A Hybrid Model for Dynamic Simulation of Custom Software Projects in a Multiproject Environment

Javier Navascués[1], Isabel Ramos[2], and Miguel Toro[2]

[1] Universidad de Sevilla,
Departamento de Organización Industrial y Gestión de Empresas,
Av. Descubrimientos s/n, 41092 Sevilla, Spain
jnavascues@us.es
http://io.us.es
[2] Universidad de Sevilla,
Departamento de Lenguajes y Sistemas Informáticos,
Av. Reina Mercedes s/n, 41011 Sevilla, Spain
iramos@us.es, mtoro@us.es

Abstract. This paper describes **SimHiProS**, a hybrid simulation model of software production. The goal is to gain insight on the dynamics induced by resource sharing in multiproject management. In order to achieve it the hierarchy of decisions in a multiproject organization is modeled and some resource allocation methods based on algorithms from the OR/AI domain are used. Other critical issues such as the hybrid nature of software production and the effects of measurement and control are also incorporated in the model. Some first results are presented.

Keywords: Hybrid simulation, multiproject resource management, hierarchical decision making.

1 Introduction

Dynamic simulation of software projects is a well established field of research and application. Software project simulation models have been able to provide significant insight on many characteristics of the software production process. Nevertheless some real-world settings do not lend themselves easily to existing models. One particular question is the problem of resource allocation to projects in the case of software developed by multiproject organizations, i.e. software built by organizations working simultaneously in several projects for different customers with shared resources. Depending on the circumstances, this kind of software projects can become extremely risky as the intrinsic uncertainty of each particular project compounds with the mutual dependency between projects.

The research work presented here aims to develop modeling tools adequate to one of these cases: a medium-sized software engineering company whose business consists in developing custom software projects for State agencies with a

relatively stable set of human resources. The work adapts state-of-the-art models and methods both from the field of Software Engineering and from Project Management. In particular it builds on dynamic models aimed at the simulation of software production on one hand, and constrained resource project scheduling algorithms on the other.

The rest of this paper is organized as follows: Section 2 presents the critical issues this research tries to address. Section 3 comments on previous work carried by the academic community which has been revised and, when appropriate, used to build the model. Section 4 introduces the model, its main features and its operation. Section 5 presents some of the first results, draws conclusions and presents the limitations of the model in its current state. Section 6 announces future work.

2 Critical Issues in Custom Software Projects

The aim of this research is to gain understanding in resource allocation methods to software development in a multiproject context. Although it is focused on a concrete software company the class of problems studied are relevant for some branches within the software industry. Four are the issues that have been judged as being critical for the case under study:

1. Uncertainty owing to the combined effect of individual project risk and resource sharing
2. Coexistence of different levels of decision in a multiproject environment
3. The need to account both for continuous and discrete features in software production simulation
4. The trade-offs of implementing more or less tight measurement and control processes and SPI programmes in this context.

2.1 Resource Allocation and Uncertainty

Custom software developed for governmental agencies consists mostly in applications and functionalities which run on top of or interface with large existing legacy systems. This means that the developers must know and understand not only the project they are involved in but also the system or systems with which it will interact. Additionally in these cases the development model is usually incremental and evolutionary as opposed to waterfall models. Another particular feature of this kind of projects is what can be termed as *volatility of requirements* meaning that requirements are a source of uncertainty through the whole development cycle due to the complexity of the larger systems involved and of interactions which cannot easily be forecast in advance.

All these circumstances call for development teams composed by (at least a core of) experienced personnel who must remain committed to the project throughout the whole development cycle. To make things worse, these projects are prone to waiting periods owing to the number of stakeholders concerned and the complexity of decision making involved. Ideally the development team

should not remain idle during these periods but this could affect productivity by switching between tasks.

This are the kind of circumstances which determine the real difficulties of managing this kind of projects, both at the single project level when coping with the perturbations inherent to its development or induced by other projects, and at the company level when facing the problem of dynamically assigning and reassigning staff to meet delivery dates while minimizing the idle periods, keeping a check on multitasking and - of course - avoiding excesses of personnel.

2.2 Multiple Decision Levels in a Multiproject Environment

In [9] the multiproject management problem is viewed under two different optics: the first focuses on the decision scope traditionally classified as *strategical*, *tactical* or *operational*. The second framework views multiproject environment in terms of *variability* and *dependency*. An organization specializing in custom software development faces a significant degree of variability at the project level. If resources are shared across projects, then a great dependency appears at is has been described in the previous paragraph. A rational management strategy typically will try to *reduce dependency between projects at the operational level coping with it at the tactical one*.

Accepting a project is an strategical decision and is left out of the problem under consideration. So projects appear earmarked with their intermediate and final deadlines conceptualized in terms of time windows. The problem then becomes:

- At the *tactical* level; allocate resources to each project so that delivery dates are respected. This should be attained using the existing capacity in the most economic possible way within quality standards. It is then a *tactical capacity planning* problem minimizing costs and respecting both temporal constraints and previous allocations.
- At the *operational* level; schedule each project within the margins imposed by the tactical planning providing for the intrinsic variability affecting each one. It is then a *resource constrained project scheduling* problem with *general precedence relations*(GRP-RCPSP) [4,6] under uncertainty where the objective function is minimizing makespan.

2.3 Projects and Processes: The Hybrid Nature of Software Production

The life-cycle of a software project is populated by concrete artifacts built, verified and validated by concrete agents. These entities are discrete in number and different from each other even if they can be grouped in categories. On the other hand the concept of process embodies the idea of commonalities among entities within categories. This allows for the possibility of aggregation into continuous magnitudes of some quantifiable features of software development. To account for both views, the discrete and the continuous one, using a hybrid approach to modeling is needed.

When the issue is the production of *custom* software in a multiproject environment the need for hybrid modeling becomes even greater. Custom software projects are essentially distinct from each other so many product metrics cannot accumulate nor average; but custom software projects developed in a multiproject environment draw the workforce used from a common manpower stock and on the other hand the current quality improvement paradigm relies on the notion of *process* associated to the idea of repeatability end, thus, akin to continuity.

2.4 Software Process Management and Improvement

A difficulty arising when facing a poorly structured or inmature process is that it does not lend itself easily to formal modeling. The mechanisms which can explain the performance remain hidden behind a conglomerate of informal and opaque practices. The very effort of modeling becomes a sort of diagnosis of the current state and can actuate as a first step toward process improvement. When simulating it seems reasonable, instead of taking for granted the existence of formal metrics and procedures to acquire them, try to model how this is done.

3 Previous Work

Software project simulation is an active research field originating in the seminal work by Abdel-Hamid and Madnick [2] based on System Dynamics which has been refined and enriched with many contributions and integrated with other lines of research and simulation paradigms.

From a professional and academic point of view the major reference consists in the series of PROSIM workshops (*Workshop of Software Process Simulation and Modeling*) starting in 1998 which became in 2006 the SPW (*Software Process Workshop*), and later a special track of ICSP, an ICSE (*International Conference on Software Engineering*) co-located event. A recent review by [21] provides a comprehensive outline of the kind of problems that are researched and the methodological approaches used. When the research here presented was in its final stage, a book by Madachy [12] was published in which the state of the art of continuous simulation of the software process is thoroughly presented.

3.1 Multiproject and Incremental Models

It is worthwhile mentioning that even though as soon as 1993 Abdel-Hamid [1] pointed out that estimation models and, by extension, simulation models would be of little use if the implications of staffing at the whole organization level are not taken into account, the amount of work produced relating to multiproject settings is quite scarce. Recently published work [21] shows that this gap in research remains to be filled. In a review of the proceedings of the before mentioned workshops between 1998 and 2007 only one out of 61 papers is classified as *multi-project* in scope. Another paper from the 2008 edition [7] focused entirely on multiproject resource allocation literature, points only six cases of simulation models.

Most published multiproject models, on the other hand, explicitly model a particular instance of a multiproject or incremental setting but they are not intended to model *any* configuration nor superposition of projects. An interesting exception is constituted by Powell *et al.* [16], a System Dynamics model of concurrent development. The authors propose an abstracted model connecting resources, time and effort defined in a modular way at several hierarchical levels: work package, phase, deliverable, project and the organization.

3.2 Simulation and Advanced Methods for Resource Allocation

Most approaches combining simulation and advanced methods for resource allocation between several projects are oriented either to assessing Operation Research based solutions using Monte Carlo simulation or simulating the state space where a heuristic method is used to find a 'good enough' solution.

The first approach, as shown in [3,13], requires statistical chraracterizing of the problem and it is oriented basically toward risk assessment. Lee and Miller [11] is another example combining dynamic simulation with project management techniques. Padberg [15] presents a line of work in which schedules are generated through an approximate *dynamic programming* algorithm optimized over a subspace of the whole solution space. Neither of the two approaches consider dynamic reallocation policies so in fact decision making is not simulated.

To simulate decision making resource allocation should change dynamically in respisne to certain events. This is implementeted by Özdamar [19] who employs priority rules in projects defined in *fuzzy* terms. Another interesting example is Joslin and Poole [10] who present an agent-based model which simulates dynamically the assignment of staff to a project with several functionalities that must be delivered within previously fixed deadlines.

3.3 Hybrid Models

During the last decade, hybrid simulation has been one of the most frequent research themes. Most of the work is based on the combination of continuous (System Dynamics) and discrete-event simulation.

Rus et al. [18] and Martin and Raffo [17] are examples where the work environment, productivity and resources are treated as continuous variables while the dynamics of the work products and artifacts is presented as essentially discrete. The model presented in [14] follows the same scheme although it also implements a hierarchical approach when representing global software development.The model presented by Donzelli and Iazolla [8] treats the work process as basically continuous and resources as queue servers which obviously are discrete.

The DEVS formalism established by Ziegler [20] is used by Choi *et al.* [5] in what seems the more methodologically consistent proposal for hybrid modeling at least as far as the authors of this paper have been able to identify. Based on this formalism the cited authors formulate a model obviously inspired in the classical one by Abdel-Hamid and Madnick [2] in which the usage of an adequately small time step and the so called QSS - *quantized-state system* - as an alternative to

interpolation for numerical integration provides at least theoretically a natural way to hybrid simulation.

4 The SimHiPros Model

SimHiProS is an hybrid model which simulates the production of software in a multiproject environment through a hierarchy starting from the most elementary work processes up to the wohle organization. The first version has been coded in Modelica simulation language and implemented in a MathModelica[1] simulation environment. The hierarchy is instantiated through a modular architecture comprising three levels: *package*, *project* and *multiproject*. Apart from these hierarchically encapsulated modules which represent the hierarchy of the production process the model comprises a module of *environment* which allows to simulate the dynamics of the workforce in the organization and a *functional* module which implements functions and algorithms and is called by the former modules. Each of the three hierarchically ordered modules has got three components: *activity*, *allocation* and *measure and control*. The block diagrams presented in figures 1 and 2 represent these components as blue, orange and green boxes respectively.

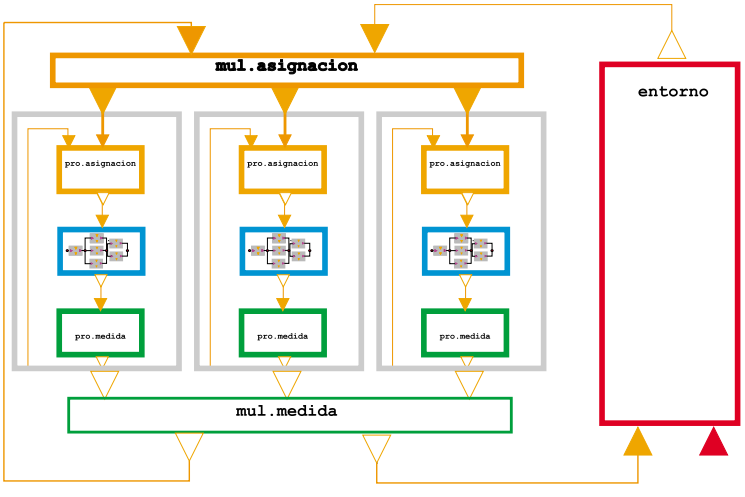

Fig. 1. Block diagram of the general architecture of SimHiProS

4.1 The Basic Level: Package Module

The *package module* simulates the work process as a number of elementary work units, *tasks*, which must be carried out to obtain a certain artifact. The *package size* is the number of tasks that must be completed. The *activity* component within the package simulates the continuous work process as tasks moving

[1] MathModelica is a Trademark of MathCore Engineering AB.

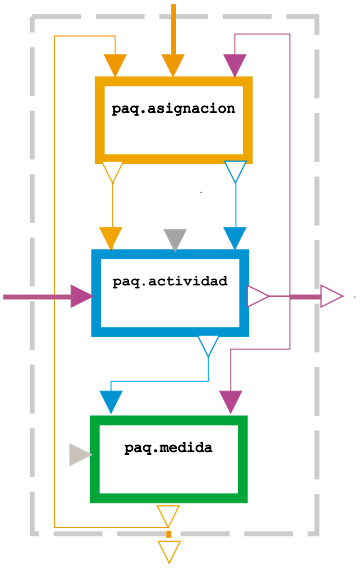

Fig. 2. The *package* module

through four consecutive *backlogs*: initial tasks, tasks to build, tasks to verify and eventually tasks to iterate and accumulating in a finished tasks level. The rates represent the actual work in preparation, construction, verification, error detection and correction and are determined by the resources allocated and the productivity of these resources. The productivity and the fraction of tasks (error production) to iterate is fixed in the current implementation although both are typically modeled as non-linear phenomena in state-of-the-art models. These non linearities have been intentionally set aside because the emphasis in the current model has been set upon the dynamic effects of the allocation policies.

The *allocation* component at the package level distributes the resources available to the package, which are themselves assigned at the *project* level, to the different rates acting upon the backlogs. Several elementary intra-package allocation methods have been implemented such as a FIFO dispatching rule or allocating personnel in proportion to pending backlogs. The assigned resources to each rate vary in a discrete manner in a double sense: the assignments do not change continuously but at fixed time intervals and the number of resources (developers, engineers, ...) assigned is always integer.

The *measurement* component of this module samples at fixed intervals the levels (backlogs) and records the resource usage. These data are used by the *allocation* component and are also passed to the *project* module to be used as is described in the following paragraph.

4.2 The Operational Level: The Project Module

At the operational level, the *activity* component is a *network of packages* (see figure 3) logically connected through precedence relations representing the sequence

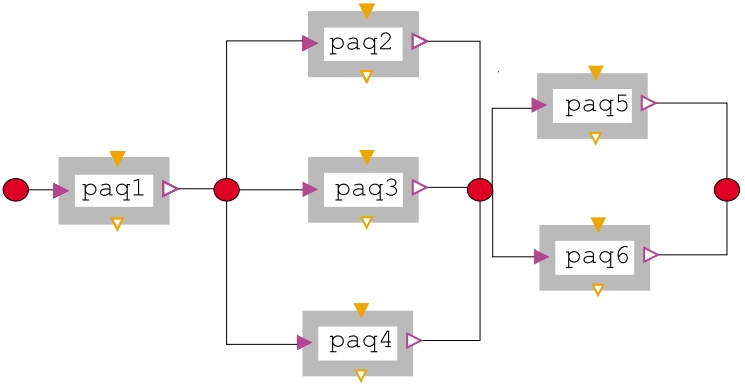

Fig. 3. Activity component of the *project* module

of artifacts in which the whole development process is organized. A package is activated when the logically preceding artifacts are complete. This gives the model enough flexibility to represent different WBS (work breakdown structures) and even different development cycle models.

The *measurement* component gathers the output of the corresponding components at the package level and calculates project metrics designed following an Earned Value Management framework. This metrics are passed on to the tactical (multiproject) level and are also used to activate the reallocation algorithms if necessary. In order to trigger the reassignment procedures, the *measurement* component makes estimates of the finish dates and compares them to the planned values. In case the deviation grows bigger than a previously set value, resources are reassigned.

The *allocation* component distributes between packages the resources allocated to the current project following a *squeaky wheel optimization* algorithm aiming to reduce makespan subject to time-window constraints. If this algorithm is not able to find a solution respecting the due dates, a message is passed to the tactical level to allow for resource reallocation across projects. The non-linear effects of schedule pressure on productivity and error generation which have been laid apart in the initial implementation of the model would naturally be modeled within this module as the relevant factors pertain to the project level.

4.3 The Tactical Level: Multiproject Module

At the tactical level, the *activity* component is made up of the projects currently active in the organization. The *measurement* component keeps track of the metrics of each individual project and their deviations and provides signals to the *allocation* component which reassigns resources between projects.

4.4 The Environment and the Functional Modules

The *environment* module models in a traditional System Dynamics fashion the evolution of human resources in terms of new personnel hiring, dismissals,holiday

or illness leaves, or resources temporarily unavailable due to infrastructural tasks, training schemes, etc. Although effective personnel is modeled in an homogeneous category it is possible to divide this into several career levels accounting for different degrees of experience and maturity.

The *functional* module contains the different functions called by the *allocation components* thus implementing the resource allocation algorithms and other auxiliary routines such as extrapolation of forecast values, precedence rules maintenance, WBS network topology description, etc.

4.5 Operation of the Model

The model is initialized with the projects under study, their WBS and size; the initial resource allocation provided by the tactical planning; and a forecast for staff demography (new contracts, people on leave, ...). In absence of any disruption, the model will follow the planned course allocating resources to projects following the tactical plan, and to work packages within projects following the operational model. In this sense, each project (itself an instantiation of the project module) actuates as an autonomous agent. All this is constrained by the dynamics of available staff simulated by the environment module.

Once the model has completed a base run, the results translate into planned time profiles for the metrics of each individual project. These values are recorded and experiments of disruptions can be carried out. These experiments consist in unexpected events affecting staff (such as people leaving in the middle of a project) of affecting projects (new requirements, errors discovered, ...). The operational agent will try to mend the schedule within the corresponding project initially by expediting all feasible resources to the affected tasks and subsequently exploring better alternatives with the randomized *squeaky wheel algorithm*. In case it is not enough, a message will be passed to the tactical level. In this case, the tactical plan is repaired with a simple neighborhood heuristics.

5 First Results, Conclusions and Limitations of the Model

5.1 First Results

The model's parameters have been estimated with data obtained from a sample of projects carried out by a middle sized company working for public agencies in Spain. In fact the model was developed to gain insight in the production management problems of this company. With this parameters and a set of 'stylized' events representative of the kind of disruptions the projects are liable to, a series of simulation experiments has been carried out.

Figures 5 and 6 show the responses of the model in a simplified case concerning two projects (whose activity networks are depicted in figure 4). In the first case there is a reassignment between activities within the first project. The second figure shows what happens when the resources assigned to the first project are not enough to repair the schedule and they must be borrowed from the second project.

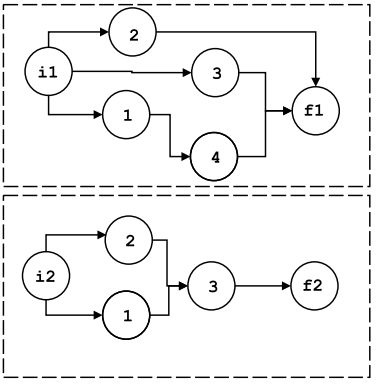

Fig. 4. Two sample projects on *activity on node* notation

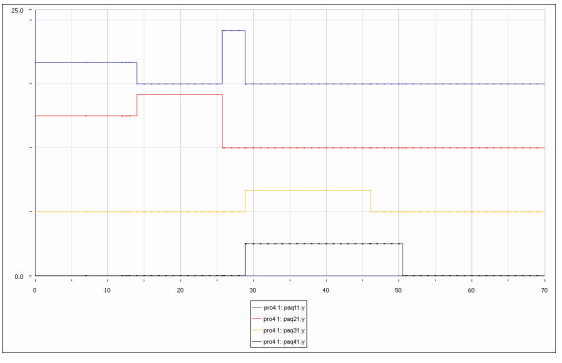

Fig. 5. Resource reassignment between activities of the first sample project

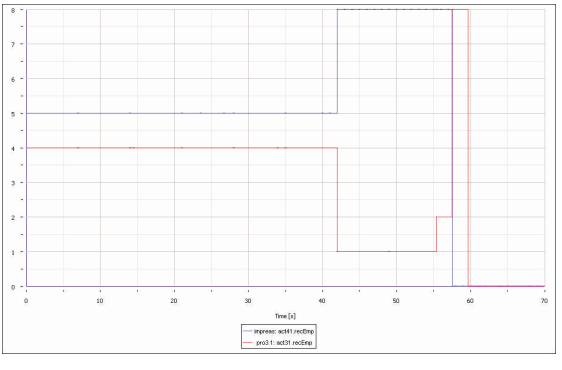

Fig. 6. Resource reassignment between the two sample projects

5.2 Preliminary Conclusions

Although the model is still in a first version, some conclusions can be drawn in the sense that the features which were conceptualized as critical for the kind of software projects under study have been successfully incorporated to the model. In particular the model represents, to the best of our knowledge, a step forward in the state of the art of dynamic simulation of the software process at least in the following issues:

- It is a *naturally* hybrid model based on DAE *differential algebraic equations* with a numerical solver which manages the integration step dynamically allowing for very short intervals in the vicinity of discrete events
- It has got a modular architecture allowing for simulation of various life-cycle models (waterfall, incremental, evolutionary, ...) and capable of representing hierarchies
- It implements three levels of hierarchically differentiated decision making levels (work package, project and multiproject)
- The resource allocation decisions at project and multiproject level are dynamically activated by the results of the simulation and are based on algorithms and models from the OR/AI community
- Measurement effort is explicitly modeled and associated costs are recorded and accumulated as a first step towards assessing the trade-offs of control policies.

5.3 Limitations of the Model

In its current version, the model has got the following limitations:

- The programming language is a declarative one and its algorithmic possibilities are not very reaching so the model implements relatively simple allocation algorithms; at the operational level serial generation schemes based on priority rules and SWO, at the tactical level a very simple neighborhood heuristic.
- Many state-of-the-art non linearities have not been implemented: such as the impact of schedule pressure on productivity and percentage of errors, communication overheads, ... These effects have intentionally been laid apart because the research interest was focused on dynamics provoked by resource sharing, but the model is prepared to incorporate them.
- The WBS of the projects remain static during the simulation although the work contents of any particular task can change; this way a change in the structure of a project which can be the outcome of a decision by the project manager in response to some disruption cannot be simulated.

6 Further Developments

Further developments planned consist basically in overcoming some of the limitations stated above and exploiting the model's features and capabilities as a

support tool for the concrete project management problems under consideration in incumbent company.

In the first strand, the immediate tasks will be incorporating the results of previous work concerning non-linearities in relation to productivity, quality, ... to the model. Secondly, implementing more sophisticated algorithms of resource scheduling. The Modelica language is oriented toward scientific simulation of models of physical and/or technical systems and lacks powerful algorithmic features. The only possibility to employ more sophisticated procedures is through calls to C++ or FORTRAN coded functions.

A more ambitious goal is developing the possibility of dynamical reconfiguration of the WBS of projects as response to disruptions, something that is consistent with rational management practices but it is very difficult to implement with the tools used up to now. Both this and the afore mentioned issue of the algorithmic power calls for a reconsideration of the modeling language.

Concerning the usage of the model as a tool for production management a debate is currently under way with the incumbent company on how to evolve the model to integrate it with the current project control system to use it in a Decision Support Tool for project management.

Acknowledgments

The work here presented has been supported by the Spanish Ministry of Science and Innovation grants no. TIN2004-06689-C03-03 and TIN2007-67843-C06-03. The authors wish to thank the useful remarks and suggestions received from three anonymous reviewers of a previous version of this paper.

References

1. Abdel-Hamid, T.K.: A multiproject perspective of single-project dynamics. Journal of Systems and Software 22(3), 151–165 (1993)
2. Abdel-Hamid, T.K., Madnick, S.E.: Software Project Dynamics An Integrated Approach. Prentice-Hall, Englewood Cliffs (1991)
3. Antoniol, G., Cimitile, A., Di Lucca, G.A., Di Penta, M.: Assessing staffing needs for a software maintenance project through queuing simulation. IEEE Transactions on Software Engineering 30(1), 43–58 (2004)
4. Brucker, P., Drexl, A., Möhring, R., Neumann, K., Pesch, E.: Resource-constrained project scheduling: Notation, classification, models, and methods. European Journal of Operational Research 112(1), 3–41 (1999)
5. Choi, K., Bae, D., Kim, T.: An approach to a hybrid software process simulation using the DEVS formalism. Software Process: Improvement and Practice 11(4), 373–383 (2006)
6. Demeulemeester, E.L., Herroelen, W.: Project scheduling, vol. 49. Kluwer Academic Publishers, Boston (2002)
7. Dong, F., Li, M., Zhao, Y., Li, J., Yang, Y.: Software Multi-project Resource Scheduling: A Comparative Analysis. In: Wang, Q., Pfahl, D., Raffo, D.M. (eds.) ICSP 2008. LNCS, vol. 5007, pp. 63–75. Springer, Heidelberg (2008)

8. Donzelli, P., Iazeolla, G.: Hybrid simulation modelling of the software process. The Journal of Systems and Software 59(3), 227–235 (2001)
9. Herroelen, W.: Project scheduling-theory and practice. Production and Operations Management 14(4), 413 (Winter 2005)
10. Joslin, D., Poole, W.: Agent-based simulation for software project planning. In: Winter Simulation Conference, pp. 1059–1066 (2005)
11. Lee, B., Miller, J.: Multi-project management in software engineering using simulation modelling. Software Quality Journal 12, 59–82 (2004)
12. Madachy, R.J.: Software process dynamics. Wiley, IEEE Press, Hoboken, Piscataway (2008)
13. Meier, C., Yassine, A.A., Browning, T.R.: Design process sequencing with competent genetic algorithms. Transactions of the ASME 129, 566 (2007)
14. Setamanit, S., Wakeland, W., Raffo, D.: Planning and improving global software development process using simulation. In: GSD (2006)
15. Padberg, F.: On the potential of process simulation in software project schedule optimization. In: 29th Annual International Computer Software and Applications Conference, 2005. COMPSAC 2005, vol. 2, pp. 127–130 (2005)
16. Powell, A., Mander, K., Brown, D.: Strategies for lifecycle concurrency and iteration â a system dynamics approach. Journal of Systems and Software 46(2-3), 151–161 (1999)
17. Raffo, D., Martin, R.H.: A model of the software development process using both continous and discrete models. Software Process Improvement and Practice 5, 147–157 (2000)
18. Rus, I., Collofello, J., Lakey, P.: Software process simulation for reliability management. The Journal of Systems and Software 46(2–3), 173–182 (1999)
19. Özdamar, L., Alanya, E.: Uncertainty modelling in software development projects (with case study). Annals of Operations Research 102(1-4), 157–178 (2001)
20. Zeigler, B.P., Kim, T.G., Praehofer, H.: Theory and practice of modeling and simulation. Academic Press, New York (2000)
21. Zhang, H., Kitchenham, B., Pfahl, D.: Reflections on 10 Years of Software Process Simulation Modeling: A Systematic Review. In: Wang, Q., Pfahl, D., Raffo, D.M. (eds.) ICSP 2008. LNCS, vol. 5007, pp. 345–356. Springer, Heidelberg (2008)