A simplified model of software project dynamics

Mercedes Ruiz , Isabel Ramos, Miguel Toro

b

1. Introduction

The publication, in 1991, of a dynamic model for managing software development projects (SDPs) by Abdel-Hamid and Madnick (1991), led to a new field which is allowing a better understanding of the different variables to be considered, and the complex relations produced between them. The multiple applications that dynamic simulations have and the current available simulation environments (like Estella, Vensim, Ithink, etc.) have opened some new working lines inside SDPs, where significant advances have not been produced in the past few years and where traditionally used methods and tools were becoming obsolete for the complexity and magnitude of the actual software projects.

Significant progress has been made in the area of software process simulation which is gaining an increasing interest among academic researchers and practitioners. As a result of this increasing interest, several dynamic models have been developed over the last decade (Rodrigues and Williams, 1997; Kellner et al., 1999). Each of these dynamic models contains a set of parameters and functions used to model the organisation and the management policies which can be applied to the software projects.

The size, complexity, and detail of a model are interrelated. Together, these characteristics form a delicate balance between a model's appropriateness of power and ease-of-use for its intended audience. A powerful model, one rich with possibilities for experimentation, will require more size, complexity, and detail than a less powerful model. With power, however, comes the likelihood for greater difficulty in understanding and using a model.

More precisely, one of the obstacles that seems to impede a more frequent application of the modelling and simulation approach in the software engineering community is the large number of parameters and functions which require to be initialised in order to run simulations. This fact can be an important drawback at the early stages of a project when the amount of information is little (for instance, in one's first contacts with a client).

* Corresponding author.
E-mail addresses: mercedes.ruiz@uca.es (M. Ruiz), isabel.ramos@lsi.us.es (I. Ramos), mtoro@lsi.us.es (M. Toro).

The model described in this paper brings a new contribution to the software process modelling and simulation work. The modelling approach has been determined by the simplification of Abdel-Hamid and Madnick's model using the works of Eberlein (1989) about understanding and simplification of models, as an alternative to the general modelling approach.

This paper has two main objectives. The first is to justify the development of a reduced dynamic model (hereinafter RDM) which should be easy to learn and understand and could be used at the initial phases of a project where the available or known information about the project is little. This model has three goals: to allow project managers to make initial estimations when the amount of information is little; to allow researchers and software project managers to experiment with strategies for reducing software development cycle time and cost through process improvements; and to provide a tool for hands-on training. The second objective of this paper is to present the results obtained from the validations of this RDM.

The structure of this paper is as follows. In Section 2, a brief analysis and comparison of static and dynamic cost models are outlined. The reasons found to develop a simplified model with a smaller number of parameters but still with a good capability of reproducing software project dynamics are presented in Section 3. Sections 4 and 5 list the reduction process followed and describe the model in detail. The testing activities and the validation of the model are presented in Section 6. Section 7 summarises the main results obtained and draws conclusions.

2. Cost models

Software cost estimation is the process of predicting the amount of effort required to build a software system. Cost estimates are necessary throughout the software life cycle. Preliminary estimates are required to determine the feasibility of a project, whereas detailed estimates are needed to assist with project planning. The actual effort for individual tasks is compared with estimated and planned values, enabling project managers to reallocate resources when necessary.

Analysis of historical project data indicates that cost trends can be correlated with certain measurable parameters. This observation has resulted in a wide range of models that can be used to assess, predict, and control software costs on a real-time basis.

Models can be categorised as either static models or dynamic models. Static models are derived using regression analysis on data collected from past software projects and they require the number of code lines or the number of function points as a primary input to provide direct estimates of effort. Dynamic models capture the dynamic and uncertain nature of software projects providing information about not only the values of the principal variables but their evolution.

We present a brief description of a traditional static cost estimation model like the constructive cost model (COCOMO) (Boehm, 1981) and of dynamic cost estimation models. Also a hierarchical classification of dynamic models similar to that of static ones is proposed.

2.1. Static cost models

These models use empirically obtained formulas (from a small sample of projects) to predict the required data. These formulas are represented in the following equation:

$$\text{Resource} = a \cdot (\text{estimated feature})^b, \tag{1}$$

where Resource can be the effort, development time, human resource level, or technical documentation lines; a and b are empirical constants derived from the calibration of the model; "estimated feature" can be the number of source code lines or function points, or another previously estimated software pattern. Among the traditional static cost estimation models, COCOMO (Boehm, 1981), is considered as the most complete and used model. COCOMO assumes that software requirements are not going to be altered after the plan and requirement analysis phases. In this sense, it is similar to Abdel-Hamid and Madnick's model, which also considers static client requirements. COCOMO includes the analysis, design, code and test phases and it has been formulated as a hierarchy of models.

The following gives a brief summary of each one of the COCOMO models, detailing the aspects that are interesting for this paper:

- Basic COCOMO is a cost model to make an approximate prediction of the effort of software projects as a function of program size. This model is applicable at early project phases because of the little information it requires and it follows a similar relation to Eq. (1).
- Intermediate COCOMO computes the effort from a similar relation to (1) multiplied by an effort adjustment factor (EAF). In order to obtain the EAF it is necessary to evaluate 15 cost drivers. This model can be applied when we have not only the number of code lines but also a better understanding about the project.
- Advanced COCOMO is a model that contains all the characteristics of the previous models but makes an evaluation of each of the cost drivers over each phase of the project (analysis/design, code, and test). Therefore, it is applicable when enough specific information about each of the basic phases of the software project is known.

Boehm and his colleagues have refined and updated COCOMO to COCOMO II (Boehm et al., 2000), that accounts for recent changes in software engineering technology. Whereas COCOMO is reasonably well matched to custom, build-to-specification software projects, COCOMO II is useful for a much wider collection of techniques and technologies. COCOMO II provides up-to-date support for business software, object-oriented software, software created via spiral or evolutionary development models, and software developed using commercial-off-the-shelf application composition utilities.

2.2. Dynamic cost models

SDPs can be considered as complex socio-technological dynamic systems whose patterns of behaviour will mainly come determined by their internal structure as well as the relations established inside the working team. This fact allows the development of multi-attribute dynamic models to describe the feedback structure of the system being modelled as well as the mental process followed by project managers in decision making. Decision making has been traditionally based on the manager's experience. The simulation of a dynamic model offers the possibility of exploring the impact of a change of technology and/or different management policies over the project and the organisation before beginning the development. It also makes possible the accomplishment of post-mortem analyses. Applying dynamic models to simulate the development process leads one to consider two principal questions: What do we have to do to fulfil the objectives imposed on the project? And, what should we have done to fulfil the objectives or to improve the final results?

The dynamic models for SDPs include a set of parameters and functions to help to investigate different behaviours. These behaviours are controlled by the management policies applied to the project, which can be related to the project environment (initial estimations, complexity of the software, etc.) and to the organisation and its maturity level. Table 1 shows the different groups of parameters classified according to their role.

The parameters and functions related to the initial estimations (such as the number of tasks, time, cost, manpower, etc.) and project complexity (such as potential productivity, effort allocated to training activities, etc.) are included within the group called "project environment". The parameters and functions related to the different management policies which can be evaluated and those related to the maturity level of the organisation are placed in the "organisation environment" group. Within the subgroup "management policies", the management policies related to the effort allocation and daily manpower are included. Finally, the parameters and functions related to average hiring, to rotation or dismissals, to the average assimilation delays or to the all kinds of restrictions over the completion time are placed inside the "maturity level" subgroup.

All the parameters and functions of a model must be initialised before the simulation begins. This is one of the principal drawbacks of present dynamic models. The other drawback is the difficulty understanding and validating complex and large models. In many circumstances the project manager has a high level of uncertainty with regard to the proper values for these parameters. This is especially true at the early stages of a project.

2.3. Comparing static and dynamic cost models

Comparing static and dynamic cost models has made us consider that current dynamic models would be placed at the intermediate level proposed by Boehm (1981). That is, present dynamic models need to know an important number of project and organisation attributes (modelled as parameters and functions in the dynamic model) before being simulated.

Table 2 illustrates the equivalence we found between static and dynamic cost models. The corresponding box to dynamic models which are applicable at the early stages, when we have little information about the project but we need to have a rough idea of the evolution of the fundamental variables, could be occupied by the RDM. This is so because the RDM constitutes a significant reduction of Abdel-Hamid and Madnick's one, so that the amount of information required to simulate the RDM is approximately half the amount needed to

Table 1
Classification of the parameters and functions of dynamic models for software project dynamics

Related to		
Project environment	Organisation environment	
	Management policies	Maturity level
Initial estimations	Effort allocation	Average delays
Project complexity	Personnel management	Nominal values
	Completion time	Others

Table 2
Comparison between COCOMO models and existent dynamic models

Static cost models (COCOMO)	Dynamic cost models
Basic COCOMO	RDM
Intermediate COCOMO	Abdel-Hamid and Madnick (1991) Draper Laboratory (Clough et al., 1992) SEPS (Lin et al., 1997), etc.
Advanced COCOMO	Madachy (1996)

simulate the Abdel-Hamid and Madnick's model (hereinafter the Extended Model).

3. Justification to develop a RDM

When considering how to apply Abdel-Hamid and Madnick's model to a local company, we realised that it was very difficult or almost impossible to determine the initial values for all the parameters and functions of this model. The absence of historical databases and the large number of parameters and functions (some of them hard to understand by the project manager) made the application unfeasible.

Although this was the principal factor that lead us to develop the RDM, there were other factors:

- The existing hierarchy of empirical estimation models such as COCOMO (static and single attribute) suggested us the possibility of creating a similar hierarchy of dynamic estimation models (multi-attribute) depending on the information one has about the project.
- The existence of a generic common causal template for R&D projects and software projects (Ramos and Ruiz, 1997). This common template is described in the next section.
- The work of Eberlein (1989) about simplifying dynamic models. The simplification proposed by Eberlein consists of obtaining a reduced model from an extended one by eliminating the feedback that is not considered essential for the behaviour to be analysed.

3.1. Generic common causal template

The first dynamic models for SDPs were based on other dynamic models developed to simulate R&D projects (Roberts, 1978). Both types of systems have enough similarities to proceed that way. A previous comparative analysis elaborated with dynamic models for both SDPs and R&D ones, has led us to obtain a generic common causal template for these two kinds of projects (Ramos and Ruiz, 1997). Fig. 1 illustrates this diagram. It is composed of four principal feedback loops: two positive and two negative. A brief description of each loop follows:

- *First feedback loop*. From an original measure of software size, initial estimations for cost and development time can be obtained by applying a static model such as COCOMO. With these estimations the required manpower is acquired through hiring activities. Here it is important to notice that the dynamic models analysed model and simulate a single project, so that it is not possible to consider the reallocation of existing staff from other projects. As the project runs, one gets information about the real progress. Comparing the values obtained with those originally estimated may lead to change some of the estimations, such as time or cost, and possibly to decide to hire manpower.

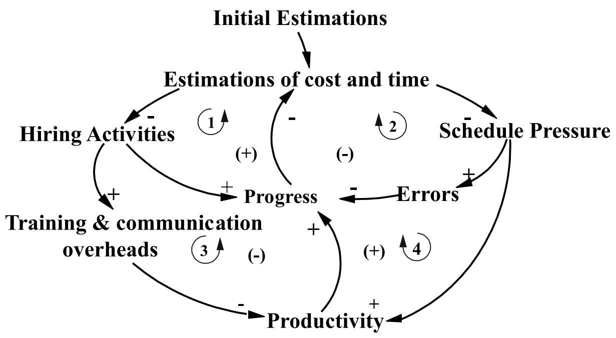

Fig. 1. Generic common causal template.

- *Second feedback loop*. This loop illustrates the effects caused by schedule pressure on the quality of the products of SDPs. If the perceived completion time is greater than the planned one, the project has schedule pressure. To solve this, the project manager can decide to hire more personnel or to do overwork, since the possibility of reducing the size of the completed system to meet the deadline is not considered in the models analysed. However, a sustained situation of overwork may result in a high exhaustion level of personnel, which increase the number of errors in the project. This amount of new committed errors requires a big effort in error detection and rework activities, which restricts the progress.
- *Third feedback loop*. If the project manager decides to hire new personnel he/she must take into account the negative effects of this action. Apparently, the increment of the human resource level contributes to the growth of productivity. However it is important to remember that the relation between the level of personnel and the productivity obtained is not linear: it is influenced by the increasing of communication overhead. This is the well-known Brook's Law which states that "adding manpower to a late project makes it later" (Brooks, 1995). It is also important to notice that new personnel productivity is significantly less than that of expert personnel (that is, the personnel who are used to the current project and technology, and to the procedures of the organisation). Hence, it is common to dedicate some effort of expert personnel to train the people just hired. These training activities contribute to decrease the net productivity of the working team.
- *Fourth feedback loop*. This is to illustrate the known effect of creative pressure. When personnel know that the project is behind schedule, they tend to increase their efficiency. This is normally reflected in a reduc-

tion of dead times and, sometimes, the beginning of overwork.

4. Reduction process of the extended model

4.1. Reduction methodology

Our approach to model simplification is based on behaviour, with simplified models generating only selected behaviours of the full model. The reduction process followed is based on the work of Eberlein (1989) about simplification and understanding of models. It is possible to develop a smaller model, based on selected dynamics generated by the original model (Abdel-Hamid and Madnick's model, in this case) and containing only a set of the feedback loops in that one. The smaller model has the feedback important in generating the selected behaviour. Model simplification as used here is similar, in spirit, to the eigenvalue-based loop analysis of Forrester (1982).

The heart of simplification is the selection of levels to include in the simplified model. All feedback paths pass through levels, so that retaining only selected levels results in retaining only selected feedback.

One of the criteria we have followed when making the choice of the levels to be retained has been keeping "the five numbers" over which project management is based (Putnam and Myers, 1996):

- a measure of the quantity produced,
- a measure of the remaining time to complete the project,
- a measure of production cost,
- an indication of product quality, and
- a measure of the average productivity.

In the RDM these measures are, respectively, the accomplished tasks and the size of the project in tasks, the size of the project in days, the size of the project in effort (technician-day) as the cost of the project can be obtained by multiplying the required effort and the average daily cost of the personnel, the number of committed errors, and the average working team productivity (task/technician-day).

To obtain the selected levels from the Extended Model, an aggregation of some of the original levels from the Extended Model has been made. That is,

- No difference between expert and non-expert personnel. Both types of personnel have been included as a single level called Personnel, which contains the total number of people working in the project.
- No difference between active and passive errors in the same sense of the Extended Model. Undetected errors that may produce more errors in the system are called active errors (for instance, a design error is a typical example of an active error as it produces new errors in the following phase). Errors that not produce other errors and are detected and corrected in the testing phase are called passive errors. In the RDM, both types of errors have been included as a single level called Errors.
- We consider that at the early stages of the project it is very important to know, at each moment, the work done (tasks accomplished), without distinguishing between the tasks developed and the tasks tested, and we have therefore combined development and testing into one activity.

The last assumption has been the one that has had the most influence in the reduction of the Extended Model.

Once the levels have been selected, the next stage of simplification is the creation of a simplified model. This is done by cutting feedback links in the Extended Model. The RDM retains the variables and nonlinearities of the original model but only the feedback judged to be essential to the behaviour of interest. Other variables that are required but not included in feedback are turned into exogenous variables.

The levels of the RDM are shown in bold type in Fig. 2, and coincide with the former values plus another essential variable in any SDP: the human resource level.

4.2. Reduction obtained

Applying the previous criteria of reduction, the size of the RDM is approximately the half of the Extended Model size. The RDM does not contain some of the parameters and functions hard to estimate by our local project managers, such as the exhaustion level of the personnel or the time required to recover.

Table 3 shows the corresponding percentages of reduction for variables, parameters, functions, and equations. As it can be seen, the reduction obtained for the parameters and functions is higher than 50%.

The parameters and functions related to the production and control subsystems have suffered the greatest reduction compared with the Extended Model.

5. Description of the RDM

The RDM supports the same internal structure that Abdel-Hamid and Madnick's dynamic model and has been implemented in Vensim which is a flexible and easy-to-use simulation environment.

Fig. 2 shows the simplified causal diagram of the RDM. [1] It contains the main feedback and the main variables of the dynamic model for SDPs. A description

[1] Further information about the model can be found at http://www.lsi.us.es/~informes/mdr.pdf.

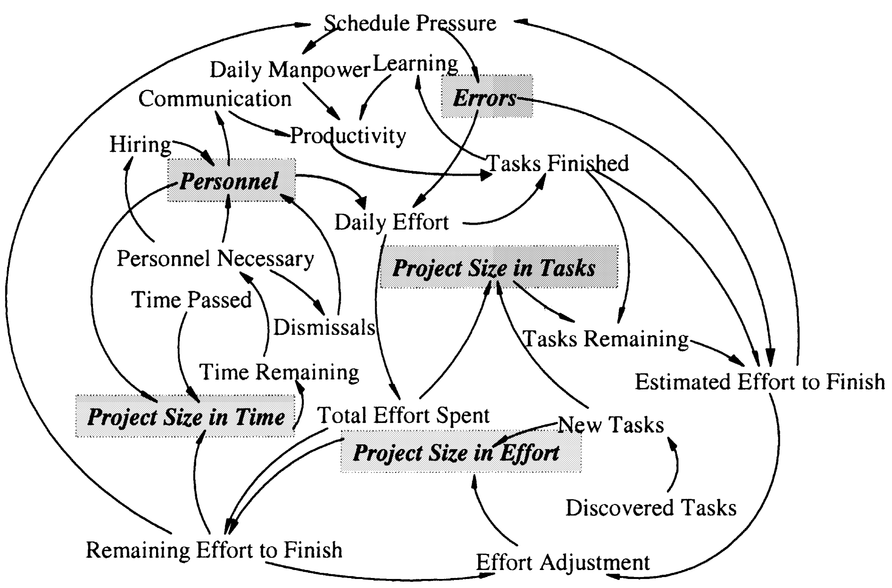

Fig. 2. Simplified causal diagram of the RDM.

Table 3
Obtained reductions of the RDM compared to the Extended Model

	RDM	Extended Model	Percentage of reduction
Variables	67	138	48.5
Parameters	19	37	51.3
Functions	16	27	59.3
Equations	127	237	53.6
Total	95	189	50.3

of this causal diagram and of the most common ways of behaviour of the variables follows.

Initial estimates are obtained based upon experience and/or traditional estimation models (such as COCOMO (Boehm, 1981), Walston-Felix (Walston and Felix, 1977) etc.). These initial estimations are for the completion time, the number of tasks to be accomplished and the required effort and manpower. These values are used to set the initial values for the levels of the dynamic model.

As the project progresses, information about accomplished and remaining tasks becomes available. This information is compared with the initial estimates in order to decide if it is or is not necessary to update them, especially those related to effort, time, and manpower. This is because the real size of the project (measured in number of tasks) is not known at these early stages of the development.

Initial estimates are revised when schedule pressure appears. The schedule pressure can be defined as the difference between the present estimated effort to complete the project on time and the remaining effort which was initially estimated (see Eq. (2)). The former depends on actual progress and the newly discovered tasks which were not initially estimated.

$$\text{Schedule pressure} = \frac{\text{Estimated effort} - \text{Remaining effort}}{\text{Remaining effort}} \tag{2}$$

So, if schedule pressure is positive it will indicate that the project is delayed and if it is negative, it will indicate that the project is advanced according to the initial estimates. A negative or positive pressure may affect the human resource level which can vary by hiring, dismissals or turnover activities, as Eq. (3) shows.

$$\frac{\Delta\ \text{Human resource level}}{\Delta t} = \text{Hiring} - \text{Dismissals} - \text{Turnover} \tag{3}$$

The value of the human resource level directly affects other important variables as the available effort to accomplish tasks, staff productivity, and thus the project progress itself.

The relationship between the human resource level and the obtained productivity is not linear. When the human resource level grows, losses of productivity appear. These losses are due to the communication lines established inside the working team and to the training activities performed by the experts. Productivity is also affected, in a positive way, by growing manpower which is produced by eliminating dead times, a greater level of experience and by seeing the progress that the project has made (see Eq. (4)).

$$\begin{aligned}\text{Productivity} = {} & \text{A function of (human resource}^{+}, \\ & \text{daily manpower}^{+}, \text{experience}^{+}, \\ & \text{communication and training overheads}^{-}) \end{aligned} \tag{4}$$

On the other hand, when schedule pressure is positive the model adjusts the detected delay by modifying not only the human resource level. Changes may also affect the daily manpower and the reduction of quality levels of the final product, in the same way as the general software development practices control positive schedule pressures (Abdel-Hamid and Madnick, 1991). These changes directly affect the error generation rate caused by the new people just hired, by the exhaustion level of personnel and by the reduction of the effort applied to revision activities. However, as the error generation rate grows, the required effort to detection and rework activities will be greater. This directly affects the effort assigned to development activities, which is reduced.

As in real projects, the typical modes of behaviour of dynamic models for SDPs depend on the value of pressure at each moment, just as much in time as in effort. These types of behaviour are also seen in the RDM and can be summarised as follows:

When the schedule pressure is positive there are three possible actions:

1. To increase the human resource level through hiring activities. The amount of people hired will depend on the manpower required and the time remaining to complete the project. As it has been said before, the immediate effects over the project will be positive or negative. The positive effect is that productivity will grow because of increasing manpower in the project. The negative effect is that productivity may decrease because of communication and training overheads.
2. To increase the daily manpower by raising the number of working hours in the project or reducing dead time. However, increasing the number of working hours has an important drawback, which must be noticed: the growth of the exhaustion level of people, which affects the error generation rate.
3. To decrease the effort assigned to revision activities and allocate it to development ones. This decision contributes to an increment in productivity, but the quality of the final product decreases because of the greater probability of errors.

When the schedule pressure is negative it is possible to decrease the number of people assigned to the project through dismissals. Furthermore, the RDM models the motivational role of negative schedule pressure which is to increase the project members' slack time. By slack time we mean the fraction of project time lost on non-project activities. The loss in productivity due to motivational factors may lead, if not controlled, to unexpected delays in the project.

6. RDM testing

A simulation is a simplification of the real world, and thus inherently an approximation. As indicated in (Robertson, 1997) it is not possible that a model is absolutely correct. Therefore model testing is concerned with creating enough confidence in a model for its results to be accepted. RDM testing focused on three activities: verification, validation, and evaluation.

6.1. Verification

Verification of a model consists of activities that focus on its internal workings. The RDM verification activities fall into two categories:

- Verifying the structure of the model. The tests performed were focused on exploring the dimensional consistency of equations in the model, the ability of these equations to handle extreme values and whether the structure of the model is adequate to address the problems to be studied.
- Verifying the behaviour of the model. The tests made to verify the model behaviour were of two kinds: parameter (in)sensitivity and structural (in)sensitivity. The final results of these tests were positive.

6.2. Validation

Validation of the RDM consisted of activities that compared it to a real system. In order to validate the RDM, the example case from Abdel-Hamid and Madnick (1991) was firstly used. The results verified that the RDM was able to reproduce the same evolution patterns as the Extended Model. Then, we used the RDM to simulate the behaviour of a real SDP jointly undertaken by two local software development companies. The assistance of the project manager was invaluable in this second test.

The validation activities fall into two categories: validating the structure of the model and validating the behaviour of the model. Validating the structure of the RDM consisted of comparing the implementation of the model with a real system. Validating the behaviour of the model consisted of executing the RDM and comparing its output with what would be expected from the real system or what has been observed in the real system. When comparing the model's output with what was expected by the project manager, the general behaviour trends of the model were tested. A summary of the results obtained follows.

6.2.1. Behaviour of the RDM compared with the Extended Model: Example case

Figs. 3 and 4 illustrate the evolution of different variables of the project in both models. The former shows that the number of errors at the end of the project is greater in the RDM than in the Extended Model. Nevertheless, both curves have similar evolution patterns, gradually growing at the initial and intermediate phases to quickly grow at the final ones. In Fig. 4 we observe that the human

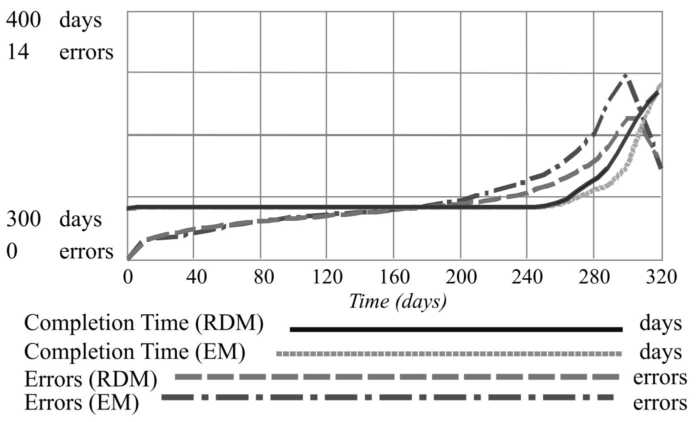

Fig. 3. Completion time and errors evolution in the RDM and in the Extended Model.

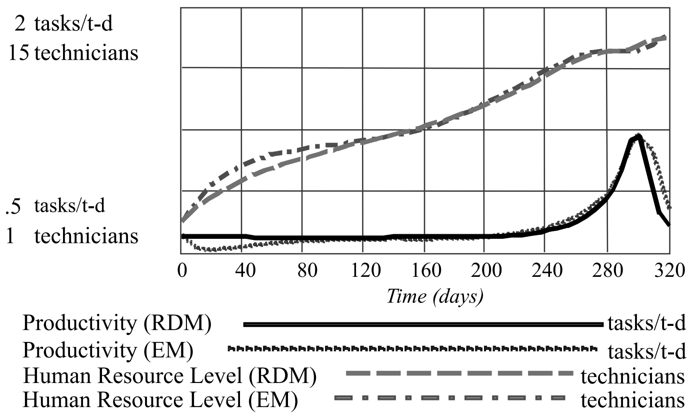

Fig. 4. Productivity and human resource evolution in the RDM and in the Extended Model.

resource and productivity evolution in the RDM is similar to that of the Extended Model.

From these results, it is possible to say that the RDM has lost precision compared to the Extended Model. This is due to the process of simplification which has not considered some development aspects that the Extended Model includes and which have been enumerated in Section 4. Nevertheless, the RDM has avoided the difficulties of estimating some parameters and functions at the initial project phases. At these phases, when the project manager has little information about the SDP, the RDM provides an important support, showing the behaviour of the main project variables.

6.2.2. *Post-mortem analysis of a SDP*

The project selected for studying in this work is a personnel management system (hereinafter PMS) jointly carried out by two local software development companies.

The data for initialising the parameters and levels of the RDM have been collected from the tracking document of the original project and from the experience of the project manager. These data are for the design, code, and testing phases. Therefore, the analysis phase and other final activities are not considered. The programming environment used was COBOL and a relational database.

The initial estimated effort to carry out the project was 208 technicians-day (t-d) whereas the actual final effort was 404 t-d. The estimated completion time was 101 days; the real value was 141 days. Therefore, the underestimation in effort and time for this project were respectively, 48% and 28%. The average number of technicians participating in the project was 6 and the number of lines of code (LOC) was 67,800. The project manager defined a task as 270 LOC. On this basis, the project was composed of, approximately, 251 tasks.

85% of the real effort was allocated to the development activities (design and code) and the remaining 15% to the test phase. On the basis of the project manager's experience, 10% of the development effort was spent in revision activities, so that the total effort spent in development was 309 t-d. From this, the development productivity for this project was 219.4 LOC/t-d.

Development productivity

$$= \text{size in LOC/development effort} = 67,800/309$$
$$= 219.4\ \text{LOC/t-d}$$

Other data for the real project are shown in Table 4. These values are used to set the initial values of the parameters and levels of the RDM for simulation runs. For each one, the name of the parameter in the model, a short description of its meaning, and its value at the beginning of the project are shown.

The evolution of necessary effort, of completion time, and of remaining tasks to finish the PMS obtained with the RDM are shown in Fig. 5. Comparing these results with the real values, it can be seen that completion time is 151 days in the simulation (instead of the real 141 days) and the effort is 410 t-d (instead of 404 t-d). Therefore, the deviations have been, respectively, of 7% and 1.5%.

Also, we can see from Fig. 5 that the first important adjustments were made about half-way through the project when it was detected that the number of remaining tasks was greater than might have been expected. These adjustments coincide with adjustments made in the actual PMS project because more of half of the estimated time had been spent but more than the half of the tasks still remained to do. If the slope of the curve representing the remaining tasks had not changed, it would have seriously damaged the time and the final effort. Two additional considerations can be observed in Fig. 5: as in the majority of the projects of this organisation, the effort forecasts are modified before the time ones; and the modifications of effort and time are done simultaneously.

If the results of the simulation are compared with the real behaviour of the project (this was possible thanks to the help of the project manager), we can see that the estimated time and effort obtained by means of the simulation agree with the real time and effort. This is especially true in two aspects:

- The early revisions were made when half of the time had been spent.
- The revisions carried out for adjusting the detected deviations, simultaneously affected both the time and the effort.

Hence, the RDM not only follows the behavioural patterns of Abdel-Hamid and Madnick's model. It is also useful for simulating projects of our local software development organisations.

6.3. Evaluation

The evaluation activity focused on the usefulness of the model. The usefulness of the RDM was determined through observation of its use and the perceived benefits

Table 4
Representative parameters of the project and organisation environment

Name	Description (Units)	Value
ADMPPS	Average daily manpower per staff (day/day)	0.4
DESRWD	Average delay for QA and rework (days)	5
HIASDY	Average hiring and assimilation delay (days)	20
INUDST	Initial understaffing factor (dimensionless)	1
MNHPXS	Most hires per full time equivalent staff (technicians/technicians)	1.5
MXSCDX	Maximum schedule completion date extension (dimensionless)	50
TRNSDY	Time delay to transfer people out (days)	1
DLINCT	Average delay to incorporate discovered tasks (days)	5
AVEMPT	Average employment time (days)	1000
TRPNHR	Number of trainers per employee (dimensionless)	0.15
UNDESM	Man-days underestimation fraction (dimensionless)	48
UNDEST	Tasks underestimation fraction (dimensionless)	15

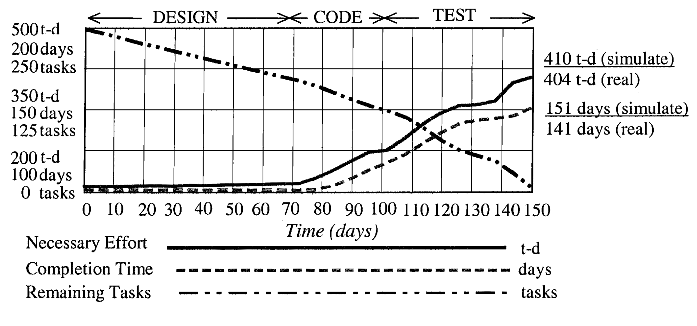

Fig. 5. Necessary effort, completion time, and remaining tasks evolution for PMS.

of interaction with it, and was evaluated with respect to the model structure and its behaviour. Evaluating the structure of the model focused on the level of detail in the model. Evaluating the model structure consisted of one test to determine the appropriateness of the model's characteristics for its intended audience. The purpose of the evaluation of the model's characteristics, with respect to its intended audience, was to examine whether the size, complexity, and detail of the model were appropriate.

As stated in Section 1, researchers and practitioners are the intended audience of the model. The RDM is considered to be appropriate for this audience as it allows both researchers and practitioners to experiment with different strategies when they do not have enough information to calibrate a more detailed model. Hence, the RDM can be used as a tool for hands-on training to illustrate the advantages and disadvantages of the strategies experimented.

Evaluating the behaviour of the model focused on the insight gained through use of the model. No unexpected counterintuitive behaviour was exhibited by the model. The model does, however, allow known paradoxical behaviour to be examined (for example Brook's Law (Brooks, 1995)). Our industrial collaborator was able to verify that the model correctly simulated a number of scenarios that were not in mind when the model was built.

7. Conclusions

Software process simulation models represent a meaningful advance compared to static models because they help us to understand the *evolution* of projects.

Dynamic models use a number of parameters and functions to characterise the project and the organisation environment. Having the initial values for all of these parameters and functions is a requirement to run simulations of it. Unfortunately, the absence of historical project databases and the uncertainty of software projects make it difficult to set these initial values. So a simplified model, with a smaller number of parameters and functions, but still producing useful results, is necessary. To obtain this reduced model, a theory of the simplification of dynamic models (Eberlein, 1989) has been applied. It should be noted that the simplified model is itself a complete model, which can be used for simulation and for policy experimentation. The model presented has been validated against the results obtained for Abdel-Hamid and Madnick's Example Case, and has been used to make a post-mortem analysis of a real project in an industrial environment.

The RDM is an useful tool which can be used both to make estimations and to experiment with different management policies, particularly in the following cases:

- When there is little information about the SDP.
- When the organisation does not have a historical project database or this has not been completed enough to define an important number of project and development process attributes, especially at the early project phases.
- When a fully, simple, and easy to learn model is necessary, for example, in training activities for software project managers.

Acknowledgements

The authors wish to thank Dr. Paul Wernick and the anonymous reviewers for their valuable comments which greatly helped in improving the quality of the paper. We are also grateful to our industrial collaborator who has actively participated in the interpretation and analysis of the data and the simulation results. Thanks also to the Comisión Interministerial de Ciencia y Tecnología, Spain, (under TIC1999-0351) for supporting this research effort.

References

Abdel-Hamid, T., Madnick, S., 1991. Software Project Dynamics: An Integrated Approach. Prentice-Hall, Englewood Cliffs, NJ.

Boehm, B., 1981. Software Engineering Economics. Prentice-Hall, Englewood Cliffs, NJ.

Boehm, B., Abts, C., Brown, A.W., Chulani, S., Clark, B., Horowitz, E., Madachy, R., Reifer, D., Steece, B., 2000. Software Cost Estimation with COCOMO II. Prentice-Hall, Englewood Cliffs, NJ.

Brooks, F.P., 1995. The Mythical Man-Month, 20th Aniv. edition 1995, Addison-Wesley, Reading, MA (first edition 1975).

Clough, A., Nguyen, N., Ahmed, S., Smith, B.J., Vidale, R.F., 1992. The Software Process Model Technical Report, Draper Laboratory, Cambridge, Massachusetts.

Eberlein, R.L., 1989. Simplification and understanding of models. Syst. Dyn. Rev. 1 (5), 51–68.

Forrester, N.B., 1982. A dynamic synthesis of basic macroeconomic theory: implications for stabilization policy analysis. Ph.D. dissertation, Sloan School of Management, MIT, Cambridge, MA.

Kellner, M.I., Madachy, R.J., Raffo, D.M., 1999. Software process simulation modeling: why? what? how? J. Syst. Software 46, 91–105.

Madachy, R., 1996. System dynamics modeling of an inspection-based process. In: Proceedings of the 18th International Conference on Software Engineering, pp. 376–386.

Roberts, E.B., 1978. A simple model for R&D project dynamics. In: Roberts, E.B. (Ed.), Managerial Applications of System Dynamics, Productivity Press, pp. 293–314.

Lin, C.Y., Abdel-Hamid, T., Sherif, J.S., 1997. Software-engineering process simulation model (SEPS). J. Syst. Software 38, 263–277.

Ramos, I., Ruiz, M., 1997. Análisis de las estructuras dinámicas comunes a los proyectos de desarrollo de software y los proyectos de investigación y desarrollo. In: Proceedings of the III Jornadas de Informática, pp. 127–136 (in Spanish).

Robertson, S., 1997. Simulation model verification and validation: increase the user's confidence. In: Proceedings of the 1997 Winter Simulation Conference, pp. 53–59.

Rodrigues, A.G., Williams, T.M., 1997. System dynamics in software project management: towards the development of a formal integrated approach. Eur. J. Inf. Syst. 6, 51–56.

Putnam, L.H., Myers, W., 1996. Executive Briefing. Controlling Software Development. IEEE Computer Society Press, Silver Spring, MD.

Walston, C.E., Felix, C.P., 1977. A method for programming measurement and estimation. IBM Syst. J. 16 (1), 54–73.