Workshop on Databases and Software Engineering

A Capacity Measurement of Workflow Management Systems

Jian-Chang Cheng, Jian-Wei Wang, Feng-Jian Wang
Department of Computer Science and Information Engineering, National Chiao Tung University
Room 510, EC Building, 1001 Ta-Hsueh Road, Hsinchu City, Taiwan, ROC
Telephone: (03)5712121 x 54718
Fax: (03)5724176
Email: {jccheng,jwwang,fjwang}@csie.nctu.edu.tw

Abstract

Workflow management systems (WFMSs) are the essential software systems that facilitate the automation of business processes. The capacity of a WFMS is the maximum throughput that the WFMS provides with acceptable performance. The information of capacity is not only valuable for system administrators to tune the system, but also helpful for managers to make a decision in WFMS procurement. The objective of the paper is to develop a method for measuring the capacity of a WFMS with certain combinations of hardware, software, and workflow applications. Besides, a set of software tools are constructed.

Keywords: workflow, workflow management systems, capacity measurement

1. Introduction

A business process consists of a set of related tasks that are performed cooperatively by a group of people to achieve a certain goal. With increasing complexity of business processes, the workflow management technology, workflow model, is needed for facilitating the automation of business processes. The workflow model consists of a set of tasks and rules as when and who to execute the tasks. Workflow management systems (WFMSs) are software systems that are built to support the workflow management technology. They are used for definition and

execution of workflows.

The performance of a WFMS is critical to the throughput of carrying out business processes. When a workflow is executed, the WFMS dispatches the tasks to corresponding users; furthermore, the WFMS must have acceptable response time when the human workers issue request; otherwise, the throughput of business processes will be bad because the human workers spend too much time waiting the WFMS for response. In order to have acceptable performance, the WFMS should not receive too much load of requests. The capacity is the maximum throughput that the WFMS can provide with acceptable performance. The paper presents a method to measure the capacity of a WFMS.

A WFMS receives various levels of workload. The workload consists of a sequence of requests, and such a sequence might be defined in the form of Poisson process. The level parameter of workload is the average request arrival rate of Poisson process. When the level of workload exceeds the capacity, the WFMS cannot provide any more throughputs and thus the capacity can be determined. Software tools are constructed for the measurement. A workload generator is built to generate the workload specified to model the scenario of human behaviors in the context of executing a business process. Furthermore, a monitor is also developed to collect the observed performance data for analysis.

The remainder of the paper is organized as follows. Section 2 introduces the concept of a WFMS as well as related work and motivation of the paper. Section 3 introduces proposed method to measure the capacity of a WFMS. In Section 4, the design and implementation of the measurement tools constructed to support the method

is presented. An example is shown in Section 5 to illustrate the method. Finally, concluding remarks is discussed in Section 6.

2. Background

In this section, the work discussed in the paper is described in details. First, the concept of a WFMS is introduced. Then, the related work is presented. Finally, the motivation is shown.

2.1. The Concept of a Workflow Management System

A business process is a sequence of activities that perform a specific function. A business process can be as simple as applying to leave of absence in a general office environment, which may consist of only four activities – applicant request, approval of manager, approval of personnel, and applicant acknowledgement. It can also be as complex as procurement in a governmental agency that consists of hundreds of activities and involves millions of dollars. A business process is usually repeated over and over in an organization [1]; therefore, it is profitable to incorporate information technology in a business process to achieve some degree of automation.

Workflow is "the automation of a business process, in whole or part, during which documents, information or tasks are passed from one participant to another for action, according to a set of procedural rules" [2]. A workflow process is the process definition of a business process that applies workflow technology. A workflow process consists of activities as logical steps. Some activities are manual while others are automated. Manual activities require the participation of human users, and automated activities are processed by computers alone. Applications that are built based on workflows are called workflow-based applications. Workflow-based applications require the consideration of activity coordination.

A WFMS is a computer program that facilitates the coordination of workflow activities. Workflow-based application can be designed and executed without leveraging a WFMS. However, it will be more difficult just as data management is much easier with the help of a database management system. The main job of a WFMS is to determine the execution path of a workflow and to assign activities to corresponding users accordingly. Hence, a WFMS is a critical component of an information system that contains workflow-based applications. Therefore, the performance issue of a WFMS cannot be ignored.

2.2. Related Work

The capacity of a system is the maximum throughput of the system. Capacity is one of the most important performance metrics of a system. The information of the system capacity is used in capacity planning and system tuning. Capacity planning ensures that computer resources are adequate for future circumstances; hence, it is needed to know future resource requirements and weather the capacity of proposed solution is higher than the required resources. System tuning, on the other hand, is the process of adjusting parameters of current systems to provide the highest performance. The capacity of a system aids to determine the best appropriate system parameters.

The three techniques for performance evaluation of computer systems are analysis, simulation, and measurement [3]. Gillmann et al. [4] assess the performance of a WFMS with analysis technique. Analysis gets the results quickly with low cost; however, the results are not as precise as the other two techniques. Further, analysis models of complex systems are difficult to build and the solutions are hard to find if not impossible. Miller et al. [5] evaluate the performance of a WFMS using simulation. Simulation is less restricted in the modeling and the

results are more accurate than analysis. Nonetheless, the time required for simulation is longer and the cost is much more than analysis. Gillmann et al. [6] proposed a benchmark for measuring the performance of a WFMS. Although the measurement technique costs most and requires most time among the three techniques, it can obtain more accurate results than the other two. The limitation of the measurement technique is that it cannot be conducted before the prototype of the system is being built.

2.3. Motivation

Although capacity is an important performance metric of a WFMS, few of prior approaches for performance evaluation of a WFMS is dedicated to evaluating capacity of a WFMS. The benchmarks for WFMSs proposed in the literature generate workload and thus the capacity with respect to the benchmarks can be obtained. However, the workloads generated by the benchmarks lack think time and hence are unrealistic. Applying these benchmarks can only get capacity of WFMSs with respect to the benchmarks for comparison but cannot get the capacity in real situations. Therefore, an approach to evaluate the capacity of WFMSs in real situations is necessary.

3. Capacity Measurement Framework

The structure of the capacity measurement is described in the first subsection. Each component of the measurement framework is introduced separately. The technique for capacity measurement is then explained in the second subsection. The steps to conduct the measurement are provided in the last subsection.

3.1. The Structure

During the execution of a workflow system, the components involved in capacity measurement might be depicted as in figure 1. There are five programs: *workload script editor, workload generator, monitor, a WFMS,* and *data analysis and presentation tool.* Workload script and collected raw data are the place to store the data used

by these programs. The programs are applied in the preparation, execution, and interpretation phases of the capacity measurement, respectively. The following paragraphs describe the programs in detail.

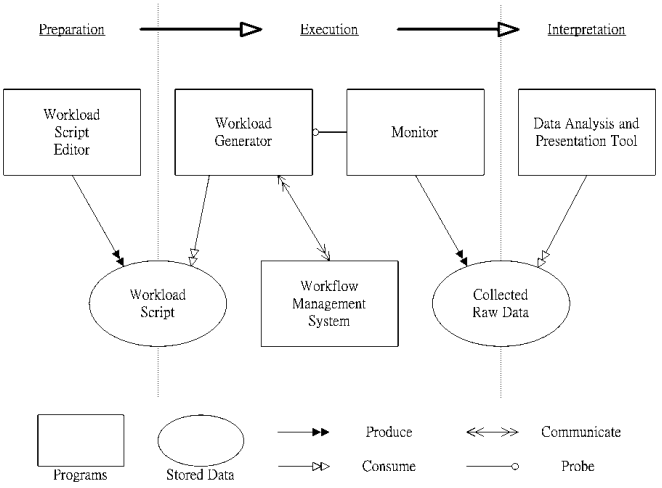


Figure 1. The Structure of Capacity Measurement

A workload script editor allows users to describe the workload in capacity measurement in plain text.

A workload generator produces workload to the system under measurement. It emulates real users of a WFMS. The behaviors of the emulated users are recorded in workload scripts. Employing workload generator instead of real users allows workload to be repeatable, controllable, and cheaper. The design of the workload generator used in this paper is explained in the next section.

The monitor observes the system performance. It resides on a WFMS for collecting server-side data such as throughput, utilization, and such alike. It can accompany with the workload generator in order that the performance metrics on the user side, for example, response time, are collected. The raw data require further processing for interpretation. The technique of statistical analysis is employed to summarize raw data, and then graphs and charts are used to present the summaries. The actual analysis will be described in the next section. There are already many utilities available for the above tasks, for example, Microsoft Excel, SPSS, and SAS.

3.2. The Technique

Capacity measurement of a WFMS includes generating workload to the WFMS and collecting the performance data at the same time. I.e. given a collection of raw data, capacity of the WFMS is then determined. There remain three questions. First, what kind of workload should be generated? Second, what performance data should be collected? That is, how should the experiment be designed? Third, how does the collected data be interpreted to determine the capacity? These questions are answered in the following paragraphs.

Workload Model. Workload model is a model for specifying the workload to be generated. There are two approaches [3] for workload generation, based on trace or model. The trace gets the records of every request that a WFMS receives. It can then be replayed to generate the workload for the WFMS. Trace-based workload [3] is more realistic; however, it is not flexible because this kind of workload cannot be adjusted. Model-based workload [3] is a model that intends to represent the real workload.

A workload model contains parameters. By adjusting the parameters, different levels of workload can be generated. A workload model can be general or specific. General models can accommodate more situations than specific ones. The most generally applied model is Poisson process. On the contrary, specific models are described more accurately, but huge amount of statistical data are required to build. Because the statistical data and characteristics of workload of WFMS are not available currently, the paper selects the Poisson process as the workload model.

From the viewpoint of Poisson process, there are a sequence of data can set as input. By the definition of Poisson process, the time between successive requests is exponentially distributed.

Experimental Design. The interaction of workload generator and a WFMS is depicted in figure 2. The workload generator generates a request and sends it to the WFMS. The WFMS then processes the request and gives the workload generator a response. Obviously, the request/response forms a cycle.

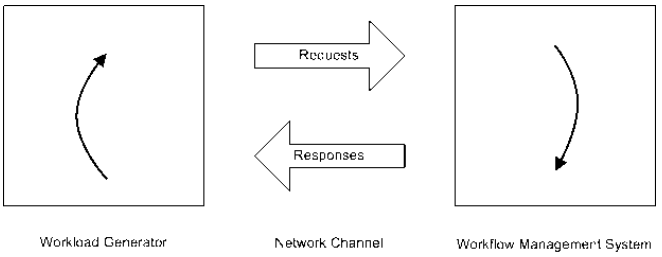


Figure 2. Workload Generator and Workflow Management System

The workload generator, the network channel, and the WFMS might have their own maximum processing rates. Due to the cyclic property of the processing, the maximum overall system-processing rate can be deemed as the least of these three processing rates Thus, the tactic is to make the workload generator and network channel possess more processing rate than the WFMS does.

In order to calculate the overall maximum system-processing rate, it is suggested that the experiment start with an arbitrarily small level of workload, and then doubles the values, each turn of observation. When the level of workload exceeds the maximum system-processing rate, the desired level of workload cannot be generated because the WFMS can not generate such level of workload. The maximum system-processing rate is then between the last two levels of workload in experiment. Experiments with a binary-search-like parameter setting are then conducted to get the maximum system-processing rate with accepted precision.

Since the maximum system-processing rate is achieved, the load level between zero and the maximum system-processing rate is then divided into ten or twenty levels, depending on the desired precision. The response

time of requests, that is, the time between sending the request and receiving the response, under the workload with each level of request arrival rate are observed. The collected data is interpreted later to determine the capacity.

Data Analysis. By plotting the experimental data, a graph like figure 3 can be obtained. The request arrival rate at point A is the maximum system-processing rate and thus represents the capacity of the WFMS. The response time at point A is the response time when the workload equals the capacity of the WFMS. Usually such response time is larger than desired. Suppose the response time at point B is the desired response time, the request arrival rate at point B is then the capacity of the WFMS that provides accepted performance.

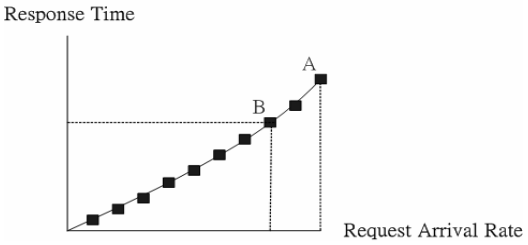


Figure 3. Data Analysis of Capacity Measurement

3.3. The Procedure

The procedure to conduct a capacity measurement of a WFMS includes the following steps.

1. Write Workload Script. The workload script consists of sequences of requests, which are called scenarios, for a WFMS. The scenarios represent the real usage of the WFMS. The workload generator produces workload according to the workload script. In the course of a capacity measurement, the workload script needs write only once.

2. Check Experimental Equipment. Before making the workload generator produce workload, the

computers that the workload generator resides on must be checked for effectiveness. Otherwise, when the system-processing rate is achieved, one cannot decide whether the bottleneck is made by the workload generator or by the WFMS. The technique for checking experimental equipment is that conducting the capacity measurement and making the request arrival rate exceeds the system-processing rate. Then, add another computer to run workload generator to increment just a little workload. If the total system-processing rate increases compared with the previous one, one can concludes that the bottleneck is the workload generator; otherwise, the bottleneck is the WFMS, and the experimental equipment is adequate. If the experimental equipment is not adequate, add another computer and do the preceding procedure again. The above procedure can stop only when that more computer is added does not increase the system-processing rate.

3. Conduct the Experiment. The experiment proceeds by adjusting different levels of request arrival rate for observing system performance as described in previous sections. To improve the emulation result, the network channel need be light load; the computers run the workload generator should not run other programs, which would compete the hardware resources. Furthermore, the WFMS should have no other workload than which the workload generator produces; otherwise the experimental conclusion would be misleading.

4. Analyze Experimental Result. Given the collected experimental data, there are two steps to interpret the data, one for the experimenter himself, and one for the other people. The data are first analyzed with statistical techniques. The data are much reduced at this point, and can be interpreted to get conclusions. However, the presentation is not applicable for non-specialists in capacity measurement. Hence, it is more appropriate to

represent the result in graphical forms, as in figure 3. Finally, the managers, purchasers, analysts, and related people, can take advantage of the results for their jobs.

4. The Measurement Tools Constructed

In this section, the design and implementation of a workload generator and a monitor are presented. First, architecture of the workload generator is described. Then, design of the monitor is presented. Finally, some important issues of implementation are explained in the last subsection.

4.1. Workload Generator

The workload generator consists of workload script interpreters, the library of workflow client application interface, and a barrier as depicted in figure 4. The *workload script interpreters* interpret workload script to generate workload. There are in general more than one workload script interpreters in a workload generator. The *workflow client application interface* is a collection of functions that can be called in the workload script in order to make requests to a WFMS. The *barrier* synchronizes the workload script interpreters so that the workload generated by the workload script interpreters as a whole is in an ordered fashion. The details of the three components are described below.

Workload Script Interpreter. Workload script interpreter generates workload as specified in workload script. The simplest form of workload script is a sequence of requests to a WFMS [3]. In this case, the function of workload script interpreter is to issue the requests one after another. However, this form of workload script is not convenient for complex workload programming. In order to be practically applicable, the workload script should support basic control structures, such as loop and branch. Furthermore, variables and functions are also useful for

workload script. Therefore, a general-purpose script language would be more suitable for workload script.

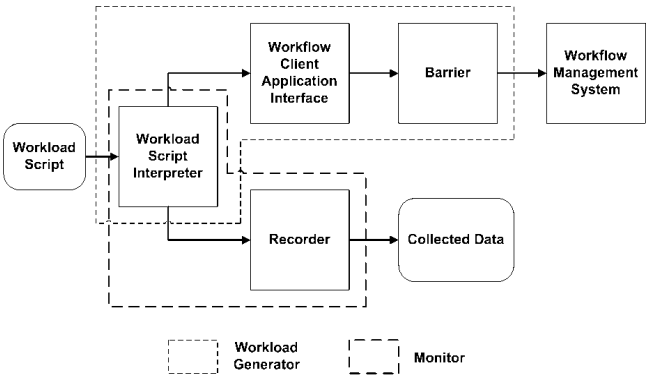


Figure 4. The Structure of Workload Generator and Monitor

In the workload generator, a group of workload script interpreters executes concurrently. If there is only one running workload script interpreter, it would be idle while the WFMS is processing the received request. Besides, after the WFMS responses, the workload script interpreter takes time to prepare the next request and the WFMS would be idle during this moment. Thus, the WFMS would not be able to sustain sufficient workload as desired. On the contrary, multiple workload script interpreters can take full advantage of the computer resources where they execute. When one workload script interpreter is waiting for response, other workload script interpreters can continue executing to prepare for the next requests.

Workflow Client Application Interface. The workflow client application interface provides a facility to make requests to a WFMS in workload script. The interface is in the form of a set of functions for the structured programming paradigm, or a set of methods encapsulated as a class for object oriented paradigm, such that it can be called in workload scripts. For example, the Workflow Application Programming Interfaces (WAPI) [7] contains methods like **WMCreateProcessInstance** for creating a new process, **WMConnect** for connecting to

a WFMS, and **WMGetWorkItem** for retrieving a specific task.

There are standards of workflow client application interface. Workflow Management Coalition developed a workflow management application programming interface specification [7]. A part of the specification is dedicated to the interface between client application and a WFMS. Object Management Group published a workflow management facility specification [8]. It is mainly used in the CORBA environment. Almost all WFMSs provide libraries of their proprietary interfaces. Therefore, making requests to a WFMS in workload script is convenient.

Barrier. The barrier synchronizes the workload script interpreters so that the produced requests to a WFMS form a Poisson process. A Poisson process of requests means that the requests are ordered in a sequence and the distribution of inter-request time is exponential. When the workload script interpreters are ready to send request to a WFMS, they are waiting in a queue managed by the barrier. After sending a request, the barrier would wait for an interval drawn from the exponential distribution, and then allow the next request to be sent. The barrier sustain this behavior during the workload generation.

To make the inter-request time exponentially distributed, a random variate generator of exponential distribution has to be designed. The technique employed in the paper is the inverse transform method [3]. In practice, it is sometimes annoyed that a sequence of very small inter-request time is generated. To alleviate this situation, a more complex exponential distribution, which has an additional parameter that specifies the smallest number in the distribution, is employed.

4.2. Monitor

The performance of a WFMS can be observed either on the client side or on the server side. Almost all

WFMSs have built-in facilities to log predefined performance metrics. To collect the performance data on the client side, one can insert extra codes in workload scripts. For example, in order to collect the response time of a request, the statements to record the current time are inserted before and after the request statement. Then the response time of that request is obtained from the difference of the two times.

The monitor contains a workload script interpreter and a recorder. Besides generating workload, the workload script interpreters execute the code inserted in workload scripts to collect performance data as well. The recorder is used to store the observed performance data for later analysis. It is accessible in workload scripts. Because a unique recorder serves multiple workload script interpreters, the recording tasks of workload script interpreters are synchronized to maintain data consistency. Besides storing the raw data, the recorder is responsible for organizing the data in a form that is accessible to data analysis programs.

4.3. Implementation

The workload generator and monitor are implemented in the Java programming language because they may be run on many computers with heterogeneous configurations. ECMAScript [9] is chosen as the language for workload script. ECMAScript is a standardized scripting language whose syntax is similar to JavaScript. There is a freely available ECMAScript interpreter, Rhino [10], which is written entirely in Java and so it can be combined with the other components of workload generator and monitor seamlessly. The workflow client application interface, barrier, and recorder are implemented as Java classes. They are made accessible in the workload script using the mechanism provided by Rhino. While running, the main program of the workload generator creates multiple workflow script interpreters and executes them in a thread, respectively.

5. Application Example

An example of the capacity measurement is illustrated in this section as a proof of concept to the method presented in the paper. At first, the hardware, software, and other equipments used in the experiment are described. Then, the experimental result are presented and discussed.

5.1. Experimental Setup

Workflow Application: The example workflow is S. J. Liu's vacation application [11]. In the example, only the most frequently executed path, whose performance dominates the entire performance, is measured in order to reduce complexity. The sequence of tasks of the above execution path is depicted in figure 5.

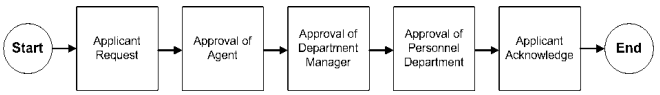


Figure 5. An Execution Path of Vacation Application

Workflow Management System: The WFMS used in the example is Y. S. Chen's Agentflow [12], which is also used in S. J. Liu's work. Agentflow provides a workflow client application interface called *Workflow Common Interface* (WFCI). In the example, the functions of WFCI are called in workload script to make requests to Agentflow workflow management system.

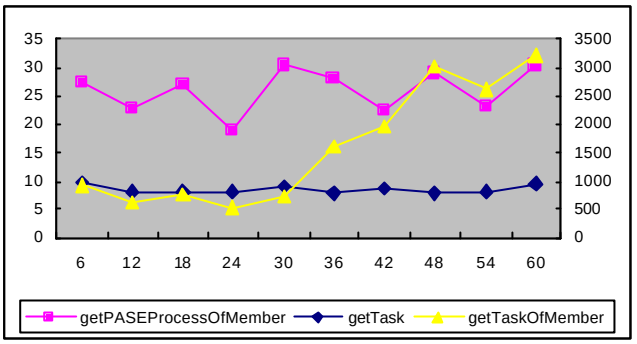
Hardware and System Software: Sun's Java development kit v1.3.1 is used to run the constructed measurement tool as well as Agentflow since both of them are written in Java. The database server for Agentflow is Microsoft SQL Server 7. The PC for Agentflow has an Athlon 800 CPU and 328MB memory, running Microsoft Windows 2000. All PCs for measurement tool have an Athlon 800 CPU and 1GB memory, running FreeBSD 4.5. The machines for measurement tool and Agentflow are connected in a fast Ethernet environment.

5.2. Experimental Result

The experimental result is shown in figure 7. The response times of three WFCI functions are measured. When a user wants to create a process, the method *getPASEProcessOfMember* is called to retrieve all process definitions that the user can create. When a user begins to work, the method *getTaskOfMember* is called to retrieve all tasks belonged to the user. When the user picks a specific task to carry out, the method *getTask* is called to retrieve that task.

As seen in figure 7, the maximum number of average requests per minute is 60. In the experiment, if the average requests per minute is set to be larger than 60, there will be no workload script interpreter waiting to send request when the barrier is selecting one interpreter to send the next request. This phenomenon means that the system cannot process requests at a higher rate; besides, the measured data is invalid because the requests are no longer in the form of a Poisson process. Thus, the capacity of the WFMS in the experiment is 60 requests per minute in average. After that, the average requests per minute from zero to the capacity, that is, 60 are partitioned into 10 segments. The system performance of ten levels of workload is measured respectively.

As seen in figure 7, as long as the workload is under the capacity, the response time of methods *getPASEProcessOfMember* and *getTask* are less than one second. Besides, there is little increment of response time when the workload increases. However, the response time of the method *getTaskOfMember* increases drastically from about half of the capacity. When the average requests per minute reach the capacity, the response time of the method *getTaskOfMember* increases up to 3.2 seconds. A wait for 3.2 seconds is acceptable for most people; therefore, the capacity would not be cut back in order to obtain acceptable response time.



x-axis: average requests per minute

left y-axis: Response time of getPASEProcessOfMember and getTask in milliseconds

right y-axis: Response time of getTaskOfMember in milliseconds

Figure 7. Diagram of Experimental Result

6. Conclusion

In the paper, a method to measure the capacity of a WFMS is presented. The design and implementation of measurement tools supporting the method are also described. To measure the capacity, the WFMS receives requests in the form of a Poisson process with various levels of load. Due to the cyclic property of interactions between a WFMS and workflow client applications, the WFMS cannot produce responses with the same rate as the requests if the workload is higher than its capacity; thus, the capacity of the WFMS can be determined.

References

[1] Frank Leymann and Dieter Roller. *Production workflow: concepts and techniques*. Prentice-Hall, Inc. 2000

[2] Workflow Management Coalition. *Terminology & Glossary*. WFMC-TC-1011, June 1996

[3] Raj Jain. The art of computer systems performance analysis: techniques for experimental design, measurement, simulation and modeling. John Willey & Sons, Inc. 1991

[4] Michael Gillmann, Jeanine Weissenfels, Gerhard Weikum, Achim Kraiss. *Performance and Availability Assessment for the Configuration of Distributed Workflow Management Systems*. VII. Conference on Extending Database Technology (EDBT), Konstanz, Germany, 2000

[5] J.A. Miller, A.P. Sheth, K.J. Kochut, X. Wang , and A. Murugan. *Simulation Modeling within Workflow Technology*. Winter Simulation Conference, Arlington, VA, December 1995

[6] Gillmann M, Mindermann R, Weikum G. *Benchmarking and configuration of workflow management systems.* Fifth International Conference on Cooperative Information Systems (CoopIS), Eilat, Israel, 2000

[7] Workflow Management Coalition. Workflow Management Application Programming Interface (Interface 2&3) Specification. WFMC-TC-1009, July 1998

[8] Object Management Group. Workflow Management Facility Specification, V1.2. April 2000

[9] Standardizing Information and Communication Systems. *ECMAScript Language Specification 3rd edition.* Standard ECMA-262, December 1999

[10] The Mozilla Organization. *Rhino.* [http://www.mozilla.org/rhino/]

[11] S. J. Liu. *Testing a 3-Tier Workflow-based Application*. Master Paper of Department of Computer Science and Information Engineering, National Chiao-Tung University, Taiwan, 2001

[12] Y. S. Chen. *Building an Edit and Enactment System for Process Definition.* Master Paper of Department of Computer Science and Information Engineering, National Chiao-Tung University, Taiwan, 2001