Web Intelligence and Agent Systems: An International Journal 7 (2009) 1–21
DOI 10.3233/WIA-2009-0152
IOS Press

Evaluation of properties in the transition of capability based agent organization

Eric T. Matson [a,*], Scott A. DeLoach [c] and Raj Bhatnagar [b]

[a] *Department of Computer and Information Technology, Purdue University, West Lafayette, Indiana, USA*
[b] *Department of Computer Science, University of Cincinnati, Cincinnati, Ohio, USA*
[c] *Department of Computing and Information Sciences, Kansas State University, Manhattan, Kansas, USA*

Abstract. It has been said that the only constant in life is change. This rule can also be directly applied to the lives of organizations. Any organization of non-trivial size, scope, life expectancy or function, is destined to change. An organization without the ability to transition is not robust, evolvable or adaptable within its environment. These basic preconditions to human organizations must also hold in viable agent organizations. To model an adaptable agent organization, the capability must be present to transition from one state to the next over the life of the organization. The organization model must include not only the structural objects, but also the ability to facilitate change. The ability to change empowers the organization to transition from one state to the next, over its useful life. To enable transition, we must formally capture and define what triggers an organization transition. In this paper, we will define the properties to formally model the ability of an adaptable organization to transition throughout its useful life. The properties will be instantiated, using an implemented system, allowing the evaluation of internal and external stimuli to cause transition to the organization. These transitions will be evaluated from several perspectives to determine their effectivity on basis of design and use.

Keywords: multiagent systems, transition, organization, reorganization, properties

1. Introduction

No matter what aspect of life that we consider, change continuously occurs. We switch jobs, gain new friends, lose old friends, become more educated and eventually lose physical vigor and function as we grow older. The results of these changes impact larger aspects of our lives. Our lives are connected to the others around us to whom we are related by family, work or other community-based relationships.

Because we are all linked to a number of simple and complex organizations such as families, corporations, universities, sports teams and friends, we also understand these organizations change over time. We tend to reason about these changes at a summary level, not considering the individualized transactions occurring to initiate or complete these changes to those around us. These changes are transitions that alter our relationships and the organizations which constitute the extent of our relationships and the people involved.

The process of transition can either be the initial organization formation or a reorganization that takes place when triggered by some event, causing a change. To model an artificial organization, based upon our understanding of a human model, we must formalize the intricacies and understand the complexity of all transactions [2]. In this case, the transaction of interest is that of transition. This study of transition results in a research effort which is an extension of work first proposed in a previous effort [17].

There is no reasonable manner in which to understand the basic transition processes of organization and reorganization, without first understanding what triggers these processes. We can view the phenomenon of transition, from a mechanistic perspective, without understanding the properties that drive the change to take place. This lack of understanding allows us to see how an organization may change, but does not imply

*Corresponding author. E-mail: ematson@purdue.edu.

any deeper perspective into why the organization must change due to some change in capability, relationship or modification of structure. This effort proposes to approach not just the mechanical operations that describe the change, but the properties that provide reason for the organization to transition.

Transition processes, relating to agent organizations, are basic environmental stimuli, or sets of stimuli, that engage the organization to behave in some non-deterministic manner. The stimuli can either originate from within the organization or external to the organization. We will establish *organization properties* to capture the nature of these stimuli. While the organization properties are abstract, *transition predicates* can be created to allow direct instantiation of transition stimuli. These predicates are then instantiable in a logical system to model the organization and the transition properties to create a realistic example of organizations, which can be tested, evaluated and validated. This approach satisfies the requirement to quantify transition properties and the establishment of a direct relationship between an organization and the task environment in which it exists and acts.

An *internal force* comes from some element currently participating in the organization. An *external force* comes from some element that is not currently acting within the organization. Transition is the process undertaken when some force acts upon the organization with such dramatic effect as to alter its basic constitution. The force that acts upon the organization can either be an internal or external force. Figure 1 exhibits the nature of external and internal forces in relation to the organization. There is a great deal of research with agent organizations, self-organization and societies. There are numerous organizational models such as *OperA* described by Dignum [7], *MOISE+* from Hubner, Sichman and Boissier, or the *OMACS* model from DeLoach and Matson [5], to cite a few. These and many other models are summarized and described by Horling and Lesser [15]. The missing element in these models is the formalization in how they transition from one state to the next based on some stimulus, as approached by Dignum, Sonenberg and Dignum [8]. Costa and Demazeau presented the need for dynamic elements to organizations [4]. There have been computational approaches to organization, such as by Carley [3] but most describe the abstract notion of transition. Omicini, Ricci and Viroli [21] provide a basis for algebraic representation covering the details of organization and coordination, particularly examining dynamic relationships between agents within an

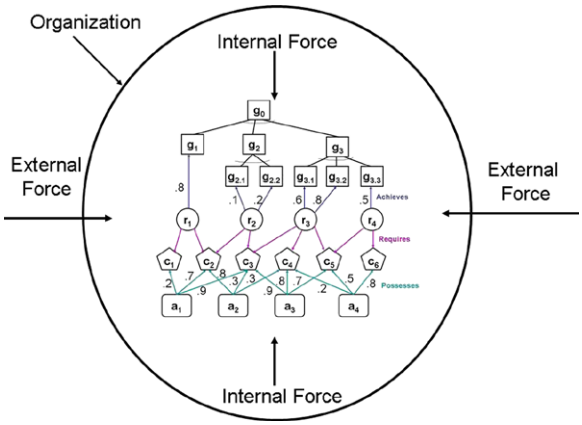

Fig. 1. Transition forces.

organization. Glaser and Morigot [12] provide a useful, but incomplete, idea of reorganization, although many basic agent structure elements are represented. A more complete work of modeling organizational change, specific to multiagent system, is described by Hoogendoorn generally [13] and also applied to more specific cases [14], which draws closer to a complete solution.

While the models, theory and algebraic proposals provide a solid background in describing and modeling organization operation, they do not provide a method to capture the properties required to capture the essence of what begins an organization's transition from one state to the next. Without a method, along with structure, to transition from one state to the next, an organization cannot exhibit the global capability to adapt or overcome impediments in its achievement of goals. Our approach not only specifies the theoretical elements of an organization model the required transition elements, but implements the model into a complete system. We propose a transition-capable model system that is first formally defined, in general as a transition system. Then a system for capturing stimuli is defined and translates the transition property into a usable predicates, as the input for the system. We will use predicates as rules to model the transitions as proposed by Zambonelli, Jennings and Wooldridge [19]. This paper will provide an intersection to formalize transition into a simple set of predicates. This work is intentionally general to insure it is applicable to any logically formed organization model. We then add an organization engine as an implementation which is evaluated using a number of different organization sizes. The end result is a complete system to capture and realize multiagent organization transition,

capable of adaptation and survival, over a series of environmental changes.

The basic theory and formalizations to capture the transition processes of organization and reorganization are discussed in Section 2. Section 2 additionally covers the theory of transition, first shown as abstract transition properties and discussed intuitively, followed by a translation into specific transition predicates that can be formally used to describe agent organization transition. An applied example of transition properties is detailed in Section 4 using the organization model and transition predicates, including a instantiation and implementation of an organization for evaluation. The example shows how a small conference workshop organization can transition. Section 4 further extends the example simulation to look at organization properties and the effects on an organization from a number of different perspectives. Finally, Section 5 provides some concluding thoughts and summaries.

2. Organization, properties and transition

In organizational research, organizations have commonly been modeled using agents to play roles within a structure in order to satisfy a given set of goals. This research has a basic foundation of dividing the elements of our organization model into *structural*, *state* and *transitional* elements. Our organizational model (O) includes a structural element, a state element and a basic transition function [6].

$$O = (O_{structure}, O_{state}, O_{transition}) \tag{1}$$

Transition is formally described as an abstraction of two distinct computational processes; initial organization and reorganization. While seemingly the same idea, there are some distinct differences between the two processes. For either process, the symbol $\Rightarrow$ will represent a single transition from one state to the next. Formally, we can describe this as:

$$Transition = \{init.\,organization, reorganization\} \tag{2}$$

The process of *initial organization* begins at an initial state O_0 where there exists no pre-existing organization and an organizational structure results O_1. The resulting organization structure is the first organization instance.

$$init.\,organization(O) \equiv transition(O_0) \Rightarrow O_1 \tag{3}$$

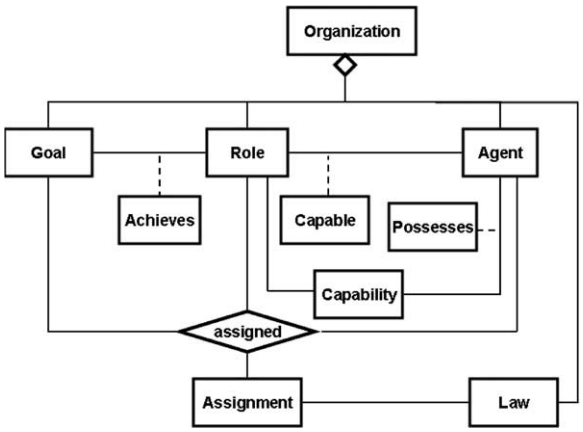

Fig. 2. Organization structure.

The process of *reorganization* begins with an already existing organization O_n and transitions to a new state O_{n+1}, representing a different and unique organization instance.

$$reorganization(O) \equiv transition(O_n) \Rightarrow O_{n+1} \tag{4}$$

Before looking at the details of transition, we must first review the details of the other elements of the organization model, *structure* and *state*. Without an understanding of these elements, as described in [5], transition will be difficult to understand. Figure 2 shows the objects and relationships, via a UML diagram of the organization model elements. In the two sections, *structural* and *state* elements are discussed.

2.1. Structure

The *structure* is defined by:

$$O_{structure} =< G, R, L, C, ach, req, sub, con > \tag{5}$$

where G describes the set of *goals*, R is the set of *roles*, L is the set of *laws* or rules required, C is the set of *capabilities*, *ach* is *achieves*, *req* is *requires*, *sub* is *subgoal*, *con* is *conjunctive*.

The organization structure also contains a set of relations. The achieves relation, $achieves : R, G \rightarrow [0..1]$, states the relative ability of a *role* to satisfy a given *goal*. *Roles* require *capabilities* to satisfy a set of *goals* and this is captured by the $requires : R, C \rightarrow Boolean$. The organization may contain subgoal relationships $subgoal : G, G \rightarrow Boolean$. The conjunctive relationship between *goals* is $conjunctive : G \rightarrow$

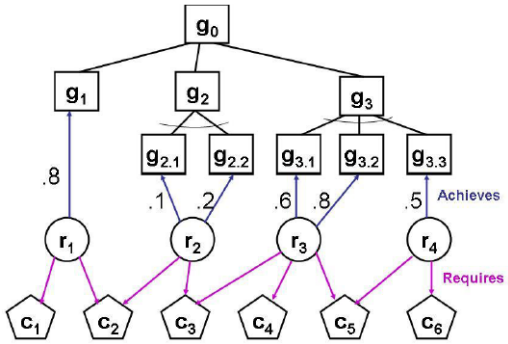

Fig. 3. Organization structure graph.

$Boolean$. Figure 3 displays the graphical relationship using the objects and relationships of the structural element.

2.2. State

The *state* is defined by:

$$O_{state} =< A, possesses, capable, assigned > \quad (6)$$

where an A defines a set of *agents* available to participate in the organization. There are several relationships in the state element of the organization. An *agent* capable of playing a certain role possesses the necessary *capabilities* described by the possesses relation, $possesses : A, C \rightarrow [0..1]$. An *agent* is capable of playing a *role* in the organization as described by the capable relation, $capable : A, R \rightarrow [0..1]$. The assigned relation, $assigned : A, R, G \rightarrow [0..1]$, is used to match the best *agent*, *role*, *goal* combination that maximizes the capability of the organization.

2.3. Transition

The general form of our theoretic organization model approach is expressed by:

$$O_{transition} = (O, \Phi, \delta, s_n, S_{optimal}, S_{possible}, S_{final}) \quad (7)$$

where O is the organization over which the transition will occur, Φ is the set of properties that can trigger a transition of the organization, δ is the transition function, s_n is the set of relative states of the organization, $S_{optimal}$ is the set of optimal states that result from transition and $S_{possible}$ are states that are possible to

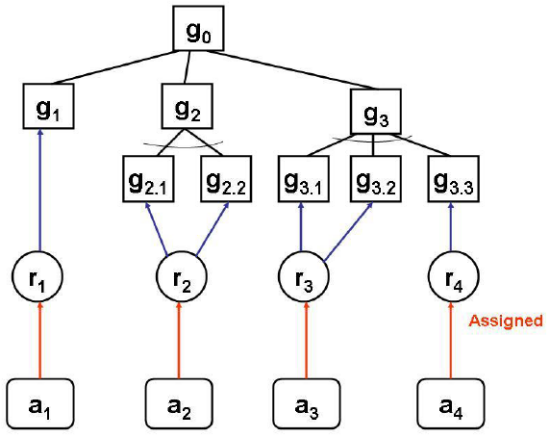

Fig. 4. State.

reach, from the current state. S_{final} is a set of organization states where all goals are satisfied, or the first goal is satisfied, or it is determined that not all goals can be satisfied. Even though the outcomes are different, each final state draws a conclusion to the organization's set of transitions. Because an organization can only exist as a single entity or instance, the current state s_n is always a unique value [16].

The basic transition is defined as a product of the O, Φ and S resulting in a set of reachable organization states:

$$\delta : O \times \Phi \times S \Rightarrow S \quad (8)$$

So the transition function will be of the form:

$$\delta(O, \phi, s_n) \Rightarrow S' \quad (9)$$

where transition function δ takes the organization O, a *specific* transition property ϕ, and a state of the organization s_n and can transition to a set of new states S' where:

$$S_{optimal} \subseteq S_{possible} \quad (10)$$

$$S_{optimal} \subseteq S' \quad (11)$$

$$S_{final} \subseteq S_{possible} \quad (12)$$

These equations apply for both finite and infinite transition organizations, with the added constraint that $|S_{final}| \geqslant 1$ for finite transitions and $|S_{final}| = 0$ for infinite transitions, indicating that finite transitions have one or more end states and infinite transitions have no states defined as end states.

Where a *finite state automaton* uses a string of symbols to transition, as normally used to validate lan-

guages, our transition function takes as input a string of transition properties ϕ as input. This string of properties is not predetermined, but will be generated dynamically as the organization interacts in its environment as represented by $\{\phi_0, \phi_1, \ldots, \phi_n \mid \phi \in \Phi, n \in N\}$.

Anytime any element of the organization changes, a new organization state will be instantiated. Even the smallest change in the organization state changes the structural integrity of the organization state, and therefore, a transition must occur.

When a property change initiates a reorganization, the organization may transition to a new state s_{n+1} or it can also result in the same state (s_n) where, even though the property changed, the values and relationships of the organization instance did not change.

Definition. *Finite Transition* formally expresses the definition of a finite organization transition. The condition must hold for the set of transitions to eventually result in a final state being reached, where $(\Rightarrow^*)$ defines a set of transitions from a state to an end state. The transition:

$$\delta(O_i, \phi, S) \Rightarrow^* S_{final} \tag{13}$$

states that there is a set of transitions that will lead from some initial machine state to a S_{final} or final state. This will also indicate that, if designed correctly, a finite transition organization will terminate. An example of a finite transition will be that of a software project. A project has a definite start and end. While there may be a number of transitions during the project, the project will come to an end and therefore, end any transition after that time.

Definition. *Infinite Transition* is expressed in a similar manner to the finite. By definition, an infinite organization, never reaches a final state, so the expression will result in S_∞, instead of S_{final}.

$$\delta(O_i, \phi, S) \Rightarrow^* S_\infty \tag{14}$$

This indicates that an infinite organization will not reach a final state and will continue on transitioning indefinitely. For this reason, $|S_{final}| = 0$ will be an enforced constraint for any infinite transition organization. An example of an infinite organization is that of a software company. An organization is put in place that has no plans to stop. While some companies obviously do terminate, they never plan to stay in business for a while and then plot their own demise. They tend to plan as a perpetual organization.

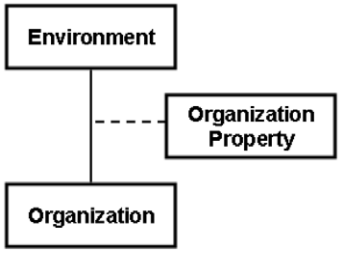

Fig. 5. State.

2.4. Properties

Organization transition properties must be developed in a formal description. First, we describe what transition properties are in an abstract sense. In this section, we will describe abstract organization properties and specific organization properties in an intuitive manner. Then, we will proceed to translate the intuition into a formal set of properties. Figure 5 shows the direct relationship between an organization and the environment in which it works.

Changes in organization structure and participants will drive transition activities. Transition properties can be triggered internally or externally. The general transition properties can be split into properties that are external and those that are internal.

Definition. *Internal Force* represents any stimulus that is part of the organization which causes the organization to transition. An example is an agent who loses capability, thereby not being able to fulfill the requirements of some role.

Definition. *External Force* is any stimulus that is not part of the organization which causes the organization to transition. An example is the entry of a new agent, more capable to play a role than an agent already playing that role.

2.4.1. Organization properties

An organization property Φ is somewhat of an abstract theoretical term. It is abstract to capture the generic nature of what it can define. In general terms, an organization will need a set of properties Φ, for example, *capabilities* or *agents*, which by their existence can be the reason for a transition. A major element of defining transition will be the definition of these properties such that individual properties ϕ can be identified as transition triggers. Any individual property ϕ in Φ is eligible to act as a reorganization trigger. Some examples of ϕ include a change in the real value of a capability, the loss of overall capability or agent func-

tion, loss of an agent, the reentry of an agent, or the addition of a new agent.

The basic *assumption* is the system will act rationally as it will always act in its own best interest. Its collective best interest will be defined as always maintaining the highest capability score.

The *intuition* of ϕ is some intrinsic or extrinsic force which will trigger a potential transition in the organization such that $\phi \in \Phi$ where ϕ represents an individual transition property.

There are a number of task specific transition properties that exist in any domain problem. What we intend to capture here are general transition properties that can be applied to any organization. These general properties can be instantiated to fit specific examples, as will be shown in the next section. The general transition properties are:

1. Loss of an agent participating in the organization
2. An agent loses capability required to play some role
3. A new agent becomes available
4. Capability of an agent increases
5. Capability of an agent decreases
6. A goal is removed
7. A goal is added
8. A goal is relaxed (changed)
9. Change in goals to roles achieves relationship
10. Change in role to capability requires relationship

Organization properties are by their nature, abstract and general. To formalize the premise of transition, transition predicates are formed to be distinct and specific outcomes of transition properties.

2.4.2. Transition predicates

Using transition predicates allows a way of formalizing the individual transition properties of a unique organization.

A key assertion in formalizing transition predicates is that there is no requirement or need for temporal specifications to model the predicates. As each predicate represents a reason to consider reorganization, there is no need to apply a larger temporal language. While there are temporal problems associated with transition, the granularity of properties and the associated predicates remove the need to attach temporal specifications.

Transition predicates can be abstracted in several forms:

$$\Phi = \{\phi_1 \dots \phi_n\} \tag{15}$$

In general, Φ can be expressed as a set of standard, abstract predicates:

$$\Phi = \{\phi_{lose}, \phi_{add}, \phi_{change}\} \tag{16}$$

where ϕ_{lose} is the abstract property dealing with loss, such as losing an agent from the organization or an agent losing capability to play a role. The add property ϕ_{add} describes the action when an object or relationship is added to the organization. For example, $\phi_{add}agent$ an agent becomes available for invitation to the organization. The change property ϕ_{change} can either be an increase or decrease and further specializes the change predicate:

$$\phi_{change} = \{\phi_{decrease}, \phi_{increase}\} \tag{17}$$

Definition. *Primitive Predicates* can be used to formalize single properties. The *primitive predicates* exhibit polymorphic behavior as each can be applied to different organization elements to capture different properties. If there is a loss of an agent participating in the organization, it can be formalized as the predicate $\phi_{lose}agent(a)$. An agent a losing some capability can be captured as $\phi_{lose}capability(c, a)$.

Definition. *Complex Predicates* are the combination of primitive predicates. Some predicates will encompass others, but in some cases two properties can be successfully combined to form a single property of transition. The complex predicates will be logically constructed using primitive predicates and the common *and* ($\wedge$) and *or* ($\vee$) binary relations. Examples of complex predicates are shown in Table 1.

In the case that an agent exits an organization, it can be reasoned that all capability of that agent will also exit. Combining the two previous predicates of losing an agent and losing a capability by an agent are redundant, in respect to the capability predicate $\phi_{lose}agent(a) \wedge \phi_{lose}capability(c, a)$, as long as the capabilities are not possessed by another agent or required by a role. Deleting an object will involve deleting any relationships which are dependent on the object.

In another situation, an organization may lose two agents simultaneously. If agents a and b both leave, we can capture that by $\phi_{lose}agent(a) \wedge \phi_{lose}agent(b)$, where one primitive predicate does not contain the other. Complex predicates are unlimited in their scope. They may be used to create a set of relationships and

Table 1
Complex predicates

Predicate	Description
$\phi_{add}agent(a) \wedge \phi_{add}agent(b)$	Add new agents a and b
$\phi_{add}goal(g) \wedge \phi_{add}goal(h) \wedge \phi_{add}subgoal(g,h)$	Add goals g, h and a subgoal relation
$\phi_{add}role(r) \wedge \phi_{lose}role(s)$	Add role r and delete role s

Table 2
Object predicates

ϕ	Object	Predicate
add	agent	$\phi_{add}agent(a)$
	goal	$\phi_{add}goal(g)$
	role	$\phi_{add}role(r)$
	capability	$\phi_{add}capability(c)$
	law	$\phi_{add}law(a, max(r,2))$
lose	agent	$\phi_{lose}agent(a)$
	goal	$\phi_{lose}goal(g)$
	role	$\phi_{lose}role(r)$
	capability	$\phi_{lose}capability(c)$
	law	$\phi_{lose}law(a, max(r,2))$
change	agent	$\phi_{increase}capability(c,a)$
	agent	$\phi_{decrease}capability(c,a)$
	goal	$\phi_{change}goal(g)$

Table 3
Relationship predicates

ϕ	Relationship	Predicate
add	achieves	$\phi_{add}achieves(r,g)$
	requires	$\phi_{add}requires(r,c)$
	possesses	$\phi_{add}possesses(a,c)$
lose	achieves	$\phi_{lose}achieves(r,g)$
	requires	$\phi_{lose}requires(r,c)$
	possesses	$\phi_{lose}possesses(a,c)$
change	achieves	$\phi_{change}achieves(r,g)$
	requires	$\phi_{change}requires(r,c)$
	possesses	$\phi_{change}possesses(a,c)$

objects greater than the size of the existing organization.

As there are primitive and complex predicates, predicates can be further defined as either *object* or *relationship* predicates. Object predicates represent objects of the organization such as goals, roles, capabilities or agents. Relationship predicates represent the link between two objects such as achieves, possesses or requires.

Definition. *Object Predicates* are defined as predicates where the property represents an object of the organization, such as an agent being added or a goal being deleted. An example of an object predicate is a goal addition, $\phi_{add}goal(g)$. This is a single predicate only involving an organization object. Examples of object predicates are shown in Table 2.

Definition. *Relationship Predicates* are defined as properties where a relationship between two objects is added, lost or altered. Relationship predicates can be primitive, as long as the objects in which they bind already exist in the organization. Object predicates may be complex as the object must collaborate with a relationship to connect to the organization. Object and relationship predicates will typically be combined in complex predicates. Relationship predicates such as when g_i is a subgoal of g_j is $\exists_{g_i,g_j}\phi_{add}subgoal(g_i,g_j)$ where the relationship can only exist if both g_i and g_j exist prior. Another relationship predicate g is achieved by r if $\exists_{g,r}\phi_{add}achieves(g,r)$ where the relationship can only exist if both g and r exist prior. Examples of relationship predicates are shown in Table 3. The general constraint for a relationship predicate is stated by $\exists_{x,x}\phi relationship(x,y)$.

3. Example

In this section, we will generate a simple, but not trivial, organization and then show a progression through a number of transitions. Each transition will be modeled with a specific predicate. After each transition the impact will be shown, with changes reflected from the previous predicate. To bring the organization example to life, we will use an example of a small organization organizing a workshop at a conference.

3.1. Organization description

To demonstrate how we can apply the formalizations of transition, properties and predicates, we will define an organization then apply the predicates to the organization. As each transition predicate is applied, a new organization state will be formed that will have differences from its predecessor. The example organization is shown in Fig. 6, the *Organization Graph*.

Once the organization structure is defined, the initial organization step will be shown. The initial organiza-

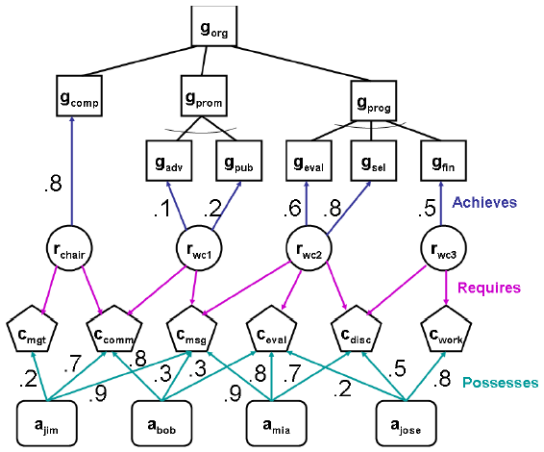

Fig. 6. Organization graph.

tion transitions from $O_0 \Rightarrow O_1$ and is a single transition process. Beyond state O_1 all transitions will be *reorganizations* where the organization transitions from some state to the next state, $O_n \Rightarrow O_{n+1}$.

3.2. Organization structure

The *goals*, *roles* and *capabilities* of the organization, $O_{example}$, must be defined first. In this example, we will not utilize organization *rules* for the sake of simplicity.

The organization in charge of managing the conference workshop will have the following elements:
The *goals* of the organization are:

$$G = \{g_{org}, g_{comp}, g_{prom}, g_{adv}, g_{pub}, g_{prog}, g_{eval}, g_{sel}, g_{fin}\}$$

where:

g_{org} : organize a conference workshop
g_{comp} : completing a workshop proposal
g_{prom} : promoting the workshop
g_{adv} : successfully advertising the workshop
g_{pub} : publishing the workshop details on the web
g_{prog} : preparing the program
g_{eval} : evaluating papers
g_{sel} : paper selection
g_{fin} : organizing the final workshop program

The *roles* of the organization are:

$$R = \{r_{chair}, r_{wc1}, r_{wc2}, r_{wc3}\}$$

r_{chair} : workshop chairperson
r_{wc1} : workshop committee person 1
r_{wc2} : workshop committee person 2
r_{wc3} : workshop committee person 3

The *capabilities* are:

$$C = \{c_{mgt}, c_{comm}, c_{msg}, c_{eval}, c_{disc} c_{work}\}$$

where:

c_{mgt} : management skills
c_{comm} : skill of communication
c_{msg} : organizing the workshop message
c_{eval} : knowledge to evaluate research
c_{disc} : discerning the best papers
c_{work} : ability to work with workshop chair

The achieves relationships defined are:

$achieves(r_{chair}, g_{org}) \rightarrow .8$
$achieves(r_{wc1}, g_{adv}) \rightarrow .1$
$achieves(r_{wc1}, g_{pub}) \rightarrow .2$
$achieves(r_{wc2}, g_{eval}) \rightarrow .6$
$achieves(r_{wc2}, g_{sel}) \rightarrow .8$
$achieves(r_{wc3}, g_{fin}) \rightarrow .5$

There are a number of *requires* relationships. Only the relationships resulting in *true* are listed. The *requires* relationships are:

$requires(r_{chair}, c_{mgt}) \rightarrow true$
$requires(r_{chair}, c_{comm}) \rightarrow true$
$requires(r_{wc1}, c_{comm}) \rightarrow true$
$requires(r_{wc1}, c_{msg}) \rightarrow true$
$requires(r_{wc2}, c_{msg}) \rightarrow true$
$requires(r_{wc2}, c_{eval}) \rightarrow true$
$requires(r_{wc2}, c_{disc}) \rightarrow true$
$requires(r_{wc3}, c_{disc}) \rightarrow true$
$requires(r_{wc3}, c_{work}) \rightarrow true$

Only the *subgoal* relationships where the result is *true* are listed. The *subgoal* relationships are:

$subgoal(g_{org}, g_{comp}) \rightarrow true$
$subgoal(g_{org}, g_{prom}) \rightarrow true$
$subgoal(g_{org}, g_{prog}) \rightarrow true$
$subgoal(g_{prom}, g_{adv}) \rightarrow true$
$subgoal(g_{prom}, g_{pub}) \rightarrow true$
$subgoal(g_{prog}, g_{eval}) \rightarrow true$
$subgoal(g_{prog}, g_{sel}) \rightarrow true$
$subgoal(g_{prog}, g_{fin}) \rightarrow true$

There are conjunctive *goal* relationships in this organization. Only relationships where the result is *true* are listed. The *conjunctive* goals are:

$conjunctive(g_{prom}) \rightarrow true$
$conjunctive(g_{prog}) \rightarrow true$

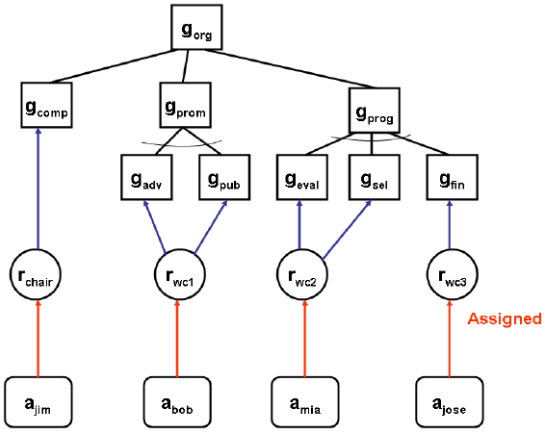

Fig. 7. Organization assignments: s_1.

3.3. Transition – initial organization

When an initial organization is computed, the *agents* must be added to the structural elements such as *goals*, *roles* and *capabilities*. The agents for our example organization are:

$$A = \{a_{jim}, a_{bob}, a_{mia}, a_{jose}\}$$

The *possesses*, *capable* and *assigned* relations will be stated. The *possesses* are:

$possesses(a_{jim}, c_{mgt}) \rightarrow .8$
$possesses(a_{jim}, c_{comm}) \rightarrow .7$
$possesses(a_{jim}, c_{msg}) \rightarrow .9$
$possesses(a_{bob}, c_{comm}) \rightarrow .8$
$possesses(a_{bob}, c_{msg}) \rightarrow .3$
$possesses(a_{bob}, c_{eval}) \rightarrow .3$
$possesses(a_{mia}, c_{msg}) \rightarrow .9$
$possesses(a_{mia}, c_{eval}) \rightarrow .8$
$possesses(a_{mia}, c_{disc}) \rightarrow .7$
$possesses(a_{jose}, c_{eval}) \rightarrow .2$
$possesses(a_{jose}, c_{disc}) \rightarrow .5$
$possesses(a_{jose}, c_{work}) \rightarrow .8$

For an *agent* to play a *role*, they must be capable. Only two are listed because there are many and the space is limited. The *capable* relations are:

$capable(a_{jim}, r_{wc1}) \rightarrow .8$
$capable(a_{bob}, r_{wc1}) \rightarrow .55$

The final step in the computation of an organization is the assignment of an *agent* to play a *role* that is charged with achieving a *goal*. The computational aspects of this process are described in previous work [5]. The *assigned* relations capturing the assignments computed for the initial organization state, shown in Fig. 7, are:

$assigned(a_{jim}, r_{chair}, g_{comp})$
$assigned(a_{bob}, r_{wc1}, g_{adv})$
$assigned(a_{bob}, r_{wc1}, g_{pub})$
$assigned(a_{mia}, r_{wc2}, g_{eval})$
$assigned(a_{mia}, r_{wc2}, g_{sel})$
$assigned(a_{jose}, r_{wc3}, g_{fin})$

where the assignments are based on which agent is most capable of playing a specific role. This result is arrived at by computing the numerical value of the relationships between the role and the agent.

3.4. Transition – reorganizations

This is a simple organization. From the definitions, it is apparent that there are not a great number of different combinations of *agents* to play the different *roles*. Based on the various relationships, a_1 is the only agent able to play the role of chairman r_1 which is the only role able to achieve the goal of completing a workshop proposal, g_1. This organization is constructed as a simple model to show the effects of predicate based transitions. Most organizations are much more complex, but this example shows the effects of transition properties being used as stimuli.

There are different effects of transitions. Some transitions will cause no structural change to the organization and externally appear as nothing has changed, while other, more drastic changes, will cause the organization to fail or go to an end state. In this section, we will show the effects of a *minimal*, then a *moderate* and finally, a *catastrophic* transition.

Definition. *Minimal Effect Transition* is a transition in which the structural elements are not added or subtracted from the organization graph. An example of minimal effect transition is the value change of an achieves relationship.

Definition. *Moderate Effect Transition* is a transition where structural or state elements are physically changed and the organization graph is impacted by addition or loss of objects or relationships. An example of moderate effect change is the loss of an agent from an organization.

Definition. *Catastrophic Effect Transition* occurs when the transition property outcome disables the organization from continuation and sends the organization to an end state. An example is the loss of an agent required to play a role to satisfy a goal that is critical to the organization.

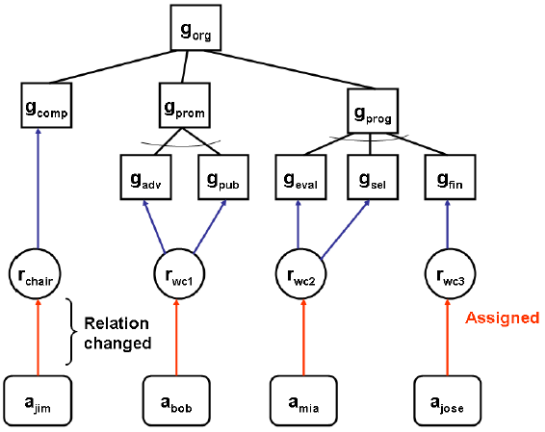

Fig. 8. Organization assignments: s_2.

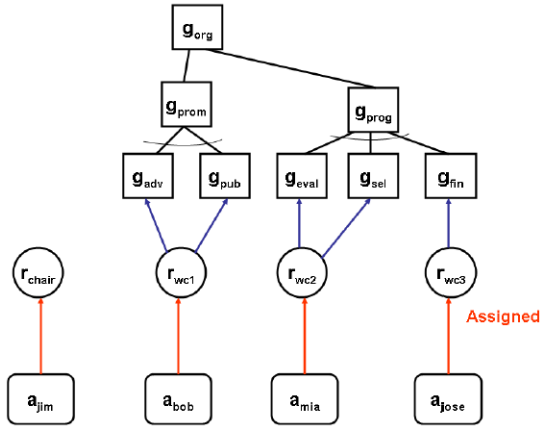

Fig. 9. Organization at s_3.

The first transition is a *minimal effect* transition. It is shown by the transition equation:

$$\delta(O_{example}, \phi_{change}(c_{mgt}, a_{jim}), s_1) \Rightarrow s_2$$

The effects of this transition reflect no change to the organization structure, as shown in Fig. 8. The relationship from the agent a_{jim} to the capability c_{mgt} do not force the organization to change structure or assignment. The organization must be computed, because there is the potential that a change in capability may change assignments of agents to roles. This is an internal effect, as the capability is changed by the agent involved, without external intervention.

The second transition is a *moderate effect* transition. It is shown by the transition equation:

$$\delta(O_{example}, \phi_{lose}(g_{comp}), s_2) \Rightarrow s_3$$

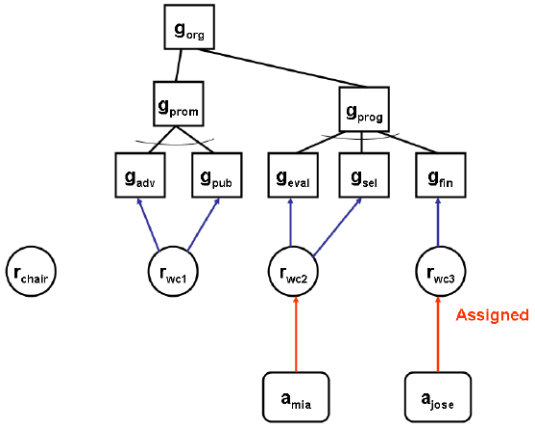

Fig. 10. Organization at s_4.

In this transition, with results shown in Fig. 9, the organization deletes one of the goals, g_{comp}. In real terms the goal, g_{comp} is no longer required and drops from the organization. Since g_{org}, the overall workshop organization does not conjunctively require g_{comp}, the proposal, a transition can be made. The agent a_{jim} is still attached to the workshop chairperson role, although neither is currently utilized in the organization. This changes the structure of the organization but has no effects of making the organization non-functional to complete the other goals. So while it changes the structure, it does not have a catastrophic effect. This is an external effect as the goals are not developed within the organization.

The third transition, as shown in Fig. 10, is a *catastrophic effect* transition. It is represented by the transition expression:

$$\delta(O_{example}, \phi_{lose}(a_{jim}) \wedge \phi_{lose}(a_{bob}), s_3) \Rightarrow s_4$$

This transition removes agents a_{jim} and a_{bob} from the structure. The chairperson and one of the program committee members are no longer available for the organization. While the loss of a_{jim}, the chairperson, has no real effect, the loss of a_{bob}, a program committee member, has dramatic consequences. This is an internal effect and is catastrophic to the organization. The organization cannot survive with the loss of both agents a_{jim} and a_{bob}. Even though g_{comp} is already deleted, which was dependent on r_{chair}, played by a_{jim}, the g_{prom} structure is the problem. The r_{wc1} satisfies both g_{adv} and g_{pub}. With both a_{jim} and a_{bob} gone, the capability no longer exists to play r_{wc1} and therefore no way to achieve either g_{adv} and g_{pub}, which are conjunctive. Because g_{prom} cannot be

achieved, the overall goal g_{org} cannot be achieved, which renders the organization ineffective.

3.5. Simulation of example

In this section, the system is implemented for evaluation. An initial organization is first built, followed by organization properties arriving and involving several reorganization states. The simulation is written in the Java programming language using the *JESS* package [9] to form the organization core. This combination is common for systems development, such as used by Sambasivam and Davies [22]. Each of the objects and relationships were modeled using logical constructs called *deftemplates* from the *JESS* language. The interactions and computational constructs were created using rules. Finally, the instantiations of each object and relationship were realized using facts. These elements compose the organization core of the simulation.

There are several reasons for utilizing *JESS* as the core of the organization implementation. The primary reason is the need for a rules engine which fit well within the design of the agent organization. *JESS* was designed as a rules engine architecture for use in mobile agents.

This rule engine is a part of a much larger agent organization architecture, which includes graphical user interfaces, special domain task interfaces and structural implementations of the agents. The complete system is written in *Java*, so the match of *Java* and *JESS* is natural for our implementation.

JESS is very fast to resolve rules, as shown based on work by Forgy [11] and results using the *Rete algorithm* produced by Albert [1]. The *Rete* algorithm [10] is the center of the *JESS* architecture and is adept in pattern matching as it has a memory of past results and does not recompute completely through each rule loop.

While there are a number of possible solutions, such as Prolog or CLIPS, to satisfy the rule engine requirement for our agent organization. *JESS*, based on the reasons above is the best fit and solution.

The goal of the simulation is to validate the model. A secondary goal is to show that the transitional processes of initial organization and reorganization are computationally viable, regardless of the size of the property that is added to the organization. The property may be a single object or relationship. It may, alternatively, be a complex property whose elements exceed the size of the current organization. Either of these are common and realistic organization scenarios, so the model must be capable of supporting these transition property inputs, if it is to be considered a viable model.

Each predicate of the organization model's structure and state can be directly represented by a template in *JESS*. For example, the structural templates are:

(deftemplate goal (slot goal))
(deftemplate goal (slot role))
(deftemplate goal (slot capability) (slot score))
(deftemplate achieves (slot role) (slot goal) (slot score))
(deftemplate requires (slot role) (slot capability))
(deftemplate subgoal (slot goal) (slot goal))
(deftemplate conjunctive (slot goal))

The state templates are:

(deftemplate agent (slot agent))
(deftemplate possesses (slot agent) (slot capability) (slot score))
(deftemplate assigned (slot agent) (slot role) (slot goal) (slot score))
(deftemplate capable (slot agent) (slot role) (slot score))

Each JESS deftemplate represent the structure for each fact in the organization core. So a ϕ property of adding a goal, $\phi_{add}(g_0)$, will exist in JESS as $(goal(goal\ g0))$ added by a rule in *JESS*. The set of JESS facts work with a set of rules that provide all action for the system. The premise is a pattern matching system where the structural and state elements match patterns to construct the organization graph. For example, given a $(goal(goal\ g0))$, $(goal(goal\ g1))$, and a $(subgoal\ x?\ y?)$, where $x?$ and $y?$ are variable, the result is a fact $(subgoal\ g1\ g0)$.

3.5.1. Initial organization – example

To begin any valid, non-trivial organization instance, we must add some initial objects and relationships. For this example, which illustrates the formation of a simple workshop program committee, we begin by adding 19 object and 25 relationship structural elements and 4 object and 12 relationship state elements. Once the objects are added, then we can compute the initial organization taking us from $state_0 \rightarrow state_1$. Tables 4 and 5 contain each of the primitive predicates and JESS fact statements for structural objects and relationships, respectively. Tables 6 and 7 contain each of the primitive predicates and JESS fact statements for state objects and relationships, respectively. When the initial organization algorithm is executed, the relationships between all facts are instantiated in the *organization core*.

The initial organization process yields additional facts, through pattern matching of the pre-defined structural and state rules. The format of this new fact is $(capable(slot\ agent)(slot\ role)(slot\ score))$. Only

Table 4
Initial organization – structural objects

Predicate	JESS Statement
$\phi_{add}goal(g_{org})$	(assert(goal org))
$\phi_{add}goal(g_{comp})$	(assert(goal comp))
$\phi_{add}goal(g_{prom})$	(assert(goal prom))
$\phi_{add}goal(g_{prog})$	(assert(goal prog))
$\phi_{add}goal(g_{adv})$	(assert(goal adv))
$\phi_{add}goal(g_{pub})$	(assert(goal pub))
$\phi_{add}goal(g_{eval})$	(assert(goal eval))
$\phi_{add}goal(g_{sel})$	(assert(goal sel))
$\phi_{add}goal(g_{fin})$	(assert(goal fin))
$\phi_{add}role(r_{chair})$	(assert(role chair))
$\phi_{add}role(r_{wc1})$	(assert(role wc1))
$\phi_{add}role(r_{wc2})$	(assert(role wc2))
$\phi_{add}role(r_{wc3})$	(assert(role wc3))
$\phi_{add}capability(c_{mgt})$	(assert(capability mgt))
$\phi_{add}capability(c_{comm})$	(assert(capability comm))
$\phi_{add}capability(c_{msg})$	(assert(capability msg))
$\phi_{add}capability(c_{eval})$	(assert(capability eval))
$\phi_{add}capability(c_{disc})$	(assert(capability disc))
$\phi_{add}capability(c_{work})$	(assert(capability work))

two capable facts resulting from this initial organization are listed and they are:

$(capable(a_{jim}, r_{wc1}, 0.8))$
$(capable(a_{bob}, r_{wc1}), 0.55)$

While there will typically be numerous capable agent to role options, the best are selected based on the best score or most capable of each agent to play a role to fulfill a goal. The format of this new fact is $(assigned(slot\ agent)(slot\ role)(slot\ goal))$. The assignments in this example are:

$(assigned(a_{jim}, r_{chair}, g_{comp}))$
$(assigned(a_{bob}, r_{wc1}, g_{adv}))$
$(assigned(a_{bob}, r_{wc1}, g_{pub}))$
$(assigned(a_{mia}, r_{wc2}, g_{eval}))$
$(assigned(a_{mia}, r_{wc2}, g_{sel}))$
$(assigned(a_{jose}, r_{wc3}, g_{fin}))$

After the initial organization transition is complete, the organization core contains 60 initial facts, with 5 additional capable and 5 additional assigned facts generated by matching. Even a small organization, such as this with 70 facts, is not trivial to compute. The time it takes to compute this initial organization is shown in Table 9 for state 0. After this step the organization is at state 1. Table 9 also shows the number of objects and relationships after the transition property is considered.

Table 5
Initial organization – structural relationships

Predicate	JESS Statement
$\phi_{add}subgoal(g_{comp}, g_{org})$	(assert(subgoal(comp,org))
$\phi_{add}subgoal(g_{prom}, g_{org})$	(assert(subgoal(prom,org))
$\phi_{add}subgoal(g_{prog}, g_{org})$	(assert(subgoal(prog,org))
$\phi_{add}subgoal(g_{adv}, g_{prom})$	(assert(subgoal(adv,prom))
$\phi_{add}subgoal(g_{pub}, g_{prom})$	(assert(subgoal(pub,prom))
$\phi_{add}subgoal(g_{eval}, g_{prog})$	(assert(subgoal(eval,prog))
$\phi_{add}subgoal(g_{sel}, g_{prog})$	(assert(subgoal(sel,prog))
$\phi_{add}subgoal(g_{fin}, g_{prog})$	(assert(subgoal(fin,prog))
$\phi_{add}conjunctive(g_{prom})$	(assert(conjunctive(prom))
$\phi_{add}conjunctive(g_{prog})$	(assert(conjunctive(prog))
$\phi_{add}achieves(r_{chair}, g_{comp})$	(assert(achieves(chair,comp,0.8)))
$\phi_{add}achieves(r_{wc1}, g_{adv})$	(assert(achieves(wc1,adv,0.1)))
$\phi_{add}achieves(r_{wc1}, g_{pub})$	(assert(achieves(wc1,pub,0.2)))
$\phi_{add}achieves(r_{wc1}, g_{eval})$	(assert(achieves(wc2,eval,0.6)))
$\phi_{add}achieves(r_{wc2}, g_{sel})$	(assert(achieves(wc2,sel,0.8)))
$\phi_{add}achieves(r_{wc3}, g_{fin})$	(assert(achieves(wc3,fin,0.5)))
$\phi_{add}requires(r_{chair}, c_{mgt})$	(assert(requires(chair,mgt)))
$\phi_{add}requires(r_{chair}, c_{comm})$	(assert(requires(chair,comm)))
$\phi_{add}requires(r_{wc1}, c_{comm})$	(assert(requires(wc1,comm)))
$\phi_{add}requires(r_{wc1}, c_{msg})$	(assert(requires(wc1,msg)))
$\phi_{add}requires(r_{wc2}, c_{msg})$	(assert(requires(wc2,msg)))
$\phi_{add}requires(r_{wc2}, c_{eval})$	(assert(requires(wc2,eval)))
$\phi_{add}requires(r_{wc2}, c_{disc})$	(assert(requires(wc2,disc)))
$\phi_{add}requires(r_{wc3}, c_{disc})$	(assert(requires(wc3,disc)))
$\phi_{add}requires(r_{wc3}, c_{work})$	(assert(requires(wc3,work)))

Table 6
Initial organization – state objects

Predicate	JESS Statement
$\phi_{add}agent(a_{jim})$	(assert(agent jim))
$\phi_{add}agent(a_{bob})$	(assert(agent bob))
$\phi_{add}agent(a_{mia})$	(assert(agent mia))
$\phi_{add}agent(a_{jose})$	(assert(agent jose))

3.5.2. Reorganizations – example

In this example, there are 3 organization properties sent to the organization. The predicate and assertions are sown in Table 8. Each results in a transition as previously described and each will require a recomputation of the organization to determine if a better assignment list exists, each will have different outcomes. The time for each reorganization is listed in Table 9.

The first reorganization is relatively minor but does require a recomputation. The number of object and relationship facts are not altered.

The second reorganization, based on the property of losing goal g_{comp}, has the effect to force recomputation, but the organization can continue. The net effect is the loss of g_{comp} which also causes the loss of all relationships connected to or depending on g_{comp}. That results in the loss of an achieves, possesses and assigned relationships.

Table 7
Initial organization – state relationships

Predicate	JESS Statement
$\phi_{add}possesses(a_{jim}, c_{mgt})$	(assert(possesses(jim,mgt,0.2)))
$\phi_{add}possesses(a_{jim}, c_{comm})$	(assert(possesses(jim,comm,0.7)))
$\phi_{add}possesses(a_{jim}, c_{msg})$	(assert(possesses(jim,msg,0.9)))
$\phi_{add}possesses(a_{bob}, c_{comm})$	(assert(possesses(bob,comm,0.8)))
$\phi_{add}possesses(a_{bob}, c_{msg})$	(assert(possesses(bob,msg,0.3)))
$\phi_{add}possesses(a_{bob}, c_{eval})$	(assert(possesses(bob,eval,0.3)))
$\phi_{add}possesses(a_{mia}, c_{msg})$	(assert(possesses(mia,msg,0.9)))
$\phi_{add}possesses(a_{mia}, c_{eval})$	(assert(possesses(mia,eval,0.8)))
$\phi_{add}possesses(a_{mia}, c_{disc})$	(assert(possesses(mia,disc,0.7)))
$\phi_{add}possesses(a_{jose}, c_{eval})$	(assert(possesses(jose,eval,0.2)))
$\phi_{add}possesses(a_{jose}, c_{disc})$	(assert(possesses(jose,disc,0.5)))
$\phi_{add}possesses(a_{jose}, c_{work})$	(assert(possesses(jose,work,0.8)))

Table 8
Reorganizations

State	Predicate ϕ	JESS Statement
1	$\phi_{change}capability(c_{mgt}, a_{jim})$	*rule change*
2	$\phi_{lose}goal(g_{prom})$	(retract(goal prom))
3	$\phi_{lose}agent(a_{jim})\wedge$	(retract(agent jim))
	$\phi_{lose}agent(a_{bob})$	(retract(agent bob))

Table 9
Example organization times

State	Object	Relationship	Time
0	23	37	0.020446174
1	23	47	0.009058972
2	23	47	0.004945321
3	22	44	∞

The last transition, beginning with state 3, results in an infinite time ∞ to recompute, as the organization has catastrophic effect transition. The loss of agents a_{jim} and a_{bob} also causes a loss of numerous other relationships bound to a_{jim} and a_{bob}. This property disallows continuation of the organization, as all required goals cannot be completed. This results in the incapability to transition to a new valid state and produce a transition time.

The simulation, described in this section, is further used in the next section to provide an in depth evaluation of transition properties and the ability to change organizations. While this simulation was small in nature, the following simulations will feature larger organization instances.

4. Evaluation

In this section, we extend the implemented organization to show the reaction to larger numbers of organization properties being sent to the organization and how the various arrangements affect the organization as it grows and shrinks, due to the changes. There are a number of ways an organization may change. The important factor is how long it takes to recompute if it is indeed possible to recompute. As shown in the workshop example, there are organization properties that will force the organization to a premature exit, in which all goals may not be accomplished.

Transition has a progression of requirements with a dependency chain. The first is the ability to transition, meaning the objects and relationships exist in the organization to form a new organization instance. Without the basic ability to transition to a new state, further requirements are not evaluated. Secondly, even if an organization can transition, it must be able to recompute in a time which fits the domain problem in which it is working. For a reorganization, where there is no time requirement, it does not matter how long the organization takes to recompute. For the application of the organization model where the time to recompute must be minimized, it is critical to understand the effects of recompute time based on the organization size.

As time to recompute is important to consider, we are compelled to consider the time required to recompute organizations that get very large. It is important to consider the shape of data that results from large incorporation of transition properties into an existing organization. In this section, we will simulate initial organization and the reorganization with large sets of properties. Then analysis will be made, based upon the size and type of properties used.

4.1. Initial organization

While the process of initial organization will only occur once in the life of any organization, we must look at the computational cost of instantiating the organization. In simulating the initial organization, the minimum initial organization, that is valid, must have at least one goal, role, capability and agent. There also must exist an achieves, requires and possesses relationship between the objects. While there is a single initial organization transition, the number of predicates involved will vastly change the time to initially reorganize. If there is a minimal organization of 4 objects and 3 relationships, the time will be quite small. If a large number of objects and relationships are included in the complex predicate as input to initial organization, the time can be quite large. The results form the baseline from a comparative result, on a similar struc-

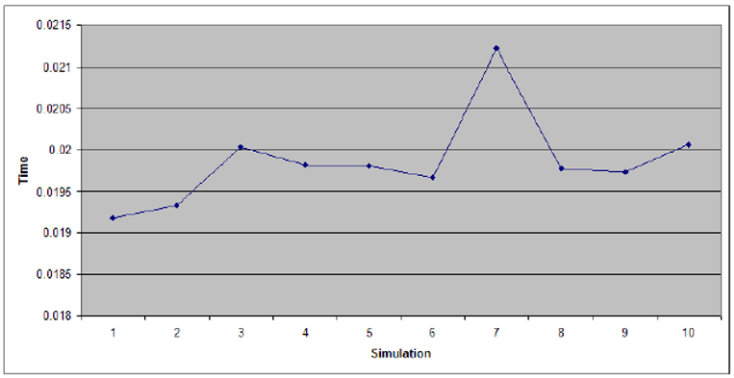

Fig. 11. Initial organization time.

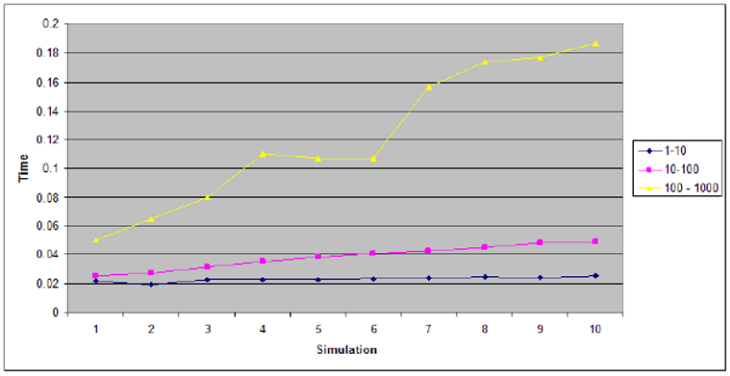

Fig. 12. Initial organization comparison on ranges of predicates.

tural model [23] and a previous minor result with this model [18].

Table 10 shows 10 configuration simulation states of an initial organization computation. The minimum number of objects and relationships is 7 comprised of 4 objects and 3 relationship predicates. This is shown in simulation 1. Each of the following simulations adds either one object or one relationship prior to computing the initial organization state. Figure 9 shows the general trend as the time generally increases from an initial organization with 7 predicates to a larger initial organization computation with 16 predicates, as shown in simulation 10 of Table 10.

Figure 12 compares the time to initially organizations on three ranges of differing complex predicate sizes. The simulations were executed in groups of 10. For this comparison, each simulation will contain 4 object predicates and 4 relationship predicates. The lowest plot shows the time to compute an initial organization with a single group of 8 simple predicates, representing a single simulation. Each subsequent simulation, adds 8 predicates. For example, the second simulation will contain 16 simple predicates, the third simulation will contain 24 simple predicates all the way to the tenth simulation which will contain 80 simple predicates. The middle plot shows the simulations with the initial complex predicate containing 10 sets of 8 simple object and relationship predicates. The simulations then step by 10 sets per simulation until the final simulation contains 100 groups of 8 simple predicates

Table 10
Initial organization

Simulation	Object	Relationship	Time
1	4	3	0.019182323
2	5	3	0.019335698
3	5	4	0.020033945
4	6	4	0.019820116
5	6	5	0.019809222
6	7	5	0.019670378
7	8	5	0.021213310
8	8	6	0.019777095
9	8	7	0.019732396
10	9	7	0.020067076

each. This shows the time to initially compute an organization with a maximum of 800 elements. The upper plot steps from 100 sets to 1000 sets, giving the time to compute large initial organizations. In this set, the time to initially compute an initial organization containing 4000 objects and 4000 specified relationships is 0.187157 seconds. This organization will also compute additional relationships where possible. The results indicate the capability of the initial organization algorithm to compute a very large initial organization, in minimal time.

4.2. Reorganization

Reorganization requires that a single transition property ϕ be present as a catalyst to enable execution. In the model, transition properties are implemented as predicates. The predicates can be represented as either simple predicates or complex predicates. The reorganization process will be simulated from a number of perspectives. The first view will be that of adding objects and deleting objects.

4.3. Simple properties

The first evaluation will be using simple properties for each transition. This is defined by a single simple predicate contained within each ϕ transition property. In this section, we look at adding simple object predicates followed by the use of complex predicates. Within using simple predicates, we look at adding objects and relationships individually, adding objects and relationships mixed together and deleting simple predicates. Then, we look at the organization transition using complex predicates of differing size up to transition properties that contain more objects and relationships which are larger than the original organization to which they are added. For each of these simulation experiments, the transitions are measured over the time it takes to compute the new state, from the existing state. Because the transition processes are computationally intensive, the time required to reach a new state is critical in creating valid, realistic organization models and corresponding transitional algorithms.

4.3.1. Adding objects

While it is legal to add an object to an organization, if subsequent relationships do not bind the object to other objects within the structure, the object has no bearing on the outcome. It will have an effect on the reorganization computation process.

Table 11
Reorganization: *Objects only*

State	Object	Relationship	Time
1	4	3	0.019421184
2	5	3	0.009218770
3	6	3	0.004433245
4	7	3	0.004319823
5	8	3	0.004259200
6	9	3	0.004261994
7	10	3	0.004357537
8	11	3	0.004076216
9	12	3	0.003963074
10	13	3	0.003939887
11	14	3	0.003874795
12	15	3	0.003565257
13	16	3	0.003514134
14	17	3	0.003386185
15	18	3	0.003680914
16	19	3	0.004629080
17	20	3	0.003534248
18	21	3	0.003352102
19	22	3	0.003302934
20	23	3	0.003168279
21	24	3	0.003366908
22	25	3	0.003882058
23	26	3	0.003133918
24	27	3	0.003138387
25	28	3	0.003203480

Table 11 illustrates the data for simulation of adding strictly objects to an organization. The initial simulation 1, which represents the initial organization, is higher than the remainder of the simulations. There are 3 relationship predicates, required to form a minimal organization, along with 4 objects. In addition, each subsequent predicate is an object and is not related to any other object with a relationship predicate. Figure 13 shows the relationship of each simulation. Even though the number of predicates grows, the trend is reasonably flat, even decreasing minimally.

4.3.2. Adding relationships

We cannot, by definition, add relationships, without having objects that correspond to the relationships, so in this simulation, objects and relationships will be added one at a time. If relationships are added without corresponding objects, they result in orphan predicates that cannot participate in the organizational structure.

Table 12 illustrates the data when adding objects and relationships for the objects. In this simulation, objects

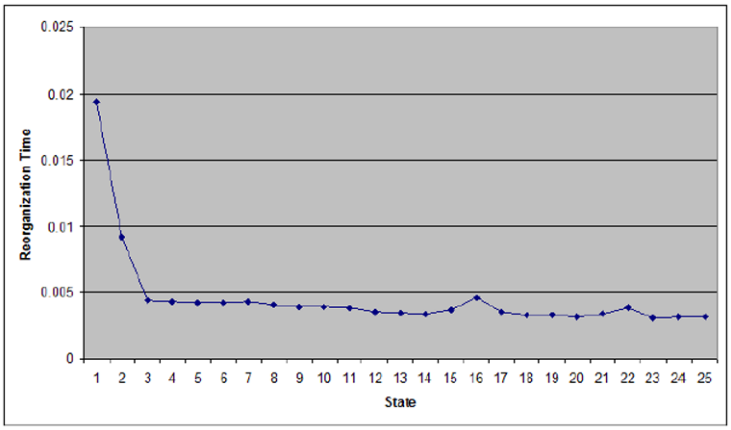

Fig. 13. Reorganization: Objects only.

Table 12
Reorganization: *Objects and relationships*

State	Object	Relationship	Time
1	4	3	0.020193069
2	5	3	0.011930008
3	6	3	0.004773232
4	7	3	0.004273728
5	8	3	0.004492470
6	8	4	0.004786642
7	8	5	0.004592204
8	8	6	0.004573765
9	9	6	0.005894045
10	10	6	0.004533816
11	11	6	0.004406147
12	12	6	0.003848254
13	12	7	0.004614833
14	12	8	0.004278197
15	12	9	0.005519975
16	13	9	0.004215340
17	14	9	0.004423467
18	15	9	0.004098845
19	16	9	0.003946032
20	16	10	0.004273448
21	16	11	0.005125791
22	16	12	0.004109460
23	17	12	0.004183772
24	18	12	0.004255848
25	19	12	0.004841118
26	20	12	0.003981233
27	20	13	0.004317588
28	20	14	0.004254172
29	20	15	0.004432407

and relationships added relate to each other. No orphan objects or relationships are considered or added. The initial organization simulation begins with 4 objects and 3 relationships, the minimal organization. Then, each simulation adds 4 objects then three relationships to tie the objects together. The last simulation contains 20 objects and 15 relationships. While the organization transition time decreases from the initial organization to the next set of reorganizations, the overall time to reorganize to the last simulation is reasonably flat. There is some variation as the organization grows in size. This is reasonable for a small organization.

Figure 15 shows the trend of time for a large number of reorganizations. In this set of simulations, 390 reorganizations were completed adding even numbers of objects and relationships. The trend is fairly flat until approximately 40 reorganizations. With the relations added, there are additional relationships generated with the existing properties. In this case, the end result is the initial organization and 390 additional reorganizations. During the reorganizations, yielding 390 rules, 199 additional relationships were constructed, yielding a total of 509 total objects and relationships. The toggling of times is explained by additional new relationships being generated and computed as planned relationships are added. Time for the last reorganization is 0.109313614 seconds.

4.3.3. Deleting objects

When objects are deleted from the organization, there is a compound effect on the relationships bound to the objects. Since a relationship requires two objects, the removal of the two objects also requires removal of any relationships currently bound to that object. Adding 390 objects and relationships, then deleting 390 objects and relationships will return the organization state to the original minimal organization.

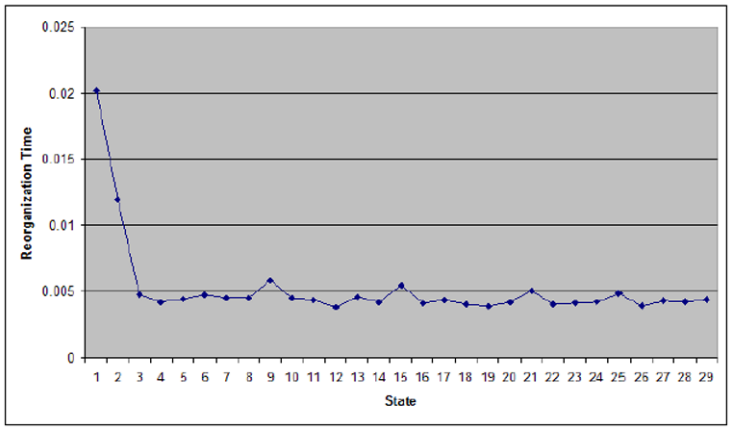

Fig. 14. Reorganization: Objects and relationships.

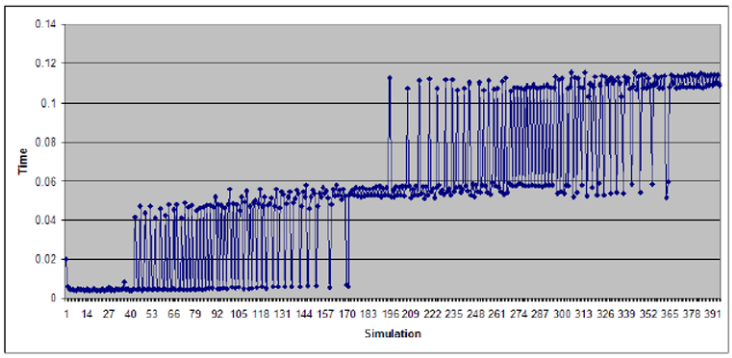

Fig. 15. Reorganization: Adding objects and relationships.

4.3.4. Adding complex predicates

Complex predicates consist of a set of simple predicates, of any size. In Table 13, the number of objects and relationships versus time is described. In this simulation, an initial minimal organization of 4 objects and 3 relationship predicates is computed on the first organization. The initial organization also adds a global goal, for which all other goals are subgoals. For each additional transition, sets of 4 objects and 4 relationships are added. The size of the organization grows quickly.

Figure 17 show the curve for adding complex predicates, of size 8, to an existing organization. The increase in transition time, each state, rises slightly upward. The initial organization, which includes objects and relationships, decreases to a lower time on the reorganization processes.

Figure 18 shows the simulation of an initial organization with 4 objects and 4 relationships, then increasing by 4 objects and 4 relationships through 125

Table 13
Reorganization – complex predicates

State	Object	Relationship	Time
1	4	4	0.019922364
2	8	8	0.005504051
3	12	12	0.006399417
4	16	16	0.007452902
5	20	20	0.006987201
6	24	24	0.007305118
7	28	28	0.007605436
8	32	32	0.007381943
9	36	36	0.008508065
10	40	40	0.007355124
11	44	44	0.008310833
12	48	48	0.007521347
13	52	52	0.008910071
14	56	56	0.007806020
15	60	60	0.011343062
16	64	64	0.008287087

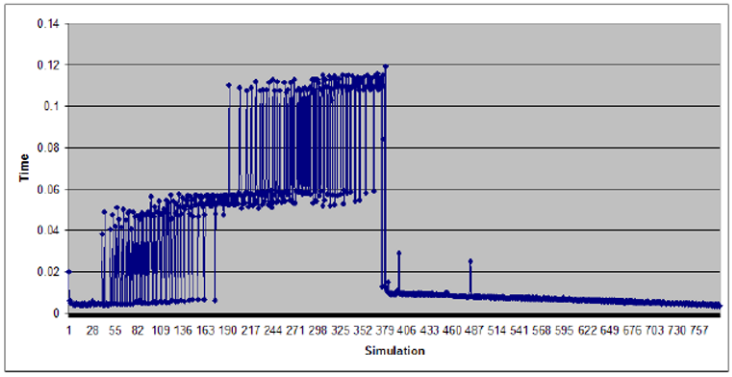

Fig. 16. Reorganization: Adding and deleting objects and relationships.

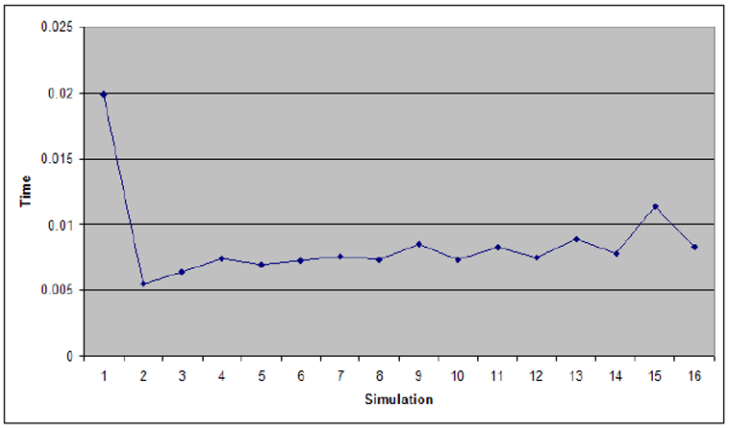

Fig. 17. Reorganization: Complex predicates.

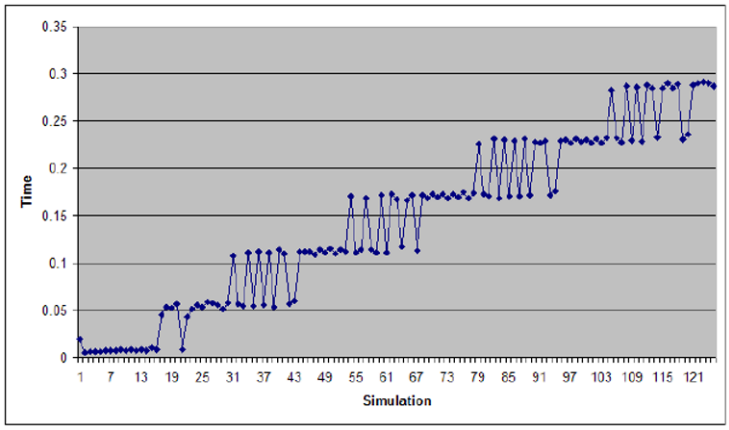

Fig. 18. Reorganization: Complex predicates – 1000 maximum.

states until the organization has a size of 1000 total added objects and relationships. The total number of relationships is actually larger. The upward trend stair steps at certain points, when the addition of new objects forces a set of new relationships to be asserted, which were not included in the predicate. The nature of these new assertions are assignments and capable relationships.

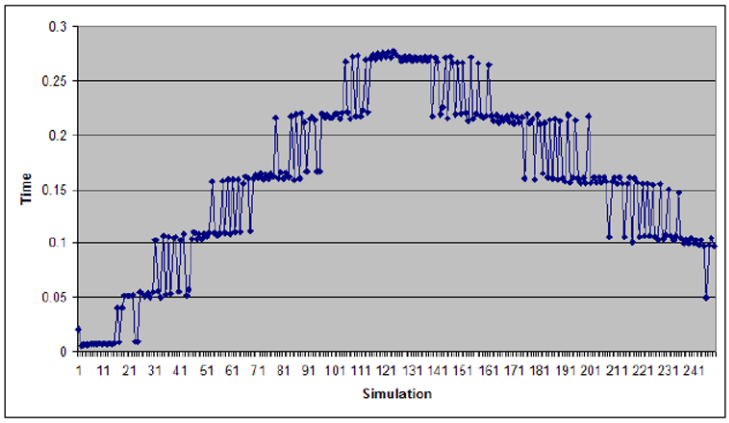

Fig. 19. Reorganization: Complex predicates – add and delete sets.

4.3.5. *Deleting complex predicates*

Figure 19 shows the simulation of an initial organization with 4 objects and 4 relationships, then increasing by 4 objects and 4 relationships until it has a size of 1000 total added objects and relationships, which is a continuation of the simulation shown in Fig. 18. Once the organization is at a size of 1000 complex predicates deleting objects and relationships, at the same rate of 4 objects and 4 relationships occur. The trend to recompute the organization progresses downward. At the end, the time is not symmetric to the starting times, as there are still relations which exist in organization as a by product of the complex add predicates.

4.3.6. *Adding complex predicates equal to existing organization*

Often, if an organization is forced to merge with another organization, then organization may grow by a non-trivial size. For example, if two equally sized organizations merged and became one, each organization doubles its original size. In this simulation, we studied the time effects of send predicates, to the organization, that double the size of the organization each transition. Table 14 shows the data used in this simulation. The predicate indicates the size of the predicate where each predicate has 4 new objects and 4 new relationships to add. After the initial organization, the predicate size doubles each time. The only caveat to this is a subgoal relationship was added in the initial organization, which slightly lessens the doubling factor for relationships. As the organization grows, the numbers of internally added relationships, capable and assigned, also grows. The added components totals 8193 objects and relationships.

Table 14
Reorganization – doubling predicates

Simulation	Pred	Object	Relationship	Time
0	1	4	5	0.030100753
1	1	8	9	0.01913232
2	2	16	17	0.012051531
3	4	32	33	0.012455214
4	8	64	65	0.019561704
5	16	128	129	0.021259406
6	32	256	257	0.034361909
7	64	512	513	0.053398966
8	128	1024	1025	0.254627893
9	256	2048	2049	0.587807947
10	512	4096	4097	1.199390984

The time curve, shown in Fig. 20, indicates the doubling of the organization takes an greatly upward trend as the organization transitions from a size of 1025 to the next size of 2049. Future evaluation will test the limits of organization size. While an organization size of 4096 objects is not trivial, larger organizations do exist. Recomputing an organization of size 8193 in 1.199 seconds is reasonable. Most organization transitions will not involve all objects and relationships during a transition.

4.3.7. *Deleting complex predicates larger than existing organization*

As organizations may grow at levels doubling the current size of the organization, or more, they may also shrink at the same or greater rates. If a company splits off divisions, the company may be only half the size of its previous organization. In this simulation, the organization will decrease by half with each transition property. Table 15 shows the size of the pred-

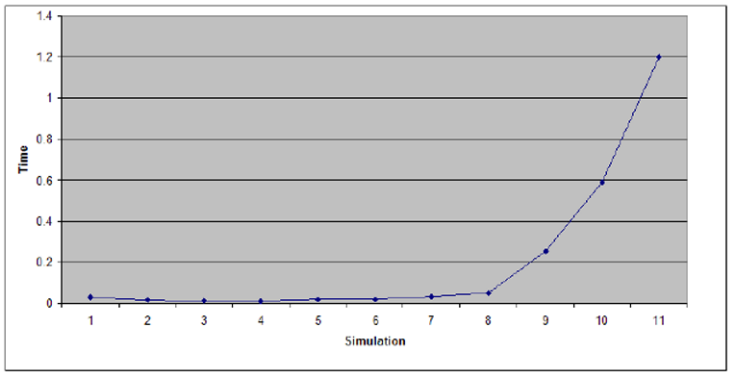

Fig. 20. Reorganization: Complex predicates – double size.

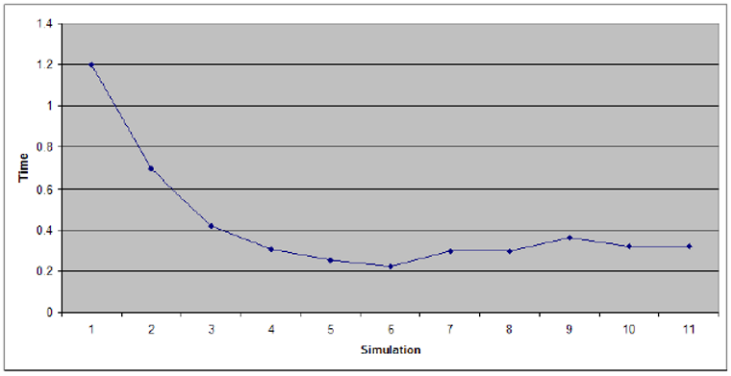

Fig. 21. Reorganization: Complex predicates – reduce by half.

icates, number of predicate object and relationships, and transition time. Figure 21 shows the transition time from state to state. As the time increased rather quickly as the organization doubled, the time to recompute decreases quickly as the organization size halves each transition. The transition time does toggle after a steady decrease.

5. Conclusions

The ability to formalize the basic processes of organization transition is fundamental to the understanding, capture and application of evolved biological organization models to multiagent systems. The process of transition, organization or reorganization, is seemingly very simple, but the reality is of complexity.

In the paper, we have shown the theoretical premises of translating agent organization properties into formal predicates. These predicates can then be applied to specific transition processes for instances of agent organizations. There are two categories of transition property predicates, primitive and complex, which can represent any formal property of agent transition. A simple transition predicate represents a change in a single transition property. A complex predicate is a logical set of simple predicates joined by conjunctive or disjunctive relationships.

Table 15
Reorganization – halving predicates

Simulation	Pred	Object	Relationship	Time
1	512	4096	4097	1.199390984
2	256	2048	2049	0.694899212
3	128	1024	1025	0.418405691
4	64	512	513	0.308071557
5	32	256	257	0.256492654
6	16	128	129	0.225215216
7	8	64	65	0.30098574
8	4	32	33	0.300371695
9	2	16	17	0.362175081
10	1	8	9	0.322663076
11	1	4	5	0.322274759

It is important to validate the concept of transition properties as they pertain to instantiated organizations. To satisfy the requirement of validation, the organization, properties and simulation environment were developed in a combination of Java and JESS. Simulations which evaluate all important perspectives of transition were executed for small to rather substantial organizations. The simulations show organization properties to be a viable and flexible concept to introduce change, and therefore transition, to organizations.

The properties and predicates have been created to be flexible in approach. The reason for this is the applicability to any organization model. Any organization model based on logic can use this basic set of predicates to model the transition processes associated with initial organization or reorganization processes. Our design is meant to work with any number of organization models.

References

[1] Luc Albert, Average Case Complexity Analysis of Rete Pattern-match algorithm and Average Size of Join in Databases, Rapports de Reserche, No. 1010, Institut National de Reserche en Informatique and Automatique, Rocquencourt, France, April, 1989.

[2] P.M. Blau and W.R. Scott, *Formal Organizations*, Chandler, San Francisco, CA, 1962, pp. 194–221.

[3] K.M. Carley, Organizational Adaptation, *Annals of Operations Research* **75** (1998) 25–47.

[4] A.R. Costa and Y. Demazeau, Towards a formal model of multi-agent systems with dynamic organizations, in: Proceedings of the International Conference on Multi-Agent Systems, MIT Press, Kyoto, Japan, 1996.

[5] Scott DeLoach and Eric Matson, An Organizational Model for Designing Adaptive Multiagent Systems, in: Agent Organizations: Theory and Practice at the National Conference on Artificial Intelligence (AAAI-04), San Jose, CA, July 25–29, 2004.

[6] Scott A. DeLoach, Walamitien Oyenan and Eric T. Matson, A Capabilities-Based Model for Artificial Organizations, *Journal of Autonomous Agents and Multiagent Systems* **16**(1) (February 2008) 13–56.

[7] Virginia Dignum, A model for organizational interaction: based on agents, founded in logic [S.l.] : [s.n.], Thesis, Proefschrift Universiteit Utrecht, 2003.

[8] Virginia Dignum, Liz Sonenberg and Frank Dignum, Dynamic Reorganization of Agent Societies, in: Proceedings of CEAS: Workshop on Coordination in Emergent Agent Societies (ECAI 2004), Valencia, Spain, 22–27 September 2004.

[9] Ernest Friedman-Hill, *JESS in Action: Rule Based Systems in Java*, Manning Publications, Inc., Greenwich Connecticut, USA, 2003.

[10] Charles Forgy, Rete: A Fast Algorithm for the Many Pattern, Many Object Pattern Match Problem, *Artificial Intelligence* **19** (1982) 17–37.

[11] Charles Forgy, Allen Newell and Anoop Gupta, High-Speed Implementation of Rule-Based Systems, *ACM Transactions on Computer Systems* **7**(2) (May 1989) 119–146.

[12] Norbert Glaser and Philippe Morignot, The Reorganization of Societies of Autonomous Agents, in: Multi-Agent Rationality – Proceedings of the 8th European Workshop on Modeling Autonomous Agents in a Multi-Agent World (MAAMAW '97), Magnus Boman and Walter Van de Velde, eds., Springer-Verlag, Berlin, Germany, 1997.

[13] M. Hoogendoorn, C.M. Jonker, M.C. Schut and J. Treur, Modeling Centralized Organization of Organizational Change, *Computational and Mathematical Organization Theory* **13** (2007) 147–184.

[14] M. Hoogendoorn, C.M. Jonker and J. Treur, Redesign of Organizations as a Basis for Organizational Change, in: COIN 2006 Workshops, P. Noriega et al., eds., LNAI 4386, 2007, pp. 46–62.

[15] Bryan Horling and Victor Lesser, A Survey of Multi-Agent Organizational Paradigms, *The Knowledge Engineering Review* **19**(4) (2005) 281–316, Cambridge University Press.

[16] Eric Matson and Scott DeLoach, Formal Transition in Agent Organizations, in: IEEE International Conference on Knowledge Intensive Multiagent Systems (KIMAS '05), Waltham, MA, April 18–21, 2005.

[17] Eric Matson and Raj Bhatnagar, Properties of Capability Based Agent Organization Transition, in: 2006 IEEE/WIC/ACM International Conference on Intelligent Agent Technology (IAT-2006), Hong Kong, December 18–22, 2006.

[18] Eric Matson and Raj Bhatnagar, Knowledge Sharing Between Agents in a Transitioning Organization, Coordination, Organization, Institutions and Norms in Agent Systems (COIN), in 2007 Multi-agent Logics, Languages and Organizations Federated Workshops (MALLOW '07), Durham University, Durham, U.K., September 3–7, 2007.

[19] Franco Zambonelli, Nicholas R. Jennings and Micheal Wooldridge, Organisational rules as an abstraction for the analysis and design of multi-agent systems, *Int J. of Software Engineering and Knowledge Engineering* **11**(3) (2001) 303–328.

[20] Jomi Hubner, Jaime Sichman and Olivier Boissier, MOISE+: Towards a structural, functional and deontic model for MAS organization, in: Proceedings of the 1st International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS '02), 2002.

[21] Andrea Omicini, Alessandro Ricci and Mirko Viroli, An Algebraic Approach for Modelling Organization, Roles and Context in MAS, *Journal of Applicable Algebra in Engineering, Communications and Computing* **16**(2–3) (2005) 151–178.

[22] Samuel Sambasivam and Chris Davies, SMARTVIEW – An Intelligent Expert System Tool using JAVA and JESS Framework, *Information Systems Education Journal* **3**(31) (2005), http://isedj.org/3/31/, ISSN: 1545-679X.

[23] Christopher Zhong and Scott A. DeLoach, An Investigation of Reorganization Algorithms, in: Proceedings of the International Conference on Artificial Intelligence (IC-AI'2006), June 2006, Las Vegas, Nevada, CSREA Press, 2006.