Turkish Journal of Electrical Engineering and Computer Sciences

Volume 12 | Number 2 Article 3

1-1-2004

A Platform for Software Engineering Course Projects

BİROL AYGÜN

Turk J Elec Engin, VOL.12, NO.2 2004, © TÜBİTAK

A Platform for Software Engineering Course Projects

Birol AYGÜN
Yeditepe University, School of Engineering and Architecture, Dept. of Computer Engineering, 34755 Kayışdağı, İstanbul-TURKEY
e-mail: baygun@cse.yeditepe.edu.tr

Abstract

The importance of projects in software engineering courses is well known. Both synthetic and real-life projects have various advantages and disadvantages. Our aim was to create a framework where students can develop projects which reflect some of the complexities of real-life, involving many concurrent, interacting, asynchronous processes, each in a different stage of development, with wide temporal differences among them - some occurring within millisconds of each other and others executing sporadically over much longer periods. In this project, which was carried out in different arrangements in several software engineering courses in three universities, the students developed both the sub- and super-structures required, with varying degrees of success. The projects were performed in parallel with and subsequent to one-semester courses in software engineering. The development was performed in accordance with the principles established in the lectures. The sub-structure consists of a discrete-time event simulator and a message passing mechanism. It can support many different super-structures. The super-structure we created was an e-business community simulator where a manufacturer, its trading partners and the transactions among them were simulated. In this paper we summarize the project and our experiences during the development. A literature search for similar projects for software engineering education did not yield any hits. However several reported industrial projects for virtual supply chain management were examined. Our project on the other hand was tailored for implementation by student groups in one semester with the primary purpose of getting experience in complex, multi-group software development rather than immediate industrial use of the software.

1. Introduction

The primary audience of this paper is software engineering educators. The primary emphasis is on a particular education technology, not advanced technology or research in e-business systems per se.

Given the importance of project-based work in introductory software engineering courses, we are often inclined to believe that real-life projects, i.e. those which come from an actual user community, such as a company or public organization, would be preferable to synthetic projects, developed expressly for the course. As is well known by software engineering instructors however, real-life projects offer several difficulties which make them less advantageous than they seem. Some of these are:

- The organization owning the project may not be able to designate an official to work with the academic project who can devote sufficient time and effort to it,

- The project may be too complex or too routine to be used in the course,
- The project may not be sufficiently defined as yet by the owner organization for students to start work on it immediately,
- The hardware and software requirements of the project may not be available in the university,
- Meetings may be difficult to arrange, which will hinder project's progress.

A synthetic project on the other hand, can be a real-life project adapted to the conditions of the course such as complexity, number of workers, hardware requirements, length of development and other attributes. It might be argued that students will take a real-life project more seriously. However an instructor with sufficient real-life experience can add the required realism to the definition of the project and the development process to make it feel life-like.

The projects we developed were based on a real-life like scenario adapted to numbers of students in each class, the length of the term, the level of experience of the students and the software platforms available. A number of features that go beyond those used in the industry were incorporated and some feature details which would be required in a real-life project but would not contribute to the experience gained by the students were excluded. In the development process, our priority was for the students to gain the experience of working in a group, in fact, a multi-group project. Thus strong inter-group communication and coordination were required. We also wanted the students to use relatively state-of-the-art tools to gain an additional benefit for working on the project. To clarify some terminology, by "project" we mean a typical one of the several similar projects developed by independent student groups, having some variations such as the platform or implementation techniques. Each organizational unit (a company or a department) simulated in the project is called a "sub-project" and assigned to a specific group of 3-6 members. As a result there were nine sub-projects in each project, one being the Master Process which implemented the sub-structure described below, the others being application programs representing the various organizations included in the simulation.

In the literature, we did not find any reports of projects using a similar approach in software engineering education. However, there are reports of discrete-event simulation being used in real-world applications [3,4].

2. General Description of the Project

The central idea in the project, as title states, was to create a framework in which the interactions among various organizational units can be simulated based on a distributed process model with possibly very different scales of time, some interactions taking milliseconds, others hours or days. This subject may be considered an adaptation of the author's earlier work related to dynamic analysis of program behaviour by simulation of the machine and programs' execution on the machine. Details can be found in [1,2]. A literature search of undergraduate software engineering course project reports yielded no projects of a similar nature. However a number of industry projects on the subjects of virtual supply chains and virtual enterprises using discrete-event simulation have been reported ([3,4]). The latter includes a long list of other references. Our approach was tailored for a one-semester undergraduate course where the load can be distributed relatively evenly across groups. Therefore we examined the industrial project reports for ideas but decided to create our own optimized approach.

This type of simulation requires a time-grain independent scheduler and a discrete time-simulator which can activate a process anywhere in a network at a desired time. We also required that processes communicate by a standard interface. We chose to use XML formats agreed upon by the communicating parties for messages between themselves. Given such a sub-structure, super-structures simulating different organizations or scenarios could be developed. The super-structure we created for the project is illustrated in Figure 1.

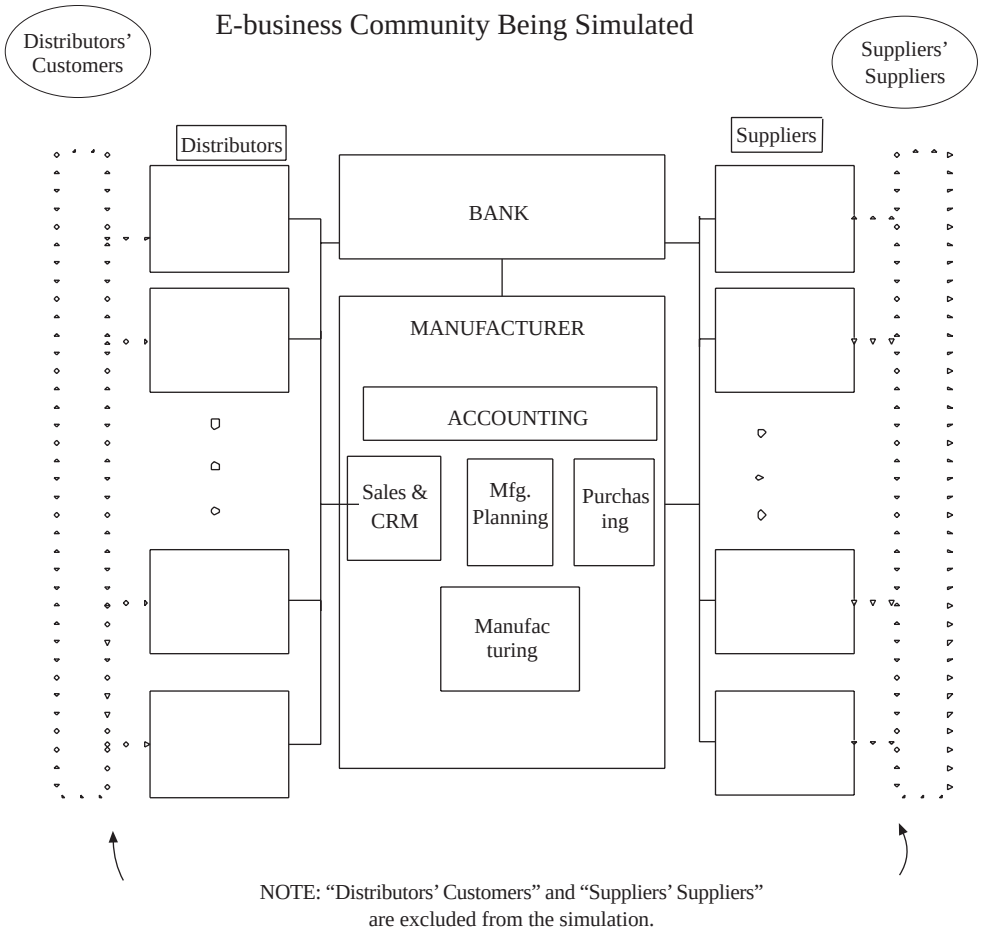

Figure 1. The super-structure of the project.

In the super-structure developed, a simulation of an e-business community has been created. The community consists of a manufacturing company (in the actual project it was a manufacturer of hard-disks), its suppliers of parts and customers (i.e. distributors of harddisks) and a bank in which all the companies have an account. The bank pays the customers' bills automatically from their accounts. Detailed provisions are provided in case where there are insufficient funds in the buyer's account. Other business-related features include rating of suppliers based on observed (simulated) failures of parts purchased from them, awarding premiums to distributors who surpassed their sales quotas, etc. In short, quite a bit of detail is included in every application to make them life-like, indeed more like a vision of how e-business firms might operate in the near future. It is possible that if the simulation component were removed, the application programs can be expanded to become the nucleus of a real-life e-business software. Documents exchanged between every pair of stakeholders is has a unique type code and serial number. Every document is acknowledged by its receiver. Thus it is possible to check if there were any missing links in the document chains.

The application sub-projects were described rather generally in the initial project description document handed out to the development groups. The students were then required to augment their respective sub-project descriptions through their own research and group discusssion. The groups were then required to submit a Project Plan followed by a Software Requirement Specifications (SRS) for their sub-projects using a format handed out in the lectures, and present them to the class. A part of the SRS describes the content of the interfaces with other groups. Often there was a lot of feedback from the audience and the SRSs had to be revised multiple times. Similarly, at the end of the design phase, each group had to submit and present a Design Document, which went through a design review in class. This document included a detailed description of the group interfaces in XML format. The groups also presented their test plan and test results documents in their unit test phase. An effort was made to dovetail these presentations with the coverage of the subject in the lectures.

An important point about testing in multi-group projects is that some of the groups with which a particular group interfaces may not yet be ready to provide the interface required for the test. Therefore the groups were required to simulate arrival of documents from other groups based on the interface descriptions provided in the SRS and Design Documents by creating those documents manually. This way each group could proceed with development until the sub-structure was ready to be used by sub-projects.

The project has been implemented by three different groups: one in Windows .NET and two different implementations in Linux.

3. The Sub-Structure of the Project

The sub-structure of the project is illustrated below in Figure 2. As can be seen, it consists of three concentric layers. Proceeding from inside out, these are:

1. The Time-based Simulation Layer,

2. The Messaging Layer,

3. The Application Programs Layer.

The innermost two layers are implemented by the Master Process (MP) sub-project. The third layer is implemented by the application program sub-projects.

3.1. The time-based simulation layer

This layer receives event notices from the application programs indicating when they would like to be awakened in the timestamp field of the event notice. The time is of course simulated time; it could be thirty seconds, or one day later, or 8:00 A.M. next Friday. The event notice is described in more detail below. The Master Process keeps track of legal holidays so that erroneous event notices falling on those days can be returned. It does not keep track of the beginning and ending times of shifts; it leaves that up to the application programs.

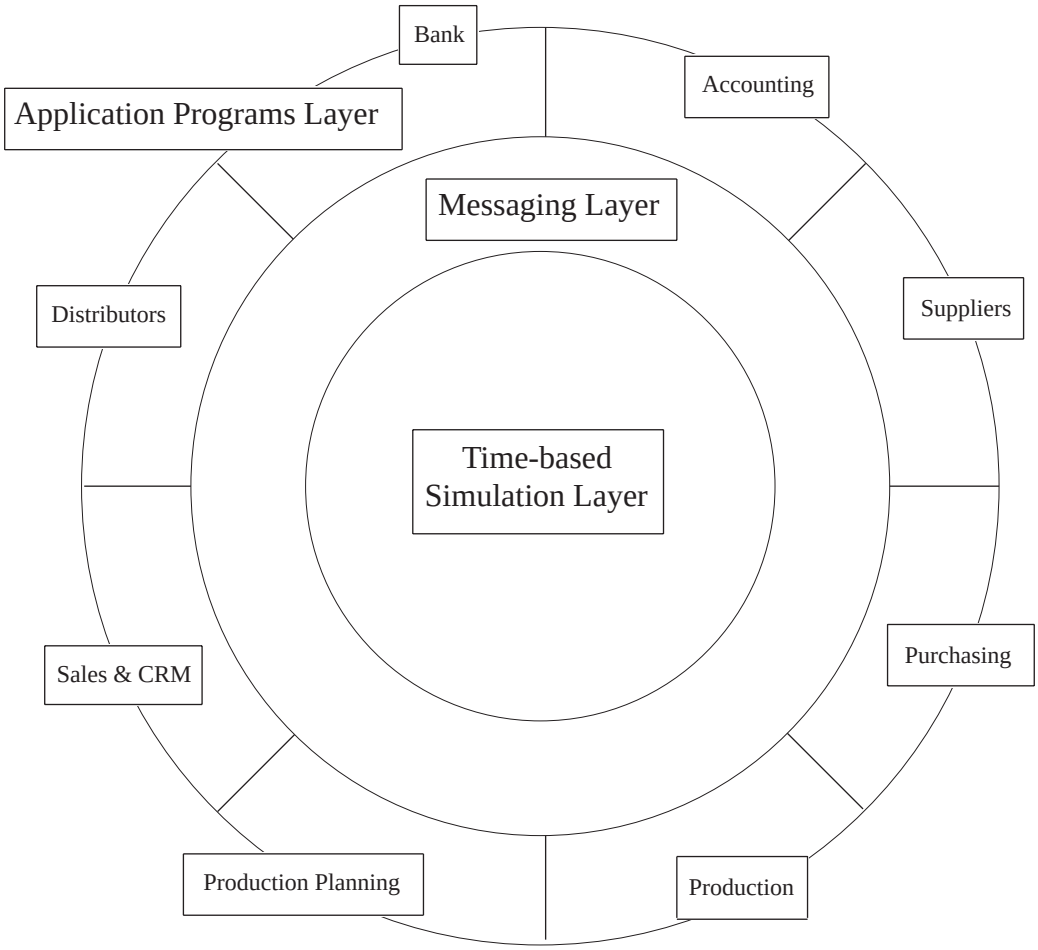

Figure 2. The sub-structure of the project.

3.1.1. Activation and synchronization of processes

All the processes may be on different machines or some may be on one machine. They communicate with each other using XML messages representing business documents such as orders, invoices, bank account statements, payment overdue notices etc. Timestamps may be in "relative time" format (e.g. 1 day, 12 hours and 5 minutes from now) or "absolute time" format (e.g. "10/9/2003 0805").

When the Master Process (MP) wants to activate a process as a result of an event notice, it does the following:

1. Locate the earliest event notice.
2. Check if all processes are idle.

If there is an active process, it is possible that it can issue an earlier event notice than the current event notice selected. Therefore MP sends a poll to see if they have an event notice to posts with a pre-set time-out period for response. If it does not get a reply in the allowed time, it tries this two more times. If it still doesn't get a reply, the process is assumed to be dead or malfunctioning. The MP writes a message to the **Event Log.** That process is ignored and the MP checks for other active other processes, handling each one as described above. When the MP is convinced that no other processes can send an event notice

for an earlier time than the earliest event notice in the schedule, it awakens the target process indicated in the event notice and gives the event notice to it. Thus a sending process can communicate with the receiver in two ways: the XML file being sent and the parameters in the event notice. The timestamp is present in the event notice; however the processes can also ask for the current time from MP.

All the event notices and error messages are written to an Event Log which can be queried for debugging or reporting purposes. In some cases the Event Log has been implemented a database file, in others a sequential XML file. If a process is presumed dead, as mentioned above, all processes sooner or later run out of things to do since orders can not be completed and that tends to prevent new orders from being generated, and thus the schedule becomes empty and the simulation ends.

Issuing Process ID	Target Process ID	Synchroni zation Code	Time stamp	Parameters

Figure 3. The format of an event notice.

3.1.2. The event notice

The format of an Event Notice is given below.

Issuing Process ID: The ID of the process issuing the Event Notice,

Target Process ID: The ID of the process to be awakened at the time indicated in the Timestamp field. This could be the same process as the Issuing Process.

Synchronization Code: "B" = Busy, "I" = Idle. If this code is "B", the issuing process will be marked "Busy" after the event notice is put in the schedule, indicating it has other work to do and that it may issue more event notices before it goes idle. If the code is "I", the process is marked "Idle", meaning it has no other work to do at this time.

Timestamp: The time at which the Target Process should be awakened, in absolute or relative time format

Parameters: Additional information that needs to be passed to the target process.

3.2. The messaging layer

This layer provides for transfer of XML files among application processes. Each application process has an input box and an output box. It also has a Listener Process, a Main process and an Output process. The Listener Process constantly monitors the input box looking for messages. When it finds one or more, it sends a confirmation for each one, and passes the messages on to the Main Process, which performs all the processing required by the messages, possibly using multiple threads for independent activities, and creates output XML files to be sent to the next process, if any. It then sends one or more event notices to MP to be awakened again at certain times to send out those messages. The time whose passage is simulated by two successive event notices corresponds to the time it would take in real life to perform the associated task(s) and it is usually randomized around an average time. For example, if the incoming message was an order for goods sent by a distributor to the Sales and Customer Relations Management department, the time passage to be simulated would correspond to the time it would take in real life to the processing of the order in that department before the order is posted in a daily order list or sent to Manufacturing Planning, depending on how the department operates. In the latter case, the department would create an XML document passing the order on to Manufacturing Planning. When the Main Process of the Sales and CRM department is

re-awakened as a result of its event notice, it calls its Output Process to send the XML files created as a result of the task to the receiver processes' input boxes.

3.3. The application programs layer

This layer contains the code for all the organizational units taking part in the simulation. These programs have been written in Java for Linux and C# for NET. These programs communicate with each other by sending business documents in XML format and with the MP through event notices and error messages when the latter arise. The structure of a typical application program is shown below in Figure 4.

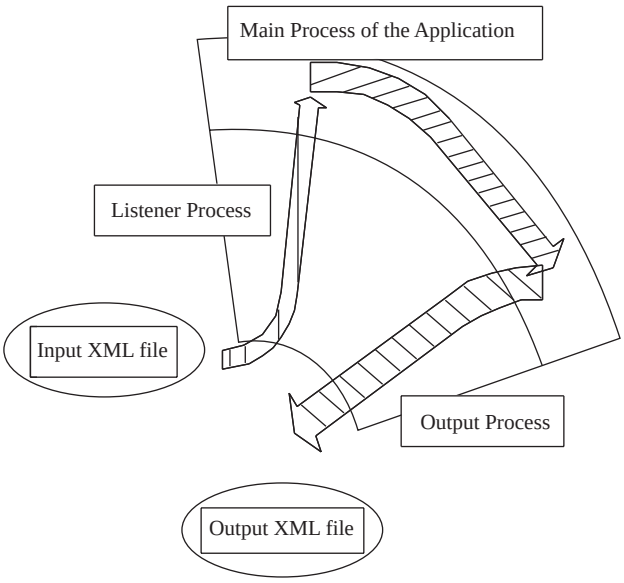

Figure 4. Structure of a typical application.

4. System Testing

It is generally difficult to test time-based simulations thoroughly due to the complexity presented by the temporal relationships of the random interactions among the many asynchronous processes. The same is true in this case, compounded by the fact that many different scenarios are needed to cover all the business cases which can arise. Some of these are as follows:

The distributors, who sell goods to dealers, keep a certain amount of stock of their own. If new orders can be met from this stock, the distributor has the choice of doing so or ordering new stock at current prices,

To meet demand for goods, the Manufacturing Planning department can use a number of different policies involving amount of stock on hand, minimum order levels and average order delivery period of various suppliers, the current workload on manufacturing, and possibly other criteria.

A large number of test cases were designed to cover some of these possibilities, but only a few could be tested due to academic time constraints.

The simulation was designed to run automatically to completion with pre-specified initial customer order inputs. However, for testing purposes members of all groups were present to monitor their application's execution and to intervene if needed, e.g. when there is a system problem in the network or one of the machines. We should point out that overall (black-box) system test plan design is an important area which

should be carefully planned due to the complicating factors mentioned above in a time-based simulation approach to group projects. The test case design itself becomes an important learning tool for the students.

5. Composition, Operation and Grading of Development Groups

Members of groups were chosen at random in principle. The reason for this was to minimize complaints later that particular group was favored or biased against, and to reflect the fact that in real life you often can not select the teammates you have to work with. Once the initial groups were formed, exchanging members between groups to facilitate collaboration, such as arranging meetings, was permitted. Only the Master Process group was hand-picked due to the higher technical complexity of that sub-project and its importance to get the simulation to run. That proved to be a good decision. However, it did not go far enough. It would have been better if we could get the Master Process group start work, say three weeks, ahead of the other groups.

There were other difficulties, as is common in group projects, to get the groups to work together and in assigning grades. The approach to handling the first problem was to make some class time in addition to lab time,as well as group mail and a web site available for communication. The second problem was handled by, in principle, giving the same grade to all members and enforcing division of labor as much as possible in all phases of the project, including the requirements analysis, design, development testing and documentation. Each of these phases were graded seperately. Members who grossly violated this arrangement were given lower and sometimes failing grades. It was felt that trying to assess the individual contributions of all the members in detail would create more heat than light.

6. Development Platforms

As mentioned, two different platforms were used: Linux and .NET. In the Linux platform, the project was developed using the Jakarta/Tomcat server, Java, MySQL, SOAP and XMLRPC. One group tried using PHP for its application program but although the program was developed without much problem, there were difficuities in getting the process to be awakened by the MP which is coded in Java, and in using the Java-based parsers to parse the XML files.

In the case of .NET, after much investigation, it was decided to assign a Web server to each application program as well as the MP, and to use the Web Service approach to invoke their services. On hindsight this does not seem unreasonable since each application program represents an independent set of threads possibly running on different servers in real life.

Almost 200 project documents were produced and 200KLOC were produced over four semesters and three different versions of the project.

7. Evaluations and The Learning Experience Achieved

Sevaral evaluation mechanisms were used to evaluate the effectiveness of this approach in the software engineering project course.

One is the traditional course evaluation mechanism used by the universities. Another is a specific student feedback session used in place of the final exam in one of the sections.

The responses in these two types of evaluation fell in three categories: Some students wanted more technical instruction on the tools used. Indeed this was a fair criticism since a number of new tools were used

on which not all the students had used before. This included the Java parsers, Java itself, SQL database, XML and of course SOAP on Linux, and on the .NET side, C#, Web service and XML. Of course the upside of this is that if it weren't for this project, they might not have had a chance to use these tools.

A second category of criticisms were related to difficulties in intra- and inter-group communication and collaboration. We tried not to interfere in this area during development but provided class-time and resources to accomplish these. We feel that this too was a good lesson for them to learn about working in group and multi-group projects.

The third category was one of high applaudit - where the students said this was the best and most life-like project they had worked on and that they believed it gave them a good feel for what to expect when they go to work. Our feeling is that the project went beyond what most will probably face in their initial assignments at work, and that it gave them a vision which will be helpful for quite a while to come for many of them–which in fact was the aim of the project.

There are two other ways in which the educational effectiveness of this approach in teaching of a software engineering project course can be evaluated:

1– The guidance and help it provides graduates in, especially the beginning part of, their careers,

2– Comparison of the achievement of students who took this course using this approach, and those that didn't.

We are unable to make the first evaluation on a systematic basis. However, some feedback from graduates indicates that they got a very good exposure to project development from this course. In fact, some felt that the project environments they worked in was much more informal and less rigorous than what they experienced in our project.

The latter kind of evaluation is done by comparing the understanding of the students who took part in this project with regard to how to approach development of such an information system as opposed to those who did not use simulation-based approach, based on other similar projects, indicates that the former group in fact got a better understanding of the overall process.

8. Future Work

We conjecture that this project possibly provides another software development process model for top-down development of distributed sets of applications where there are very diverse lengths of time between executions of processes – e.g. where some execute constantly, some are user-initiated, some are periodical and some are quite asynchronous, executing only when conditions warrant. Our event-notice and time-based simulation-based approach with well-defined process interfaces provides a method where interaction of these application processes, each of which may be in a different stage of development, can be observed, tested and debugged.

Future work associated with this project will proceed in two directions:

1– Training new software engineering students to use the framework to develop new applications suitable for time-based simulation and to implement software engineering tools for testing, performance measurement, debugging and case analysis.

2– To cooperate with other departments or schools in the universities to find new application areas in their curricula which can be modelled using this approach. Obviously, due to the first application area we developed, business information systems would be a likely candidate. Other groups which have processes which interact over differing periods of time may find this framework useful since we are able to accomodate such processes and provide a standard way of communication among them regardless of the length of time

between their interactions.

Acknowledgements

The real heroes of the projects are, of course, the students. I thank my students in all three universities and my graduate assistants Sadi Şeker Evren (Yeditepe University), Zerrin Işık and Murat Genç (Sabancı University) and Coşkun Gündüz (İstanbul Bilgi University) for their contributions to the project.

References

[1] B.O. Aygun, Dynamic Analysis of Execution: Techniques and Problems, Ph.D. thesis, Carnegie-Mellon University, Department of Computer Science, 1973.

[2] B.O. Aygun, "Environments for Dynamic Analysis of Program Behaviour", Proceedings of Conference on Simulation of Computer Systems, 1973.

[3] J.A. Joines, R.R. Barton, K. Kang, and P.A. Fishwick, eds. Proceedings of the 2000 Winter Simulation Conference: Distributed Supply Chain Simulation Across Enterprise Boundaries.

[4] S. Umeda, A. Jones, Virtual Supply Chain Management : A Re-engineering Approach Using Discrete Event Simulation, National Inst.of Standards, Maryland, USA.