LNAI, 3913: 3 – 17, 2006.

[6] V. Dignum, A model for organizational interaction. PhD. Thesis, Dutch Research School for Information and Knowledge Systems, ISBN 90-393-3568-0, 2004.

[7] R. Milner, Communication and Concurrency, Prentice Hall, 1989.

[8] R. Milner, J. Parrow, D. Walker, A calculus of mobile processes, Information and Computation, 100 (1): 1-77, 1992.

[9] M. Bugliesi, S. Crafa, M. Merro and V. Sassone. Communication and Mobility Control in Boxed Ambients. Information and Computation, 202(1): 39-86, 2005.

[10] D. Gorla, Comparing calculi for mobility via their relative expressive power. Technical Report 09/2006, Dipartimento di Informatica, Universit`a di Roma “La Sapienza, 2006.

[11] L. Cardelli, G. Ghelli and A. D. Gordon, Types for the Ambient Calculus. Information and Computation, 177(2): 160–194, 2002.

[12] M. Esteva, Electronic Institutions: from specification to development. Number 19 in IIIA Monograph Series. PhD Thesis, 2003.

[13] M. Esteva, D. de la Cruz, C. Sierra, ISLANDER: an electronic institutions, editor. In: Proc. of the 1st International Joint Conference on Autonomous Agents and Multi-agent Systems, pp.1045–1052, 2002.