AN EXPERIENCE OF MODELING AND SIMULATION IN SUPPORT OF CMMI PROCESS

Yung-Hsin Wang
Department of Information Management
Tatung University
40 Chungshan N. Rd., 3rd Sec, Taipei 104, Taiwan
E-mail: ywang@ttu.edu.tw

Yuan-Fan Chen
Department of Computer Science and Engineering
Tatung University
40 Chungshan N. Rd., 3rd Sec, Taipei 104, Taiwan
E-mail: yuanfanc@hotmail.com

KEYWORDS

Modeling and Simulation, Software Development, Process, CMMI.

ABSTRACT

In this paper, we will present our experience of applying simulation technology to assist in the CMMI (Capability Model Maturity Integration) introduction through an industrial practical case. We use the SES/*workbench* software to model and simulate the proposed software development process model and compare the simulation results through different scenario plots of resource allocation, which provides managers with processes adjustment experiences and estimation basis during introduction. This demonstrated the feasibility and ability of modeling and simulation technique to support CMMI related process improvement for organizations.

INTRODUCTION

Being essentially a non-industry that never existed forty years ago, software development and application in practice has now become an over several billion dollar business. However, with the involvement of software industry and other industries, the whole environment had become much more complicated and unpredictable. As can be seen, software has replaced hardware as the most accountable part for much of the functionality that systems provide and has become the brain of the systems we are using. Besides, with the increasing software complexity and the increasing customer demands that oriented to be "better, faster, and cheaper", software developers now are forced to seek any possible performance improvement. The industry had attempted several plots to work on this challenge to achieve their goals, in the form of various case tools, new computer languages, and more advanced and sophisticated machines. A key question is, "How can the tools, technologies and people work together in order to achieve these increasingly challenging goals?" It is noted that one potential answer to this question is through changes to the software development process or software organization. Thus, companies are addressing these issues with an emphasis on software process performance improvement and increased process maturity (Raffo 1999).

How can one be sure of the change plot that is going to be adopted and applied to be clearly correct? The answer is "You never know." Basically, the process behavior can be too complicated and the time span of the project can be too large so that potential problems couldn't be easily perceived. By the way, possible changes will require a significant amount of resources to implement and have significant implications on the firm. How can organizations gain insights into potential solutions within the processes to these problems and their likely impacts? One area of research that has attempted to address these questions and has had some success in predicting the impact of some proposed solutions is software process simulation.

In industrial practice, usually due to a shortage of time to meet market needs, processes are often changed at will without considering possible coming impacts the change could bring. This is most likely to be seen in fresh companies without much experience regarding processes, and even some mature companies without proper preservation of their past knowledge and experiences. If anyone wants to know the relationships between process changes and the possible outcome, it would need several dry runs under different conditions and examine the results to gain a better understanding of the process. Unfortunately, this is not feasible and practical since involving lots of staffs in experimental processes not producing marketable results is definitely out of the question. Hence, simulation comes to rescue for being a widely used popular method for studying complex system (Christie 1999; Kellner et al. 1999).

On the other hand, the SEI (Software Engineering Institute) Capability Maturity Model Integration (CMMI) was developed to help organizations improve their software engineering management practices (Paulk et al. 1993). It provides benchmarks evaluators can use to grade the ability of an acquisition or programming organization to produce reliable, maintainable software that meets customers' needs. Simulation and CMMI are both fundamentally process focused with common objectives to enhance process capabilities and performance. Some researches had applied simulation to the business practices to help achieve higher CMMI maturity level (Miller et al. 2002; Raffo et al. 1999).

Proceedings 15th European Simulation Symposium
Alexander Verbraeck, Vlatka Hlupic (Eds.)
(c) SCS European Council / SCS Europe BVBA, 2003
ISBN 3-936150-28-1 (book) / 3-936150-29-X (CD)

The objective of this study is to model and simulate the software development process in order to support the introduction of CMMI in an enterprise. In the course of refining software development, the decision maker faces a huge amount of information, which may come from internal executive experiences or external user communication. It is hard to use the traditional linear programming or statistical analysis tools to handle them. With the dynamic analyzing ability, simulation technology can be used to assist in this kind of decision problem more powerfully. The enterprise can take advantage of adopting simulation results for the evaluation of new designed business processes without taking any risk of changing any physical part of a real business.

CMMI INTRODUCTION PHASES AND EXPERIENCES

This study is based on the case of a major software development enterprise in Taiwan. Within the group there exists a lot of different business units and companies of various industries. We focus on the examination of internal activities and required process in this enterprise attempting to find out possible problems in its business process. Currently the business organization is conducting the CMMI introduction and expecting to pass level 2 in a year. Lab CMMI was established as the facilitator at the very beginning, where our study started. Firstly the role we play here was being an observer to the process improvement activities. Data was collected through attendance to training sessions, observation of requirements analysis and discussion sessions, administration of two questionnaires, inspection of relevant documents, as well as interviews with engineering development and management staffs.

CMMI was just beginning to become a springing concept and getting more attention in Taiwan recently. Now top-level managers in more and more companies are getting the idea that CMMI can be a solution to the chaotic software environment they are in. Generally speaking, companies wait too long before they attempt software process simulation and refinement. They then expect more from the process models than that may be reasonably provided. Software process simulation can help companies at early stages of process maturity. First, the graphical capabilities of software process simulation can help companies plan and define their processes and process changes such as that in CMMI Level 2 Activities. These models can be developed to a fine level of granularity and can provide a great deal of assistance in communicating and understanding a company's software development process.

The structure of the CMMI related attendants within or outside the case organization of this study contains the SEPG (Software Engineering Process Group), three business units, one CMMI LAB and a consultant group. Basically, each business unit represents an independent company or simply one of several business units inside the group. These business units are selected from the running enterprise as a pilot to the CMMI introduction activity. The SEPG is the steering committee of the project assigned directly from high-level managers to be in charge of entire project execution and monitoring control. In addition, outside the organization an entity called CMMI LAB was established in support of the SEPG for the introduction project. It is important for being the third facilitator, the communicating bridge to the consultant group, and also the preserver of the introduction experience and knowledge for future knowledge management implementation.

Before the process started, Gap Analysis was firstly employed within the organization as a pre-evaluation of the CMMI assessment. The purpose was to find out how much difference there would be of the current processes from CMMI regulated processes. Figure 1 shows the CMMI adoption phases. A series of questions were issued based on the official CMMI articles, and a number of suitable objects were selected from the business unit including roles like senior managers, project managers and technicians who were in charge of the relevant processes.

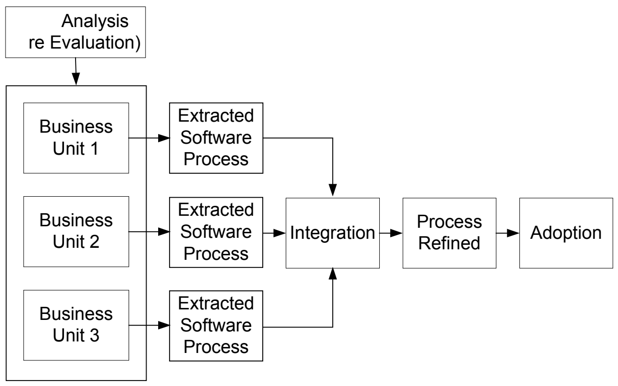

Figure 1: CMMI Process Refinement and Adoption

Before CMMI conduction, the organization needs a major integration of different processes fell in each sub unit. This is because each sub unit within the organization has their own business scope and style, and that even in the same industry, the same business practices may tend to be conducted in various ways, causing a great deal of redundancy. In order to meet the ultimate goal of process refinement and reengineering, these redundancies should be carefully eliminated. Thus, each different and unique software process must be extracted along with the relevant forms, applications and role definitions in preparation for the coming integration.

The processes extracted here would then be trimmed and nourished properly. With the unnecessary activities being cut down and required practices being replenished, integration can be successfully done. After certain adjustment and tailoring, the adoption can be then more smoothly conducted.

SOFTWARE DEVELOPMENT PROCESS MODEL

We created the software development process model according to the real processes conducted within the company and with a reference to the typical software lifecycle models (Mead et al. 2000) as example. Then the whole model was divided into four different submodels, each representing a group of specific workflows and relevant processes representation (See Figure 2 for illustration).

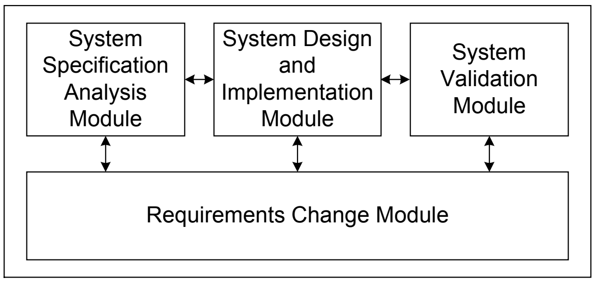
System Specification Analysis Module
System Design and Implementation Module
System Validation Module
Requirements Change Module

Figure 2: Proposed Model for Software Development Process

The entire model consists of four individual component modules: System Specification Analysis Module, System Design & Implementation Module, System Validation Module and Requirements Change Modules. Processes and information travel between all the sub-modules, while outside users, the clients or any other possible service objects outside the company, communicate with them through certain interfaces created. Within System Specification Analysis Module, all relevant information about the pre-requirement developing phase was gathered and analyzed, as a basis for further system design and implementation. System analysis, design and the actual coding phases were modeled through the creation of the System Design & Implementation Module, while necessary information were collected and transmitted throughout to other modules. The System Validation Module basically handles information of the final confirmation with outside users. During the lifecycle of the software development process, users' requirements change request can be received through the Requirements Change Module, and pass on to other possible sub-modules after detail analysis.

The model was simplified and established according to the software life cycle relations with process engineering areas as shown in Figure 3. As can be seen, different projects contain different requirements, while they all have the same goal to develop the high quality product in order to meet the clients' needs within limited budget and time. Unfortunately, due to the perplexity of software development process, there is no such pattern or model suitable for all situations. Note that on CMMI level 2, there are seven KPAs (key process areas): Requirements Management, Project Planning, Project Monitoring and Control, Supplier Agreement Management, Measurement and Analysis, Process and Product Quality Assurance and Configuration Management. In the figure we can see clearly the relationship between the software development lifecycle and the engineering areas. For example, technical solution covers two phases: Design, and Program Implementation and Testing. These issues should all be taken into consideration during the process modeling.

Due to space limitation, detailed workflow of the four sub-modules of our proposed software development lifecycle model is not depicted here. According to the problems that managers and the concerning participants stated, users' requirements change rapidly and are hard to track. Any request for a system requirement change during the later phases such as system integration and testing may cause serious effect and lead to an inability of delivery. Because a response to this request not only affect the around activities, sometimes also cause a specification redesign and lead to an unexpected and out of control situation (Lowry and Bebbington 1998).

Thus, in the modeling process, we tend to create the model in a more loosely fashion. While collecting and defining the processes, we tend to break these necessary tasks into smaller activities in order to break out the task dependency potentially lies within, which makes the model a little slightly compact and more loosely coupled. This is to eliminate the uncertainty and to increase the traceability throughout project management when receives users' requests for any change, also to make the affected area as tight and compact as possible.

According to the facts we found from the interview in the case company, the scope of the software project team varies from size to size due to the uncertainty of project scope in engagement. A small project sometimes takes only a few people while a mid-sized project takes up to a dozen of people. Normally the project team is formed not too early before the project is initiated, and team members are sometimes selected according to the needs and manager's experiences. Thus, management and the project quality can be very difficult to maintain. This is exactly why the rules of CMMI strongly address importance of the issue on the employment training. If all the employees were properly trained and skilled, once the project is initiated, managers can easily draw out needed persons and also enhance the project quality (Conwell et al. 2000; Car and Mikac 2002).

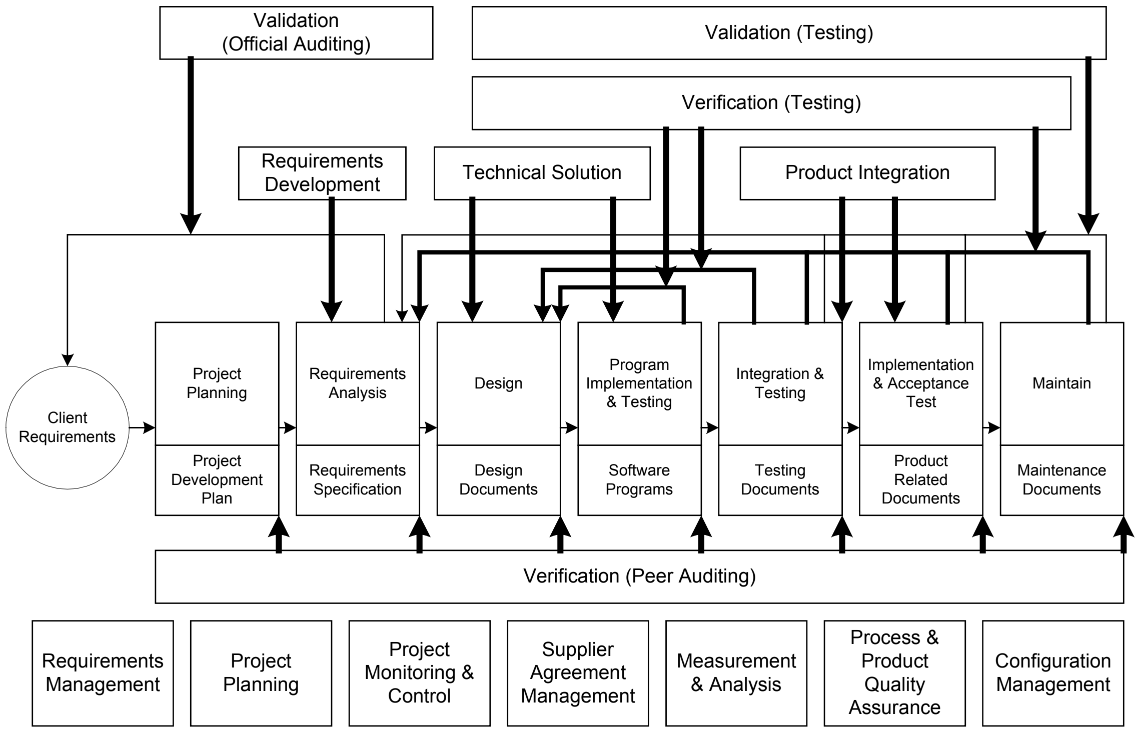

Figure 3: Software Life Cycle Relations with Process Engineering Areas

SIMULATION MODELS & RESULTS ANALYSIS

Software Development Lifecycle Model

The entire software development lifecycle model is designed and created in SES/*workbench* (SES, Inc. 1994) based on our discussion above for the case business enterprise. Instead of depicting the abundant submodels, we shall present some selected models in what follows.

Figure 4 shows one of the Design and Implementation Submodels. It represents the model activity processes from coding, code inspection to further function tests. In our case study, there is a testing group fully responsible for the entire code test, while in other cases we found most companies still use peer group test to ensure code product quality, which is not effective. For modeling convenience, here we simply put a node to represent it.

Figure 5 shows the model of the requirements change. It handles the activity processes from requirements change receiving, analysis and validation to its final execution approval. In this study, the organizations usually use certain methods to decide which action should be taken during change analysis. Generally, changes requested from the clients are most likely of extra functions that only few programs need to be modified or created. This kind of requirements change does not affect the system drastically and normally can be solved easily. If the change effects a wide range of project extent, such as contradiction to the original specification or design, it should be taken as a new requirement and starts from the very beginning in avoidance of over time and cost.

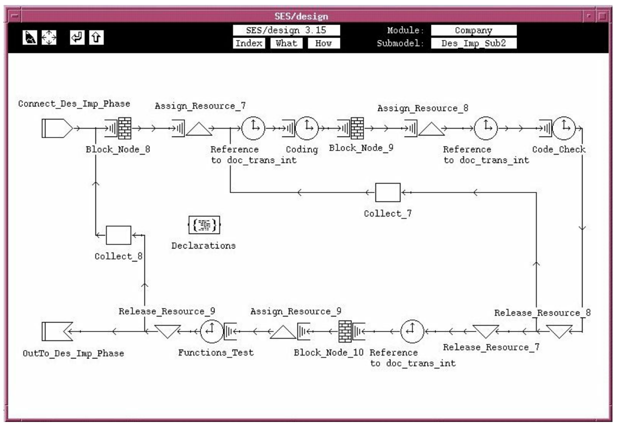

Figure 4: One of the System Design and Implementation Sub Models in SES/*Workbench*

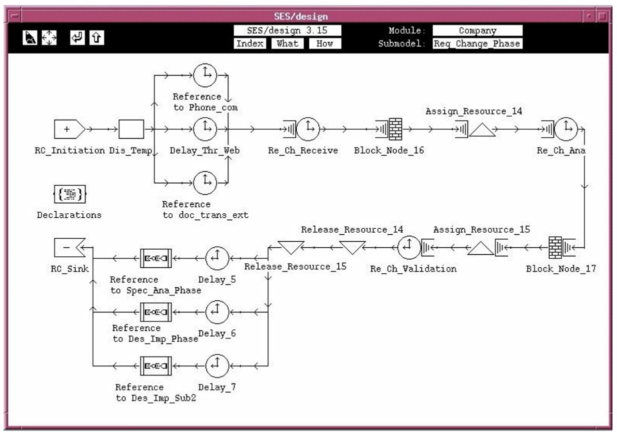

Figure 5: The Requirements Change Model

Simulation and Analysis

In this case, we tend to find out the effects of different employment methods through some sketched scenarios with adjustment of resources allocation. A few scenarios were selected and conducted in the simulation in order to address the issues and seek the best, and relevant parameters were defined and set to distinguish the characteristics in each scenario.

In this study, we found that a small project may take only a few people while a mid-sized project may take up to a dozen of people. But mostly the size of the software development project is mid-large, which normally takes 10-20 (usually around 15) people and usually takes 3 to 6 months to complete. The project team members cover commonly from management level to downward technicians, including project manager, technical engineer, system analyst and testing technicians, etc. Some organizations may have their own project team structure according to their organizational policy or composition factors, and do a slight change. A project team is established while the project is still at the preliminary stage in contacting the clients, not long before the project initiates. The actual members of the project team were selected under considerations of their workload and technical factors. Sometimes the decision can be very hard for the managers to make.

In our simulation study, we try to find out an evaluation way and provide an analysis basis of statistical results for managers to take and see if the project plan needs to be changed. If the project were taken, would the project team be able to accomplish it efficiently? Should more technical engineers be assigned to share the unbearable workload while still keeping in budget? These are all serious issues that managers seek to answer. We use two different scenarios of resource allocation examples for the simulation and then compare their results. Table 1 and Table 2 describe the general simulation information for both scenarios. The total simulation time is set to 9140 hours, i.e., more than 4 working years.

Table 1: Simulation Settings

Name	Content
Simulation Time	9140 hours
Project Scope	28 mid-large sized projects
Project Team	10-15 people
Project Duration	3-6 months

Table 2: Resource Allocation Scenarios

Project Team Structure	Scenario 1	Scenario 2
Project Manager	1	1
System Analyst	3	2
Technical Engineer	6	8
Testing Group	3	2

In scenario 1 the parameters and the relevant settings were set as the original plot we discovered from within the company. A usual mid-large project normally takes one manager, 3 system analysts, 6 technical engineers and 3 testing technicians. Simulation results (Table 3 and Table 4) show that there exists possible working overload for the technical engineers, which managers should take into consideration to adjust team structure or the task assignment. Should the project include more people to digest unconsumed workload within budget, or does the inclusion here bring only extra expense without effective contribution to the project progress? These are dangling and unsolved if not being practiced. Accordingly, scenario 2 was developed where a system analyst and a testing technician were assigned to support technical engineers to share the unbearable overload. The results in Table 5 show that the utilization of technical engineer was cut down clearly in scenario 2. Compare Tables 4 and 6, the average cycle work hour and average rework time have about 11% decreases. This significance is due to the improvement of some possible lock knot process. These example results of simulation provide a good basis for managers to control and monitor the project progress more aggressively instead of passively waiting for possible crisis to come.

Table 3: Staff Utilization of Scenario 1

Team Member	Utilization
Project Manager	0.8900
System Analyst	0.8788
Technical Engineer	0.9661
Testing Group	0.6410

Table 4: Other Simulation Results of Scenario 1

Performance index	Value
Average Cycle Work Hour	5980.79 hrs
Average Rework Time	2318.24 hrs
Average Requirements Change Time	1783.36 hrs
Number of Requirement Change	82

Table 5: Staff Utilization of Scenario 2

Team Member	Utilization
Project Manager	0.8803
System Analyst	0.9217
Technical Engineer	0.8412
Testing Group	0.7221

Table 6: Other Simulation Results of Scenario 2

Performance index	Value
Average Cycle Work Hour	5321.32 hrs
Average Rework Time	2072.25 hrs
Average Requirements Change Time	1802.19 hrs
Number of Requirement Change	83

CONCLUSIONS AND FUTURE WORK

Conclusion

The process of developing software is dynamic and rapidly evolving due to the competitive pressures to deliver software products of high quality, on time and at reasonable cost. But implementing process changes along with, it is expensive, risky and unrealistic for the requirements change constantly. Thus, simulation provides a viable and inexpensive approach for organizations to gain insights into potential solutions within the process, predicting the impact of some of these proposed solutions by assessing and reducing this risk through the quantitative analysis of process performance. The software process model we proposed here was constructed from the current processes conducted within the case enterprise with the aid of the SES/*workbench* simulation tool. Before conduction, the standard process was extracted from several companies. Simulation here is used as an estimation approach for better project planning. From the research results, simulation proves to be an effective and valuable way to support both initial planning and re-planning activities. In initial planning, resource need and cost estimates must be established. And if the result doesn't go well, pitfalls should be notified and erased in the re-planning activities. In the way a better solution is made, CMMI can be introduced more smoothly with less resistance and difficulty, thus saving its cost.

Future Work

A business organization should keep on pursuing higher software capability maturity not only for the underlying benefits it can bring, but also for being able to deal with the rapidly changing era. According to the CMMI definition, any level-5 organization should have the ability of self-adjusting to the optimal state. Thus, we will keep on the work on using simulation to help achieve higher software development maturity, and smoothen the CMMI introduction. In our research, since we are now focusing on an about-to-be level 2 company, only fairly experience-based data associated with costs and schedules are likely to be available. Some information was collected from the widespread survey or interview. But these issues can be improved while moving onto higher level of CMMI. On level 3, a library of processes that can be reused should be created and maintained. On pursuit of level 3, information like requirements changes, design-defects and code-defects can also be acquired as an input to the simulation model. Therefore, detailed performance measures like software or service quality and development time to adding or changing functionality can all be elaborately calculated and estimated. In the practical case, the question of how can we conduct CMMI standard without sacrificing past efforts is a serious issue to the organization.

REFERENCES

Car, Z. and B. Mikac. 2002. "A method for modeling and evaluating software maintenance process performances," In *Proceedings of Sixth European Conference on Software Maintenance and Reengineering*, 15-23.

Christie, A.M. 1999. "Simulation in Support of CMM-based Process Improvement." *Journal of Systems and Software* 46, No.2/3 (Apr), 107-112.

Conwell, C.L.; R. Enright; and M.A. Stutzman. 2000. "Capability Maturity Models support of modeling and simulation verification, validation, and accreditation." In *Proceedings of Winter Simulation Conference*, Vol.1, 819-828.

Kellner, M.I.; R.J. Madachy; and D.M. Raffo. 1999. "Software Process Simulation Modeling: Why? What? How." *Journal of Systems and Software* 46, No.2/3 (Apr), 91-105.

Lowry, G.R. and P.R. Bebbington. 1998. "Towards enterprise transition to object technologies: a CMM-based methodology." In *Proceedings of International Conference on Software Engineering: Education and Practice*, 78-87.

Mead, N.R.; R. Ellison; R.C. Linger; H.F. Lipson; and J. McHugh. 2000. "Life-Cycle Models for Survivable Systems." In *Proceedings of the Third Information Survivability Workshop* (Boston, MA, Oct. 24-26).

Miller, M.J.; D.M. Ferrin; and F. Pulgar-Vidal. 2002. "Achieving Higher Levels of CMMI Maturity Using Simulation." In *Proceedings of the Winter Simulation Conference*, Vol.2, 1473-1478.

Paulk, M.C.; C.V. Weber; S.M. Garcia; M.B. Chrissis; and M. Bush. 1993. "Key Practices of the Capability Maturity Model for Software, Version 1.1." Technical Report SEI-93-TR-25. Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA (Feb).

Raffo, D.M. 1999. "Getting the Benefits from Software Process Simulation." In *Proceedings of 1999 Conference on Software Engineering and Knowledge Engineering* (Kaiserlautern, Germany, June).

Raffo, D.M.; J.V. Vandeville; and R.H. Martin. 1999. "Software Process Simulation to Achieve Higher CMM Levels." *Journal of Systems and Software* 46, No.2/3 (Apr), 163-172.

SES, Inc. 1994. *SES/workbench Creating Models*, Release 3.1. Scientific and Engineering Software, Inc., Austin, TX (Feb).

AUTHOR BIOGRAPHIES

YUNG-HSIN WANG is an associate professor of the Department of Information Management at Tatung University, Taipei, Taiwan. He received his MS and PhD degree in Electrical and Computer Engineering from the University of Arizona in 1987 and 1992, respectively. His research interests include modeling and simulation methodology, web-based simulation, simulation-based intelligent systems, decision support systems, and e-Business related area.

YUAN-FAN CHEN received his MS degree from the Department of Computer Science and Engineering at Tatung University in August 2003. His research interests are in the area of modeling, simulation, and CMMI related software process improvement.