Integrating Dynamic Models for CMM-Based Software Process Improvement

Mercedes Ruiz[1], Isabel Ramos[2], and Miguel Toro[2]

[1] Department of Computer Languages and Systems
Escuela Superior de Ingeniería. University of Cádiz
C/ Chile, nº1. 11003 – Cádiz, Spain
mercedes.ruiz@uca.es
[2] Department of Computer Languages and Systems
Escuela Técnica Superior de Ingeniería Informática. University of Seville
Avda. Reina Mercedes, s/n. 41012 – Seville. Spain
{isabel.ramos,mtoro}@lsi.us.es

Abstract. 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 static 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 paper describes the concepts underlying this framework, its implementation, the dynamic approach followed to systematically develop the dynamic modules, and an example of its potential use and benefits.

1 Introduction

World-wide the demand for high complex software 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. The rapid pace in which this software is required, 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. Over the last few decades, the software industry has received a significant help by means of CASE tools, new programming languages and approaches, and more advanced and complex machines. However, it is widely accepted that the potential benefit of a better technology cannot be translated into a more successful project if the processes used to execute it are not

M. Oivo and S. Komi-Sirviö (Eds.): PROFES 2002, LNCS 2559, pp. 63-77, 2002.

well defined, established, and executed. Proper processes are essential for an organization to consistently deliver high quality products with a high productivity.

Currently, many frameworks are available for software processes, being CMM [1] and ISO 9001[2] the most influential and widely used. Although ISO 9001 is a standard, and has been interpreted for a software organization in ISO 9000-3 [3], it has been written from the customer and external auditor's perspective. On the other hand, CMM is not a binary certification process, but a framework that categorizes software process in five levels of maturity and provides roadmaps to evaluate the software process of an organization as well as planning software process improvements.

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. However, simulation is only effective if both the model and the data used to drive it, accurately reflect the real world. As a consequence, it is possible to say that the construction of a dynamic model for software process itself points to what metric data must be required and, hence, collected, providing clear guidelines on what to collect.

After having applied the system dynamics approach to model and assess software process improvement in a local organization [4], lessons learnt from this experience moved us towards the development of, initially, a framework and, then, a working environment which combines the traditional and static techniques used in project management with the process dynamic modeling and simulation.

The aim of this paper is to present this combination 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 [1]. 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 underestimate of cost or schedule. During the simulation metric data are saved. These data conform to the SEI core measures [5] recommendation and are mainly related to cost, schedule and quality.

The structure of the paper is as follows. Section 2 describes in detail the dynamic framework proposed. It includes the motivation we found to design and develop it, the conceptual approach followed, the potential uses of the framework, its architecture, a brief description of the different techniques which have been integrated in the framework, and how it has been implemented to develop a fully functional working tool. In Section 3 an example of how the framework may be used to design and evaluate the effect of a certain process improvement is presented. The example shows the use of the integrated techniques together with the dynamic modules inside the framework. Finally, Section 4 summarizes the paper and draws the conclusions and lessons learnt.

2 Description of the Framework

2.1 Motivation

The Dynamic Integrated Framework for Software Process Improvement (DIFSPI) has been designed with the aim of creating both a conceptual framework and a working environment to help in the achievement of higher maturity levels according to CMM. Traditionally, system dynamics and simulation have been considered as helpful tools to design and evaluate software process improvements. In fact, some real applications can be found in the literature regarding the application of this approach inside software development organizations [6]. All these applications share a common characteristic: they conclude that for a successful system dynamics application, it is a requirement for the organization to have their processes well defined and structured, as well as defined metric programs to provide the models with accurate data. It is absolutely true that dynamic models are only useful if the data used to drive their numerical parameters are available in the organization, and that the availability of these data is only possible if a metric process is applied to the software process executed at the organization. As a consequence, many of the applications of system dynamics in the field of process improvement have required from organizations a certain level of maturity in their processes, typically level 3 or above according to CMM.

In order to design a software process dynamic model is necessary to know the features of the process which is going to be modeled. It becomes apparent that the higher the maturity level, the better the process is defined and established, the easier to obtain information to develop the dynamic model, and, therefore, the better results are obtained when using the model as a tool for evaluating software process improvements. Bearing this statement in mind, our interest is to help organizations with less mature processes to design and execute the process improvements which could help them to achieve upper levels of maturity.

Thus, we propose an approach which is mainly focussed on the development of dynamic models, using them to trigger a process inside the organization aimed to increase the level of knowledge it has about its software process, and to design a software metrics collection program which can be applied to obtain the required data to drive the parameters of the dynamic model. This metric 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. The utilization of dynamic models is the main technique in the framework, but it is not the only. Static or traditional algorithmic models are integrated with the dynamic models in order to supply important information during the early stages of the life cycle simulated. Other recommendations and techniques are also used inside the framework to provide useful tools of analysis and evaluation of the results obtained through simulation.

2.2 Conceptual Approach of the Framework

Static models use empirically obtained formulas to compute some project estimates as a function of some project initial estimation, typically the size. The empirical data that support these models come from a limited sample of projects, and this requires a careful utilization of these models, as only if the features of the project under estimation are similar to those of the projects used to calibrate the static models, the results obtained will be reliable. Although static methods for software project management have revealed during the last decades their weaknesses, there is common agreement that they are still useful. At least, in lower maturity organizations they are useful to establish an initial baseline for the project with which evaluate the results of the project and the process improvement tentative. Static and dynamic models have similar objectives, yet the perspective under which they work is completely different. Static models are normally based on a top-down decomposition of the software project, while the dynamic models can be characterized by the aggregation process they are 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. By combining both approaches in a common framework, a useful methodology and even a working tool can be developed to provide insight into the software process and to help in the achievement of upper maturity levels. Nevertheless, the achievement of higher maturity levels can only be possible if the data that enable the evaluation of the results of the process improvement practices are available. Therefore, the design of a proper metrics collection program for the organization is mandatory. The metrics collected are useful in many different aspects:

1. The metrics collected must 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 completely necessary to begin identifying the general processes and the information that has to be collected about them. The questions of what to collect, at 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 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.
2. 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. Thinking from a perspective of causal loops, what has been described is no more than the translation of a positive reinforce causal loop which is shown in Figure 1.

The benefits of using dynamic models have been widely discussed in the literature [7]. It can be concluded that the use of these models leads to better prediction activities, more accurate cost predictions, and more effective process improvement initiatives. Three factors which are known to drive organizations towards upper maturity levels according to CMM.

2.3 Potential Use of the Framework

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 [1]. From a general point of view, it could be said that projects are composed of processes which fall in 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 lifecycle of the project. 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 it is structured attending to project management and engineering processes. In both levels, the utilization of dynamic models to simulate the real processes and to define and develop a historical database will be the main feature.

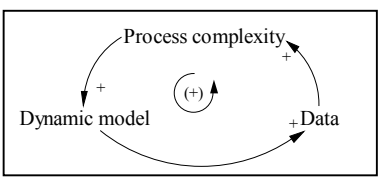

Fig. 1. Causal diagram

2.3.1 Engineering Processes in the Framework

On this level the dynamic models simulate the life cycle of the software product. Figure 2 shows a schematic representation of how DIFSPI may be used inside an organization. In low maturity organizations, the amount of information required to begin running simulations is relatively small and mainly focussed on the initial estimations, that is the estimated size of the project, and the initial size of the working team. Depending on the paradigm followed to develop the software product and the maturity level of the organization, the suitable dynamic model is simulated. The main paradigms that can be simulated inside the framework are the traditional waterfall and COTS paradigms. Depending on the paradigm chosen, different dynamic modules will be joined in order to create a final and fully operational dynamic model. Once the simulation has been run, it provides data that are saved in a simulated database. These initial data contain the results of the simulation together with a set of initial estimations resulting from the computation of the static models. These initial estimations establish the base line for the project, and the simulated data obtained represent the dynamic evolution of the project variables along the whole life cycle. As well as the initial baseline and the simulated data, the so-called simulated database contains a third component. This third component contains the results of applying some other techniques during the simulation of the project, which are oriented towards the gaining of insight into the process under simulation. These techniques, which have been integrated with the dynamic modules, are described in Section 2.5.

As it was mentioned before, the process of modeling the software process requires a good knowledge of the software process itself, and triggers a metrics collection program which can then be used to initialize the parameters of the model and increment the level of visibility the model has about the process. All that has been simulated so far, must be taken into practice. After having determined the initial estimates and run the simulations to establish the initial base line, it is possible to run different scenarios in order to find out the outcomes of different initial values for the project estimates. This reflects, of course, the level of uncertainty low maturity organizations have at the initial stages of a project. When the real project begins, the metrics collection program may be applied to gather real information about the progress. These real data are also saved in the database, enabling the development of a historical database. As these data become available, it is possible to perform analysis and calibrate the functions and parameters of the dynamic modules so that their accuracy may be improved. Improving the accuracy of the dynamic modules may require an improvement in the knowledge we have about the software process and, this way, the loop is closed.

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 one of the models capable of simulating the engineering processes of, for instance, level 4 organizations.

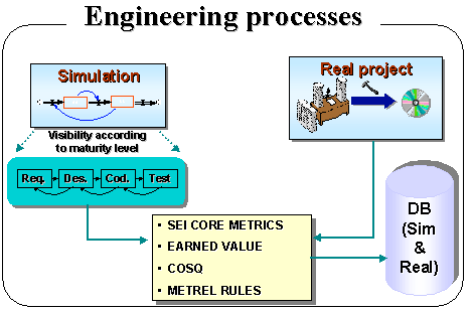

Fig. 2. Engineering processes in DIFSPI

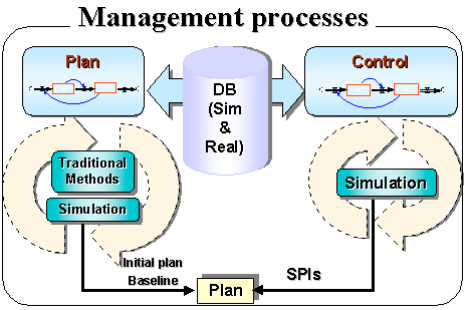

Fig. 3. Management processes in DIFSPI

2.3.2 Project Management Processes in the Framework

Inside the framework, management processes are divided into two main categories:

Plan. It 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 an enormous importance.

Figure 3 shows the utilization of DIFSPI at this level. As it was mentioned before, the initial baseline for the project is established using the static models built inside the framework. The dynamic modules that model the planning activities performed in the organization have not only the differential equations to model these activities, but the equations of the traditional static estimation models. In order to gain a useful information from these static models, the same knowledge about the software process which is required to use these models is necessary at this moment.

The control modules model and simulate all the activities which determine the progress of the project, and make the corrective decisions which are required to meet the project objectives. These modules have a great importance in the design of the process improvements.

2.4 Dynamic Modules Architecture

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

1. 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.

2. 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 scheduled too.

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 to achieve a generic model and avoided modeling specific behaviors of concrete organizations which might limit the flexibility of DIFSPI. To initialize the functions and parameters of the initial model, data originating of historical databases collected in the available literature were used [8]. Figure 4 shows the main structure of the initial model. Four dynamic modules are joined together to develop an operational model that provides the set of final differential equations to generically simulate the software process in low maturity organizations.

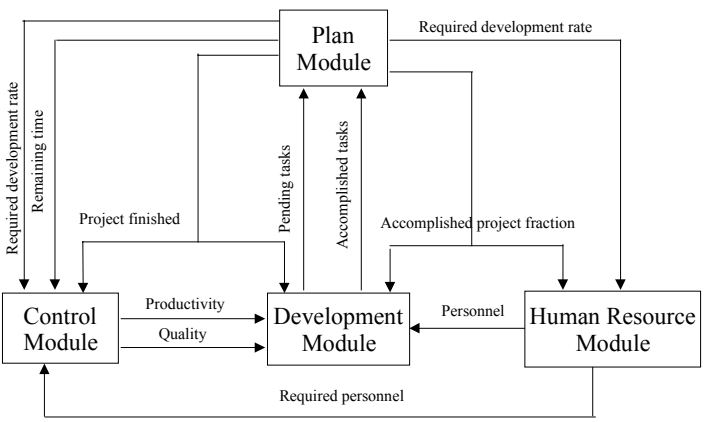

Fig. 4. Submodules architecture of the initial model

By replicating some of the equations of the initial model it is possible to model the progress to higher maturity levels. The initial model can be used to simulate software projects developed in organizations progressing to level 2. Generally speaking, the software product development process can be considered as follows. The number of tasks to be developed is determined from an initial estimate of 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 the waterfall paradigm). 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.

2.5 Integrated Techniques

As it was mentioned before, our aim is to develop a working environment where the simulation of different scenarios can be used to generate the simulated database where managers can experiment different process improvements and activities focussed on the implementation of metrics programs and value analysis. The following techniques and methods are currently successfully implemented in DIFSPI:

Traditional Estimation Techniques. Traditional algorithmic estimation models have been implemented inside this framework with the aim of providing an initial baseline for software projects carried out in low maturity level organizations [9] and [10].

SEI Core Measures. Recent studies and experiences highlight the benefits of the application of these four core measures to the software life cycle. The main aspects of the product and process (quality, time, size and cost) are monitored and tracked to facilitate project success and higher maturity achievement. Inside this framework theses four measures constitute the basics for both, the dynamic models and the graphical representation of the process performance [5].

Metrel Rules. Given the dynamical nature of the DIFSPI proposed, we consider it could be useful to integrate a taxonomy of software metrics which is derived from the needs of users, developers, and management. Among all the advantages that can be obtained with the use of this system of metrics, we would like to point out the dynamic performance of these metrics, that is, how their accuracy, precision, and utility changes over the duration of a project, the life of a product or the strategic plan of an organization. In DIFPSPI Metrel rules have been used as an efficient method for

depicting on one graph the information needed for management, staff, and customers to view or predict process performance results. We consider that Metrel rules are particularly important in the field of software process modeling as their application provides a formal procedure for the expansion and transformation of models. By employing simple mechanisms as derivative or integration (over time, phases or even projects), a mathematical model for one level can be transformed into another for another level, providing a simple but powerful extension for the analysis processes [11].

CoSQ. The basis for the Cost of Software Quality (CoSQ) is the accounting of two kinds of costs: those which are due to a lack of quality and those which are due to the achievement of quality. These costs have an inverse relationship that is normally shown as a set of two-dimensional curves that plot costs against a measure of quality. We think that CoSQ can help not only to justify quality initiatives, but also to a number of other benefits. Among them, we would like to point out that CoSQ provides the basics for measuring and comparing the cost effectiveness of the quality improvements undertaken by an organization [12].

Earned Value Analysis. Earned value analysis has been chosen as the method for performance measurement as it integrates scope, cost, and schedule measures to help managers assess process performance. The three main values and the derived efficiency indexes are used in combination to provide measures of whether or not work is being accomplished as planned. Furthermore, the earned value analysis is used to evaluate the performance of different software process improvements inside DIFSPI [13].

Statistical Process Control. Current software process models (CMM, SPICE, etc.) strongly recommend the applications of statistical control. In the framework, Statistical Process Control (SPC) is used to obtain run charts and control charts with the aim of helping software managers to find an answer for questions such as "How do I know if my software development process is in control?" SPC is also used to test the capability of the process. In order to do that, SPC and earned value techniques have been merged in the way [14] suggests.

2.6 Implementation of the Framework

The conceptual ideas presented above were firstly implemented using VemSim® [15] which was used for developing and analyzing the different dynamic models. However, there are some limitations to using this tool. This simulation environment provides a crude way of modularization, there is no easy way to both overlay objects for abstraction and to generate a generic sub-model so that it can be instantiated multiple times without duplicating the effort, and since there is no scoping mechanism, all the elements are global to each other. Like traditional programming languages, a mechanism to allow data encapsulation and modularity is essential to handle the complexity in large and complex models. Due to this reason, the complete framework is currently being re-engineered using Java™ technology. The purpose of this process is to develop a library of classes, each of which represents a simple

dynamic module. When using this tool, a suitable dynamic model is made of the required objects. This way, the abstraction aspect and standardization of the interface of these defined modules may be taken to the point where project managers could transparently „plug-in“ the modules regarding the software process improvement they would like to analyze. This approach implies a special effort at the interfacing mechanism of these different modules when they are plugged together. Nevertheless, the results obtained after the first functional prototype have been successful and encouraging.

3 Example of Utilization of the Framework

In this section we present the results obtained when using the framework to design and evaluate a specific software process improvement, as well, as quantifying the return on investment of tackling this SPI and its benefits. This example pretends to show the utilization of the different integrated techniques in the SPI field. The approach to the SPI considered is the realization of software inspection activities which is a key factor for level 3 organizations.

Inspections are aimed to objectively identify the maximum number of software defects. Different studies have mentioned the problems and the benefits of these kinds of challenging activities [16]. Figure 5 uses system dynamics notation to illustrate the dynamics of the inspection process module.

The inspections module implements the general steps of the inspection process, the allocation of manpower to the inspection activities, the amount of errors found during inspection, and the defect prevention resulting from the inspection process. The most important factor to be considered here is how the module is obtained, that is, the

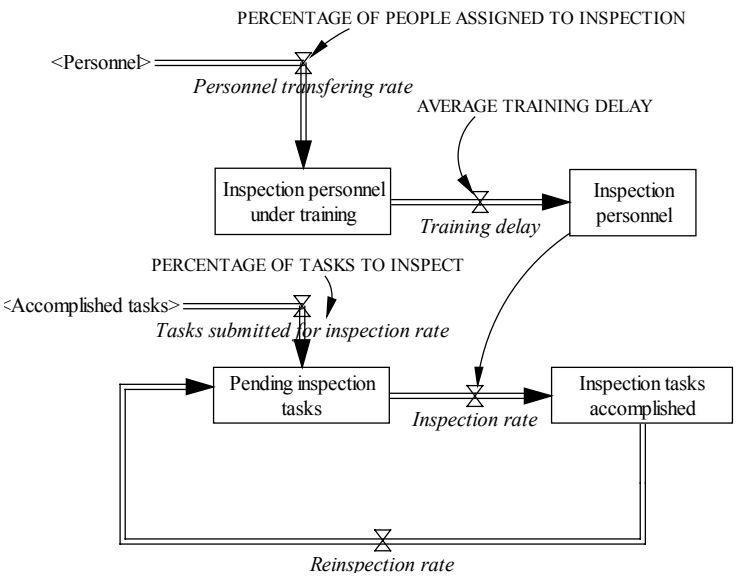

Fig. 5. Inspection module dynamics

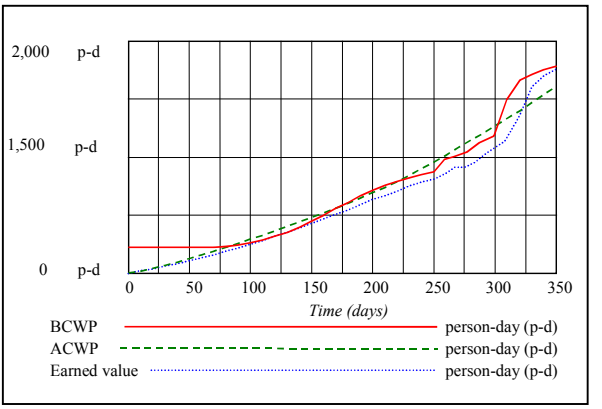

Fig. 6. Average earned value evolution

revision process module itself is the instantiation of other dynamic modules. In fact, the modeling of the workforce allocation, and personnel training which is oriented towards providing them the required abilities to perform the inspection activities, is no more than an instantiation of the generic human resource workforce of the initial model. The modeling of the development of the inspection activities is an instantiation of the generic development model, described in Section 2.4. Other parameters are necessary to model and determine the costs of performing this activity. These parameters have not been graphically represented to simplify the figure but have been coded in the equations of the model.

The final dynamic model results from the integration of the dynamic modules for a level 3 organization together with the module that models the SPI under study. To analyze the effect of this SPI over the project, as well as the effect of its different parameters, DIFSPI drove Monte Carlo simulations. These capability enables the database to be fed with a sufficient amount of information originating from a high number of projects simulated.

Figures 6 and 7 depict the results obtained in this example. Figure 6, shows the average evolution of the earned value associated to the SPI through the projects simulated. This average earned value can then be compared to the one obtained after having simulated the project with the parameters that reflect the current software process.

The application of the Metrel taxonomy can be observed in Figure 7. In the scenario of the example, the evolution of the percentage of detected and corrected errors (which is the process metric in this example) when the parameter „percentage of people assigned to inspection“ is varied. The curve represents the final value of the metric along two hundred simulated projects. All the projects implement the SPI with a different value for the parameter.

In Figure 7, *c* represents the current value of the metrics when applied to the software process of the organization, and *b* represents the desired value for the previously mentioned.

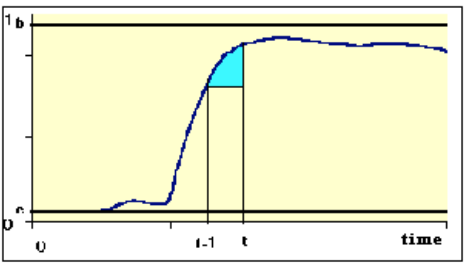

Fig. 7. Number of defects detected and correct due to the SPI

Once the model is simulated and their results graphically represented, it becomes apparent that the potential effect of the SPI in the whole software process follows a model which is determined by the Equation 1.

$$f(t) = b - (b - c)(1 - g)^t \quad . \tag{1}$$

According to this model $f(0) = c$ and $f(1) = c + (b\text{-}c)*g$, where g is the gain of the SPI. Let's assume that currently 60% of software errors are detected and corrected and that the objective of the organization is to get 80%. According to this objective, $b = 0.8$, and $c = 0.6$. Let's assume that the final value of the metric is $f(1) = 0.7$. With these values, the quantified improvement of the process is: $g = (0.7 - 0.6) / (0.8 - 0.6) = 0.5$.

This value is computed during the simulation of each project of the set that integrates the Monte Carlo simulation. It serves as an indicator of the capacity of the organization regarding this process improvement. Besides, and according to the Metrel taxonomy, a process metric as the one that has been used here, can be easily transformed by means of integration and derivative processes, in a metrics of the organization or the product. Equation 2 describes the model of the metric for the organization.

$$f'(t) = -(b - c)\,(1 - g)^t \,\mathrm{Ln}\,(1\text{-}g)\,. \tag{2}$$

In this example, given the values for b, c, and g, the obtained model is: $f'(t) = 0.14\,(0.5)^t$

$$\int f(t)dt = bt - (b - c)\,(1 - g)^t \,/\, Ln(1 - g)\ . \tag{3}$$

As far as the metric of the product is concerned, the integral of f(t) along a time interval provides a metric for the number of software errors detected and corrected during that time interval. This value is represented by the shadowed area in Figure 7.

In fact, this same Figure represents the three metrics at the same time, according to the taxonomy integrated. That is, the main curve represents the process metric, the tangent of this curve represents the organization metrics, and, finally and as it has been said before, the shadowed area, the product metrics.

4 Conclusions and Further Work

Motivated by lessons learnt from another System Dynamics application in an industrial environment, the development of a framework to combine the traditional static 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 some important benefits are expected to be obtained. First, during the process of model building, the project manager may gain 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, it allows project managers to better understand the underlying dynamics of the software process. As a consequence, several process improvement suggestions may easily be designed and, most importantly, analyzed using simulation of scenarios. Third, templates and guidelines for a metrics collection program may almost automatically derived from the requirements of the dynamic modules. Fourth, the approach of abstraction and encapsulation followed to develop the dynamic modules makes it possible to easily instantiate a dynamic model using different dynamic modules which can be plugged in the final model. Finally, the combination of the dynamic approach with other techniques allow project managers to perform complete analysis and quantification of the effects and the benefits of different software process improvements. All these features combined in the framework intend to help organizations to design and execute more mature process and, therefore, to increase their maturity level.

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, (under grant TIC2001-1143-C03-02) for supporting this research effort.

References

1. Paulk M., Garcia S.M., Chrissis M.B., Bush. M.: Key practices of the capability maturity model. Version 1.1 Technical Report CMU/SEI-93-TR-25. Software Engineering Institute, Carnegie Mellon University, Pittsburg, PA (1993)
2. International Standards Organization.: ISO 9001, Quality Systems – Model for Quality Assurance in Design/Development, Production, Installation, and Services (1987)
3. International Standards Organization.: ISO 9000-3, Guideliness for the Application of ISO9001 to the Development, Supply, and Maintenance of Software (1991)
4. Ruiz M., Ramos I., Toro M.: A simplified model of software project dynamics. Journal of Systems and Software, Vol. 59, No 3 (2001) 299-309
5. Carleton A., Park R.E., Goethert W.B., Florac W.A., Bailey E.K., Pfleeger S.L.: Software measurement for DoD systems: recommendations for initial core measures. Technical Report CMU/SEI-92-TR-19. Software Engineering Institute, Carnegie Mellon University, Pittsburg, PA (1992)
6. Prosim 200 workshop Proceedings. July, 12-14 2000. London UK
7. Christie AM. Simulation – An enabling technology in software engineering. http://www.sei.cmu.edu/publications/articles/christie-apr1999/christie-apr1999.html
8. Putnam L.H.: Measures for excellence: reliable software, on time, within budget. Prentice-Hall, New York (1992)
9. Boehm B.: Software Engineering Economics. Prentice-Hall Inc. (1981)
10. Boehm B., Horowitz E., Madachy R., Reifer D., Clark, B.K., Steece B., Brown A.W., Chulani S., Abts, C.: Software Cost Estimation with COCOMO II. Prentice-Hall Inc. (2000)
11. Woodings T.L.: A Taxonomy of Software Metrics. Software Process Improvement Network (SPIN) (1995)
12. Knox S.T.: Modeling the Cost of Software Quality. Digital Technical Journal, Vol. 5, No 4 (fall 1993), 9-16
13. Fleming Q.W., Koppelman JM.: Earned Value Project Management, 2nd Edition, Newton Square, Project Management Institute (1999)
14. Lipke W., Jennin, M.: Software Project Planning, Statistics, and Earned Value. Crosstalk (December 2000)
15. Vensim. Ventana Simulation Environment. Reference Manual, version 4, Belmont, MA (2000)
16. Siy H.P.: Identifying the mechanisms driving code inspection costs and benefits. Unpublished doctoral dissertation, University of Maryland, College Park (1996)