Automatic Formation and Analysis of Multiagent Virtual Organization

Qinhe Zheng, Xiaoqin Zhang
Department of Computer and Information Science
University of Massachusetts at Dartmouth
x2zhang@umassd.edu

November 15, 2004

Abstract

Virtual organization (VO) refers to the temporary teaming of enterprises. By sharing physical, human and knowledge resources via information technologies, a virtual organization enables member enterprises to share skills, costs, access to one another's markets and, at the same time decrease the risk of investments. To realize this new generation of business model, the ability to form, operate, and dissolve of virtual enterprise is of most pivotal importance. The paper describes our experience gained by implementing a multi-agent system that simulates an artificial marketplace, for which we have derived mechanisms for the decision-making process in various stages of a virtual organization. We presented a negotiation protocol and a bid selection algorithm for agents to form a virtual organization. We adopted the MQ framework to model the agent's reasoning process when it is involved in more than one organizations. In order to better understand the organizational problem, we adapted a statistical model that predicts the expected rewards of individual agents and the performance of the virtual organization. The comparison and analysis of the simulation results and the model predictions are also presented in this paper.

1 Introduction

A virtual organization or enterprise can be defined as "*a cooperation of legally independent enterprises, institutions or individuals, which provide a service on the basis of a common understanding of business. The cooperating units mainly contribute their core competences and they act to externals as a single corporation. The corporation refuses an institutionalization e.g., by central offices; instead, the cooperation is managed by using feasible information and communication technologies.*" [1] This concept is illustrated in Figure 1, which shows a new organization formed by the contributions of resources from four independent enterprises. Of the participating enterprises, a member is designated as initiator agent, who is responsible for task allocation and coordination among the members.

A classical example of a virtual organization is the Agile Infrastructure for Manufacturing Systems (AIMS) project founded by the U.S Government's Advanced Research Project Agency (ARPA). With participating members that include Lockheed, Texas Instruments, and several universities, the goal of AIMS includes the development of mechanisms in both business and technology infrastructures, using national information highways, that would allow companies to very rapidly put together partnerships for the development of complex projects. The set of mechanisms was ultimately referred as AIMSNet [4].

This concept of partnership turned out to be what the business world has been looking for. By sharing physical, human and knowledge resources, a virtual organization enables member enterprises to share skills, costs, and access to one another's markets, at the same time decrease the risk of investments. Through the use of information technologies, the member companies of a virtual

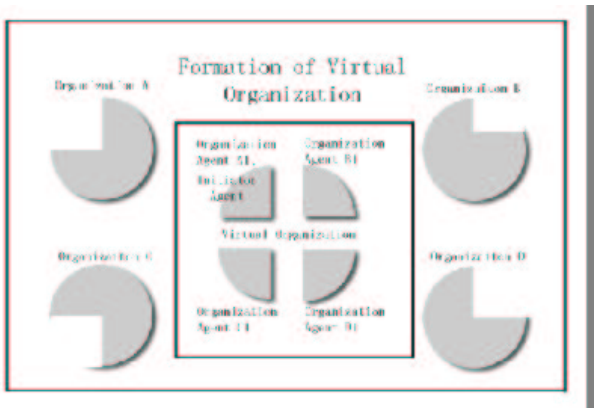

Figure 1: Formation of a virtual organization

Table 1: The Virtual Organization Life Cycle

Identification Phase	Description of product or service to be delivered by the virtual organization, which guides the conceptual design of the virtual organization.
Formation Phase	Rational selection of the individual organizations (partners), which will compose the virtual organization, based in its specific knowledge, skills, resources, costs and availability.
Operation Phase	Control and monitoring of the partner's processes, including resolution of conflicts, and possible virtual organization reconfiguration due to partial failures.
Dissolution	Breaking of the virtual organization, distribution of the obtained profits and storage of relevant information.

organization can work seamlessly across distances, organizations and business boundaries, which enable members to potentially address markets to a global scale. Another benefit for a company to join a virtual organization is the potential of leveraging unused assets. In this case, a company could utilize of otherwise unused resources without interference with its core business.

Generally, the life cycle of a virtual organization can be decomposed in four phases [1], described in the following Table 1.

In order to automate the formation and operation processes, an electronic market infrastructure is needed. Such an infrastructure can be seen as the virtual marketplace where business participants that are geographically distributed can meet each other and cooperate in order to achieve a common business goal. Within this virtual marketplace, individual organizations are the participants and the common business goal is the formation of a VO as the most favorable group of organizations that can satisfy a specific business need. To automate the formation and operation process of virtual organizations, agent seems to be an appropriate metaphor and a mean of system development methodology. A multiagent system consists of a set of agents that are autonomous or semi-autonomous entities, which can perform tasks in complex and dynamically changing environments. There are many similar aspects between multi-agent systems and human organizations: they are constructed of intelligent individuals; there are different relationships among these individuals; each individual has only limited knowledge and is resource-bounded; the individuals interact with each other, they coordinate, negotiate, share knowledge, transfer information, and form and dissolve all kinds of organizations and groups. The traditional development methodologies, such as object-oriented programming, are insufficient in capturing the essences of agents in terms of autonomy and pro-active nature of agents. The recent studies of agent-based engineering have enhanced the development methodology for multiagent system, and made it easier to implement of virtual

organizations. Therefore, it is natural to view virtual organizations as multi-agent systems. The virtual building company described in this paper is a multiagent system where each individual agent exhibits capabilities to be (semi-) autonomy, has the ability to interact with other agents in the virtual market and make rational decisions under changing environments.

The objectives of this research include: to implement a multiagent system that supports the simulation of artificial marketplace, to derive mechanisms for the decision-making process in various stages of a virtual organization, and to perform experiments to evaluate and verify those mechanisms in order to better understand the organizational problems. Our works are mainly focused on the decision-making process of the member agents (each organization is represented by a software agent) during the various stages of a virtual organization, such as the partner selection during the formation process and each agent's task selection process during the operation of a virtual organization. We use Motivational Quantities (MQ) framework to support the agent's local decision-making process. Additionally, we have adapted a modified statistical model to predict the task conflicts and expected reward for each agent during the lifetime of a virtual organization.

In the remaining of this paper, we first introduce a virtual building organization scenario that is used as an example through this paper. We then describe the detailed process in the various phases of the VO's life-cycle: a negotiation protocol for the formation phase, a recursive best-first search algorithm (RBFS) for the partner selection process, an agent's utility mapping function based on MQ framework, penalty policy for less commitment, and an analytical model to predict the behavior of agents and the organization. Finally we will present our experimental work and results.

2 Scenario: Virtual Building Organization

One of our main objectives was to implement a multi-agent system that supports the simulation of artificial marketplace; in that aspect we have developed a scenario as the base for our model. A real estate developer, named Concrete Developer, has recently won the right to develop a large suburban area for residential use. Concrete Developer has always relied on a single outside contractor, who in turn enlists a group of sub-contractors, to construct the residential buildings. However, after a careful analysis, it decided that it would be much more profitable and effective if it forms a virtual organization. The developer partitions the building process into 5 partial processes, namely framing, foundation, electrical work, plumbing, and finishing, assuming they must complete in sequential order, and makes the initial proposal of formation of Virtual Building Company to the sub-contractors in its marketplace. The individual enterprises can then bid for these partial processes; after the initiator has received substantial bids from the individual contractors. The developer then selects a group of bids that meets its highest expectation based on multiple criteria, such as competence, availability and etc. Once the virtual organization is formed, it goes into the operational phase. During the operational phase, a buyer may request for a house at any given time (the negotiation process between the buyer and the developer is omitted for simplicity). After receiving a buyer request, the developer notifies individual participants of the virtual organization, who may or may not commit to a subtask based on their own decision-making mechanisms. The developer accepts a buyer's request only when it has all the commitments necessary to complete the whole construction task. Only when all the subtasks are completed, the developer can collect money from the buyer. From the individual agents' perspective, an agent may receive service request from the developer agent and/or from direct buyers and possible another initiator agent of another virtual organization where the agent also belongs.

There are three type of agents in our model, the initiator agent (the developer), individual "worker" agent (an agent that is capable of partial process), and the buyer agent. The initiator agent is the one who takes the initiative in the formation of virtual

organization and is responsible for task allocation and management during the operation. An individual agent is a self-sustained entity, it may receive service request from direct buyer and is free to join any virtual organization. The buyer agent is the simplest one; its sole purpose is to send service request to any virtual organization or any individual agent. Each of the three types of agents can be instantiated into any number of distinct agents by specifying its characteristic (name, competencies, availability and etc.).

We have successfully implemented the system that models the virtual building company from its formation to the dissolution phase. It is implemented using Java Agent Framework [2] and runs under the MASS simulator [3]. While this system models the work-flows of a construction company, the same procedures could be used for any virtual organization. When implementing the system, we have ignored the business rules that are associated with the specific industry, but mainly focused on areas where agents need to make rational decisions. Discussion of such decision-making is summarized in the following sections.

3 The Formation and Operation Process

3.1 Negotiation Protocol

When the initiator is planning to make the initial proposal, it concerns on two major issues: a set of evaluating criteria is needed to select the most favorable group of member organizations (each organization has a set of attributes, such as cost, quality, and availability, etc.). It also must realized the resulting virtual organization's competence may inter-related by constraints attached with each members' attributes. For example, if an agent can perform only n unit of a partial processes, then the maximum achievable VO output couldn't exceed n units. Therefore, the initiator must determine the set of evaluation criterion and impose a preference order on the attribute values and/or attribute itself. On the other hand, "worker" agents are faced with the decision of whether or not to join a particular virtual organization. Joining a virtual organization is beneficial but it is not always cost free; for instance, in a practical situation, a member agent may required to adopt to use standard business process and information technologies in order to facilitate communication and transaction process in these virtual organizations. Therefore, it must carry out a cost-benefit analysis, based on information provided by the initiator agent and the degree of belief it has in the initiator agent, before joining a virtual organization.

A negotiation protocols is needed for both initiator and individual agents in support of their decision-making during the identification phase. In our model, the proposal sent by the initiator agent includes the following information: the type of task (building construction) needed for the organization, the estimated work load for each type of subtask (partial processes), and the estimated net profit to the organization. The bid from the potential participant includes the following information: the type of partial process the agent is capable of, the number of units it will contribute to the organization (capability), and its profit sharing rate (how much it requests from the virtual organization's profit). For our simple model, this protocol is sufficient for individual agent to decide whether or not to join a virtual organization and for an initiator to evaluating a bid.

3.2 Partner Selection Process

Before the proposal can be made, the initiator needs to decompose the whole product/service process into partial processes. Most often this is to be done by human being assisted by a suitable modeling techniques (we assume this has already been done in our model). Once the initiator agent has identified the partial processes, the partial processes need to be distributed to the agents so that each agent can make its own contribution depending on its specialty, hence an allocation process of partial processes to

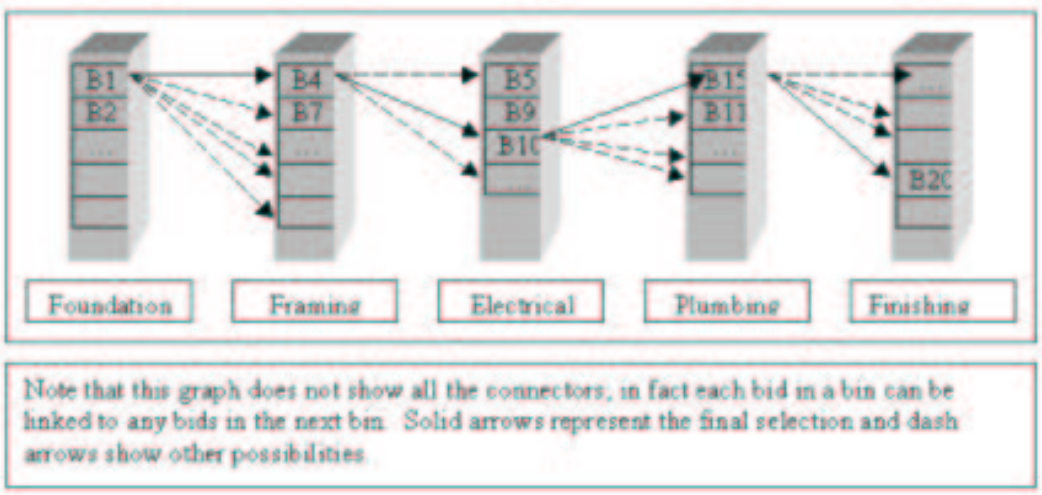

Figure 2: Partner Selections

appropriate agents needs to be found. In our marketplace, the individual agents complement each another in their service offer, i.e., different enterprises cover different partial processes, which resembles a horizontal allocation of a real world business practice. To find the members, we could first search for an agent that will deliver a partial process, then continue looking for other agents that are able to deliver those complementary services, and so on. The search is finished when the organization is self-contained and does not require any more services from other services provided by the marketplace.

However, the objective of the selection process is not only to select a group of members that would form a self-contained virtual organization, but also to form a virtual organization that would maximize the profit. There are two different approaches in terms of maximizing the profit given the set of partial processes (foundation, framing, electrical, plumbing, and finishing), and the estimated workload L. The initiator may try to maximize the profit of the virtual organization. Given the profit is proportionate to the workload, so a maximized workload L will generate maximized profit $R * L$. In this case, the initiator will prefer agents with higher capabilities. However, being a self-interested agent, the initiator could be more interested in maximizing it own profit. The initiator's profit depends on the profit of the organization and the profit to be handed out to other agents in this organization. Assume that a bid B_i contains the following information: type of task T_{bi}, work load commitment count L_{bi}, profit sharing rate S_{bi}, then the profit for the initiator would be: $R \cdot L' \cdot (1 - Sum(S_{bi}))$ where L' is the practical workload ($L' = min(L_{bi})$, and $L' <= L$).

We chose the second approach in partner selection process for our virtual building company. The core of this procedure is a recursive best-first search algorithm (RBFS) (Figure 3) with some heuristics. It works in the following way. First it groups the bids into different bins according to which task they are bidding on (Figure 2). Then it selects one bid from the first bin which will maximize the initiator's profit, but it also remembers the second best choice. It goes to the next bin and finds the best bid of the current bin: if the initiator's expected profit at current bin is no less than the second best choice of the previous bin, it continues to the next bin; otherwise, it will unwind to the previous bin and choose the second best and proceeds from there. It continues this process until an optimal solution is found (Figure **??**).

In our model, the selection criteria includes only the agent's capability (commitment count) and the profit sharing rate it asks. It is obvious that these two criteria are inadequate in real world application. In a more realistic application, we shall also consider other attributes of a bidder, such as the quality of its contribution and the time required to compete a partial process. Each

function RECURSIVE BEST FIRST SEARCH(bid_vector)

RBFS(bid_vector, Make − Tree(bid_vector),$best\_sequence$, $0, \infty$);

functionRBFS($bid\_vector, tree, sequence, depth, f\_limit$)returnsasolution,

orfailureandanewf − costlimit

if$depth$==bid_vectorthenreturn$sequence$

tree ← $SUCCESSOR(bid\_vector, tree, depth, sequence)$;

while(!FINAL − STATE)

$best$ ← thelowestf − valuenodeinsuccessorsorthebestnode

if$f[best] > f_limit$thereturnfailure,$f[best]$

$second\_best$ ← thesecondbestbestsuccessors

$sequence[depth]$ ← $BID[best]$

$f[best] \leftarrow RBFS(bid\_vector, best, sequence, depth + 1, \min(f\_limit, f[second\_best]))$

repeat

returnf_limit

Figure 3: Recursive Best First Search Algorithms

additional variable will inevitably increase the complexity of the selection process, thus, this process could easily turn into a NP-problem. One way to get around this is to use a set of screening filters to eliminate undesired bids from the selection pool before a search algorithm is applied to find an acceptable solution.

3.3 Penalties for Lack-of-commitment

An additional problem in virtual enterprise is that the agents participate in the marketplace are self-interested, trying to maximize their local utilities. Therefore, an enterprise may need an incentive to encourage the agents to maximize the profit for the enterprise. This, however, implies that, whenever it is beneficial, agents may lie during the selection process in order to be more attractive to an initiator agent. During the bidding process, an agent is required to specify the number of the partial processes it is capable/willing to perform to a virtual organization. However, during operation phase the agent may, in self-interested fashion, favor other opportunities and leave the commitment to the virtual organization unfulfilled. Unless there is an established mutual trust among the member companies, especially between the initiator agent and the individual agents, a penalty for less commitments than what it has promised is the most straightforward and easily to implement. Depending on how the penalty is calculated, there are different penalty policies. A linear penalty policy has a fixed penalty rate for each unfulfilled commitment. A progress-based penalty policy has a decreasing penalty rate as more commitments has been fulfilled. It charges a heavy penalty if the agent can not fulfill a minimum percentage of its promise, and it charges a much less penalty if the agent has fulfilled a certain percentage of obligation. For instance, a progress-based penalty policy can be stated as the following: if the agent can not fulfill 30% of its promised commitments, there is 100 units penalty for each unfulfilled commitment; if the agent has fulfilled 90% of its promise, there is only 10 units penalty for each unfulfilled commitment.

To react relationally toward the penalty of lack-of-commitment, the agent needs to incorporate the penalty policy into its local decision-making process. In our model, this is implemented by introducing a control parameter in the utility mapping function

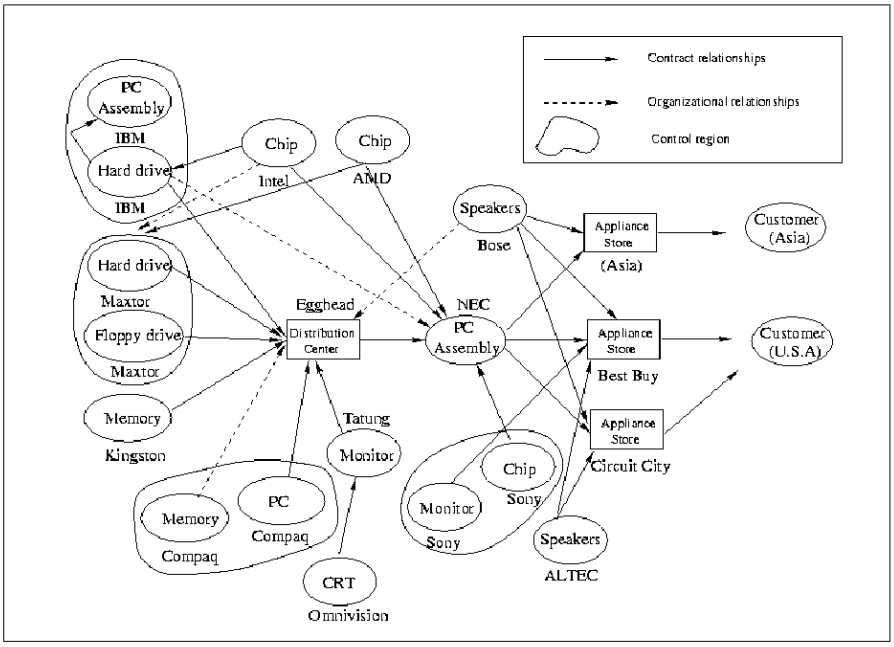

Figure 4: Complex Agent Organization Relationships

(See Section **??** for more detail) associated with the organization task. By adjusting this parameter, different penalty policies can be reflected into the agent's decision-making process, so the agent can balance the profit and penalty when making selection on different types of tasks.

3.4 Motivational Quantities

For an organizationally situated agent, it must interact with agents within and out of its organization; therefore, it is essential that agents must model their organizational relationships and reason about the value of utility of interacting and coordinating with particular agents over particular actions [4]. In a more complicated situation, an organizational agent may belong to multiple virtual organizations with different, or even conflicting, goals and objectives, and the cooperation attitude between a local agent and others may range from fully self-interested to full cooperative. To further complicate the matters, real agents are hindered by bounded rationality, limited resources, and only have imperfect knowledge of the environment. The interactions among the agents may result in a complex relationship diagram similar to Figure 4. When examining a scenario like this, there is the need to relate the different motivational factors that influence agent decision-making, such ability is a requisite for the agents to act rationally given their social context.

Having different relationships with multiple agents means that at any given time, an agent may receive service request from different agents in order to make progress toward different goals. If the agent cannot perform all the tasks, it has to select a subset of the tasks to perform and and determine an appropriate sequence in which to perform them. This problem faced by an agent in the electronic marketplace can be categorized as a real-time action-selection-sequencing problem where an agent has n candidate tasks and alternative different ways to perform the tasks. Tasks have deadlines and other constraints as well as different performance properties, e.g., consuming different resources or producing results of varying quality. How to perform them where the appropriate choice depends on the agent's context, which includes its relationships with other agents, shared organizational goals and individual goals, commitments made with other agent, and resource limitations. It is in this context that [6, 7] suggests

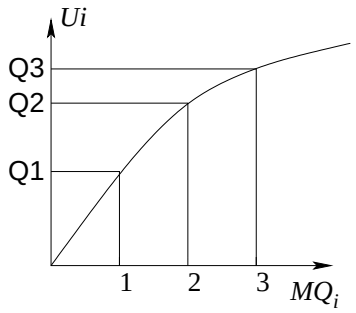

Figure 5: Motivational Quantities and Utilities

the need to quantify motivational factors using Motivational Quantities (MQ) Framework.

The MQ framework is an agent control framework that provides the agent the ability to reason about which tasks should be performed and when to perform them. The reasoning is based on the agent's organizational concerns. The basic assumption is that agents are complex, with multiple goals related to the multiple roles they play in the agent society. The progress toward one goal cannot substitute for the progress toward another goal. Motivational Quantities (MQs) are used to represent the progress toward organizational goals quantitatively. Each agent has a set of MQs which it is interested in and wants to accumulate. Each MQ_i in this set represents the progress toward one of the agent's organizational goals. Each MQ_i is associated with a preference function (utility curve), U_{f_i}, which describes the agent's preference for a particular quantity of the MQ_i. The MQ framework thus provides an approach to compare the agent's different motivational factors through a multi-attribute function. Not all agents have the same MQ set. Different agents may have different preferences for the same MQ.

MQs are consumed and produced by performing MQ tasks. The agent's overall goal is to select tasks to perform in order to maximize its local utility through collecting different MQs. MQ tasks are abstractions of the primitive actions that an agent may perform. The agent compares and selects tasks that are associated with different organizational goals. Each MQ task T_i has the following characteristics: earliest start time (est), deadline (dl_i), and process time needed to accomplish task T_i (d_i).

The MQ scheduler schedules current potential MQ tasks, and produces a schedule for a set of MQ tasks, specifying their start times, and finish times. The scheduler takes the following factors into consideration: the MQ consumed and produced by performing task T_i, duration d_i, the earliest start time est_i and the deadline dl_i of each MQ task, and the agent's current accumulation of MQs. Notice that MQ is always being evaluated in the context of agent's current MQ accumulation state. For example, Figure 5 shows a single utility curve for a single MQ_i. The first one unit MQ_i brings the agent Q_1 units of utility U_i . After the agent has collected 2 units of MQ_i, the additional one unit of MQ_i brings the agent additional $Q_3 - Q_2$ units of utility U_i. $Q_3 - Q_2$ is not necessarily equal to Q_1, they are all calculated based on the utility curve associated with MQ_i.

The MQ framework provides the comparison of tasks that need to be performed for different reasons: for different organizational goals, for other agents to gain some financial benefit or favors in return, for cooperation with other agents, etc. It also supports different utility functions that relate the execution of tasks to the importance of organizational goals. In summary, the motivational qualities (MQ) framework provides an agent with the capability to reason about different goals in an open, dynamic and large-scale multiagent system.

3.5 Utility Mapping Function of MQ

In a virtual organization, each member agent receives service requests not only from this organization (referred as *organization task*), but also from other organizations (if the agent belongs to multiple organizations) or from direct customers (*outside task*). When there is conflict between different tasks, the agent needs to decide which task to commit. The MQ framework provides such a mechanism for keeping the different motivational concerns separate, because they represent quantities that are not interchangeable. For example, the completion of task a is a progress toward objective A, but does not necessarily present a progress toward objective B. MQ enables agents to compare different types of tasks, the costs and the benefits of a particular courses of action.

Based on MQ framework, we assume that each different type of task produces a different type of MQ. For instance, tasks from organization A produce $MQ\_organizationA$, tasks from organization B produce $MQ\_organizationB$, and tasks from direct customers produce MQ_direct. There is a utility mapping function associated with each type of MQ, and it reflects how the agent evaluates this task in terms of the contribution to its local goal and objective. To focus on the study of organization, we assume the outside tasks directly produce monetary value, the mapping function for MQ_direct is expressed as $f(x) = y$, which maps each unit of monetary value into one unit of local utility.

It is more complicate to evaluate an organization task, the agent needs to consider a number of issues: how important the organization's achievement is to this agent, how many commitments it has made for this organization, what the penalty policy is, and etc. Here we propose a mapping function that takes into the consideration of the number of commitments the agent has promised to a virtual organization, the maximum expected reward from a virtual organization and the penalty policy of the organization. The utility gain from performing a virtual task can not be simply measured by an monetary unit, since the agent not only gets money in return or as penalty, but also the utility of having a good relationship with the initiator agent. The mapping function is expressed as $f(x) = a \cdot \frac{1}{b} \cdot (1 - ((1-b)^{\frac{1}{c}})^x)$, where a is the expected utility from/of the virtual organization depending on how important the agent feels about the organization's achievement. For example, if the expected utility of the organization is 1000, and the profit sharing rate for this agent is 20%. a can be set as 1000, which means the agent views the organization's achievement as its own, a can be set as 200 (1000*20%) if the agent only cares about its own gain, and a can be set as any other number between 200 and 1000, depending on how much emphasis the agent has on the organization's achievement. c represents the number of commitments the agent has made towards this organization. b is a control parameter and $0 <= b <= 1$, which works with c together to reflect the penalty policy (See details in the following example). The function is a derivation of a general function $a * (1 + b^{(} - x))$ which produces an upward decreasing curve. By adding a third variable c to the formula, we would have more control over the way that how an agent would fulfill its promise to the organization. The intention is the agent would try to fulfill its promise to the organization; afterward the utility gain from performing virtual tasks would slow down. The mapping function can be illustrated in Figure 6.

In the first case we have a=90, b=0.9 and c=10. As we can see from the graph, it has an upward decreasing curve; the agent has the tendency of preferring organization task to the outside task, which can be visualized as having a linear curve. This is especially true in the beginning, and eventually the utility gain became flat as the first few commitments have been fulfilled. This actually reflects the progress-based penalty policy, where the penalty rate decreases significantly after the agent has fulfilled a certain percentage of obligation. Depending on the organization's penalty policy, one can modify the value of parameter b in order to adjust the agent's attitude toward the organization task. For example, if the organization has a linear penalty policy, the first mapping function may not be a good idea since it only favors the first few commitments. By changing the b value from 0.9 to

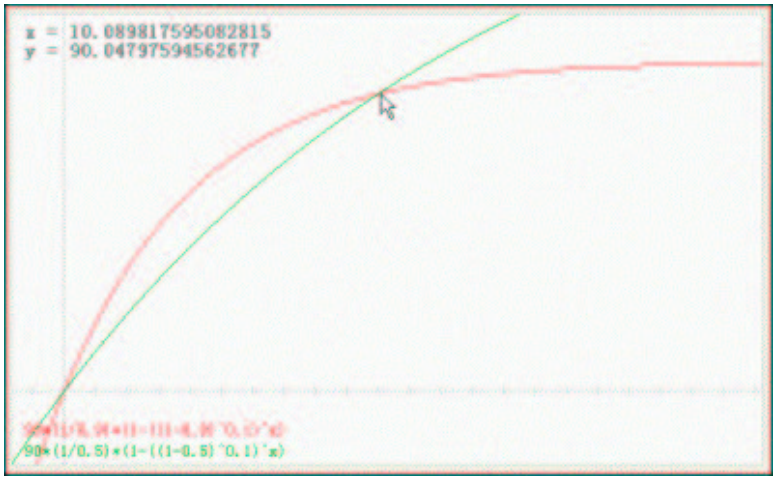

Figure 6: Mapping Functions in Motivational Quantities

0.5, the agent would have the preference on the organization task before all the commitment has reached (represented by an near linear curve, assuming to be steeper than linear curve of the outside task).

4 Analysis Based on a Statistical Model

4.1 The original model

The expected reward for an organizationally situated agent is straight forward if the agent is expecting one type of task without scheduling conflicts. However, this is rarely the case, since an agent may coordinate with multiple agents and/or belong to multiple organizations. Consequently, an agent may receive multiple service requests at any given time. Being self-interested by nature, an agent must make decision on which tasks to perform and in what order to perform them. [5] proposed a generic statistical model that anticipates the probabilities of conflict between any two types of tasks for a local agent and the expected reward for the agent[1]. This generic model assumes a simple agent organization, which is made up of three agents, A_1, which is the initiator agent, has task T_1 coming in, of which there are two subtasks $sub2$ and $sub3$ that need to be sub-contracted to A_2 and A_3 respectively. At the same time, T_i arrives at A_i with a probability of $\frac{1}{r_i}$ at each time unit (which means each agent may have two possible arriving tasks at any given time). For each task there is a number of parameters associated with it, as shown in Figure 7 (which also illustrates their relationships). For task T_i, e_i, dur_i and sl_i are uniformly distributed within the ranges $(a_i^e, b_i^e]$, $\left(a_i^d, b_i^d\right]$, $and\,(a_i^s, b_i^s]$; and for task sub2 and sub3, e_i, dur_i and sl_i are uniformly distributed within the ranges of $(a_{1i}^e, b_{1i}^e]$, $\left(a_{1i}^d, b_{1i}^d\right]$, $and\,(a_{1i}^s, b_{1i}^s]$.

An agent needs to choose which task to execute when and only when there is a conflict between tasks. A task of type i is in conflict with a task of type j (whether it comes before task i or after) if and only if the following two inequalities are both true:

$dli - estj <= duri + durj,$

$dlj - esti <= duri + durj$

By rewriting the two inequalities in term of est, dur and sl, we get: $sl_i - dur_j \leq est_j - est_i \leq dur_i - sl_j$

To calculate the expected reward for an agent at any given time, we need first calculate the probability that an arriving task type i is in conflict with a task of type j.

[1]The material presented in this section is quoted from [5], the purpose is to give readers a brief overview of this model in order to understand our following work. Each equation can be solved using the basic statistical parameters. Details are omitted here and can be found in [5].

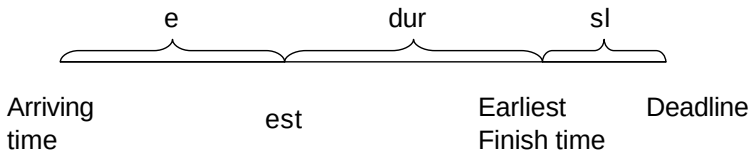

Figure 7: The relationship of the different parameters of a task (Source: [5])

$Pcij = P(sli - durj <= estj - esti <= duri - slj)$

Given the probability of conflict, we can calculate the expected reward for each agent. For A_2 and A_3, there may be two types of tasks coming in at any moment: the direct task (can be viewed as outside task) T_i and the subcontract task (can be viewed as organization task) sub_i with a probability of $1/r_i$ respectively, where $i = 2, 3$. Let us look at them one by one.

When a direct task T_i for A_i arrives, it accumulates reward only under one of the following circumstances:

1. There is a conflict between T_i and a subcontract task sub_i and there is not conflict with other direct tasks. In addition, the direct task reward is greater than the utility of the subcontract task that it is in conflict with, i.e., $R_i > Rn_i = R_{1i} + k_i \cdot \frac{1}{2} \cdot R_{11}$. The expected reward gained by executing the new task in this case is:

$$ER_i^{(1)} = Pc_{1i,i} \cdot (1 - Pc_{ii}) \cdot E(R_i | R_i > Rn_i) \tag{1}$$

2. The only conflict caused by this task is with another direct task T_i'. In addition, the new reward is higher than that of T_i'. The expected reward gained by executing this task under this condition is:

$$ER_i^{(2)} = (1 - Pc_{1i,i}) \cdot Pc_{ii} \cdot [E(R_i | R_i > R_i') + \frac{1}{2} E(R_i | R_i = R_i')] \tag{2}$$

3. There are conflicts with both another direct task and a subcontract task. In addition, the reward gained by the new direct task is the highest.

$$ER_i^{(3)} = Pc_{1i,i} \cdot Pc_{ii} \cdot [E(R_i | R_i > Rn_i \& R_i > R_i') + \frac{1}{2} E(R_i | R_i > R_{ni} \& R_i = R_i')] \tag{3}$$

4. There is no conflict caused by the new task.

$$ER_i^{(4)} = (1 - Pc_{1i,i})(1 - Pc_{ii}) \cdot \frac{ar_i + br_i}{2} \tag{4}$$

Similarly, when a subtask sub_i arrives at A_i, A_i will choose to commit to it under four conditions, but it can accumulate this reward only when the other agent decides to commit to the other subtask as well. Therefore the expected reward will be:

$$ER_i^{(5)} = Pcommit_2 \cdot Pcommit_3 \cdot R_{1i} \tag{5}$$

where $Pcommit_i$ is the probability of agent A_i commits to the subtask sub_i ($i = 2, 3$).

Now we have the expected reward that A_2 or A_3 collects at each time unit:

$$ER_i = \frac{1}{r_i}(ER_i^{(1)} + ER_i^{(2)} + ER_i^{(3)} + ER_i^{(4)}) + \frac{1}{r_1} ER_i^{(5)} \tag{6}$$

function create LookupTable($organizationTask, outsideTask$) returns a lookup table

$table \leftarrow \text{MAKE} - TABLE(empty)$

$arrival\_number \leftarrow$ expectednumberofvirtualtasks

$current\_accepted \leftarrow 0$

$conflict\_with\_outside\_task \leftarrow \text{CONFLICT}(organizationTask, outsideTask)$

$conflict\_with\_itself \leftarrow \text{CONFLICT}(organizationTask, organizationTask)$

loopfor$i$from1to$arrival_number$do

$reward \leftarrow \text{UTILITY}\_GAIN(organizationTask, current\_accepted)$

$probability\_accept \leftarrow (1-conflict\_with\_itself) \cdot (1-conflict\_with\_outside)$

$+0.5 \cdot (1-conflict\_with\_outsideTask) \cdot conflict\_with\_itself$

$+0.5 \cdot conflict\_with\_outsideTask \cdot conflict\_with\_itself \cdot \text{P}_{reward\_greater\_than\_outsideTask}$

$+conflict\_with\_outside \cdot (1-conflict\_with\_itself) \cdot \text{P}_{reward\_greater\_than\_outsideTask}$

$table \leftarrow MAKE\_ROW(i, current\_accepted, probability\_accept, reward)$

$arrival\_number \leftarrow arrival\_number++$

$current\_accepted \leftarrow current\_accepted + probability\_accept$

returntable

Figure 8: Lookup Table Algorithms

Although this model does not explicitly express the expected reward for virtual organization, we could derive the reward by first calculating the probability P of commitment from all agents, then the expected reward for the virtual organization ER_v at any given moment would be: $ER_v = \frac{1}{r_1} \cdot P \cdot Rn_i$.

4.2 The modification: introduce a look-up table algorithm to handle the dynamic mapping of MQ

In order to model the Virtual Building Company, we need modify the original statistical framework described above to reflect the difference between our virtual organization and the generic multiagent system described in the statistical model. In the original model, it is assumed that the reward of a task is uniformly distributed within the range a_i^r, b_i^r. In our framework, it is only true for outside tasks. For organization task, the agent measures its reward using a utility mapping function based on the MQ produced by performing this task. Because MQ is always being evaluated in the context of agent's current MQ accumulation state, so the reward of an organization task also depends on the agent's current contribution toward the virtual organization. The first organization task with 1 unit $MQ_organization$ and the second organization task with 1 unit $MQ_organization$ may produce different amount of local utility for the agent, depending on how much $MQ_organization$ the agent has collected.

To solve this problem, we create a lookup table, which calculates the expected reward for an organization task at a given time (Figure 8) that describes how much local utility is generated by an organizational task at certain point of time. This calculation is based on the estimation of the agent's current MQ accumulation state, which is based on the estimation of how many organization tasks have been accepted previously. To estimate how many organization tasks have been accepted, we need calculate the probability of conflict and compare the rewards of different tasks.

This algorithm would create a table similar to Table 2. Once we have this lookup table, we can have a function $R(t)$, which

Arrival#	Current accepted	P (accepted)	Utility Gain
1	0	0.74	1,159.79
2	0.74	0.74	1,092.74
3	1.48	0.74	1,029.57
4	2.22	0.74	970.0469
5	2.96	0.74	913.9677
6	3.7	0.74	861.1305
7	4.44	0.74	811.3478
8	5.18	0.7082	764.4431
9	5.8882	0.6703	722.0952
10	6.5585	0.6364	684.1748
11	7.195	0.6059	650.017
12	7.8009	0.5782	619.0842
13	8.3791	0.5531	590.9373
14	8.9322	0.5301	565.2137
15	9.4622	0.509	541.6114

Figure 9: A Sample Lookup Table

calculates the expected reward for that given time when provided with the current time unit t. As with the original model, the assumption is that we have prior knowledge of all agents, including the frequency of the arriving organization task and the duration of the virtual organization. To find the expected reward at a given time, we calculate the estimated arrival number using $arrival\_number = (duration+1)/\mathrm{r_j}+1$, and use the reference lookup table to find the estimated accumulation of MQs (current_accepted) and the corresponding utility gain (expected reward at time t). The overall expected reward for the agent during the operation of a virtual organization is then expressed as: $ER_i = \sum_{t=0}^{duration} \frac{1}{r_i}(ER_i^{(1)} + ER_i^{(2)} + ER_i^{(3)} + ER_i^{(4)}) + \frac{1}{r_1}ER_i^{(5)}$, Where $\lfloor Rn_i \rfloor$ (the reward for the organization task) in each case of $ER_i^{(n)}$ is replaced with the function $R(t)$. Figure 9 shows a sample lookup table.

Based on the new formula derived we have implemented the statistical model as a stand-alone application, thus the comparisons between the simulation runs and the model predictions can be made.

5 Experiments

The experiments were designed to verify the correctness of a set of mechanisms we developed with a goal of unveil any relevant information base on the data we gather from running both the agent model and the statistical model. Furthermore, we would like to study the agent's behavior under different control settings. By alternating the parameters of different types of tasks for an agent, we would like to see the effect of mapping function on agent's promise to the organization, the agent's local utility, and the organization's utility.

5.1 Verification of Agent Model

The first set of experiments is to verify the statistical model through the simulation results. In an artificial marketplace, individual agents exist in the virtual environment and the market demands are generated by the buyers. By controlling the parameters/characteristics of agents and the frequency of market demands, we can perform analysis on the outcome of the artificial marketplace. The statistical model takes parameters $(freq, \left(a_i^d, b_i^d\right], (a_i^s, b_i^s], (a_i^r, b_i^r])$for each task type, which are corresponding

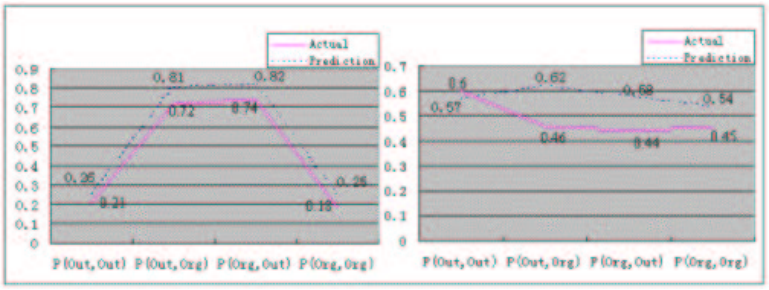

Figure 10: Comparison of Conflict Probability

to its frequency, the range of time required to complete, the range of the slack time, and the range of reward by performing such a task. These parameters enable each agent to calculate the probabilities of conflict between any two tasks and the expected reward. By changing the corresponding parameters for each agent in Virtual Building Company, we are expecting to have the matching results, or at least statistically close to the model prediction.

The experiment setting consists of five agents in a virtual organization with a running time of 3000 clock cycles. Each agent uses the following parameters: $(40, (25, 30], (0, 10], (750, 1500])$ for an outside task, and $(35, (20, 30], (0, 10], a \cdot \frac{1}{b} \cdot (1-((1-b)^{\frac{1}{c}})^x))$ for an organization task, where the reward of an organization task is replaced by the utility mapping function, with a = 30000, b = 0.9, and c = 20. The mapping function is used to calculate the utility of an organization task, so it can be compared to the utility of reward from an outside task, which is direct mapping from the monetary value. Given this set of parameters, we are expecting to have, on average, 75 occurrences of outside task and 85 occurrences of organization task. In order to calculate the conflicts in the agent model, we keep a stack of all the arriving tasks during the VO operation, regardless whether a task is being executed or not. At the end of the system run, we count the number of conflicts for a particular task type and divide it by the total number of occurrences. For instance, to calculate the probability of conflict between an outside task and an organization task, we need to find out the number of times an outside task is in conflict with an organization task, and divided by the total number of outside tasks.

The probability of conflict from both the statistical model and the agent model are summarized in Figure **??**. A similar experiment was conducted with a different set of parameters: $(80, (20, 30], (0, 10], (750, 1500])$ for an outside task, and $(25, (20, 30], (0, 10], 1000, a \cdot \frac{1}{b} \cdot (1-((1-b)^{\frac{1}{c}})^x))$ for an organization task, where a = 30000, b = 0.5, and c = 20; with expected 38 outside tasks and 120 organization tasks. The results are also summarized in Figure **??**.

We found that the probability of conflict in simulation is well predicted by the statistical model, though the prediction is a little bit higher than the simulation result. This may be explained by the fact that the simulator uses a scheduler that schedules all tasks fall into a fixed time window, hence the conflict between tasks that fall into different scheduling windows are not caught by the simulator.

5.2 Effects of MQ Mapping Function

Our second set of experiment was to investigate the effect of the mapping function on an agent's promise to the organization, its local utility, and also the overall virtual organization's performance. We used $U(x) = a \cdot \frac{1}{b}(1 - ((1-b)^{\frac{1}{c}})^x)$ as the mapping function to evaluate the organization task. As described earlier, this utility function has an upward decreasing curve with the ability to change shape as desired. Such a mapping function is useful especially in modeling the lack-of-commitment penalty and the intangible gains from the organization task. Within the context of our agent model, the function is useful only when the MQ

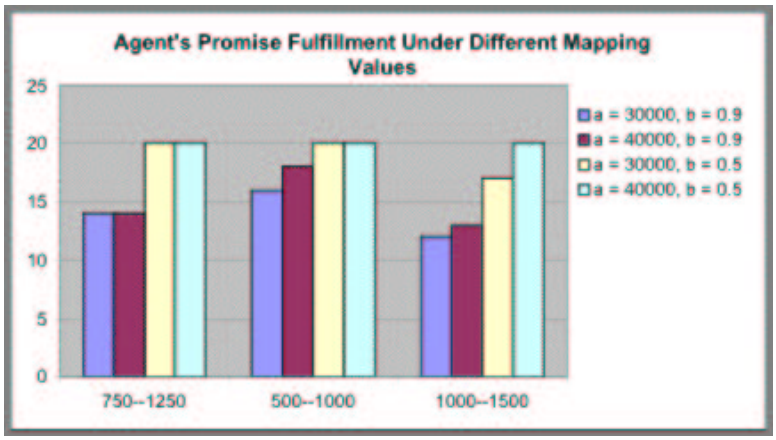

Figure 11: Effect of mapping functions on promise

scheduler is called during the operation phrase, in order to select tasks from two or more conflicting tasks.

There were several problems we encountered during this experiment. It turned out that our first set of parameters produced tasks (often an outside and an organization task at the same time) that potentially required the use of MQ scheduler; however, there was a high probability that the arriving tasks are in conflict with previously committed ones. As a result, one of the arriving tasks (or both) was rejected automatically. Therefore, the MQ scheduler was never called and so the mapping function did not come into play. Another problem arises, when the MQ scheduler selects an organization task to commit, the task is eventually canceled due to other agent's failure of commitment to other related partial process. Consequently, the agent loses an opportunity to commit to an outside task instead.

With these experiences, our next set of parameters were to model the situation where there is only one agent receives two types of tasks while others are completely dedicated to the virtual organization. Even though there are five members in the virtual organization, we are only interested in the one with the multiple types of tasks. In order to reduce the noises introduced by the randomness of the occurrences of tasks, we control the frequency of tasks so that they occur regularly with a fixed time interval between two sequencing tasks. For example, a task with a frequency of 40 means there is a new arrival in every 40-clock cycle. This setup would allow us to fully observe the effect of mapping function on the particular agent's behavior.

For this set of experiments, each run lasts 800 clock cycles. We used the following parameters: $(40, (20, 30], (0, 10], a \cdot \frac{1}{b} \cdot (1 - ((1-b)^{\frac{1}{c}})^x))$ for the organization task, assuming the agent has 20 commitments to the virtual organization and each organization task has a fixed reward of 1000. The value of a is calculated by the sum of expected reward plus the possible penalty for doing no organization task (i.e. when there is a linear penalty of 500 for each organization task, a is equal to 300000, with the expected reward 1000 * 20 = 20000 and a penalty of 500 * 20 = 10000). We varied the frequency and the reward of the outside task in order to investigate the effect of the mapping functions. In addition, we also varied the value of b in the mapping function in order to observe the behavior of the agent under both linear-based and progress-based penalty. Also in our experiment, we tried two other approaches. One is called "no mapping function", where no mapping function was used for the organization task, instead, the agent assumes a fixed monetary reward of 1000 for each organization task. Another approach is called "forced function", which means that before the agent accomplishes all its commitments to organization, it is forced to select the organization task whenever there is a conflict.

The agent has a fixed capacity of 20 tasks under our experiment settings. Figure 11 shows the number of completed organization

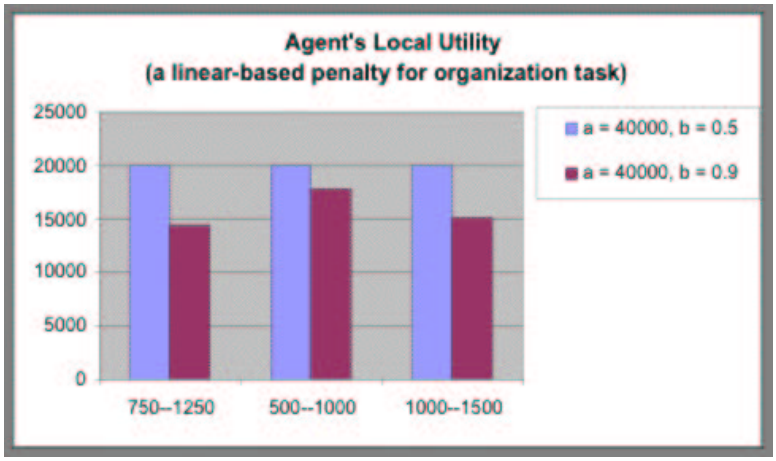

Figure 12: Agent's local utility under a linear penalty policy

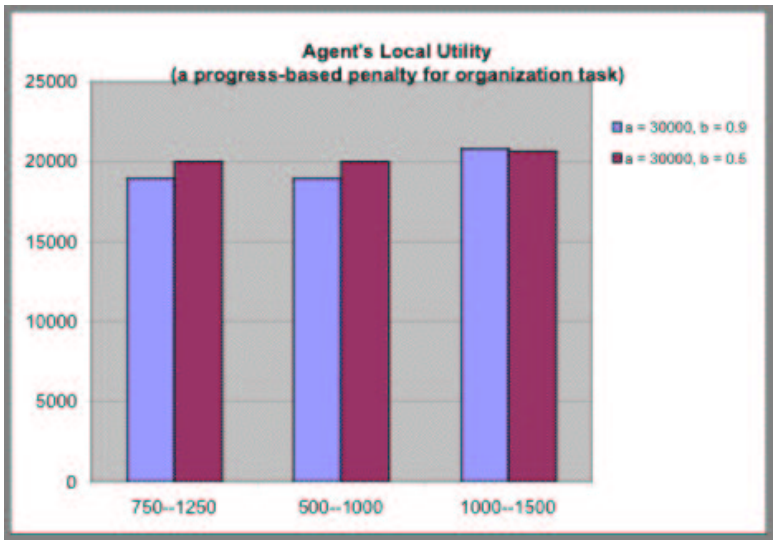

Figure 13: Agent's local utility under a progress-based penalty policy

tasks using mapping functions with different parameter values. With a reward range of 500-1000 at a frequency of 1/40, an outside task is not as favorable as an organization task; therefore the agent was able to fulfill most of its promise. On the other hand, when the outside task has a reward between 1000-1500, it becomes more competitive; consequently, fewer organization tasks were being fulfilled. The most interesting case is when the outside task has a reward of 750-1250. Under this situation, the mapping function with b = 0.9 does not guaranteed the fulfillment of its promise. With b = 0.9, the agent has the tendency of favoring the first few organization tasks but then loses its momentum afterwords. By contrast, when b= 0.5, it is most likely that the agent would fulfill its promises as it keeps its favor for organization tasks for longer time. Additionally, a bigger a value also reflects more emphasis on organization tasks and hence results in more commitments fulfilled.

As described in section 3.3, we may modify the b value in the mapping function in response to an organization's penalty policy. Given a linear-based penalty policy, we feel that a b value close to or smaller than 0.5 would be appropriated, whereas a progress-based penalty policy may require a b value close to 0.9 to ensure that the agent would at least perform the first few organization tasks. By this convention, we would be able to control the agent's behavior in response to different penalty policies. Figure 12 shows the agent's local utility using different mapping functions, assuming the organization adopts a linear penalty policy (a

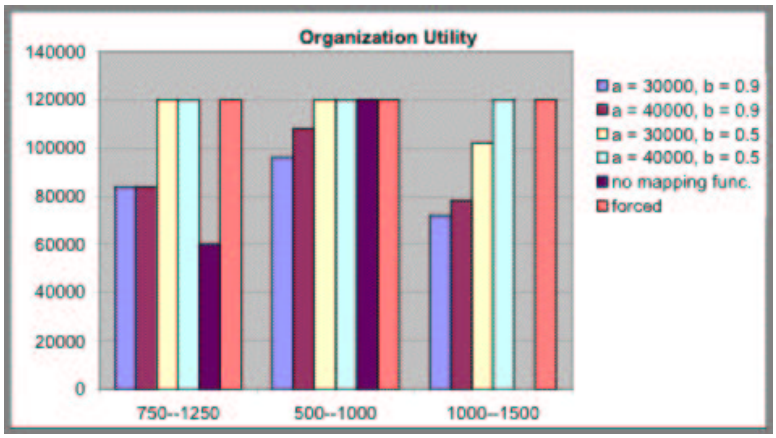

Figure 14: Organization's utility using different decision-making policies

penalty of 1000 units for each unfulfilled commitment). It is found that the mapping function with $b = 0.5$ brings the agent more local utility since it reflects the organization's penalty policy appropriately. Figure 13 shows the agent's local utility using different mapping functions, assuming the organization adopts a progress-based penalty policy (a penalty of 800 units for each unfulfilled commitment when total number of commitments is less than 10, otherwise, 200 units penalty for each unfulfilled commitment). Under this situation, it is found that the mapping function with $b = 0.9$ only outperforms the other one when the outside task generates higher reward (1000-1500). In the other two cases, the performances using these functions are close. This tells us that we need a finer turn of the parameter value to reflect the progress-based penalty policy more accurately.

The use of mapping function also has a direct effect on the overall performance of virtual organization. Since the organization utility is attributed to how cooperative the agents are, the more weights the agent put on the organization task (reflected in a and b value in the mapping function), the more organization tasks will be completed. As illustrated in Figure 14, a greater a value and a smaller b value in the mapping function have positive effect on the organization utility.

5.3 Mapping Function For Agents With Multiple Organization Memberships

Our third set of experiment was to study the effect of MQ mapping function when the agents are involved in multiple virtual organizations. We used the same settings as the previous experiment, with the parameter $(40, (20, 25], (0, 5], (750, 1000))$ for the outside task, and assuming the tasks from the two virtual organizations have the same frequency, duration, and slack time as the outside task. And once again, we focused on one particular agent who has multiple organization memberships, while others were engaged in only one virtual organization and completely dedicated to the organization tasks. Figure 15 shows the agent's commitments to different types of tasks when using different mapping functions.

As indicated by the experiment result, the mapping function with a higher expected utility (a value) or a smaller b value would ensure the agent's fulfillment of its promise to Organization A. While the tasks from Organization A may appear to be more attractive than that of Organization B (when the mapping function has greater expected utility or a smaller b value), however, this does not mean the utility gain from Organization A always exceeds that of Organization B; in fact, once the agent has accumulated a certain amount of MQs from Organization A, tasks from both organization may take turns to be selected.

From the result chart, it appears as the changes in the mapping function of Organization A have no effect on the number of tasks completed for Organization B. This is because both organization tasks are also competing with the outside task. Given a

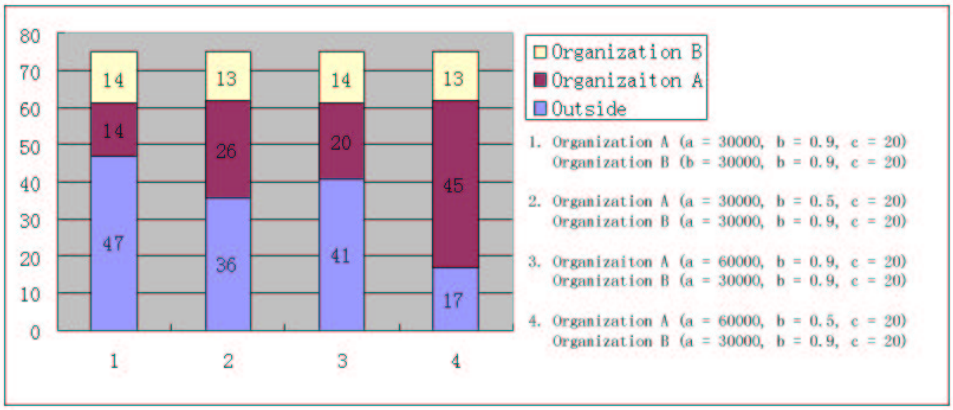

Figure 15: Effect of Mapping Function in Multiple Virtual Organization

capacity of 75 tasks, there is room for the agent to achieve at least 13 tasks for Organization B. If no outside task is considered, a change in the mapping function of one task type certainly would have an effect on the completion of another type of task. When the MQs from the two virtual organizations have the similar mapping functions, there would be an equal chance for these tasks to be accepted. In this case, it may imply that an agent could not fulfill its promises to any of the virtual organizations if its capacity is limited.

One of the recurring issues we have encountered is that few organization tasks get completed when the agents are engaged in both outside task and organization task, it is even more apparent when the agents belong to multiple virtual organizations. This is the reason why the experiments are focused on one particular agent while others are assumed to have 100% availability to the organizations. When all agents are free to join any number of virtual organization, the control power of the MQ mapping function is greatly reduced because the other agents' commitments are difficult to predict. The more immediate questions are, should the initiator agent allow agents to have multiple memberships; do multiple memberships give an agent a greater utility gain when the commitments from other agents are very uncertain?

Whether an organization task can be completed is based on how dedicate the agents are. The initiator agent accepts a building task only when there is at least one agent willing to commit for each partial process/subtask, which turns out to have a probability of $\prod_{i=1}^{n} Pcommit_i$. If a virtual organization has 5 partial processes with one member agent assigned to each partial process, and suppose each agent has a probability of 0.7 to commit to the partial process, then there is only a small chance of 0.16 that a task arrive at the initiator will actually be accepted and performed by the virtual organization.

It is clear that in considering the formation of a virtual organization, one must assess how delicate a member is to the operation of the virtual origination. The ideal form of a virtual organization is that the member enterprises have 100% availability; in this case, the probability of commitment would be 1 (assuming no conflict with a prior organization task). Aside from selecting members with the higher availability, an initiator should also try to limit the number of partial processes as much as possible. A solution to these limitations is having multiple agents who are capable of a partial process in a virtual organization. However, the fact that more agents involoves in an organization increases the communication/operation overhead, as well as decreases the overall profit to the initiator agent. Therefore, the initiator needs to have a balanced number of participating enterprises and partial processes in order to achieve acceptable probability of commitments from agents.

5.4 Summary of experimental results

From this experiment, we can make the following conclusions:

1. For an isolated instance of task selection, or in situations where there are less concern with the previously committed task, the motivational quantities framework provides an powerful tool that enables an organizational situated agent to make intelligent decision. By considering the most important factors, an agent can reason about every aspect of its actions, thus achieve its organizational goals in a rational manner. This mechanism, however, is weaker when previously committed tasks are interfering with the current decision-making.
2. The mapping function has an effect on the agent's promise to the virtual organization, its local utility and the performance of the virtual organization. An agent could change its attitude toward the virtual organization by alternating the parameters in the mapping function. For instance, an agent may react to the organization's penalty policy by adjusting the mapping function, thus change its attitude as to how to fulfill its commitments to the virtual organization. It is possible to formulate a mechanism to calculate the value of the parameters in the mapping function in order to reflect the organization's penalty policy accurately.
3. An important issue faced by the imitator agent is how to achieve a balance between the number of participating enterprises and partial processes in order to achieve acceptable probability of commitments from agents.

6 Conclusion and Future Work

With the changing landscape of business world, cooperation between the enterprises is the only way to stay on the edge of competition. Cooperation enables enterprises to share skills, costs, access to one another's markets and resources and, at the same time, decrease the risk of investments. Supported with the rapid development of information technologies, virtual organization has the potentials to be the future way of enterprise cooperation and electronic business.

This paper investigated the challenges and obstacles that we are still facing. We proposed a negotiation protocol for automatic formation of a virtual organization. We have studied the decision making of individual agent on multi-dimensional negotiation process. The partner selection process is another issue that we have focused on, we presented a RBFS algorithm to find the optimal membership for the virtual organization. This solution we applied to Virtual Building Company may not be adequate for a large number of agents and bids, some sort of heuristics or filters is needed for the screening of bids in order to reduce the complexity. We have incorporated the motivational quantities framework for the task selection process so that agents can make rational decision during their operation. We presented a utility mapping function that can model the agent's preference, promise and penalty policy of the organization. We adapted a statistical model that allows us to predict and analyze the agent's behavior and the influence on the organization utility. We have also attempted at the study of agent's local control - how the utility mapping function of the MQ affects the local time/resource allocation and the agent's overall utility achievement.

Though we have gained a big picture view of virtual organization through this study, we also find there are areas where further studies are needed. For example, we have not explored how the agent make decision in the initial bidding process, how agent should decide whether to bid and how to bid. Analytical work also needs to be done in the statical model to study how to adjust the parameter in the mapping function in order to optimize some organizational objectives. These, of course, would lead us to our future works.

7 Acknowledgment

The authors would like to express their thanks for Victor Lesser from UMass for the permission of using MASS and JAF software. Also thanks to Tom Wagner for his permission of using the MQ Scheduler. Thanks Jiaying Shen for discussion related to the statistical model.

References

[1] Ana Paula Rocha Eugenio Oliveira. Agents advanced features for negotiation in electronic commerce and virtual organisations formation processes. In *AgentLink 2001*, pages 78–97.

[2] Bryan Horling and Victor Lesser. A Reusable Component Architecture for Agent Construction. Computer Science Technical Report TR-98-30, University of Massachusetts at Amherst, May 1998.

[3] Bryan Horling, Regis Vincent, and Victor Lesser. Multi-agent system simulation framework. In *16th IMACS World Congress 2000 on Scientific Computation, Applied Mathematics and Simulation*. EPFL, August 2000.

[4] Daniel E. O'Leary. Virtual organizations: two choice problems. In *Proceedings of the international conference on Information systems*, pages 145–154. Association for Information Systems, 1998.

[5] Jiaying Shen, Xiaoqin Zhang, and Victor Lesser. Degree of Local Cooperation and its Implication on Global Utility. *Proceedings of Third International Joint Conference on Autonomous Agents and MultiAgent Systems (AAMAS 2004)*, July 2004.

[6] Thomas Wagner and Victor Lesser. Relating quantified motivations for organizationally situated agents. In N.R. Jennings and Y. Lespérance, editors, *Intelligent Agents VI (Proceedings of ATAL-99)*, Lecture Notes in Artificial Intelligence. Springer-Verlag, Berlin, 2000.

[7] Thomas Wagner and Victor Lesser. Evolving real-time local agent control for large-scale mas. In J.J. Meyer and M. Tambe, editors, *Intelligent Agents VIII (Proceedings of ATAL-01)*, Lecture Notes in Artificial Intelligence. Springer-Verlag, Berlin, 2002.