SIMULATION MODELING OF SOFTWARE DEVELOPMENT PROCESSES

G. F. Calavaro[1,2], V. R. Basili[2], G. Iazeolla[1]

[1]*University of Rome at Tor Vergata*
and
[2]*University of Maryland at College Park*

iazeolla@info.utovrm.it

ABSTRACT

A simulation modeling approach is proposed for the prediction of software process productivity indices, such as cost and time-to-market, and the sensitivity analysis of such indices to changes in the organization parameters and user requirements.

The approach uses a timed Petri Net and Object Oriented top-down model specification.

Results demonstrate the model representativeness, and its usefulness in verifying process conformance to expectations, and in performing continuous process improvement and optimization.

INTRODUCTION

Reducing the cost of large scale software projects and shortening cycle time, or time to market, is a major goal of most software development organizations.

To pursue such a goal, organizations can set productivity goals for each project, and put in place statistical productivity controls to enable developers and management to take corrective actions when there are deviations from the goal, and to distinguish a random deviation from meaningful deviations.

Simulation is one of the methods for performing such control. It can be used at various points in the software life cycle to perform risk analysis, in terms of time to product, and cost, to verify conformance to expectations, and to perform continuous process improvement and optimization.

This requires that organizations use metrics and models to evaluate and predict effort and time (Basili 1979), (Fenton 1991).

The intrinsic complexity of the software production process makes it difficult to conduct predictions using strict analytical models. It is often necessary to turn to simulation models to obtain adequate information on the dynamic behavior, the functionality and the performance of the process.

Software development organizations use several models that address the issue of estimating the effort and the time to product delivery.

Existing analytical models, such as the COCOMO model (Bohem 1981), the Mark II Function Point model (C.R. Symons, 1988) etc., generally only provide predictions of total development effort, without any information about how these are distributed throughout the process. Existing simulation models also share this deficiency. In both cases, no information is given on the instantaneous dynamic behavior of effort versus time, and on the effect of changing user requirements during development time.

Field experiences show that requirements and available resources change during development, so cost and delivery time change as well. Therefore, it is very important to have models which predict the dynamic behavior of cost and time while requirements change.

To reach this goal, this paper proposes a simulation approach to evaluate the effort spent over time by each activity of the process, the estimated delivery time, and the size of the final product.

As illustrated in Figure 1, the approach consists of a process simulation model that provides the behavior <u>over time</u> of quantities such as:

- *prod_size* : the measure of the delivered code size (in lines of code);
- *work_e* : the development work effort (in

Work partially supported by the National MURST project "Performability V&V of Products and Processes in Software Engineering",the CNR project "CE and EF in the Software process", the CERTIA Project "Software Performance V&V", and by NASA grant NSG-5123.

person-weeks);
- *delivery_t*: the time to product (in weeks);
as a function of the user requirement size (in function-points), and the organization parameters.

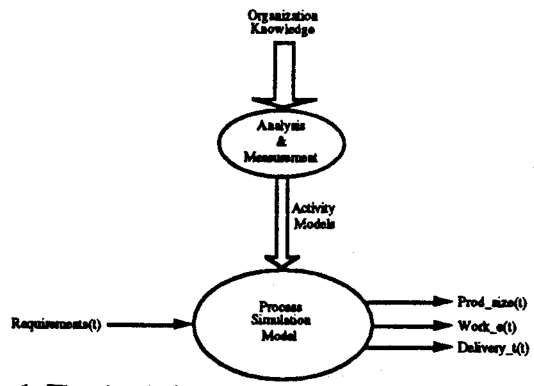

Figure 1: The simulation modeling approach

The model is parametrized on the basis of measurements and analyses of data coming from knowledge of the simulated organization.

The second Section of the paper describes the basic assumptions of this work. The third presents the considered model. The fourth presents the models of the process activities. The fifth Section synthetically illustrates the simulation model, and the model for requirements generation. The sixth presents the results of the simulation experiments.

MAIN ASSUMPTIONS

This work assumes the development process is split into a sequence of *activities* according to the Waterfall model.

An *artifact* is defined as any kind of document, paper, or file produced or used by an activity of the development process.

For each activity it is assumed there is an *input artifact* and an *output artifact*. Input artifact is the artifact the activity uses as an information source to produce the output artifact.

Example artifacts are: the requirement specification document, the requirement analysis document, the architectural design document, the detailed design document, the code after implementation, the code after system test, the code after acceptance test.

It is also assumed that:

A) There exist metrics to express the size of any artifact.

B) There exist models for the estimation of the output artifact size as a function of the input artifact size. For example, the detailed design document size can be estimated on the basis of the architectural design document size.

C) There exist models to express the amount of resources each activity requires to produce the output artifact on the basis of the input artifact size.

THE CONSIDERED MODEL

The considered process model is illustrated in Figure 2. It includes a model of standard software development process activities such as: Requirement Analysis (ReqAna), Preliminary or architectural Design (PreDes), Detailed Design (DetDes), Implementation (Impl), System Test (SysTest), and Acceptance Test (AccTest).

The input variable of each activity is the estimated size of the output artifact produced by the previous activity. The output variable of each activity is the estimated size of its output artifact and an estimated measure of the activity effort. The latter estimation is send to a *data collector* for recording and evaluating purpose (see later).

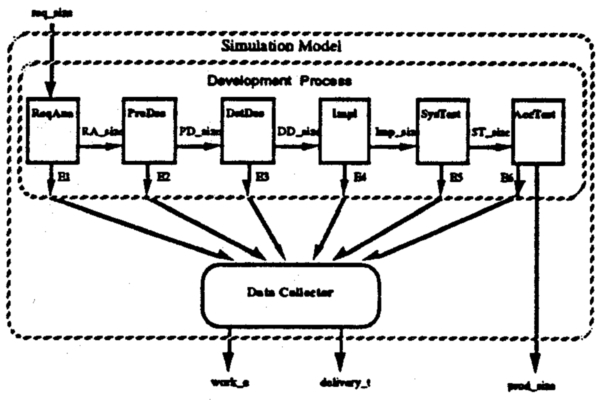

Figure 2. The considered software process model

Input and output variables dynamically change over time. In other words, each activity takes new input values at each time instant and yields the corresponding output values. There obviously exists a time delay in producing outputs, which is dealt with by the activity model.

As shown in Figure 2, the *ReqAna* block receives as input the size of the Requirement Document, *req_size*, and produces as output *RA_size* (the estimated size of the Specification Document produced by the Requirement Analysis activity), besides the measure of the

ReqAna effort (*E1*).

In a similar way, each of the following blocks receive measures of input artifact size, and yield measures of estimated output artifact size, for the following block, and of required effort for the *Data Collector* block.

The final block, *AccTest* block, receives as input the size of the code after System Test, *ST_size*, and produces as output the size of the final product, *prod_size*, with the measure of the AccTest effort (*E6*).

The *Data Collector* block obtains the *E1* through *E6* effort values and yields the total time integral of such values, *work_e,* in addition to the delivery time of the final product, *delivery_t*.

THE ACTIVITY BLOCK MODELS

This Section illustrates the details of the internal behavior of each standard activity block. Such behavior is expressed in terms of an input/output function that transforms the input artifact size into the output artifact size and into a measure of the required effort.

For the sake of conciseness the Rayleigh function is assumed as the basic activity model. Such a function can obviously be replaced by any empirically derived function, in case this is believed to better represent the organization's behavior.

The use of the Rayleigh function is supported by the large amount of literature assessing its usefulness as a good model of the software development process, (Putnam 1978), (Fenton 1991), by marketed prediction tool products as SLIM, and by the fact that empirical functions derived by large organizations, such as NASA, are very similar to the Rayleigh function, as shown in Figure 3, derived from (SEL-81-305).

As seen from the Figure, the behavior of effort versus time for each individual activity follows very closely the Rayleigh function, mathematically expressed by:

$$E(t) = \frac{W(t)}{T(t)^2}\, t\, e^{-t^2/2T(t)^2} \qquad (1)$$

where E(t) is the instantaneous effort required at time t (the equivalent of full time staffing level), W(t) is the estimated total effort of the activity (the integral of E(t) over time) expressed in person-week at time *t*, and T(t) is the estimated delivery time (weeks) of the activity for the artifact at time *t*.

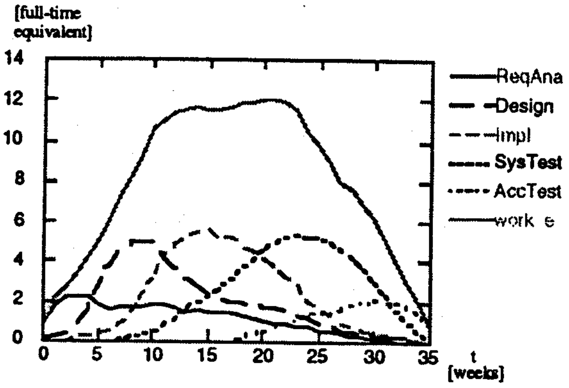

Figure 3. Example of individual activity effort and total life-cycle effort for the NASA software development process

Quantities W(t) and T(t) vary with time *t*, since it is assumed that the original input, req_size, changes over time. Their values are assumed to be given by conventional models, inspired by the COCOMO equations (SEL-81-305), as follows:

$$\begin{aligned} out\_size(t) &= a_1\, in\_size(t)^{b_1} + c_1 \\ W(t) &= a_2\, out\_size(t)^{b_2} + c_2 \qquad (2) \\ T(t) &= a_3\, W(t)^{b_3} + c_3 \end{aligned}$$

where *in_size(t)* is the input artifact size for the generic activity, and *out_size(t)* is the output artifact size, at time *t*.

According to the Putnam assumption (Putnam 78), this work also assumes that, for each activity, the outcome *out_size(t)* takes place only when the instantaneous value of E(t) reaches its peak.

The effort spent after the peak (the down sloping part of various curves in Figure 3) is for rework and updates due to requirements changes. The overlaps between down and upward sloping of curves relating to contiguous activities, implicitly and synthetically represent iterations and interactions among teams. The corresponding effort values are thus implicitly considered in the model. This concept is further refined in next Section.

Parameters a_i, b_i, and c_i, for each activity, are assumed to be derived from empirical data from the modeled organization.

THE SIMULATION MODEL

In literature there are few examples of simulation models of software development processes. Examples of such models are the *Articulator* by Scacchi and Mi (Scacchi and Mi 1993) and *System Dynamics* by Abdel-Hamid and Madnik (Abdel-Hamid and Madnik 1991).

The Scacchi work deals with a knowledge-based computing environment for modeling, analyzing and simulating complex organizational processes. Its purpose is to simulate the organizational behavior in terms of agents, tasks, and resource allocation.

The Abdel-Hamid work deals with the simulation of software development organizations based on system dynamics techniques.

Neither the former, nor the latter deal with the simulation of the effects of the requirements changes on product costs and time to delivery, which is the subject of this paper.

For the sake of conciseness, the details of the simulation model used in the illustrated approach are given elsewhere (Calavaro et al. 1995).

The model replicates the process scheme illustrated in Figure 2.

An object oriented approach is used to specify the main objects of the simulator, and their connections, and a timed Petri Net, top-down hierarchical approach is used to specify the dynamics of the simulation model and the data flows among objects (Calavaro 1995).

The combination of such approaches in the model specification, makes the simulator easily implementable and adaptable to various process organizations.

Any implementation language can be used. However, languages which are specifically meant for dynamic system simulation, such as DYNAMO and SIMNON, are preferred. For this paper, the latter has been used, since it is particularly oriented to non-linear system analysis.

Model Input Generation

In most real systems, generation of user requirements is usually performed by the development organization in conjunction with the customers, and yields the requirements of the system to be developed.

In the simulation model the requirements size (*req_size*) is the model input. This dynamically changes over time. The model assumes that the generation activity is external to the process activities, and so its effort is not part of the effort calculations.

The *req_size* value is assumed to be expressed in number of Function Points (FP). The input is assumed to start at time 0 of the process dynamics.

For the experiment in this paper *req_size* is expressed by the following equation:

$$\text{req\_size}(t) = a_4\,(1-e^{-t/b_4}) + c_4 \qquad (3)$$

where c_4 is the initial requirement size, a_4 is the changed requirement size, and b_4 is a time constant. According to this expression, the requirements increase along time by a negative exponential rate, and reach their stable value $a_4 + c_4$ asymptotically in time. Other equations could be used as well.

The model assumes a_4=50 [FP], c_4=100 [FP], and b_4=15 [weeks].

In other words, it assumes a 50% increase in the requirements size during the life cycle.

SIMULATION RESULTS

Obtained results are expressed by curves that give the effort values over time for each process activity, the global process effort, and the time to product.

Figure 4 shows the simulated effort density for various process activities (ReqAna, PreDes, DetDes, Impl, SysTest, and AccTest) and for the total modeled process (*work_e*).

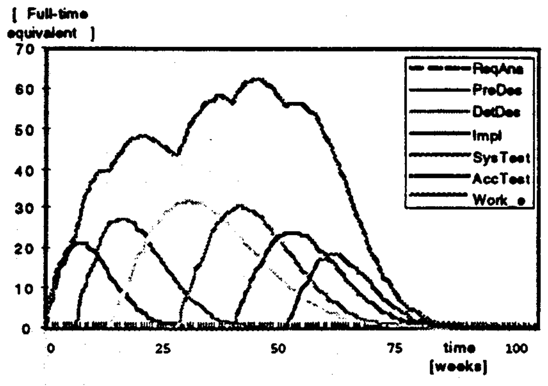

Figure 4. Simulated effort density for various process activities and for the total modeled process

The simulated total effort density for the process, *work_e*, is the sum of the individual activities' efforts. Its integral gives work_e = 4584 person_weeks for a prod_size = 103 KLOCs, in a delivery_t = 70 weeks.

The curves in Figure 4 are the results of our simulation. They are qualitatively similar to those in Figure 3, which are empirically derived.

This supports the validity of the proposed simulation model. The introduction of realistic values for the a*i*, b*i*, and c*i* parameters is the only requirement to obtain model validation on a quantitative basis.

The valleys in the top of Figure 4 *work_e* curve are a consequence of the assumption that each activity starts when the previous activity effort reaches its peak. Such valleys are not evident in the empirical NASA curve, seen in Figure 3. We believe that this difference is at least partly due to the fact that in any real environment the activity starts a little bit before the previous activity peak, since unofficial artifacts are delivered to the following activity before the official ones are ready.

This shows the representativeness of the proposed approach, for use in Waterfall-like organizations, for the prediction of the software production costs and delivery times, as well as for the analysis of the sensitivity of costs and times to changes in organization parameters, and to the variation in user requirements.

The top-down hierarchical model specification permits adaptation of the proposed simulator to non-waterfall various process models.

CONCLUSIONS and FUTURE RESEARCH

Simulation is one of the productivity control methods that enables software developers and managers to take corrective actions and perform risk analysis, in terms of time to product and cost, to verify conformance to expectations, and to perform continuous process improvement and optimization.

This paper has introduced a software process simulation modeling approach for the prediction of the software production costs and delivery times, and analysis of sensitivity to the changes in organization parameters, and user requirements.

The model is based on a top-down hierarchical model specification that can be used to adapt the proposed simulator to various process models.

This is part of future research, with the explicit modeling of the interactions between process activities and of the modeling of product iterations.

Future research also includes explicitly representing physical factors and agents that characterize various process activities, by the use of specialized software process languages.

REFERENCES

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

Basili V.R., 1989. "Software Development: A Paradigm for the Future" In *Proc.13th Int'l Computer Software and Applications Conf.*, (Orlando, Fl, Sept.), CS Press, 471-485.

Bohem B. W., 1981. *Software Engineering Economics*, Prentice-Hall, Englewood Cliffs, NJ.

Calavaro G.F., 1995. *Experience Factory and Concurrent Engineering for Software Process Optimization*, Ph.D. Dissertation, University of Rome "Tor Vergata", Rome, Italy.

Calavaro G.F., Basili V.R., Iazeolla G., 1995. "Simulation Modeling of Software Development Processes", *Technical Report University of Rome "Tor Vergata"*, RI.95.06, Rome, Italy.

Fenton N.E. , 1991. *Software Metrics, A rigorous approach* Chapman & Hall, London, UK.

Putnam L.H., 1978. "A General Empirical Solution to the Macro Software Sizing and Estimating Problem", *IEEE Transaction on Software Engineering*, (July) 345-361.

Scacchi W., Mi P., 1993. "Modeling, Integrating, and Enacting Software Production Processes" In *Proc. 3rd Irvine Software Symposium* (Costa Mesa, CA, April)

Symons C.R. , 1988. "Function Point Analysis: Difficulties and Improvements", *IEEE Trans. on Software Engineering*, (14):1, (January) 2-11

SEL-81-305, 1992. "Recommended Approach to Software Development", Revision 3, *Software Engineering Laboratory series*, SEL-81-305, NASA-GSFC, Greenbelt, MD.