An Agent-Based Simulator to Analyze Business Office Activities

Yukinao Kenjo
Tokyo Institute of Technology

Abstract

This paper presents an agent-based simulation model to analyze office persons' activities. The objectives of the model are 1) to provide managers with decision aids to improve the office work and 2) to develop scalable methods to implement large scale agent-based simulation systems. As the first step of the research, in this paper, we focus on managers' roles to improve the office performance. Experiments on the office activities with and/or without managers' roles have been carried out.

1. Introduction

This paper is concerned with the development of an agent-based simulator, which aims at analyzing office activities to uncover the characteristics of efficient and effective organizational structures of a firm. In our simulator, office activities in the organization are modeled in a bottom-up manner. This means that we define the roles of office persons, the places (spots) they work, and the interaction among the agents and spots. In the simulation, we will focus on the roles of managers' tasks and the different organizational structures: flat and/or hierarchical.

Before developing the simulator, we have monitored one week activities of a small office with half dozen of office people and replicated them using an agent-based simulator written in SOARS [Tanuma 2005]. Based on the results, the objectives of the paper are to analyze the characteristics of flat and hierarchical organization under various managers; second, we will explore the scalability of the previous model to real-scale large organizations. Based on this, in the following sections, we will explain the model description and simulation results with managers in a hierarchical structure and empowered personnel without managers in a flat organization. Future work on the simulator is also reported in the concluding remarks.

2. Related Work

There have been several studies which analyze the organization activities by computer simulation; Cyert and March [Cyert 1963] constructed an organizational behavior model in FORTRAN language. The garbage can model is well known that they cope with contingent decision making in an organization [Cohen 1972]. In addition, in the literature of computational organization theory, Carley and Gasser [Carley 1995, 1999] claim that the information processing model of Simon [Simon 1995] is enhanced in many ways and that systematic phenomena observed in a model will help to describe the differences of the organizational effectiveness. An organization is supposed to be operated, to be designed, and to be easily applied to the research and the practice of actual organizational behaviors. Our research is based on such a context. However, the KISS (Keep It Simple, Stupid!) principle by Axelrod [Axelrod 1997] is not sufficient to implement analytical tools to cope with practical office activities. Therefore, we have decided to develop an agent-based simulator for office activities.

3. Problem Description

3.1 Outline of the Model

We suppose an organization consisting of managers (defined by *m*) and staffs (defined by *n*). The purpose of the organization is to handle the assigned work efficiently, and each constituent member works his/her own best. All kinds of tasks are given to the organization every certain period. These tasks are stacked on waiting-lists in the organization, and each agent tries to finish one's duties to the full as a manager or a staff. Each task has its own volume and the deadline, and is expected to be finished by its deadline. The performance of the organization is evaluated by the number of finished and unfinished tasks when the simulation ends.

Table 1 Definition of Objects

Object	Attribute
Task	*Id (int)*
	volume (int)
	deadline (int)
Team	*waiting-list (List<Task>)*
	processing-tasks (Set<Task>)
	finished-tasks (Set<Task>)
	deadline-passed-tasks (Set<Task>)
Staff	*ability (Map<Task.id, skill>)*
	busy (boolean)
	vision (Set<Task>)
	decisionRule (Rule)
Manager	*vision (Set<Task>)*
	allocationRule (Rule)
TaskPool	*pool (Set<Task>)*

3.2 Objects

Table 1 describes the main objects and their attributes used in this model. *Team* object represents the organization, and the objects, *Staff* and *Manager*, stand for staffs and managers respectively. *Task* object is one of the various works that the organization processes, and it is generated in *TaskPool* object.

- *Task*

Task object has three attributes, i.e. *id*, *volume*, and *deadline*, each of which stands for its id variety, volume, and deadline.

- *Team*

Team object has four attributes, namely *waiting-list*, *processing-tasks*, *finished-tasks* and *deadline-passed-tasks*, and each of which means *Tasks* are before process, *Tasks* are underway, *Tasks* are safely finished, and *Tasks* are not finished by its *deadline* respectively.

- *Staff*

Staff object also has four attributes, *ability*, *busy*, *vision*, and *decisionRule*; *Ability* represents the *skill* value to the kind of the *Task*, *busy* denotes that the agent is/is not working now, *vision* maintains the *Tasks* in the *waiting-list* that the agent can confirm, and *decisionRule* makes decisions of which *Tasks* in the *vision* attribute should be started.

- *Manager*

Manager object has two attributes, *vision* and *allocationRule*. The former is similar to the *Staff's*, but *Manager's* can see more *Tasks*. While the latter decides to whom a *Task* is allocated. Details will be described later.

- *TaskPool*

TaskPool object has the attribute *pool* which generates arbitrary *Tasks* within the specified period.

3.3 Definitions of Agent Rules

The following rules described in this section are commonly used to make the agents active in the model.

- *Processing Rule*

Staff agent, handling a *Task*, tries to finish it by reducing the *volume* of the *Task* related to the *skill* value for each step. The *skill* value of the agent is defined by his/her *ability* attribute, and a unique value is given in advance according to the kind of the *Task*.

1. *Staff* agent, handling a *Task*, tries to finish it by reducing the *volume* of the *Task* related to the *skill* value for each step. The *skill* value of the agent is defined by his/her ability attribute, and a unique value is given in advance according to the kind of the *Task*.
2. *Task*, which is started once, is being kept processing by the time steps that its *volume* becomes to zero or below.
3. "Cooperation processing" means that two or more *Staffs* may deal with one *Task* together.

- *Stacking Rule*

Tasks generated from *TaskPool* are given to the organization, and are stacked on the tail of one of the *waiting-lists* in the organization. As shown in Figure 2, the number of *waiting-lists* is set to three in this model.

- *Confirmation Rule*

In *Manager's* and *Staff's* confirming their own *Tasks*, the difference is given to the cognitive ability (range of vision) of *Tasks*. This will enable us to give the level of discipline to *Staff* agents with respect to acquisition of a *Task*. In this model, the cognitive ability of *Staff* agents is assumed to be one-third of *Manager's*.

$V_{manager}(\sigma)$: *Manager* can retain up to σ pieces of *Task* from the head among *waiting-lists* in his *vision* attribute and confirm them anytime.

$V_{staff}(\sigma/3)$: *Staff* agent can retain only up to $\sigma/3$ pieces of *Task* in the *waiting-lists* in his/her *vision* attribute and confirm them anytime. In this literature, we assume that *Staff* agents can see one of three *waiting-lists*.

- *Decision Rule* (*Process-Beginning Rule*)

In starting *Task* processing rules, there exists some differences in terms of the time cost. When a *Manager* agent orders *Staff* agents to manage a *Task*, the time that hangs to transmission is considered. In this model, the delay is set to 1-step. On the other hand, in an organization without any *Manager* agent, each *Staff* agent can start handling his/her *Task* chosen in no time.

$S_{manager}$: In the organization with *Manager*, he/she allocates *Tasks*, which he/she can confirm in $V_{manager}(\sigma)$, for all *Staffs*. The way to allocate is decided by *Manager's allocationRule*. *Staff* allocated *Task* begins it from the next step.

S_{staff}: In the organization without *Manager*, the *Staff* who is idle activity (*busy*==false), selects one *Task* confirmable from *vision* attribute. The selection is decided autonomous by *Staff's decisionRule*. *Staff* begins the selected *Task* at once.

- *Beginning Rule*

The Task underway is removed from *waiting-list*, and registered in *processing-tasks* of *Team* object.

- *Process-Ended Rule*

The Task finished by *Staff*, which the *volume* becomes zero or less, is removed from *processing-tasks*, and registered in the *finished-tasks*.

- *Deadline Expiration Rule*

The unfinished-task is registered in *deadline-passed-tasks* regardless of whether it is under process or not, and is not treatable thereafter (exclusion from *waiting-list* or *processing-tasks*).

- *Temporary Maintenance Rule*

The Task temporarily reserved or interrupted by a *Manager's* judgment is moved from *processing-tasks* to the head of *waiting-list*.

Table 2 Input files of Simulation Environment

FileMaker	Attribute	
TaskPool FileMaker	*size (int)*	
	Task	*variety (int)*
		volumeMin (int)
		volumeMax (int)
AgentAbility FileMaker	*num (int)*	
	type (String)	
	Task	*variety (int)*
	Skill	*skillMin (int)*
		skillMax (int)

- *Generation Rule*

$T_{generate}(\alpha,\beta)$: The meanings of this rule are that β pieces of *Tasks* are added to *TaskPool* every α _ period (step). These additional *Tasks* are chosen from the *pool* attribute of *TaskPool* at random.

- *Deadline Setting Rule*

Deadline of a *Task* is set to the value arbitrary which is from one to twice as long as the span when a *Staff* agent with an average *skill* can finish it. This means, the *deadline* is calculated using the *volume* of the *Task*, the agent's ability for the *Task*, and the present time (now), namely

$$deadline = now + rand(1.0,2.0) \cdot \frac{volume}{skillAverage} \cdot$$

3.4 Simulation Environment

The results of our simulation model depend on both the *pool* attribute of *TaskPool* and *Staff's* ability attribute. Therefore, when the simulation begins, these elements are needed to be given by the file input. We can clearly observe the information requisite by using a common file among different models. Each file is generated by the *FileMaker* shown in Table 2.

Variety is a domain of *id* value of the *Task* object, and corresponds to kind number of *Task*s that the organization processes.

The *Task*, whose attributes are defined in the followings, is described in the file that *TaskPool-FileMaker* generates by just *size* (size of the *TaskPool*);

$$id = \{x \mid x \in Z, 1 \le x \le variety\},$$

$$volume = \{y \mid y \in Z, volumeMin \le y \le volumeMax\}.$$

Likewise, the file generated by the *AgentAbility-FileMaker* contains the *ability* data of all *Staff* agents for each of *variety* kinds of *Tasks*. The expression is the domain of *skill* value, namely,

$$skill = \{z \mid z \in Z, skillMin \le z \le skillMax\}.$$

Moreover, the *decisionRule* that the agent takes according to the type attribute is decided. This is described in the next section.

3.5 *Staff's* Decision Making: *decisionRule*

In the *Process-Beginning Rule* S_{staff}, the *Staff* agent can select one of the following rules.

D1 *Staff* agent begins a random-selected one among *Tasks* confirmed by the *vision* attribute.

D2 He/She selects one *Task* he/she is the best at among the confirmed *Tasks* and begins it.

D3 This rule gives him/her the *Task* which is waited for the longest among confirmed *Task*, and he/she starts it.

D4 He/She selects one *Task* whose *deadline* is most likely to be expired among confirmed *Tasks* and begins it.

3.6 *Manager's* Decision Making: *allocationRule*

In the $S_{manager}$ (*Process-Beginning Rule*), the *Manager* can select one of the flowing rules.

A1 The *Manager* allocates a randomly selected *Task* among confirmed *Tasks* in his/her *vision* to an idle *Staff* agent.

A2 He/She gives an idle *Staff* agent an unprocessed-task in his/her *vision* which he/she thinks that the *Staff* agent is expected to be best at.

A3 He/She gives an idle *Staff* agent an unprocessed-task in his/her *vision* which is waited for the longest among confirmed *Tasks*.

A4 He/She gives an idle *Staff* agent an unprocessed-task in his/her *vision* whose *deadline* is most likely to be expired among confirmed *Tasks*.

4. Experiments and Discussions

4.1 Comparison of agent rules

First of all, in order to show a basic benchmark of this model, the performances of *Staff* agent's *decisionRule* and *Manager* agent's *allocationRule* are compared.

- Parameter Settings

The simulation is executed with the parameters in Table 3. It is enabled that *Manager* confirms all *Task*s stacked on the *waiting-list*.

- Results

In this simulation, totally 71 *Tasks* (5 pieces arranged in the initial step are included) are stacked on the *waiting-list* for the execution periods of 100 steps. The detail is shown in Table 4. The values in Table 4 show the number of (*finished / deadline-passed*)*-tasks* by the *deadline*, and those ratios (against the number of all *Tasks*) in parentheses. Moreover, all the figures are mean values of 100 simulation runs.

Figure 1 is the first simulation among these, plotted by the time series. The horizontal axis is a time step, and the vertical axis is the number of completed *Tasks*. Figure 2 shows a snapshot of the simulation.

Both rules, A1 and D1, are random ones that both the agents begin an randomly selected *Task*, and we can see that the results depend on the differences of how long it takes *Staff* agents to start his/her *Task*. Due to the allocation time cost by the *Manager*, about 4% worth of *Tasks* are not finished and not in time for the *deadline*. Oppositely, in cases of A2 and D2 rules, i.e. *Staff* agents begin a good *Task*, the finished *Tasks* of A2 are larger than those of D2, not only because the *Manager* agent can see more *Tasks* than the *Staff* agents but also because he knows everything about the *skills* of all the *Staff* agents.

On the other hand, when it comes to the *Manager's* allocation, we observed extremely bad performance in cases of A3 or D3, and A4 or D4 rules. This is because the *Staff* agents concentrate on only one *Task* to which each rule gives priority and due to an excessive human resource allocation. The model considered this problem which will be shown in the next subsection.

4.2 Decision of talent distribution to Delivery date

To avoid an excessive human resource allocation, it is necessary to estimate whether one or more of *Staff* agents has/have enough *skill(s)* to complete a *Task* by its *deadline* and to distribute it if possible. This part of the section introduces the model in which the *Manager* agent is endowed with such *ability*.

Table 3 Parameters

Simulation steps	100
Tgenerate(α,β)	(α=3,β=2)
Vmanager(σ)	*σ=waiting-list.length*
Vstaff(σ/3)	σ/3
Task variety	5
volumeMin	30
volumeMax	80
Num	5
skillMin	1
skillMax	10

Table 4 Benchmark for Basic Rules (×100average)

task(All)	A		D	
=71	finish	deadline	finish	deadline
1	26 (0.37)	33 (0.47)	29 (0.41)	30 (0.42)
2	45 (0.63)	16 (0.23)	37 (0.52)	23 (0.32)
3	20 (0.28)	37 (0.52)	29 (0.41)	30 (0.42)
4	13 (0.18)	43 (0.61)	29 (0.41)	29 (0.41)

- **Condition**

In addition to the condition of the model in the previous subsection, the following rules are considered:

$A_{estimate \quad deadline}$(*Task*, *Resource*): The *Manager* allocates a *Task* which is expected to be finished by the deadline to a *Staff* agent, by estimating his/her *skill* value. If a *Task* is judged not to be completed by its *deadline*, even if all of the human resources, it is not allocated anymore.

- **Results**

Table 5 shows the simulation results considering this rule. The problems of A3 or D3, and A4 or D4 rules are improved by an effective allocation, and a higher performance is seen overall.

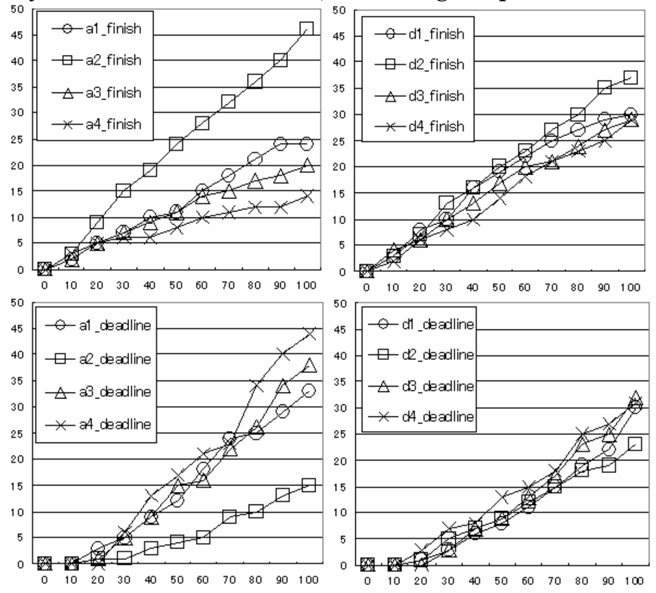

Figure 1 Performance Results of Basic Rules

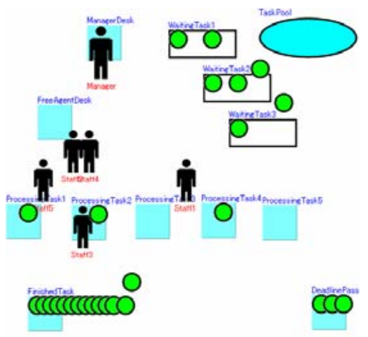

Figure 2　Snapshot of the Simulation Execution

Table 5　In case of Estimating Deadlines

task(All)	A	
=71	Finish	Deadline
1	43 (0.61)	17 (0.24)
2	45 (0.63)	14 (0.20)
3	43 (0.61)	17 (0.24)
4	47 (0.66)	12 (0.17)

5. Concluding Remarks

In this paper, we have reported on intermediate results of an agent-based simulator to aim at the analysis of the office activities. Using the simulator, we will cope with management problems of office work including task analyses, human resource management, and scheduling. Our future work includes the function enhancement of the simulator, and scaling up the models.

In this paper, we have reported on intermediate results of an agent-based simulator to aim at the analysis of the office activities. Using the simulator, we will cope with management problems of office work including task analyses, human resource management, and scheduling. Our future work includes the function enhancement of the simulator, and scaling up the models.

References

[Axelrod 1997] Axelrod, R.: The Complexity of Cooperation: Agent-Based Models of Competition and Collaboration. Princeton University Press, 1997.

[Carley 1995] Carley, K. M.: Computational and Mathematical Organization Theory: Perspective and Directions. Computational and Mathematical Organization Theory, Vol. 1, No. 1, pp. 39-56, 1995.

[Carley 1999] Carley, K. M., Gasser, L.: Computational Organization Theory .in Weiss, G. (ed.): Multiagent Systems - A Modern Approach to Distributed Artificial Intelligence. MIT Press, pp. 299-330, 1999.

[Cohen 1972] Cohen, M. D., J. G. March: A Garbage Can Model of Organizational Choice. Administrative Science Quarterly, Vol. 17, pp. 1-25, 1972.

[Cyert 1963] Cyert, R. M., March, J. G: A Behavioral Theory of the Firm. Prentice-Hall, 1963.

[Tanuma 2005] Tanuma, H., Deguchi, H., Shimizu, T.: SOARS: Spot Oriented Agent Role Simulator – Design and Implementation .in Terano, T., et al. (ed.): Agent-Based- Simulation, from Modeling Methodologies to Real- World Applications, Springer, pp.1-15, 2005.