A Dynamic Integrated Framework for Software Process Improvement

MERCEDES RUIZ mercedes.ruiz@uca.es

Department of Computer Languages and Systems, Escuela Superior de Ingeniería, University of Cádiz, Spain

ISABEL RAMOS AND MIGUEL TORO {isabel.ramos, mtoro}@lsi.us.es

Department of Computer Languages and Systems, Escuela Técnica Superior de Ingeniería Informática University of Seville, Spain

Abstract. Current software process models (CMM, SPICE, etc.) strongly recommend the application of statistical control and measure guides to define, implement, and evaluate the effects of different process improvements. However, whilst quantitative modeling has been widely used in other fields, it has not been considered enough in the field of software process improvement. During the last decade software process simulation has been used to address a wide diversity of management problems. Some of these problems are related to strategic management, technology adoption, understanding, training and learning, and risk management, among others. In this work a dynamic integrated framework for software process improvement is presented. This framework combines traditional estimation models with an intensive utilization of dynamic simulation models of the software process. The aim of this framework is to support a qualitative and quantitative assessment for software process improvement and decision making to achieve a higher software development process capability according to the Capability Maturity Model. The concepts underlying this framework have been implemented in a software process improvement tool that has been used in a local software organization. The results obtained and the lessons learned are also presented in this paper.

Keywords: software process modelling and simulation, process improvement, process maturity, dynamic models

1. Introduction

Over the past few decades software complexity has significantly increased in such a way that software has replaced hardware as having the principal responsibility for much of the functionality provided by current systems. This increasing role of software, the problems related to cost and schedule overruns, and the customer perception of low product quality have changed the focus of attention towards the maturity of software development practices. Although the software industry has received significant help by means of Computer Aided Software Engineering (CASE) tools, new programming languages and approaches, and more advanced and complex machines, there is a lack of process analysis tools for organizations interested in improving their process performance.

Dynamic modeling and simulation as process improvement tools have been intensively used in the manufacturing area. Currently, software process modeling and

simulation are gaining an increasing interest among researchers and practitioners as an approach to analyze complex business and solve policy questions.

In previous work (Ruiz, Ramos, and Toro, 2001) we attempted to apply a complex dynamic model to support process improvement in a local software development organization. The non-existence of a historical database and of measurement practices inside this organization made it impossible, as there were no numerical drivers to supply the model parameters and functions. Then, we decided to apply Eberlein's work (Eberlein, 1989) about understanding and simplification of models to obtain a reduced dynamic model capable of reproducing the software process dynamics, yet with less initial information required. But simulation is only effective if both the model and the data used to drive it accurately reflect the real world. Thus, the construction of the model itself points to what metric data must be collected and helps as a clear guideline on what to collect.

In this paper an approach is proposed that combines traditional estimation techniques with System Dynamics modeling. The aim of this combination is to build a framework to support a qualitative and quantitative assessment for software process improvement and decision making. The purpose of this dynamic framework is to help organizations to achieve a higher software development process capability according to the Capability Maturity Model (Paulk et al., 1993). The dynamic models built inside this framework provide the capability of gaining insight over the whole life cycle at different levels of abstraction. The level of abstraction used in a certain organization will depend on its maturity level. For instance, in a level 1 organization the simulator can establish a baseline according to traditional estimation models from an initial estimate of the size of the project. With this baseline, the software manager can analyze the results obtained with the simulation of different process improvements and study the outcomes of over or underestimating cost or schedule. During the simulation metric data are saved. These data conform to the SEI core measures (Carleton et al., 1992) recommendation, and are mainly related to cost, schedule and quality.

The structure of the paper is as follows. Section 2 provides a brief overview of the work conducted in the field of software process simulation. In Section 3, the justification found to develop the integrated framework is presented. Section 4 describes in detail, the fundamental basis and structure of this framework. The implementation and results obtained when applying it inside a local organization are discussed in Section 5. Finally, Section 6 summarizes the paper and draws the conclusions and lessons learnt.

2. Software process simulation

Simulation can be applied in many critical areas in support of software engineering. It enables one to address issues before these issues become problems. Simulation is more than just a technology, as it forces one to think in global terms about system behavior, and about the fact that systems are more than the sum of their components (Christie, 1999). A simulation model is a computational model that represents

an abstraction or a simplified representation of a complex dynamic system. Simulation models offer, as a main advantage, the possibility of experimenting with different management decisions. Thus, it becomes possible to analyze the effect of those decisions in systems where the cost or risks of experimentation make it unfeasible. Another important factor is that simulation provides insights into complex process behavior which is not possible to analyze by means of stochastic models. Like many processes, software processes can contain multiple feedback loops, such as those associated with the correction of defects. Delays resulting from these defects may range from minutes to years. The resulting complexity makes it almost impossible for mental analysis to predict the consequences. The most frequent sources of complexity in real software processes are:

— *Uncertainty*. Some real processes are characterized by a high degree of uncertainty. Simulation models make it possible to deal with this uncertainty as they can represent it flexibly by means of parameters and functions.
— *Dynamic behavior*. Some processes may have a time dependent behavior. There is no doubt that some software process variables vary their behavior as the time cycle progresses. With a simulation model it is possible to represent and formalize the structures and causal relationships that dictate the dynamic behavior of the system.
— *Feedback*. In some systems the result of a decision made in a certain moment can affect their future behavior. For example, in software projects the decision of reducing the effort assigned to quality assurance activities has different effects over the whole progress of these projects.

Thus, the common objectives of simulation models consist of supplying mechanisms to experiment, predict, learn, and answer questions such as: *What if* . . . ? A software process simulation model can be focussed on certain aspects of the software process or the organization. It is important to bear in mind that a simulation model constitutes an abstraction of the real system, and so it represents only the parts of the system that have been intended to be modeled. Furthermore, currently available modeling tools such as ithink® (High Performance Systems, 2001), POWER-SIM® (PowerSim Corporation, 2001), and Vensim® (Ventana Systems, 2002) help to represent the software development process as a system of differential equations. This is a remarkable characteristic, as it makes it possible to formalize and develop a scientific basis for software process modeling and improvement. Some noticeable applications of this dynamic approach to model software process can be found in (Kellner, Madachy, and Raffo, 1999).

3. Justification

Although traditional methods for software project management have revealed their weaknesses during the last decades, there is common agreement that they are still useful. We think that it is important to integrate traditional methods and process simulation under a common approach, in order to obtain a valuable tool to design

process improvements and evaluate their effects. Rodrigues and Bowers (1996), draw the conclusions obtained after having compared both approaches, and point out the necessity of integration. Traditional and dynamic approaches have similar objectives, yet the perspective under which they work is completely different. Traditional methods are normally based on a Top-down decomposition of the software project, while the dynamic method can be characterized by the aggregation process it is focussed on, according to which, some features of a project are joined together under a simulation model. In accordance with what has been said, it can be deduced that dynamic models are suitable to deal with problems placed at the strategic level, while traditional methods are useful at the operational level of software projects.

With the development of a Dynamic Integrated Framework for Software Process Improvement (DIFSPI) we offer a methodology and working environment to join both approaches and to allow project managers and members of the Software Engineering Improvement Group (SEIG) to design and evaluate new process improvements. One of the main objectives of DIFSPI is to support the evolution of the maturity level of an organization according to the Capability Maturity Model (Paulk et al., 1993). The process of design and development of both the framework and the dynamic models that integrate it, allows one to define a metrics collection program. These metrics are necessary to both initialize and validate the dynamic models. This metrics collection is not only useful for the dynamic models; it also serves as an invaluable opportunity to obtain a real knowledge of the state of the software processes inside an organization. This knowledge is essential before tackling any process improvement.

4. DIFSPI development

4.1. Conceptual approach

Using simulation for process improvement in conjunction with CMM is not a new idea. As a matter of fact, (Christie, 1999) suggests that CMM is an excellent incremental framework to gain experience through process simulation. Nevertheless, there is a lack of a dynamic framework capable of assessment in the achievement of higher process maturity. One of the main features of DIFSPI is that this assessment is provided not only by using the associated final tool, but during the development of the whole dynamic framework. The reason for this is that the benefits that can be obtained with the utilization of dynamic models inside an organization, are directly related to the knowledge and the empirical information the organization has about its processes. Figure 1 illustrates this idea. It shows the existing causal relationships among the maturity level of the organization, the utilization of dynamic models and the benefits obtained.

The positive feedback loop comes to illustrate the causal relationship that reinforces the metrics collection inside the organization. The metrics collected will be used to calibrate and initialize the dynamic models. Lower maturity organizations are characterized by the absence of metric programs and historical databases. In this case, it is necessary to begin by identifying the general processes and the information that has to be collected about them. The questions of what to collect, at

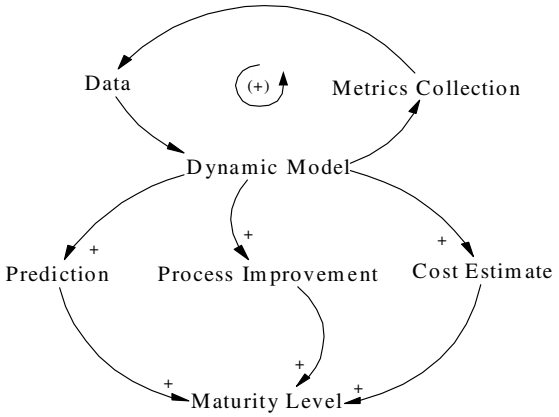

Figure 1. Causal relationships derived from the development and utilization of dynamic models.

what frequency, and with what accuracy have to be answered at this moment. The design process of dynamic models helps to come to a solution to these questions. When developing a dynamic model it is required to know: a) what is intended to be modeled, b) the scope of the model, and c) what behaviors need to be analyzed. Once the model is developed, it needs to be initialized with a set of initial conditions in order to execute the runs and obtain the simulated behaviors. These initial conditions customize the model to the project and to the organization to be simulated and they are effectively implemented by a set of initial parameters. These parameters that rule the evolution of the model runs answer precisely the former question of what data collect: those data required to initialize and validate the model will be the main components of the metrics collection program.

Once the components of the metrics collection program have been defined it can be implemented inside the organization. This process will lead to the achievement of a historical database. The data gathered can then be used to simulate and empirically validate the dynamic model. When the dynamic model has been validated, the results of its runs can be used to generate a simulated database; with this database it is possible to perform process improvement analyses. An increase in the complexity of the actions intended to be analyzed will directly lead to an increase in the complexity of the dynamic model required and, therefore, to a new metrics collection program for the new simulation modules.

The bottom half of Figure 1 illustrates the effects derived from the utilization of dynamic models in the context of process improvement. Using dynamic models which have been designed and calibrated according to an organization's data provides three important benefits. Firstly, the data of the simulation runs can be used to predict the future evolution of the project. The graphical representations of these data show the evolution of the project from a set of initial conditions (which have been established by the initialization parameters). By analyzing these graphics, organizations with a low level of maturity can obtain a useful qualitative knowledge about the evolution of the project. As the maturity level of the organization increases, the knowledge about its processes is also higher and the

simulation runs can be used as real quantitative estimates. These estimates help to predict the future evolution of the project with an accuracy that is intimately related to the uncertainty of the initial parameters. Secondly, it becomes possible to define and experiment with different process improvements by analyzing the different simulation runs. This capability helps in the decision-making process, as only the improvements which gave the best results will be implemented. Moreover, one of the most remarkable things here is that these experiments are performed with no cost and risk to the organization as they use the simulation of scenarios. Thirdly, the simulation model can also be used to predict the cost of the project; this cost can be referred to as the overall cost, or to a hierarchical decomposition of the total cost, as for instance, the cost of quality or revision activities. These three benefits are the main factors that lead to the achievement of a higher maturity level inside an organization according to CMM.

4.2. DIFSPI structure

Project management is composed of activities which are intimately interrelated in the sense that a certain action performed over a determined area will possibly affect other areas. For instance, a time delay will always affect the cost of the project but it may or may not affect the morale of the development team, or the quality of the product. The interactions among the different areas of project management are so strong that on some occasions the throughput of one of them can only be achieved by reducing the throughput of another. A clear example of this behavior can be found in the frequent practice of reducing the quality, or the number of requirements to be implemented in a certain version of the product with the aim of meeting the time or cost estimates.

Dynamic models help to understand the integrated nature of project management, as they describe it through different processes, structures, and main interrelationships. In the framework proposed here, project management is considered as a set of dynamic interrelated processes. Projects are composed of processes. Each process is composed of a series of activities designed for the achievement of an objective (Paulk et al., 1993). From a general point of view, it could be said that projects are composed of processes which fall into one of the following categories:

— *Management process*. This category collects all those processes related to the description, organization, and control of the project.
— *Engineering process*. All those processes related to the specification and development activities of the software product are collected in this category.

Both categories interact during the time cycle of the project as Figure 2 shows. From an initial plan performed by the project management processes, engineering processes begin to be executed. Using the information gathered about the progress of this second group of processes, project management processes determine the modifications that need to be made to the plan in order to achieve the project objectives. The DIFSPI proposed follows this same classification and is structured

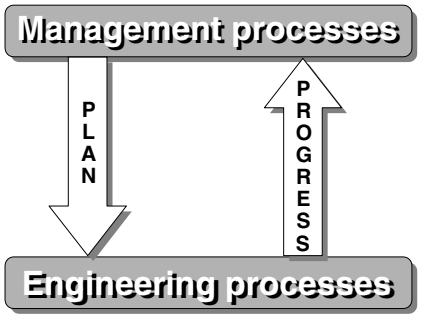

Figure 2. Classification of processes for software development.

to attend to project management and engineering processes. In both levels, the utilization of dynamic models to simulate real processes and to define and develop a historical database, will be the main feature.

4.2.1. Engineering processes in the DIFSPI. On this level the dynamic models simulate the life cycle of the software product. The benefits that simulation provides at this level are the following:

— To build a model it is necessary to improve the knowledge one has about the software development process, as it is required to establish the limits and scope of those real behaviors to be modeled and simulated.
— The parameters required by the model and the tables which determine its time behavior will constitute the main elements of a metrics collection program to define a historical database.
— The effective application of this metrics program will feed the database. The historical data gathered will help assess in the validation and calibration of the model.
— The dynamic model will finally simulate the software processes with the knowledge and the maturity that the organization has at the moment.
— The utilization of the dynamic model allows the establishment of a baseline for the project, the investigation of possible improvements, and the development of a historical database which can be fed either by real or simulated data.

The dynamic models of this level at DIFSPI should follow the levels of visibility and knowledge of the engineering processes that organizations have at each maturity level. It is obvious that the complexity of the dynamic model used in level 1 organizations cannot be the same as that of the models capable of simulating the engineering processes of, for instance, level 4 organizations.

4.2.2. Project management processes in the DIFSPI. Management processes are divided into two main categories:

— *Plan*. This groups the processes devoted to the design of the initial plan and the required modifications when the progress reports indicate the appearance of

problems. The models of this group integrate traditional estimation and planning techniques together with dynamic ones.
— *Control*. In this group all the models designed for the monitoring and tracking activities are gathered. These models will also have the responsibility of determining the corrective actions to the project plan. Therefore, the simulation of process improvements will be of enormous importance.

4.3. Elaboration of the dynamic models

The approach followed in the construction of the dynamic models is based on two fundamental principles:

— The principle of extensibility of dynamic models. According to this principle, different dynamic modules are joined to an initial and basic dynamic model. This initial model models the fundamental behavior of a software project. Each one of the dynamic modules models each one of the key process areas which conform the step to evolve to the next level of maturity. These modules can be either "enabled" or "disabled" according to the objectives of the project manager or the members of the SEIG.
— The principle of aggregation/decomposition of tasks according to the level of abstraction required for the model. Two levels of aggregation/decomposition are used:

  - Horizontal aggregation/decomposition according to which different sequential tasks are aggregated into a unique task with a unique schedule.
  - Vertical aggregation/decomposition according to which different and individual, but interrelated and parallel tasks are considered as a unique task with a unique schedule.

The definition of the right level of aggregation and/or decomposition for the tasks mainly affects the modeling of the engineering activities and principally depends on the maturity level of the process intended to be simulated.

To define the initial dynamic model the common feedback loops among the software projects were taken into account. The objective of this approach was trying to achieve a generic model and avoid modeling certain behaviors of concrete organizations which might limit the flexibility of the DIFSPI. To initialize the functions and parameters of the initial model, data originating from historical databases collected in the available literature were used (Putnam, 1992). By replicating some of the equations of the initial model it is possible to model the progress to higher maturity levels.

Figures 3 and 4 illustrate the former idea of basic modeling and structure replication. The initial model can be used to simulate software projects developed in organizations progressing to level 2. Figure 3 uses System Dynamics notation to illustrate the components developed to model the software development activity. The number of tasks to be developed is determined from an initial estimate of

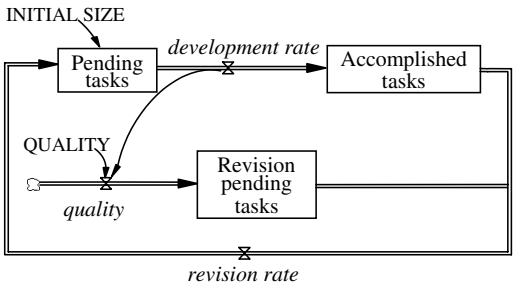

Figure 3. Basic dynamic module for software development modeling.

the size of the project. These pending tasks become accomplished tasks according to the development rate. During this process, errors can be committed. Thus, in accordance to the desired quality objective for the project, the quality rate and the revision rate are determined. These two rates govern the number of tasks that are revised. To model the progress to level 3, the model will make use of a horizontal decomposition, creating as many substructures as phases or activities are present in the task breakdown structure of the project (analysis, design, code and test, in our case). According to this approach, each time a complete model or some part of it is replicated, it will be necessary to define the new fixing mechanisms (dynamic modules) for the new structures. These mechanisms effectively implement the principle of aggregation/decomposition previously mentioned. The replication of structures also provides the possibility of replicating the modules related to the project management processes. This replication is especially useful for high maturity level organizations, which will be able to establish process improvement practices for each certain activity of the life cycle.

In Figure 4, the components of the dynamic model for level 3 organizations are shown. Each one of the four boxes labelled with the name of one phase of the project is, in fact, a complete dynamic module identical to that shown in Figure 3. The new features added to the replicated structure define the coupling structure

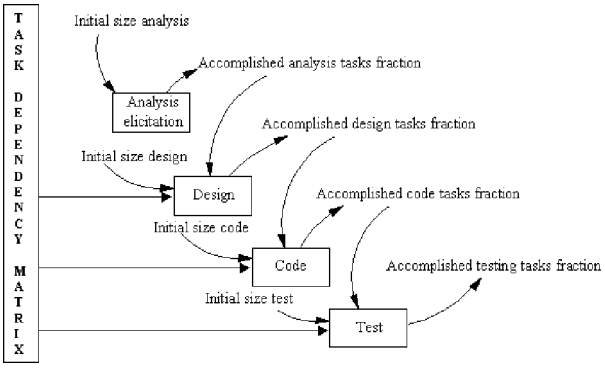

Figure 4. Replication of the basic dynamic module to model higher maturity processes.

Table 1. Main differential equations for the basic dynamic module for software development modeling (maturity level 1→2)

$\textit{Pending tasks}(t) = \int_0^{\text{Initial size}} (\text{revision rate}(t) - \text{development rate}(t))\, dt$
$\textit{Acomplished tasks}(t) = \int (\text{development rate}(t) - \text{revision rate}(t))\, dt$
$\textit{Revision pending tasks}(t) = \int (\text{quality rate}(t) - \text{revision rate}(t))\, dt$
$\textit{quality rate}(t) = \text{development rate}(t) * (1 - \text{QUALITY})$

that joins together the four dynamic modules. This coupling structure is mainly composed of a matrix that keeps the order of precedence among the phases and the percentage of a task that have to be accomplished before starting the following one. The accomplished task fraction of each phase is the value that each dynamic module shares with the others as well as the common knowledge of the matrix of precedence.

Table 2 shows the new form of the equations for this level. It is important to notice here the appearance of the double index to identify the phase of the project in contrast with the equations shown in Table 1, and also the fact that these equations are only calculated when the conditions are favourable according to the matrix of precedence.

5. DIFSPI implementation

The former conceptual ideas have been implemented to develop a tool using Java™ technology (Java 2 SDK v. 1.2 and 1.3). Three groups of initial parameters are required to drive a simulation run: parameters related to the organization (delays, average accepted overwork, etc.), parameters related to the project (size, quality objective, initial staff, etc.), and parameters to drive the simulation run. With these initial data, it is possible to run a first simulation to establish a baseline to the project. The results obtained can be graphically displayed in order to merge in a single view the static data offered by the traditional models with the dynamic data provided by the simulation runs. After this, it is possible to experiment with different process improvements and alternative plans by changing the values of the parameter(s) required and running the new simulations. All the results obtained are saved in the database. This database may be used then to feed some machine learning algorithms in order to automatically obtain management and process improvement rules (Ramos et al., 2000).

Table 2. Main differential equations of the replicated module for software development modeling (maturity level 2→3)

$\textit{Pending tasks}(i,t) = \int_0^{\text{Initial size}(i)} (\text{revision rate}(i,t) - \text{development rate}(i,t))\, dt$
$\textit{Acomplished tasks}(i,t) = \int (\text{development rate}(i,t) - \text{revision rate}(i,t))\, dt$
$\textit{Revision pending tasks}(i,t) = \int (\text{quality rate}(i,t) - \text{revision rate}(i,t))\, dt$
$\textit{quality rate}(i,t) = \text{development rate}(i,t) * (1 - \text{QUALITY})$

Table 3. Real and simulated data for the case study

Size of the project = 80,000 LOC			
REAL DATA		SIMULATED DATA	
Time	250 days	*Time*	263 days
Initial workforce	8 technician	*Effort*	4,361 technician-day
Effort	4,780 technician-day	*Quality*	80% (tasks revised)
		Workforce	9 technician

5.1. DIFSPI utilization

The potential applications of the DIFSPI have already been mentioned in the former sections. In this section some of the data obtained when DIFSPI was applied inside a local software development organization are provided. This local organization could be placed at level 1. At first the software process capability of this organization was unpredictable because it was constantly changed or modified as the work progressed. Performance depended on both the capabilities of the project manager and the technical team. Moreover, there were few stable software processes in evidence. According to level 1 organizations, the software process here was perceived as an amorphous entity, "a black box", and visibility into the project's processes was very limited. Requirements flowed into the software process in an uncontrolled manner, giving a product as a result. The purpose of this application was to ensure that the framework could reproduce the behavior observed in a real project and, therefore, could trigger a metrics collection program and help in decision making, predicting, and cost estimating. Table 3 shows the characteristics of the project that was simulated for this case study together with the data of the baseline reported by the simulation. It should be noted here that the data reported by the simulation conforms the core measures recommended by the Software Engineering Institute (SEI) (Carleton et al., 1992).

The scenario shown in Table 4 helps to analyze the impact of the size of the technical staff over the main four variables (time, effort, quality, and overall workforce). Two different cases were simulated. The first one (CASE 1) had a schedule of 250 days and 16 part-time technicians. The second case (CASE 2) had a schedule of 150 days and 16 full-time technicians.

The expected behavior for projects with a high level of personnel is that the average productivity per technician achieved will be lower. The average productivity per technician in the baseline was 0.8926 tasks/(technician $*$ day). CASE 1 and 2

Table 4. Simulated data for scenario analysis

CASE 1		CASE 2	
Time	135 days	*Time*	140 days
Effort	1,396 technician-day	*Effort*	3,596 technician-day
Quality	91%	*Quality*	91%
Workforce	18 technician	*Workforce*	16 technician

both had double the initial workforce than that of the baseline, although schedules and resource allocation were different between them. The average productivity obtained for CASE 1 and 2 was, respectively, 0.8277 tasks/(technician $*$ day) and 0.8142 tasks/(technician/day).

6. Conclusions

Motivated by lessons learnt from another System Dynamics application in an industrial environment, the development of a framework to combine the traditional estimation tools with the dynamic approach has been initiated. The main objective of this dynamic framework is to assess project managers and members of the SEIG to define, evaluate, and implement process improvements to achieve higher levels of maturity. The whole process of development of the framework also helps to design a specific metrics collection program which, once implemented, contributes to build and feed a historical database inside an organization.

With the application of DIFSPI in a level 1 organization, important benefits were obtained. First, it must be mentioned that during the process of model building, the project manager gained much new insight into those aspects of the development process that mostly influence the success of the project (time, cost, and quality). Second, having the possibility of gaming with the DIFSPI allowed him to better understand the underlying dynamics of the software process. As a consequence, several process improvement suggestions were easily designed and, most importantly, analyzed using the simulation of scenarios. Finally, templates and guidelines for a metrics collection program were almost automatically derived from the requirements of the dynamic modules.

Our future work will mainly concentrate on research towards a full development of the dynamic modules that implement the key process areas of the higher maturity levels. Once this development has been accomplished, it is intended to validate the complete DIFSPI in real industrial environments.

Acknowledgements

The authors wish to thank the Comisión Interministerial de Ciencia y Tecnología, Spain, (undergrant TIC2001-1143-C03-02) for supporting this research effort.

References

Carleton, A., Park, R.E., Goethert, W.B., Florac, W.A., Bailey, E.K., and Pfleeger, S.L. 1992. Software measurement for DoD systems: Recommendations for initial core measures, Technical Report CMU/SEI-92-TR-19. Software Engineering Institute, Pennsylvania, Pittsburgh, Carnegie Mellon University.

Christie, A.M. 1999. Simulation—An enabling technology in software engineering. *http://www.sei.cmu.edu/publications/articles/christie-apr1999/christie-apr1999.html.*

Christie, A.M. 1999. Simulation in support of CMM-based process improvement, *The J. of Systems and Software*, 46(2/3): 107–112.

Eberlein, R.L. 1989. Simplification and understanding of models, *Systems Dynamics Review.* 1(5): 51–68.

High Performance Systems, Inc., 2001. ithink 7.0, New Hampshire, Hanover.
Kellner, M.I., Madachy, R., and Raffo, D. 1999. Software process simulation modeling: Why? What? How? *The J. of Systems and Software.* 46(2/3): 91–105.
Paulk, M., Garcia, S.M., Chrissis, M.B., and Bush, M. 1993. Key practices of the capability maturity model, Version 1.1, Technical Report CMU/SEI-93-TR-25. Software Engineering Institute, Pennsylvania, Pittsburgh, Carnegie Mellon University.
PowerSim Corporation, 2001. Powersim Studio 2001. Virginia, Hendon.
Putnam, L.H. 1992. *Measures for Excellence: Reliable Software, On Time, within Budget*, New York, Prentice-Hall.
Ramos, I., Aguilar, J., Riquelme, J.C., and Toro, M. 2000. A new method for obtaining software project management rules, *SQM 2000*, United Kingdom, Greenwich, pp. 149–160.
Rodrigues, A. and Bowers, B. 1996. System dynamics in project management: A comparative analysis with traditional methods, *System Dynamics Review.* 12(2): 121–139.
Ruiz, M., Ramos, I., and Toro, M. 2001. A simplified model of software project dynamics, *The J. of Systems and Software.* 59: 299–309.
Ventana Systems, Inc., 2002. Vensim Version 5. Massachusetts, Harvard.