14

# A Framework for Business Process Simulation Based on Multi-Agent Cooperation

Wenan Tan[1,3], Wei Xu[2], Fujun Yang[3], Song Li[3] and Yi Du[1]
[1]*School of Computer and Information, Shanghai Second Polytechnic University, Pudong, Shanghai,*
[2]*College of Arts and Science, Jianghan University, Wuhan, Hubei,*
[3]*Institute of Computer Software and Theory, Zhejiang Normal University, Jinhua, Zhejiang,*
*P.R. China*

## 1. Introduction

Most enterprise information systems can be viewed as discrete event systems (DES). Workflow is the computerized business processes. Workflow technology is one of key techniques in current enterprise information system. As the analyzing tool for workflow model, process simulation can support business process reengineering (BPR) effectively. Defects and bottlenecks of enterprise processes can be found through process simulation before the deployment of business management system. After the simulation, considerable reduction of cost is achieved as the result of improved workflow model.

Compared with other workflow research areas, process simulation is still underway. Discrete event simulation is the generic method used in process simulation.

Workflow model is the abstract representation of business processes. Some workflow definition languages have been developed, such as WPDL [1], FDL, and PSL [2]. PSL is the abbreviation of Process Specification Language which defines a neutral representation for manufacturing processes.

XPDL (XML Process Definition Language), which inherits WPDL, is a standard for "interface I" defined by the Workflow Management Coalition (WfMC). The WfMC has identified five functional interfaces to a workflow service as part of its standardization program. "Interface I" is specified to support Process Definition Import and Export, which includes a common meta-model for describing the process definition and an XML schema for the interchange of process definitions. The meta-model defines the objects and attributes contained within a process definition. One of key elements of XPDL is its extensibility to handle information used by a variety of different tools.

To support workflow simulation, we define some XML-based elements containing simulation information or facilitating simulation. These elements, such as <TimeDistribution> (containing information on the distribution of activity's duration, e.g. beta distribution, standard distribution), <Utility> (containing information to facilitate utility calculation, used to support activity execution) can be inserted into other XPDL-derived workflow model schema seamlessly.



Source: Multiagent Systems, Book edited by: Salman Ahmed and Mohd Noh Karsiti,
ISBN 978-3-902613-51-6, pp. 426, February 2009, I-Tech, Vienna, Austria

In this paper, we propose a framework for business process simulation based on multi-agent cooperation. Social rationality of agent is introduced into the proposed framework. Adopting rationality as decision making strategies, flexible scheduling of activities is achieved. On the base of our newly defined XPDL elements, workflow model extended on XPDL can support simulation effectively. WfModel, a prototype application supporting workflow modeling, has been developed. JAXB (Java data binding framework proposed by SUN) and JGraph (an open source Java graphic library) are used to rapidly develop a visual workflow modeling application. The output of MfModel is a XML format compatible to XPDL.

The rest of the paper is organized as follows: Section 2 discusses the discrete event simulation and agent technology. Section 3 proposes a framework for process simulation based on multi-agent cooperation, and discusses the process agent simulation, decision-making strategies, and the multi-agent communication mechanism. A prototype system, namely WfEmulator, has developed to validate the proposed architecture. Section 5 gives a conclusion and discusses future works.

## 2. Related techniques

### 2.1 Discrete event simulation for workflow

Discrete event system is a discrete-state, event-driven system, that is, its state evolution depends entirely on the occurrence of asynchronous discrete events over time. Most systems in life are discrete event systems. Such as: traffic system, manufacturing systems.

Discrete event simulation (DES) is one way of building up models to observe the time based behaviors of an enterprise system. The range of application areas is extremely large and there are numerous examples of the use of simulation in service industries, manufacturing (batch and process) and office environments.

Inside the DES model will be a number of important concepts, namely entities:

- *Entities* are the tangible elements found in the real world.
- *Relations* link entities together, e.g. a part may be processed by a machine.
- *Simulation Executive* is responsible for controlling the time advance and executing discrete events.
- *Random Number Generator* helps to simulate different data coming into the simulation model.
- *Result and Statistics* provides the user a means of utilizing the simulation tool to gain meaningful analysis of the model.

The structure of DES is illustrated as figure 1.

Executive is responsible for ordering the events. The executive removes the first event from the list and executes the relevant model logic. Any new events that occur as a result are inserted on the list at the appropriate point. The cycle is then repeated. A central clock is used to keep track of time.

When DES is used in workflow simulation, activity instances are scheduled by execution. The executive will control the logical relationships between the activity instances and advance the clock to the new time. All the activity instances have to be scheduled by executive according to one rule, e.g. FIFO (first in first out), HPFS (high priority first serve). This mechanism is conflicting to the reality. Different departments in organization do not operate in the same manner. For example, in production department, most of activities are scheduled on time basically, sometimes on priority of work order. While in another

department, some activities may be scheduled by waiting time (retail department). A DES-based simulation system has to alter its scheduling strategies whenever workflow model is changed. A new mechanism should be introduced to implement dynamic scheduling.

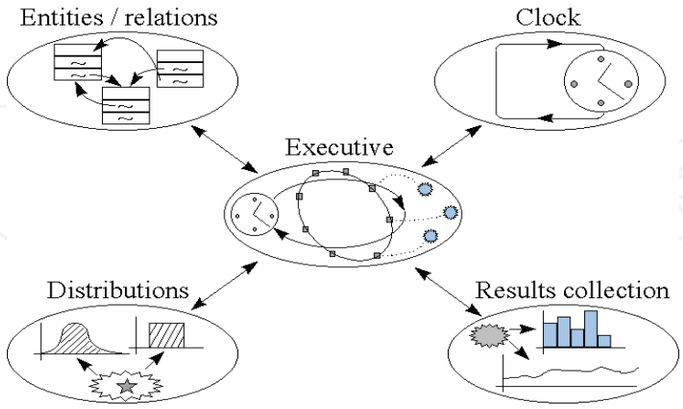


Fig. 1. Structure of discrete event simulation

### 2.2 Multi-agent theory and agent social rationality

Agent is a software entity which functions are proactive and autonomous in a particular environment. Multi-agent system (MAS) is a kind of intelligent system that interconnects separately developed agents, thus enabling the ensemble to function beyond the capabilities of any singular agent in the set-up [3].

There are two fundamental approaches used in modeling multi-agent systems: qualitative (some form of logic, e.g. BDI) and quantitative (e.g. Bayesian). Utility theory is a quantitative one to model MAS. Utility function is a mapping from states of the world to real numbers, indicating the agent's level of happiness with that state of the world. Agents in the competitive MAS potentially have different utility function.

In MAS, as to bounded resources and capability, agent does not stand alone. In accordance with behavior in reality, agent must take action based on certain strategy or rationality. Traditionally, designers have sought to make their agents rational so that they can "do the right thing". Rationality is how the rational decision is made among multiple strategies in the interaction of multi-agent [4].

The predominant theory of rational decision making in agents is that of the economic principle of maximizing the expected gain of actions [5]. Decision theoretic rationality dictates that the agent should choose an action which will maximize the expected utility of performing that action given the probability of reaching a desired state in the world and the desirability of that state [6]. The action that maximizes individual utility may conflict with overall interest (social utility), or redundant actions could be taken due to local utility preference. Hence rationality needs to be considered not only from the individual's point of view, but also from the social perspective. Jennings and Campos proposed the principle of social rationality [7] as follows:

If a member of a respective society can perform an action whose joint benefit is greater than its joint loss, then it may select that action.

Here, joint benefit is defined as the benefit provided to the individual plus the benefit afforded to society as a result of an action. Similarly, joint loss is the individual plus societal loss of performing an action. Social rationality can be expressed as follows:

$$W_i(a_j) = \lambda_i u_i(a_j) + \lambda_{soc} \sum_{k \in \{1-i\}} u_k^{'}(a_j) \quad (1)$$

Where $U_i(a_j)$ is the individual utility of agent $i$ when it takes an action $a_j$,$\lambda_i$ is the weighting given to the individual utility of agent $i$; $\Sigma U_k'(a_j)$ is the sum of utilities of other agents in the system when action $a_j$ is taken by agent $i$ , $\lambda_{soc}$ is the weighting given to the social utility part of the function.

At a coarse level, equation (1) can be rewritten as:

$$U(i,j)=k_1*selfUtility(ps)+k_2*publicUtility(pp). \quad (2)$$

Where $U(i, j)$ is the utility of agent $i$ when it takes action $j$; $k_1$, $k_2$ are the weighting given to individual utility and public utility respectively. *ps* and *pp* are the key influence parameters for individual utility function and public utility function, e.g. activity's duration, waiting time, priority. The values of $k_1$, $k_2$ can be altered to implement a wide range of decision-making strategies [8].

The proposal of social rationality is to ensure the proceeding of task planning when resource competition appears [9]. Social rationality can be used to guide an agent's decisions. In process simulation, when different activity instances could not share limited resources, competition appears. Thus agent social rationality can be introduced into process simulation to represent the decision making strategies of organizations/departments. Related organization will prefer the activity instances which maximize their predefined rationality utility functions.

## 3. Simulation framework based on multi-agent cooperation

### 3.1 Process simulation framework

Multi-agent technique is used to address the issues of complex controls and solutions through intelligent behaviors such as cooperation, competition, coordination in a set of autonomous agents under a dynamic distribution-oriented open environment [10]. These features of multi-agents make them suitable to represent the entities in the enterprise environment. Compared to the discrete event simulation, MAS is more natural to build a simulation framework for enterprise process model.

To achieve flexible simulation structure, we model workflow simulation structure as a MAS, in which agent adopts rational utility functions as its scheduling strategies.

The proposed simulation framework consists of Process agent, Sub-Process agent, Activity agent, Resource agent and Organization agent. Relational database is necessary to support data import and export in simulation process. The framework is illustrated as Figure 2.

Due to the complexity of reasoning, intentional agent can not react rapidly to the change, thus it is not suitable in our simulation system. We use a self-controllable thread to represent the weak notation of agent.

The core agent is process agent, in which maintains two lists: ready activity instance list and pre-running activity instance list. When all the former activities are completed, the instance is ready; when all the needed resources can be satisfied, the ready activity instance are pre-running, and awaiting activity agent. Process agent records the information of all activity agents, e.g. binding organization, utility value of running instance.

Activity instance is a plain object, which contains corresponding information of activity, such as activity's state, priority, utility value. Ready activity instance list and pre-running activity instance list store its reference.

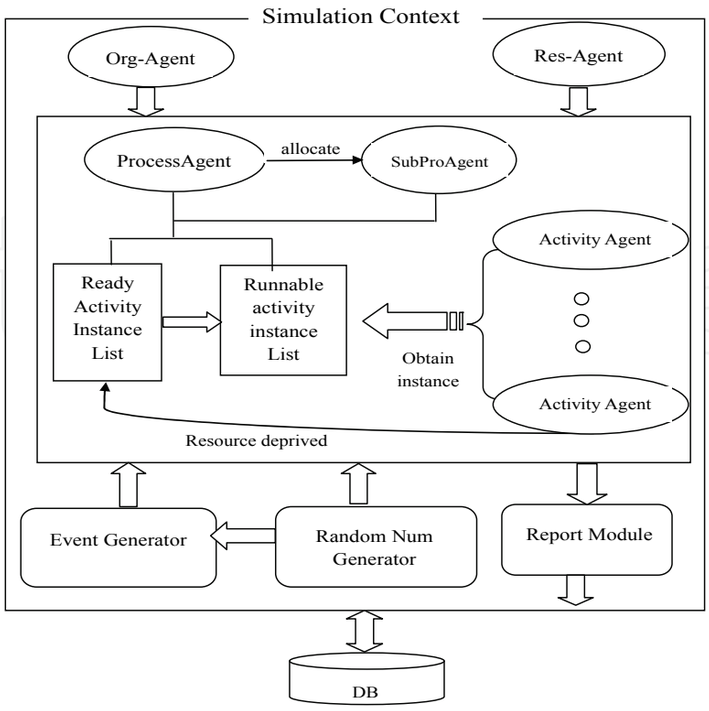


Fig. 2. Workflow simulation framework based on multi-agent cooperation

Activity agent is a continuously-running thread, which obtains activity instance from pre-running activity instance list, simulates the activity, and sends the result data to report module, meanwhile sends message to process agent to inform the end of simulation

Sub-Process agent, who shares the same ready activity instance list and pre-running activity instance list with process agent, is the agent addressing sub-process. If the type of activity instance is sub-Process (XPDL divides activity into three types: plain activity /Implementation, Route, and sub-Process/compound activity), instance will send to this agent. The inner structure of sub-Process agent is similar to process agent.

Resource agent and Organization agent act as infrastructure model in enterprise model. The two agents communicate with process agent and sub-Process agent, and monitor ready activity instance list and pre-running activity instance List. Organization agent understands all the organizational details and is responsible for the calculation of activity instance's utility.

*Event Generator* is used to create event message, which is sent to *Process agent* to activate simulation. More than one unrelated process could be defined in one process model, and different messages will activate different process.

*Random Number Generator* is used in the calculation of working time of activity instance. Different time distribution (e.g. Beta, Standard) depends on different random number generation algorithm.

*Report Module* is a common object collecting statistic information shown in dialog boxes.

*Simulation Context* is the body of simulation, which is responsible to create all the agents used in simulation.

The main steps in simulation are shown in Figure 3. The core step, process agent simulation is as follows:

1. Process agent import the data stored in database, which is converted from workflow model formatted in XML.
2. Event Generator creates message to inform process agent that one simulation is activated. Process agent instantiate Start Activity, and put it into Ready Activity instance List. Resource agent and organization agent, which monitor the two lists, will be messaged and put the instance that can be satisfied resource requirements into Pre-running Activity instance List.
3. Activity agent obtain instance from Pre-running Activity instance List, register itself in process agent, and resources are occupied.
4. Activity agent runs simulation on the instance. Random Number Generator gives its working time according to the time distribution defined in the workflow model.
5. When simulation of the instance ends, statistic data are sent to report module, and process agent is informed.
6. Once received end message from an activity agent, process agent queries database to get subsequent activities of the instance, instantiate them and put them into Ready Activity instance List.
7. Event Generator create SIMULATION_START event continuously until predefined end-simulation- condition is satisfied.

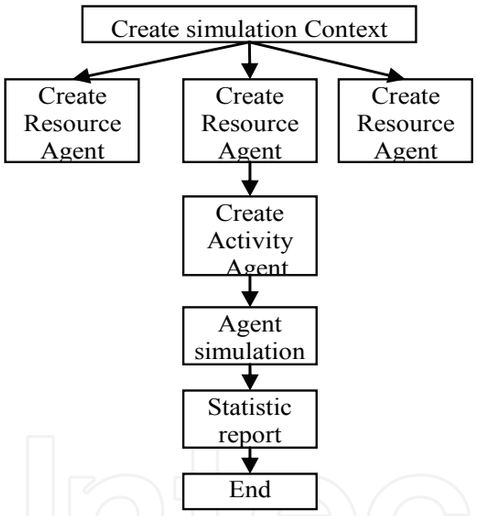


Fig. 3. Main procedure of simulation

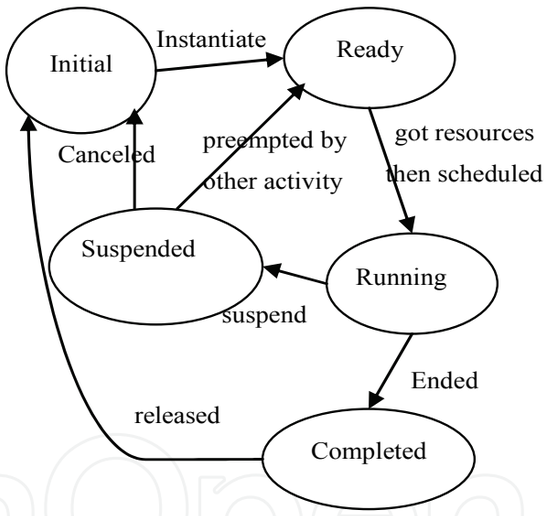


Fig. 4. State transition in process simulation

In the simulation framework, the states of agent include: Initial, ready, running, suspended, complete. States are explained as follows:

- *Initial* is the state after the activity is instantiated. As we mentioned before, activity instance is a plain object containing simulation information of the activity.
- *Ready* is the state that all the former activities of current instance are completed,
- *Running* is the state that an instance is in simulation by an activity agent. A running instance can be switched to ready when it is replaced by a higher-utility instance.
- *Suspended* state is used to support the pause operation during simulation procedure.
- *Complete* is the state that the activity instance finishes its simulation. The instance object will be disposed.

Figure 4 illustrates the relation between them. In workflow management system, there is an activating state (work-item put into working). While in activating, activity instance could not be suspended or completed. In simulation, what we care is the running features, and work-item does not work, therefore activating state is out of consideration.

When an activity instance is running, it can be replaced by another instance whose utility is higher. Then the instance's state will switch to ready, and the instance is put into Ready Activity instance List. Related information (e.g. running time, left working time) are updated in the instance. Process agent maintains the relation between activity agent and organization, as well as the utility value of running instance.

A running activity instance can be replaced, which is similar to the preemptive priority scheduling in computer operation system. When an activity instance (called A) is shifted to *Pre-running Activity instance List*, process agent check its utility value and performer, then query the registered activity agents. If the same performer is found (the two activity instance are performed by the same organization), the instance, which is running by the found activity agent (called B), may be replaced by instance A. The preemptive condition is that A's utility is larger then B's. For example, a production activity can be stopped to a new coming work order with higher priority (utility). Practical influence parameter may be activity's duration, cost, etc., but all these influence parameters are in utility function for different organizations.

In the framework, there is no global clock. Resources and roles are initiated, who maintain their own time. Here we explain the clock management in the framework in the following figure 5.

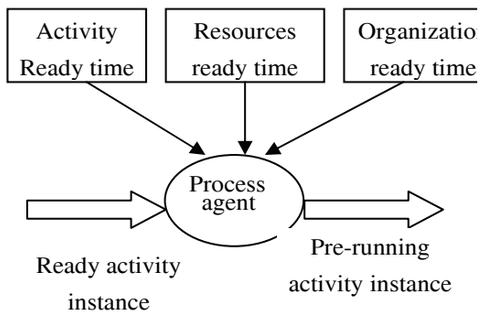


Fig. 5. Time management in the framework

When an activity instance is ready, that is, all the former activities are complete; *Activity Ready Time* is the latest complete time of these former activities. Before running ready activity, the required resources and role/role-set (in the organization) instances have to be free, so that, it can be allocated to the activity. Corresponding start time of resources and organization is the active time of the activity. When an activity is complete, resources and role instances are released, this time is the end time of service of resource agent and organization agent.

### 3.2 Decision making strategies

It's generally believed that different department has its own interest in an organization. Sometimes local interest of inner organization conflicts with overall interest of the organization. Therefore organization/ department can be described as a competitive agent,

or a self-interested agent. Utility function defined on social rationality is an appropriate form to represent the decision making strategies of organization.

An agent self-interested neither means that it wants to harm other agent nor means it only cares about things that benefit it. In simulation, resource competition is in evitable because the bounded resource could not satisfy all the needs. Since a resource always belongs to an organization/ department, the allocation of resource is determined by the interest of the organization. When two or more activities competitive on one resource, the organization agent calculates the utility value based on certain index predefined in activity and $k_1$, $k_2$ defined in the organization. The parameters $k_1$, $k_2$, which represent the orientation of organization, are constants determined through the consultation between simulation engineers and customers, and modified in a few simulation procedures.

In our implemented system, *selfUtility* and *publicUtility* are defined in <ExtendedAttributes> in each <activity> element of workflow model. <ExtendedAttribute>, which contains few <name, value> pairs, is widely used in XPDL. The utility can be a constant, or a key factor affecting the activity radically, such as the number of a specific resource, working time of the activity. For example, in the repertory department, the cubage is bounded, and deliveries should be done in time to contain newly produced products. When the delivery of the day is already scheduled, the repertory department prefers delivery according to the cubage of the delivery products. As a result, delivery priority or delivery time is the key factor in *publicUtility*, and the cubage becomes the key factor in *selfUtility*. Combined with $k_1$, $k_2$ defined in repertory, the utility of delivery activity can be calculated.

Once there is no conflict between local interest and overall interest, social rationality of agent degenerates into a singular-parameter utility function. For example, in a sale department, employee's salary is affected by sale amount directly. Receiving more customers is the same preference of local interest and overall target. Thus activities can be scheduled based on working time or customer priority.

In our simulation framework, organization agent is responsible to calculate utility. When the two conflicting instances' utility values are equal, maximum waiting time first serve or random choice can be used to determine the choice.

The steps dealing with resource competition are as follows:

1. When activity instance is put into *Ready Activity instance List*, resource agent and organization agent are informed.
2. Organization agent checks all the ready activity instances in the list and finds out all instances whose requirements in role/organization can be met.
3. Finding out conflicting instances. Only activities need to be supported by same organization or resources, conflict will appear.
4. Comparing the utility values of conflicting instances. We need to choose the instance whose value is maximal.
5. Process agent tries to allocate resource agents to the activity instances in step 4.
6. Instances whose requirements in resource are put into *Pre-running Activity instance List*.

### 3.3 Multi-agent communication mechanism

Multi-Agent coordination mechanism is achieved through the communications and information sharing between the agents [11]. The main communication mechanism is the basis on collaboration, with the current means of communication blackboard system, which is an expansion system from the agenda systems and artificial intelligence systems of the

expert system. The problem solving means of the coordination mechanism is the use of appropriate structural support distributed of the agenda. In the Multi-Agent workflow simulation system, we adopted the blackboard communication mechanism which is the provision of public work area for all agents to communicate with each other in the system. The agents can exchange information, data and knowledge with each other by the public blackboard communication mechanism. A message is written by an agent on the blackboard. The other agents can access the blackboard to read the message. When the message is the service that some agent wants, the agent should send a message to the writer, Meanwhile, a relationship message should be written to the blackboard to show the service used. An agent may access the blackboard at any time. The communication mechanism is complement by the service configuration message shown in Figure 6.
After an agent read a message from the blackboard, the agent core that is the will be activated by the message. The core is the main method or function of the call nested in coordinated inter-related tasks.

```
Struct Service{
Agent ID Source; //The service name of the source
                        agent;
Agent IP SourceIP;    //The service IP address of
                                the source agent;
Unsigned Servicetype; // Service type;
Cstring Servicename; // Service name;
Cstring Servicecontent; // Service content;
Cstring Servicepriority; // Service priority;
Cstring ServiceResource; // Service resource.
……
}
```

Fig. 6. Communication configuration message

Overall, the strengths of this approach are that the finally constructed methodology is highly attuned to system conditions and the agents in the Workflow system. The challenges are to construct the several blackboards, ensuring that a linkages accord to the local situation and that the interfaces of any pair of method fragments to be plugged together are compatible. Both of these can be facilitated by the use of software tools, the former with a process construction tool, both of which we have developed.

### 3.4 Prototype system –WfEmulator

To validate our framework, a prototype system, WfEmulator, has been designed and developed, which is written in Java. Workflow model generated by WfModel can be simulated in our prototype application. The architecture of prototype system is shown in Figure 7.
In the architecture, Wfemulator is mainly composed of four parties: *User Interface, Process Center, Wfmodel Center,* and *Simulation Engine*. *Process Center* and *Simulation Engine* form the core of the simulation system.

- *User Interface:* the module shows the Tools Bar and Menu Bar to users, The Tools bar includes many modeling figures, chars and so on to model a working follow of a

system. The Menu Bar includes load action, save action, operation action and system setting, and so on.

- *Process Center:* the module is the operation module of the system. Process Agent is the defined many current process rules. When a working following is defined, the Process Agent should creative the relevant relations for the every operation of the working following. Resource Agent manages all of the resource of the system, such as resource assignment, resource statistics, and resource effective use analysis. Activity Agent is created by the sup-process agent. An activity takes charge in one activity from the beginning to the end of the activity.
- *Wfmodel Center:* simulation configuration is the initialization of the simulation job. There are many ex-defined working following models on different operation process in various enterprises. Initially, users are either allowed to open existent working following model or define a new one and save it in the model database.

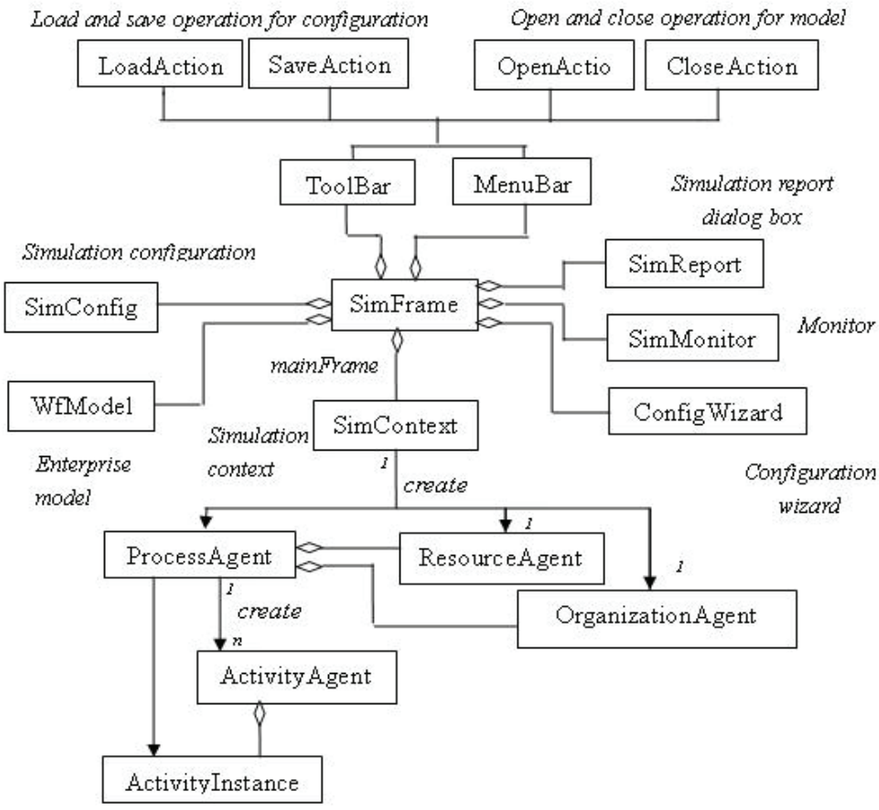


Fig. 7. Architecture of Wfemulator

- *Simulation Engine:* the module is the control center of the system. Many arithmetic, schedule rules, and analysis means are defined in the module which drives the whole of the operation of the simulation process. Simulation Report shows the simulation result and advice to the users. Simulation Monitor cooperates with the operations and communication of the different agents.

Workflow model used in our prototype system is generated by WfModel, which generates a XPDL-compatible model in XML format. Simulation configuration is an XML file containing information on simulation, e.g. halt condition, resource instantiation, activity agent number.

## 4. Conclusion and future works

We analyze discrete event simulation for workflow, and propose a framework for process simulation based on multi-agent to support flexible activity scheduling. Social rationality of agent is used to represent the utility function as the decision making strategies of organization. Physical elements such as cubage, cost can be imported to schedule activity instances, which makes simulation running in a more flexible and realistic manner. Currently, we have proposed a framework consisting of socially rational agent. If the agent has the ability to learn from previous decision making successes and failures, there is the potential that it could converge towards finding the right balance, depending on the situation, between individual and social needs. Hence a knowledge mechanism can be imported into our simulation framework to accelerate decision making when similar situation appears again.

Furthermore, due to the autonomy of agent, the number of activity agent may be varied and the activities can be distributed to a number of computers, according to the computational resources. Multiple sub-process simulations can run synchronously to hasten complex process simulation. Thus a distributed simulation environment becomes a potential alternative. JNDI and RMI can be used to help the communication among agents in this distributed simulation system.

## 5. Acknowledgment

This paper was supported by the National Natural Science Foundation of China under Grant No. 60874120 the Zhejiang provincial Natural Science Foundation of China (Grant No. Y106039), and the Science Foundation of Nanjing University of Aeronautics and Astronautics (Grant No.S0777-042).

## 6. References

[1] Workflow Process Definition Interface— XML Process Definition Language. Workflow Management Coalition Document Number WFMC-TC-1025

[2] Gruninger M., Menzel C.: The Process Specification Language. Principles and Applications, AI Magazine, 24(2003) 63-74

[3] Kary F., Timo A.R., Mikko K., Jan H.: Agent-based model for managing composite product information. Computers in Industry, 57 (2006) 72–81

[4] Russell S.: Rationality and intelligence. Artificial Intelligence, 94 (1997) 55-57

[5] J. Doyle.: Rationality and its role in reasoning. Computational Intelligence, 8 (1992) 376-409

[6] Hogg L.M., Jennings N.R.: Social rational agents- some preliminary Thoughts. Proc. of the 2nd Workshop on Practical Reasoning and Rationality, UK, Manchester, (1997) 160-162

[7] Jennings N.R., Campos J.: Toward a social level characterization of socially responsible Agent, Proc. IEEE Software Eng., 144 (1997) 11-25

[8] Hogg L., Jennings N.R.: Variable Sociability in Agent-Based Decision Making. Proc. of 6th Int. Workshop on Agent Theories Architectures and Languages (ATAL-99), (1999) 305-318

[9] Chen X.Y., Shi C.Y.: Research on Socially Rationality of agent. Journal of Software, 12(12) (2001) 1825-1829
[10] John R., Cathal H.: Process modeling for simulation. Computers in Industry, 57(2006), 437–450
[11] Tan W., Fan Y.: Dynamic workflow model fragmentation for distributed execution. Computers in Industry, 58, (2007) 381-391.


**Multiagent Systems**

Edited by Salman Ahmed and Mohd Noh Karsiti

ISBN 978-3-902613-51-6
Hard cover, 426 pages
**Publisher** I-Tech Education and Publishing
**Published online** 01, January, 2009
**Published in print edition** January, 2009

Multi agent systems involve a team of agents working together socially to accomplish a task. An agent can be social in many ways. One is when an agent helps others in solving complex problems. The field of multi agent systems investigates the process underlying distributed problem solving and designs some protocols and mechanisms involved in this process. This book presents an overview of some of the research issues in the field of multi agents. It is a presentation of a combination of different research issues which are pursued by researchers in the domain of multi agent systems as they are one of the best ways to understand and model human societies and behaviours. In fact, such systems are the systems of the future.



**InTech Europe**
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

**InTech China**
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

IntechOpen