Appendix B

ANNOTATED SYSTEM DYNAMICS BIBLIOGRAPHY

CONTENTS AND SOURCES

This section summarizes publications that involve system dynamics modeling of software processes. These same citations are listed in boldface in the references. Some grouped longitudinal bodies of work are listed with multiple citations. Only modeling with system dynamics is covered; therefore, important work with other techniques is not listed. Figure B-1 shows a Venn diagram of the current coverage. Some hybrid modeling with system dynamics and discrete event methods are included, less than 10% of the citations listed.

Publication titles in the bibliography with quotation marks are the original titles from the authors. Titles for groups of references describe the overall work and do not have quotations. Each entry in the bibliography also contains an abbreviation key for the type of reference per Figure B-2.

Books represented include the seminal *Software Project Dynamics* [Abdel-Hamid, Madnick 1991], recent edited compilations on software process modeling with chapters involving system dynamics [Acuña, Juristo 2005, Acuña, Sánchez-Segura 2006], other edited books with relevant chapters and even a novel on software project management [Demarco 1998].

The journals most frequently represented in the bibliography that publish about software process modeling with system dynamics are:

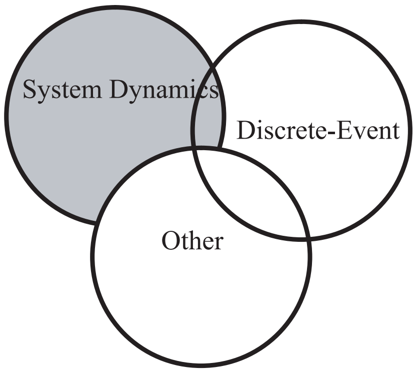

Figure B-1. Coverage of modeling techniques.

- *Software Process Improvement and Practice* (see special issues per Table B-1 at the end of this appendix)
- *Journal of Systems and Software* (see special issues per Table B-1 at the end of this appendix)
- *American Programmer* (see special issue per Table B-2 at the end of this appendix). This journal has since been renamed as *Cutter IT Journal.*

Other journals include:

- *Annals of Software Engineering*
- *Communications of the ACM*
- *IEEE Computer*
- *IEEE Software*
- *IEEE Transactions on Education*

$\boxed{\text{B}}$—Book

$\boxed{\text{BC}}$—Book chapter

$\boxed{\text{J}}$—Journal article

$\boxed{\text{C}}$—Conference or workshop proceeding

$\boxed{\text{A}}$—Academic dissertation or student course project

$\boxed{\text{TR}}$—Institutional or university technical report

$\boxed{\text{O}}$—Other type of publication, including workbooks and manuals

Figure B-2. Publication type legend.

- *IEEE Transactions on Software Engineering*
- *Information and Software Technology*
- *International Journal of Software Engineering and Knowledge Engineering*
- *Software Concepts and Tools*
- *Software Practice and Experience*

These journals in the above list are also the most likely sources for future work in the field.

Since 1998, the *International Workshop on Software Process Simulation and Modeling (ProSim)* has been the main venue for the field. It is now called the *International Conference on Software Process* after recently joining with the *International Software Process Workshop* (previously held since the 1980s) to become the leading event for systems and software process research. Each year, selected best papers are revised and published in a special journal. Table B-1 is an endnote to the bibliography that lists the year of the workshop and the corresponding journal issue with the revised papers. Only the journal versions are listed here for papers published in both places. If one is not available, then readers can consult the table to find the corresponding publication.

The *International System Dynamics Conference* is the leading venue for system dynamics and systems thinking that includes contributions from many disciplines besides software. There are other conferences and workshops represented that are related to software process improvement, cost modeling, software quality, and similar topics in software or systems engineering.

Numerous companies, government agencies, and universities have sponsored work listed in the tables. They are sometimes the sources for technical reports, dissertations, and other references. Companies have also published internal proprietary reports involving system dynamics not available to the public.

BIBLIOGRAPHY

[Abdel-Hamid 1984] $\boxed{\text{A}}$

"The Dynamics of Software Project Staffing: An Integrative System Dynamics Perspective"

This was Abdel-Hamid's Ph.D. dissertation of his integrated model for software projects. See [Abdel-Hamid, Madnick 1991] for full details of the model.

[Abdel-Hamid 1989a] $\boxed{\text{J}}$

"The dynamics of software project staffing: A system dynamics based simulation approach"

This describes the Abdel-Hamid model of software project staffing throughout the software development lifecycle. A case study is used to simulate staffing practices. The experiment provides insights into explicit and implicit policies for managing human resources. See [Abdel-Hamid, Madnick 1991] for the rest of the integrated model and more simulation experiments with it.

[Abdel-Hamid, Madnick 1989b] $\boxed{\text{J}}$
"Lessons learned from modeling the dynamics of software development"
This article presents a version of the Abdel-Hamid integrated model of the dynamics of software development, and shows how it can provide insight and make predictions for the software process. It is a forerunner of the book [Abdel-Hamid, Madnick 1991].

[Abdel-Hamid 1990] $\boxed{\text{J}}$
"Investigating the cost/schedule trade-off in software development"
In previous studies, schedule compression had been treated as a static decision that can be unambiguously measured. However, the study in this paper indicates that it is neither.

[Abdel-Hamid, Madnick 1991] $\boxed{\text{B}}$
"Software Project Dynamics"
This is the landmark first book on using system dynamics for software development, where the Abdel-Hamid model is presented in detail. It describes an integrated dynamic model for software project management based upon extensive review of the literature supplemented by focused field interviews of project managers. The integrative model captures multiple functions of software development, including management functions of planning, controlling and staffing and software development functions like designing, coding, reviewing, and testing. It uses a case study to illustrate how the model was able to accurately replicate the actual development of a software project. It utilizes the model as an experimentation vehicle to study and predict the dynamic implications of managerial policies and procedures.

Some classic process effects are included in the model such as task underestimation, progress perception delays, schedule pressure, hiring policies, overwork, staff assimilation and attrition, and more. The book synthesizes previous work including [Abdel-Hamid 1989, Abdel-Hamid, Madnick 1989a], and some of his earlier publications. This model is featured in several chapters of this book and is used for experimentation by several others in the bibliography.

[Abdel-Hamid 1991] $\boxed{\text{J}}$
"Organizational learning: The key to software management innovation"
This paper shows that organizational learning is the key to management learning. A case is made that accurate measurements are not necessarily better.

[Abdel-Hamid 1993a] $\boxed{\text{J}}$
"Adapting, correcting, and perfecting software estimates: a maintenance metaphor"
This paper contends that continuous estimation models are needed that can constantly evaluate schedules. A hybrid estimation model is proposed. A demonstration of how to use this model is also given.

[Abdel-Hamid et al. 1993] $\boxed{\text{J}}$
"Software project control: An experimental investigation of judgment with fallible information"
Heuristics are deployed to handle the problems of poor estimation and visibility that hamper software project planning and control. Also, the implications for software project management are presented. A laboratory experiment in which subjects managed a simulated software development project is conducted.

[Abdel-Hamid 1993b] $\boxed{J}$
"Thinking in circles"
Because most managers forget to think in circles, they get into trouble. Because managers continue to believe that there are such things as unilateral causation, independent and dependent variables, origins, and terminations, managerial problems persist.

[Abdel-Hamid 1993c] $\boxed{C}$
"Modeling the dynamics of software reuse: an integrating system dynamics perspective"
An integrating system dynamics approach is proposed for modeling software reuse. This approach integrates multiple functions of the software reuse process and uses feedback principles of system dynamics.

[Abdel-Hamid 1993d] $\boxed{J}$
"A multi-project perspective of single project dynamics"
The project-in-isolation assumption is relaxed to demonstrate that interactions and interdependencies among projects can have a significant influence on project behavior. A study is conducted on a multiproject system dynamics model that was developed on the basis of field studies in five organizations.

[Acuña, Juristo 2005] $\boxed{B}$
"Software Process Modeling"
This edited book focuses on new aspects of software process modeling. It deals with sociotechnological aspects, process modeling for new development types, and organization change management. See the chapters [Madachy, Boehm 2005] on software dependability applications including system dynamics, and [Lehman et al. 2005] on modeling with system dynamics for managing software evolution.

[Acuña, Sánchez-Segura 2006] $\boxed{B}$
"New Trends in Software Process Modelling"
This edited book addresses new trends in software process modeling related to open source software, system dynamics modeling, and peopleware: the importance of people in software development and by extension, in the software process. See the chapters [Pfahl et al. 2006b] on system dynamics for learning and decision support, and [Barros et al. 2006b] on system dynamics metamodels.

[Angkasaputra, Pfahl 2004], [Angkasaputra, Pfahl 2005] $\boxed{C}$
Agile and Pattern-based Software Process Simulation Modeling
One reason software process simulation is not yet widely accepted is the difficulty of delivering valid simulation models on time and within budget. As simulation models can be considered software, the authors believe that promising software development techniques such as agile methods and (design) patterns can be utilized to help solve the problem. They present a research plan that aims at enhancing IMMoS (see [Pfahl 2001]), a methodology for systematic simulation model development, by introducing agile techniques and design patterns. They provide an example of a design pattern for process simulation modeling. As research proceeds, it is expected that the new method, agile/P-IMMoS, will help shorten simulation model delivery time and cut down modeling costs without compromising model quality. In [Angkasaputra, Pfahl 2005] they identify the different possibilities to adopt the agile

practices used in XP and present the enhanced IMMoS as a potential agile method for model development. It can shorten the model delivery, and be responsive to changes in the modeled process.

[Aranda et al. 1993] $\boxed{J}$

"Quality microworlds: Modeling the impact of quality initiatives over the software product life cycle"

This work melds the concepts of Total Quality Management (TQM) with software development modeled as a stream of releases/versions. A long time horizon is used to model customer requirement evolution and released functionality separate from individual project perspectives. To no surprise, the model helps to demonstrate the leverage of long-term considerations. Both Ithink and Microworld Creator are used for implementation.

[Baik, Eickelmann 2001] $\boxed{C}$

"Applying COCOMO II effort multipliers to simulation models"

This presentation describes work at Motorola integrating COCOMO II factors into dynamic simulation models. An example is shown for the system test process and future work is discussed.

[Barbieri et al. 1992] $\boxed{C}$

"DynaMan: A tool to improve software process management through dynamic simulation"

These researchers created the Dynaman tool for implementing system dynamics as applied to software process management. This paper describes the first version of the tool.

[Barros et al. 2000a] $\boxed{C}$

"Applying system dynamics to scenario based software project management"

Scenario-based software project management is an extension of the risk management paradigm that uses system dynamics abstract models to describe potential problems that can affect a software project. Dynamic models are also used to describe possible resolution strategies applicable to eliminate or reduce the potential problem impact. Scenarios are defined by integrating combinations of these models to another based on Abdel-Hamid and Madnick's model with relations among development staff, software products, quality, project control, and planning. The original model was adapted to allow a fine-grained description of project tasks, personnel abilities, and error propagation. The modified model also allows operational project monitoring. The proposed technique allows the development of a standard problem and resolution strategy model library, to be integrated to new and ongoing projects. These models are abstract, and simulation is only accomplished when they are integrated to the project model. Their variables and equations affect the project model behavior, replicating the impact promoted by the problems and resolution strategies that they describe. The system dynamics notation was expanded to allow the definition of the integration interface. Scenario analysis is a valuable tool to predict project results, such as cost, schedule, and effort, in face of several combinations of problems and resolution actions. System dynamics complements the technique, describing nonlinear relationships and feedback loops among model elements.

[Barros et al. 2000b] $\boxed{\text{C}}$
"Using process modeling and dynamic simulation to support software process quality management"

The quality of a development process directly affects the quality of its produced artifacts. Based on this assumption, some research efforts have moved from improving just products to enhancing process quality. Different software process alternatives are possible for a software project. However, when several alternatives are made available, process improvement efforts are selected mostly based only on expert knowledge. This can be a risky selection method, due the complex nature of software development, which may induce counterintuitive reasoning and process behavior prediction. By considering the software development process as a complex dynamic system, this paper presents some applications of process modeling and dynamic simulation as tools for quality evaluation, prediction, and improvement of software processes. They explore a software project model and three simulation approaches, presenting an example of their application in project schedule prediction and quality assessment effort impact upon schedule.

[Barros et al. 2001a] $\boxed{\text{C}}$
"Explaining the behavior of system dynamics models"

This paper presents a technique called Event Tracking that helps the analysis of system dynamics models and is applied to the Abdel-Hamid model. It maps simulation trends over time to predefined events. It uses state machines whose behavior can be traced to changes suffered by selected variables in a model. Changes to variable values trigger messages, which are presented to the model analyst to help the interpretation of the underlying model behavior. The technique maps results to natural language statements. It allows a trainer assistant to define the relevant events for a model, highlighting the model major features through event messages. The technique is implemented in the ILLIUM simulator and it allows a student to track model behavior, while executing a simulation and following the presented messages.

[Barros et al. 2001b], [Barros et al. 2002a], [Barros et al. 2006b] $\boxed{\text{C}}$ $\boxed{\text{J}}$ $\boxed{\text{BC}}$
Metamodels for Software Process Modeling with System Dynamics

Metamodels are extensions to system dynamics that allow the development and specialization of domain models, providing a high-level representation for developers within domains. A domain expert develops a model, which conveys the relevant categories of elements that compose the domain and the relationships among those elements. A developer uses such model to describe a particular problem by specifying how many elements of each category exist in the model of interest and the particular characteristics of each one. Finally, the model is translated to system dynamics constructors in order to be simulated and analyzed. [Barros et al. 2001b] first shows these applied to software projects.

The authors have developed metamodels that allow the development of software process models based on high-level constructors instead of mathematical equations. These constructors represent software process domain concepts such as developers, activities, resources, and artifacts. A domain model allows the translation of these concepts to traditional stock-and-flow diagrams, which can be simulated to evaluate

the behavior of software process models. The proposed approach helps inexperienced modelers to build process models by reducing the semantic gap between system dynamics and the software process domain. Thus, resulting models are more readable and easier to understand and maintain. [Barros et al. 2002a] provides an example of a process model built with the approach, presenting its translation and simulation processes. [Barros et al. 2006b] provides more detail of metamodels and improves upon [Barros et al. 2002a].

[Barros et al. 2002b] $\boxed{C}$
"Evaluating the use of system dynamics models in software project management"
This paper presents an empirical study evaluating the application of system dynamics models in software project management. A project concerning the specification and implementation of part of an academic control system for a graduate department was proposed for several participants. The project was decomposed into an activity network and several developers were available to accomplish the activities. Each participant was asked to impersonate as the project manager and make decisions in order to finish the project in the least time possible. The results from the experimental study analysis show that, for the selected participants, managers using system dynamics models to support their decision perform better than managers who base their decisions only upon personal experience. This paper presents detailed results from the experiment and some directions to improve the application of system dynamics models in project management.

[Barros et al. 2003] $\boxed{C}$
"System dynamics extension modules for software process modeling"
Scenario models are extension modules that allow the separation of facts from assumptions in system dynamics models, as opposed to metamodels, which are system dynamics extensions that allow the development and specialization of domain models (see [Barros et al. 2001b] etc.). Application of scenario models suggests that a model must contain only known facts about its knowledge area, whereas assumptions are separately modeled as scenarios. To analyze the impact of an assumption upon a model behavior, a scenario model must be built to represent the assumption and further integrated to the model. The integration process automatically adjusts model equations according to the scenario, presenting the impact of the assumption upon the original model behavior. The authors discuss the advantages of using scenarios models instead of the traditional control parameter approach to *"what-if"* analysis and present an application of such models to support managerial decision making and process improvement in software projects.

[Barros et al. 2004] $\boxed{J}$
"Supporting risks in software project management"
This paper describes an approach to develop, retrieve, and reuse management knowledge and experience concerned with software development risks. Scenarios (see [Barros et al. 2003]) are used to model risk impact and resolution strategies efficacy within risk archetypes. A risk archetype is an information structure that holds knowledge about software development risks. A risk management process organizes the use of risk archetypes within an application development effort. The process resembles a reuse process framework, in which two subprocesses are respectively re-

sponsible for identifying and reusing risk information. Simulating the impact of the expected risks can support some of the decisions throughout the software development process. This paper shows how risk archetypes and scenario models can represent reusable project management knowledge. An observational analysis of applying such an approach in an industrial environment and a feasibility study are also described.

[Barros et al. 2006a] $\boxed{\text{J}}$

"Model-driven game development: experience and model enhancements in software project management education"

This work presents experiences in developing system-dynamics-based games for software project management. It describes a project management game intended for training purposes and the changes that were made to allow a system dynamics simulator to support game-like interactions. It also describes an experimental evaluation of the game's application as a learning-by-doing environment for management students. On the basis of the experience acquired by building such an interface, they propose models to describe the story underlying a game and its graphical presentation. Such models allow the construction of games without programming, thus hastening the development of simulation-based games.

[Burke 1996] $\boxed{\text{TR}}$

"Radical improvements require radical actions: Simulating a high-maturity software organization"

This study of process improvement dynamics in a high-maturity organization was commissioned by the SEI. Steve Burke of Computer Sciences Corporation (CSC) used the NASA SEL environment to model software development and software process improvement (SPI) processes. The model separates personnel into pro-SPI, con-SPI, and no-cares. One of the important lessons of the model is that organizations should pay attention to the attitudinal mix of personnel. This is recommended reading for those interested in using simulation to analyze SPI potential. An adaptation of the model is highlighted in Chapter 5 [Ho 1999].

[Chatters et al. 2000] $\boxed{\text{J}}$

"Modelling a software evolution process: A long-term case study"

As part of the ongoing FEAST investigation into the effect of feedback on the long-term evolution of software systems, system dynamics models of the evolutionary processes of real-world software products are being developed. The model described here simulates the evolution over 13 years and many releases of one of the constituent parts of the VME mainframe operating system. The model reflects the assumption, supported by expert opinion, that, in addition to any exogenously generated demand for VME enhancements, enhancing VME itself results in demands for further enhancements. This circular process is embodied in the model structure. Model input parameters have been calibrated using metric data and collaborator experts' judgment. Comparisons of model outputs against actual values for implemented change requests and accumulated code size are shown. The feedback-based model provides a plausible explanation of trends in VME evolution, and that the successful calibration of a model abstracting lower-level dynamics such as release schedules suggests that these may have only limited influence on

global process trends. These findings are broadly in line with the Laws of Software Evolution. The results of the work demonstrate that a simple model can successfully simulate the externally perceived behavior of the software process. See [Lehman 1996] and others for more information on FEAST and software evolution.

[Chichakly 1993] $\boxed{\text{J}}$

"The bifocal vantage point: Managing software projects from a systems thinking perspective"

The "bifocal" vantage point refers to systems thinking as keeping one eye on the big picture, and one eye on daily details. A low–medium complexity model implements some traditionally held views of software development.

[Christie 1998] $\boxed{\text{C}}$

"Simulation in support of process improvement"

This article shows how simulation including system dynamics supports software process improvement by enhancing process definition. It is shown how simulation can support activities at all five levels of CMM maturity, when the simulation capability matches the needs at each level. Simulation can start out being used qualitatively at the lower maturity levels and advance to quantitative prediction at the higher levels. At the lowest level, simulation can help improve awareness of process dynamic behavior and the sometimes insidious effect of feedback on process performance. As process maturity improves, simulation is increasingly tied to operational metrics, to validate one's understanding of process behavior and improve the simulations' predictive power. At level 5, for example, organizations now have detailed, validated models of their processes, and can quantitatively predict the impact of significant changes to these processes.

[Christie, Staley 2000] $\boxed{\text{J}}$

"Organizational and social simulation of a software requirements development process"

This work explores the organizational issues of requirements development and also the social issues, namely, how does the effectiveness of "people interactions" affect the resulting quality and timeliness of the output. It describes the modeling and simulation of requirements development since this early activity is crucial to the success of any software project. It models a Joint Application Development (JAD) process as implemented by the Computer Sciences Corporation. This process was used to identify detailed requirements for a large incremental release of a command and control system for the U.S. Air Force. Using the JAD process, system users and engineering specialists worked jointly in an electronic meeting environment and successfully reduced development time and failure risk. This model has both continuous and discrete modeling components.

[Collofello et al. 1995] $\boxed{\text{A}}$

"Modeling software testing processes"

To produce a high-quality software product, application of both defect prevention and defect detection techniques is required. One common defect detection strategy is to use unit, integration, and system testing. The authors propose utilizing system dynamics models for better understanding of testing processes.

[Collofello 2000] $\boxed{J}$
"University/industry collaboration in developing a simulation-based software project management course"
A significant factor in the success of a software project is the management skill of the project leader. The ability to effectively plan and track a software project utilizing appropriate techniques and tools requires training, mentoring, and experience. This paper describes a collaborative effort between Arizona State University and Motorola to develop a software project management training course. Although many such courses exist in academia and industry, this course incorporates a system dynamics simulator of the software development process. The use of this simulator to train future software project managers is analogous to the use of a flight simulator to train pilots. This paper describes the software project simulator and how it is utilized in the software project management training course. Feedback from the training course participants is also shared and discussed.

[Collofello et al. 1998] $\boxed{C}$
"A system dynamics process simulator for staffing policies decision support"
Staff attrition is a problem often faced by software development organizations. How can a manager plan for the risk of losses due to attrition? Can policies for this purpose be formulated to address his/her specific organization and project? The authors propose a software development process simulator tuned to the specific organization, for running "what-if" scenarios for assessing the effects of managerial staffing decisions on project's budget, schedule, and quality. They developed a system dynamics simulator of an incremental software development process and used it to analyze the effect of the following policies: to replace engineers who leave the project, to overstaff in the beginning of the project, or to do nothing, hoping that the project will still be completed in time and within budget. This paper presents the simulator, the experiments run, the results obtained, and analysis and conclusions.

[Collofello et al. 1996] $\boxed{C}$
"Modeling software testing processes"
The production of a high-quality software product requires application of both defect prevention and defect detection techniques. A common defect detection strategy is to subject the product to several phases of testing such as unit, integration, and system. These testing phases consume significant project resources and cycle time. As software companies continue to search for ways for reducing cycle time and development costs while increasing quality, software testing processes emerge as a prime target for investigation. This paper proposes the utilization of system dynamics models for better understanding testing processes. Motivation for modeling testing processes is presented along with an executable model of the unit test phase. Some sample model runs are described to illustrate the usefulness of the model.

[Cooper, Mullen 1993] $\boxed{J}$
"Swords and plowshares: The rework cycles of defense and commercial software development projects"
Rework dynamics are the focus of this modeling effort; it is well known that rework is an important consideration since it directly impacts profit margins. The

model does not represent only software aspects. An interesting presumption about the differences in perceived progress in commercial versus defense industries is made.

[DeMarco 1998] $\boxed{B}$

"The Deadline"

This fictional novel on software project management vividly illustrates principles that affect team productivity. Coincidentally, a character by the name of Dr. Abdul Jamid helps management assess their project deadline with system dynamics modeling of the team and their productivity.

[Diehl 1993] $\boxed{J}$

"The analytical lens: Strategy-support software to enhance executive dialog and debate"

In this article, MicroWorld S**4 is discussed. The three key elements of this tool that provide the needed flexibility to answer "what-if" and "why" questions are the following: flexible report generation, dynamic data analysis, and dynamic what-if analysis.

[Fakharzadeh, Mehta 1999] $\boxed{C}$ $\boxed{A}$

"Architecture Development Process Dynamics in MBASE"

The architecture of a system forms the blueprint for building the system and consists of the most important abstractions that address global concerns. It is important to develop an architecture early on that will remain stable throughout the entire life cycle and also future capabilities. The dynamics of the architecture development process were investigated on this research project at USC. They modeled architecture development in the MBASE inception and elaboration phases, and the impact of other factors such as collaboration and prototyping on the process of architecting. The primary goals were to: (1) investigate the dynamics of architecture development during early life-cycle phases, (2) identify the nature of process concurrence in early phases, and (3) understand the impact of collaboration and prototyping on life-cycle parameters. This model is highlighted in Chapter 5.

[Ferreira 2002], [Ferreira et al. 2003] $\boxed{A}$ $\boxed{C}$

Measuring the Effects of Requirements Volatility on Software Development Projects

This Ph.D. dissertation and follow-on work examines requirements volatility, a common software project risk that can have severe consequences resulting in cost and schedule overruns and cancelled projects. A system dynamics model was developed to help project managers comprehend the effects of requirements volatility. Requirements volatility and its effects were studied using various analyses and modeling techniques. Study results were used to design major simulator components that were integrated into a previously developed and validated simulator from [Houston 2000], leveraging pre-existing software-risk-related systems-dynamics research. The base simulator was also extended to provide an encompassing view of the requirements engineering process. The distributions used for stochastically simulating the requirements volatility risk effects and requirements engineering factors were derived from a survey that included over 300 software project managers. The simulator can be used as an effective tool to demonstrate the researched effects of requirements volatility on a software development project. [Ferreira et al. 2003] up-

dates the work in [Ferreira 2002] and provides empirical data on volatility. This model is highlighted in Chapter 5.

[Glickman 1994] $\boxed{C}$

"The Bellcore-CSELT collaborative project"

This presentation describes experience over a couple years at Bellcore to adapt the Abdel-Hamid model for internal uses.

[Greer et al. 2005] $\boxed{C}$

"Identifying and mitigating risk across organizational boundaries in software-intensive space system programs"

This paper describes work at the Aerospace Corporation on the use of simulation to better understand the software-intensive system acquisition process. A case-study approach is used to explore the dynamics of "disconnects" in baselines across multiple organizations in a large software-intensive space system development program. Disconnects are the latent differences in understanding among groups or individuals that can negatively affect the program cost, schedule, performance, and quality should they remain undetected or unresolved. The authors have constructed a system dynamics model of communication effectiveness and delay across four organizations that sequentially and iteratively rely on each other for requirements and deliverables. Their analyses from multiple simulations suggest that the highest points of leverage in reducing disconnects are in increasing expertise levels, improving communication clarity, and accelerating the pace of assessing the impacts of changes in partner organizations' understandings and actions. These results oppose traditional assumptions that disconnects are due to external requirements changes and that speeding up organizational processes will reduce disconnects.

[Haberlein 2004] $\boxed{J}$

"Common structures in system dynamics models of software acquisition projects"

Projects that contract third parties to develop software are even more unpredictable and underestimated by management than pure development projects and, thus, there is also a need to model and simulate these projects. This article deals with the design of system dynamics models of the unexplored domain, at least with respect to system dynamics, of software acquisition. A framework is described that captures causal structures common to all models of acquisition projects. Outputs of one concrete instance of the framework are presented.

[Hart 2004] $\boxed{O}$

"Bridging Systems Thinking and Software Quality Improvement: Initiating a Software Learning Organization"

This book draft portion shows that the systems thinking discipline is a critical part of any software organization's ability to adapt to change, how this discipline can give insights into more effective ways to enhance an organization's current software engineer and management activities, and how this skill is the foundation for developing skills in the other learning organization disciplines. It is the author's intent to also foster an awareness and appreciation in the software community for the value of replacing our existing obsession with software process maturity levels and certifications with a focus on learning as a basis of continuous performance improvement.

[Henderson, Howard 2000] $\boxed{\text{J}}$

"Process strategies for large scale software development—simulation using systems dynamics"

CMPM, the Cellular Manufacturing Process Model, is an advanced component-based process strategy that uses concurrency and distribution to reduce cycle times. The aim of this research is to provide a simulation-based tool for designing and dynamically controlling CMPM processes. In CMPM, networks of semiautonomous cells cooperate to produce a complex large-scale system, and this paper examines some of the issues that affect the ability of cells to achieve their targets. The model views development as a manufacturing activity in which systems are built from components, which are a mixture of self-built components, reused components, and brought-in components. The model is hierarchical: any component may be a product of others. Predicting the cost, quality, and schedule outcome of CMPM depends upon the behavior within the cell (intracell) and the cooperative behavior between cells (intercell) in a dynamic environment.

[Ho 1999] $\boxed{\text{A}}$

"Xerox SPI model study"

This graduate research on software process improvement was applied at Xerox. It adapts the NASA SEL process improvement model in [Burke 1996] for Xerox CMM-based process improvement. This work is highlighted in detail in Chapter 5.

[Houston 2000] $\boxed{\text{A}}$

"A Software Project Simulation Model for Risk Management"

This Ph.D. dissertation developed a system dynamics model for risk management. The SPARS model is an adaptation of Abdel-Hamid's model [Abdel-Hamid, Madnick 1991] and Tvedt's model [Tvedt 1996]. The SPARS model is comprehensive in software project management scope and updated for more modern development practices. The consolidated model also incorporates the effects of a number of risk factors. Also see [Houston et al. 2001]

[Houston et al. 2001a] $\boxed{\text{J}}$

"Stochastic simulation of risk factor potential effects for software development risk management"

One of the proposed purposes for software process simulation is the management of software development risks, usually discussed within the category of project planning/management. However, modeling and simulation primarily for the purpose of software development risk management has been quite limited. This paper describes an approach to modeling risk factors [Houston 2000] based on and simulating their effects as a means of supporting certain software development risk management activities. The effects of six common and significant software development risk factors were studied. A base model was then produced for stochastically simulating the effects of the selected factors. This simulator is a tool designed specifically for the risk management activities of assessment, mitigation, contingency planning, and intervention.

[Houston et al. 2001b] $\boxed{\text{J}}$

"Behavioral characterization: finding and using the influential factors in software process simulation models"

Most software process simulation work has focused on the roles and uses of software process simulators, on the scope of models, and on simulation approaches. Little effort appears to have been given to statistical evaluation of model behavior through sensitivity analysis. Rather, most of software process simulation experimentation has examined selected factors for the sake of understanding their effects with regard to particular issues, such as the economics of quality assurance or the impact of inspections practice. In a broad sense, sensitivity analysis assesses the effect of each input on model outputs. Here, the authors discuss its use for behaviorally characterizing software process simulators. This paper discusses the benefits of using sensitivity analysis to characterize model behavior, the use of experimental design for this purpose, the procedure for using designed experiments to analyze deterministic simulation models, the application of this procedure to four published software process simulators, the results of analysis, and the merits of this approach.

[Houston 2006] $\boxed{\text{J}}$

"An experience in facilitating process improvement with an integration problem reporting process simulation"

Software process improvement (SPI) has been a discernible theme in the software process simulation literature, which has recognized a wide variety of ways in which simulation can support SPI. This case study describes one of those ways—a very focused, retrospective modeling driven by integration delays in the development of an avionics system project. A simple simulation of the integration problem report flows offered a low-cost means of looking into the dynamics of the backlogged integration process and clearly seeing a basic development problem that was difficult to see from existing reports. The model became especially helpful when alternative scenarios were run in order to see the relative benefits that different types of actions would have provided to the integration process. These experiments clarified lessons learned from other sources and suggested a major improvement for the next release cycle. The actual results of improvements deployed in the next release cycle were a reduced problem report backlog (one-third that of the previous release) and 40% less test effort.

[Johnson 1995] $\boxed{\text{J}}$

"Dynamic Systems Modeling: The Software Management Process"

This workbook by Margaret Johnson serves as a working companion to [Abdel-Hamid, Madnick 1991]. It includes a faithful reproduction of the Abdel-Hamid model in Ithink, which has been provided for the public domain through this book. It shows how to develop the Abdel-Hamid model from scratch, integrating model sectors and running sensitivity tests. Throughout, it also explains the software process phenomenology being modeled. It is a good end-to-end illustration of the modeling process and provides valuable modeling lessons.

[Kaghazian 1999] $\boxed{\text{C}}$ $\boxed{\text{A}}$

"Dynamic process of Internet companies: An abstract model"

This graduate research project investigated the dynamics of new Internet companies. One of the aspects of current shift in technology is the dynamic process of Internet companies. Financing, prototyping, teamwork, marketing, research and develop-

ment, and the whole process of current Internet companies have evolved in a new model. Companies with a great success in business have to develop their product in as short a time as possible and make if bug free as much as possible.

The paper provides an abstract model for the dynamic process of Internet companies, with emphasis on new features for Rapid Application Development. Four major factors are explained: outsourcing, hiring personnel, early error elimination, and instant bug fixing. The model is shown to be sensitive to a number of key factors such as personnel, the level of experience, outsourcing, integration, and the number of users who may access the site in a month.

[Kahen et al. 2001] $\boxed{J}$

"System dynamics modelling of software evolution processes for policy investigation: Approach and example"

This paper describes one of the system dynamics models developed during the Feedback, Evolution and Software Technology (FEAST) investigation into software evolution processes (see [Lehman 1996] etc.). The intention of early models was to simulate real-world processes in order to increase understanding of such processes. The work resulted in a number of lessons learned, in particular with regard to the application of system dynamics to the simulation of key attributes of long-term software evolution. The work reported here combines elements of previous work and extends them by describing an approach to investigate the consequences on long-term evolution of decisions made by the managers of these processes. The approach is illustrated by discussion of the impact of complexity control activity. This model of the impact on product and global process attributes of decisions regarding the fraction of work applied to progressive and to antiregressive activities such as complexity control, for instance, exemplifies the results of the FEAST investigation.

[Kim, Baik 1999] $\boxed{A}$

"Dynamic model for COTS glue code development and COTS integration"

This research project at USC sought to understand how the glue code development process and COTS integration process affect each other. The authors analyzed the effect on schedule and effort throughout a project lifecycle of glue code development and the ongoing integration of components into the developing system. This model is highlighted in Chapter 5.

[Kocaoglu et al. 1998] $\boxed{C}$

"Moving toward a unified model for software development"

The scope and nature of software process models have been influenced by the tools available to construct the models. This work develops a unified model that unites the capabilities of both discrete and continuous simulation approaches. Although discrete modeling tools have captured the development process in rich detail, they do not capture the effect of continuously varying factors. Systems dynamics models capture the dynamic behavior of continuous project variables including their interaction and feedback, but do not capture process steps easily. A modeling tool is needed that models the creation of artifacts with attributes, modifies those attributes based on system variables, and allows the system variables to vary continuously.

[Lakey 2003] $\boxed{C}$

"A hybrid software process simulation model for project management"

This paper introduces a hybrid software process simulation model that combines discrete event and system dynamics. The model supports software project estimation as well as project management. The model was originally constructed for a specific software organization and then been modified to make it more generic in nature. The model is fully functional but is intended primarily to offer a concrete theoretical framework for a hybrid approach to modeling specific software development projects. The theoretical basis for the model is documented in detail with a few examples of its behavior for varying parameters. The model description is followed by a general discussion of the following issues: utility of the modeling approach, adaptability of the theoretical framework to project-specific conditions, important modeling considerations, and other issues. The paper offers an approach that a model developer can use as the framework for developing a realistic project-specific model.

[Lehman 1996], [Lehman 1998], [Lehman, Ramil 1999], [Lehman, Ramil 2002], [Lehman, Ramil 2003], [Lehman et al. 2006] $\boxed{J}$ $\boxed{C}$ $\boxed{J}$ $\boxed{J}$ $\boxed{J}$ $\boxed{BC}$

Software Evolution and Feedback: Background, Theory, and Practice

This body of work is based on 30 years of study of software evolution phenomenon by Manny Lehman and colleagues. It suggests that a constraint on software process improvement arises from the fact that the global software process that includes technical, business, marketing, user and other activities constitutes a multiloop, multilevel feedback system. To change the characteristics of such a system requires one to consider, design, or adapt and tune both forward and feedback paths to achieve the desired changes in externally visible behavior. [Lehman 1998] reviewed the difficulty of achieving major improvement in the global software evolution process and introduced the FEAST hypothesis that the global software process is a complex feedback system, and that major improvement will not be achieved until the process is treated as such. Results from the FEAST project support this hypothesis. The article presents results of exploring the feedback phenomena using system dynamics and black-box techniques applied to actual industrial data. See [Wernick-Lehman 1999] for a description of the system dynamics model and results, [Chatters et al. 2000] for another FEAST model, and [Kahen et al. 2001] for a later model for FEAST. [Wernick, Hall 2002] shows a model of the combined causal mechanisms for growth in the size of software products over many releases incorporating previous work.

[Lehman, Ramil 2003] describes recent studies that have refined earlier conclusions, yielding practical guidelines for software evolution management and providing a basis for the formation of a theory of software evolution. Software that is regularly used for real-world problem solving or addressing a real-world application must be continually adapted and enhanced to maintain its fitness for an ever changing real world, its applications, and application domains. This adaptation and enhancement activities are termed progressive. As progressive activity is undertaken, the complexity (e.g., functional, structural) of the evolving system is likely to increase unless work, termed antiregressive, is also undertaken in order to control and

even reduce complexity. However, with progressive and antiregressive work naturally competing for the same pool of resources, management will benefit from means to estimate the amount of work and resources to be applied to each of the two types. The systems dynamics model in [Lehman et al. 2006] can serve as a basis of a tool to support decision making regarding the optimal personnel allocation over the system lifetime. The model is provided as an example of the use of process modeling in order to plan and manage long-term software evolution. See [Ramil et al. 2005] for more details of the antiregressive work model.

[Levary, Lin 1991] $\boxed{\text{J}}$

"Modeling the software development process using an expert simulation system having fuzzy logic"

A description of an intelligent computerized tool designed to aid managers of software development projects in planning, managing, and controlling the development process of medium-to-large-scale software projects is given. The expert system having fuzzy logic handles the fuzzy input variables to the system dynamics simulation model.

[Lin et al. 1992], [Lin et al. 1997] $\boxed{\text{TR}}$ $\boxed{\text{J}}$

"Software engineering process simulation model (SEPS)"

The Software-Engineering Process Simulation (SEPS) model developed at JPL is described. It uses system dynamics to simulate the dynamic interactions among software life-cycle development activities and management decision-making processes. This simulation model of the software project-development process is designed to be a planning tool to examine trade-offs of cost, schedule, and functionality, testing the implications of different managerial policies on a project's outcome.

[Lin, Levary 1989] $\boxed{\text{J}}$

"Computer-aided software development process design"

A computer-aided software development process design is described in this paper in the form of a computerized intelligent tool. This tool is designed to help managers in planning, managing, and controlling the development process of medium-to-large-scale software projects.

[Lin 1993] $\boxed{\text{J}}$

"Walking on battlefields: Tools for strategic software management"

This article addresses the need for software management innovation and the importance of tools based on system dynamics to add a new dimension to software project management. The dual life cycle is emphasized with a feedback structure of software engineering and management processes.

[Lo 1999] $\boxed{\text{A}}$

"Reuse and high level languages"

This graduate research study investigated the dynamic effects of reuse and fourth-generation languages in a rapid application development context for reducing schedule. Using different kinds of programming languages and levels of reuse affects the schedule and effort, so understanding their effects on the software process will benefit management planning strategy. There are four phases represented in the model: requirements, design, coding, and approval phases. See Chapter 5 where this model is summarized.

[Madachy 1994b], [Madachy 1996a] $\boxed{A}$ $\boxed{C}$
System Dynamics Modeling of an Inspection-Based Process
This Ph.D. research used a dynamic model of an inspection-based life cycle to support quantitative evaluation of the process. The model serves to examine the effects of inspection practices on cost, schedule, and quality throughout the life cycle. It uses system dynamics to model the interrelated flows of tasks, errors, and personnel throughout different development phases and is calibrated with industrial data. It extends previous software project dynamics research by examining an inspection-based process with an original model, and integrating it with the knowledge-based method for risk assessment and cost estimation. The model demonstrates the effects of performing inspections or not, the effectiveness of varied inspection policies, and the effects of other managerial policies. It was also modified for other types of peer reviews [Madachy, Tarbet 2000]. See the model in Chapter 5.

[Madachy 1995b] $\boxed{C}$
"System dynamics and COCOMO: Complementary modeling paradigms"
System dynamics and static models such as COCOMO rest on different assumptions, but the two perspectives can contribute to each other in a symbiotic and synergistic way. This paper compares and contrasts the underlying formulations, and demonstrates a dynamic model with relations to the static COCOMO. Static models were used to nominally calibrate the dynamic model, the dynamic model improved on static assumptions, and simulation results were used to generate a phase-sensitive cost driver for the static model.

[Madachy 1996b] $\boxed{C}$
"Tutorial: Process modeling with system dynamics"
This tutorial covered the modeling process with system dynamics with applications to software processes. Examples of software process infrastructures and working simulation models were demonstrated.

[Madachy 1996c] $\boxed{C}$
"Modeling software processes with system dynamics: Current developments"
This article describes the current developments in modeling software processes with system dynamics. It overviews different application areas in which system dynamics is being used and future research.

[Madachy, Tarbet 2000] $\boxed{J}$
"Case studies in software process modeling with system dynamics"
This describes the use of mostly small-scale models for investigating managerial process issues and supporting personnel training at Litton's Guidance and Control Systems (GCS) Division. At the project level, these include models for planning specific projects, studying Brooks' Law and hiring issues, an interactive earned value model, requirements volatility, and a detailed peer review model. The perspective of some of the models has been at a multiproject or departmental level, including domain learning, product-line reuse processes and resource contention among projects. Insights provided by the models supported decision making at different levels and helped galvanize process improvement efforts. The models helped managers understand the key factors in complex scenarios. The training applications added

spark to classes and improved overall learning for training of software managers and leads. Topics including earned value techniques, productivity estimation, requirements volatility effects, and extrapolation of project tracking indicators have been presented with simulation models. Some include "flight training" scenarios that the students interact with to practice project control, such as the use of earned value indicators for operational decision making.

[Madachy 1999] $\boxed{\text{A}}$

"CS599 Software Process Modeling Course Notes"

This course used early versions of portions of this book. The reference contains course slides, additional articles, links to models, and student work.

[Madachy, Boehm 2005] $\boxed{\text{BC}}$

"Software dependability applications in software process modeling"

Software process modeling can be used to reason about strategies for attaining software dependability. Dependability has many facets, and there is no single software dependability metric that fits all situations. A stakeholder value-based approach is useful for determining relevant dependability measures for different contexts. Analytical models and simulation techniques including continuous systems and discrete event modeling approaches can be applied to dependability, and the trade-offs of different approaches are discussed. Development process models mainly address software defect introduction and removal rates, whereas operational process models address the probability of various classes of failure. An overview of sample applications is presented, mostly using system dynamics. An elaborated example shows how modeling can be used to optimize a process for dependability.

[Madachy et al. 2006], [Madachy et al. 2007] $\boxed{\text{C}}$ $\boxed{\text{J}}$

Assessing Hybrid Incremental Processes for SISOS Development

New processes are being assessed to address modern challenges for software-intensive systems of systems (SISOS), such as coping with rapid change while simultaneously assuring high dependability. A hybrid agile and plan-driven process based on the spiral life cycle has been outlined to address these conflicting challenges with the need to rapidly field incremental capabilities. A system dynamics model has been developed to assess the incremental hybrid process and support project decision making. It estimates cost and schedule for multiple increments of a hybrid process that uses three specialized teams. It considers changes due to external volatility and feedback from user-driven change requests, and dynamically reestimates and allocates resources in response to the volatility. Deferral policies and team sizes can be experimented with, and it includes trade-off functions between cost and the timing of changes within and across increments, length of deferral delays, and others. An illustration of using the model to determine optimal agile team size to handle changes is shown. Both the hybrid process and simulation model are being evolved on a very large scale incremental SISOS project and other potential pilots. [Madachy et al. 2007] builds on the work in [Madachy et al. 2006] by casting the results in a value-based framework accounting for mission value, and increasing the option space of the case study example. This model is highlighted in Chapter 4.

[Martin, Raffo 2000], [Martin, Raffo 2001], [Martin 2002] $\boxed{J}$ $\boxed{J}$ $\boxed{A}$

Hybrid Modeling of the Software Development Process

The Ph.D. dissertation in [Martin 2002] used a hybrid system dynamics and discrete event model and applied it to a project. Whereas models offer a means to represent the process, they also impose restrictions on those aspects of the system that can be addressed. [Martin, Raffo 2000] examined the possibility of a combined discrete and continuous simulation model that removes some of these limitations. They developed a method that combines the features of two well-known examples: the Abdel-Hamid system dynamics model and a discrete-event model of a standardized example of modifying a unit of code. The model was subsequently applied to a software development project [Martin, Raffo 2001].

[Pfahl 1995] $\boxed{C}$

"Software quality measurement based on a quantitative project simulation model"

The author outlines the system dynamics modeling and simulation approach called PROSYD, which stands for project simulation with system dynamics. The author discusses the relationship between software process modeling according to the PROSYD approach and metrics definition in addition to the relationship between project simulation, quality measurement, and process improvement.

[Pfahl, Lebsanft 1999], [Pfahl 2001], [Pfahl et al. 2002], [Pfahl et al. 2003], [Pfahl, Ruhe 2005] $\boxed{J}$ $\boxed{J}$ $\boxed{J}$ $\boxed{C}$ $\boxed{BC}$

Integrated Measurement, Modeling, and Simulation (IMMoS)

Integrated measurement, modeling, and simulation (IMMoS) is an integrated approach for system dynamics modeling and software process measurement developed and validated by Pfahl and colleagues. The Ph.D. dissertation in [Pfahl 2001] describes the development, application, and initial validation of IMMoS. The goal of IMMoS is to support systematic development and usage of process simulation models. IMMoS combines system dynamics with static process modeling and goal-oriented measurement. The result is an integrated approach to simulation-based learning. It is based on lessons learned from industrial modeling and by combining system dynamics models with static models that contain qualitative or quantitative information. IMMoS combines system dynamics with descriptive process modeling and measurement-based quantitative modeling (which includes the goal–question–metrics method). Such a framework reduces the barriers to system dynamics modeling and places simulation under the umbrella of "goal-oriented" measurement and analysis. For example, it is seen how system dynamics may impose data collection requirements and how software metrics may be used to define reference modes for simulation models.

The hybrid approach integrates the individual strengths of its inherent methodological elements and concepts. First, it enhances existing guidance for SD modeling by adding a component that enforces goal orientation, and by providing a refined process model with detailed description of activities, entry/exit criteria, input/output products, and roles involved. Second, it describes how to combine system dynamics modeling with goal-oriented measurement and descriptive process modeling, thus improving efficiency and smoothly closing the gap to established methods in empirical software engineering. The effectiveness and efficiency of IM-

MoS is supported with empirical evidence from two industrial case studies and one controlled experiment. Also see [Angkasaputra, Pfahl 2004] about enhancing IMMoS for agile modeling.

[Pfahl, Birk 2000] $\boxed{\text{TR}}$

"Using simulation to visualize and analyze product–process dependencies in software development projects"

The core element of the PROFES improvement methodology is the concept of product–process dependency (PPD) models. The purpose of PPD models is to help focus process improvement activities on those development technologies and processes that are most effective with regard to achieving specific customer-defined product quality goals. This paper describes how system dynamics simulation models can be used to check the plausibility of achieving positive effects on software product quality when implementing improvement actions derived from PPD models. This is done through extending an existing generic software project simulation model with structures that represent expected local cause–effect mechanisms of the PPD models.

[Pfahl, Lebsanft 2000a] $\boxed{\text{J}}$

"Using simulation to analyze the impact of software requirement volatility on project performance"

This paper presents a simulation model that was developed by Fraunhofer IESE for Siemens Corporate Technology. The purpose of this simulation model was to demonstrate the impact of unstable software requirements on project duration and effort, and to analyze how much money should be invested in stabilizing software requirements in order to achieve optimal cost-effectiveness. The paper reports in detail on the various steps of model building, discusses all major design decisions taken, describes the structure of the final simulation model, and presents the most interesting simulation results of a case study.

[Pfahl, Lebsanft 2000b] $\boxed{\text{J}}$

"Knowledge acquisition and process guidance for building system dynamics simulation models"

In this paper, experience with system modeling of software processes and projects within Siemens is reported. Special focus is put on problems encountered during knowledge acquisition for system dynamics model building, like inadequate guidance while conducting system dynamics modeling projects and insufficient methodical support for reusing available or generating missing knowledge. Both problems were addressed in a research project, jointly conducted by Fraunhofer IESE and Siemens Corporate Technology. One of the results of this project is a prescriptive process model for building system dynamics models. This process model, which is briefly outlined in the paper, provides guidance for a systematic development of system dynamics models in software organizations.

[Pfahl et al. 2001] $\boxed{\text{J}}$

"A CBT module with integrated simulation component for software project management education and training"

This describes a computer-based training (CBT) module for student education in software project management. The single-learner CBT module can be run using

standard Web browsers, and the simulation component is implemented using system dynamics. The paper presents the design of the simulation model and the training scenario offered by the existing CBT module prototype. Possibilities for empirical validation of the effectiveness of the CBT module in university education are described, and future extensions of the CBT module toward collaborative learning environments are suggested (see [Pfahl et al. 2004a] for the follow-on work).

[Pfahl et al. 2004a] $\boxed{C}$

"PL-SIM: A generic simulation model for studying strategic SPI in the automotive industry"

This paper presents the approach, results, and conclusions of a simulation modeling activity conducted by Fraunhofer IESE with DaimlerChrysler AG to explore the level of process leadership (PL) that can be achieved by one of DaimlerChrysler's car divisions within the next five years. A crucial issue of the study was the identification and determination of influence factors and measures expected to have an impact on the level of PL. In order to help focus discussion and provide a quantitative basis for analyses, they jointly developed a system dynamics simulation model, PL-SIM. In a first assessment, the model was found to be useful for running experiments that qualitatively reflect the fundamental dynamics of the factors influencing PL. Though additional data needs to be collected in order to increase the model's accuracy and predictive power, its generic structure can easily be transferred into other companies and serve as a framework for exploring their specific SPI strategies.

[Pfahl et al. 2004b] $\boxed{J}$

"Evaluating the learning effectiveness of using simulations in software project management education: Results from a twice replicated experiment"

This paper presents the results of a twice replicated experiment that evaluates the learning effectiveness of using a process simulation model for educating computer science students in software project management (see [Pfahl et al. 2001] for a description of the model used in the experiments). While the experimental group applied a system dynamics simulation model, the control group used the COCOMO model as a predictive tool for project planning. The results of each empirical study indicate that students using the simulation model gain a better understanding about typical behavior patterns of software development projects. The combination of the results from the initial experiment and the two replications with meta-analysis techniques corroborates this finding. Additional analysis shows that the observed effect can mainly be attributed to the use of the simulation model in combination with a Web-based role-play scenario. This finding is strongly supported by information gathered from the debriefing questionnaires of subjects in the experimental group. They consistently rated the simulation-based role-play scenario as a very useful approach for learning about issues in software project management.

[Pfahl 2005] $\boxed{BC}$

"Software process simulation in support of risk assessment"

This chapter presents a five step simulation-based method to risk assessment, ProSim/RA, which combines software process simulation with stochastic simulation. Although the proposed method is not new, it is the first time that it was systematically described in detail based on an illustrative case example that served as a

model for similar scenarios. By applying cost functions to the risk probabilities generated by ProSim/RA, the potential losses to the delivered product value can be calculated.

[Pfahl et al. 2006a] $\boxed{C}$

"Simulation-based stability analysis for software release plans"

Release planning for incremental software development assigns features to releases such that most important technical, resource, risk, and budget constraints are met. The research in this paper is based on a three-staged procedure. In addition to an existing method for (i) strategic release planning that maps requirements to subsequent releases and (ii) a more fine-grained planning that defines resource allocations for each individual release, the authors propose a third step, (iii) stability analysis, which analyzes proposed release plans with regard to their sensitivity to unforeseen changes. Unforeseen changes can relate to alterations in expected personnel availability and productivity, feature-specific task size (measured in terms of effort), and degree of task dependency (measured in terms of workload that can only be processed if corresponding work in predecessor tasks has been completed). The focus of this paper is on stability analysis of proposed release plans. They present the simulation model REPSIM (Release Plan Simulator) and illustrate its usefulness for stability analysis with the help of a case example.

[Pfahl et al. 2006b] $\boxed{BC}$

"Software process simulation with system dynamics—A tool for learning and decision support"

This chapter provides an overview of some issues in learning and decision support within the scope of software process management. More specifically, the existing work done in the field of software process simulation is presented, and the application of software process simulation models for the purpose of management learning and decision support is motivated and described. Examples of simulation modeling applications in the automotive industry are reported to illustrate how process simulation can become a useful management tool for the exploration and selection of alternatives during project planning, project performance, and process improvement. It concludes with a discussion of limitations and risks, and proposes future work that needs to be done in order to increase acceptance and dissemination of software process simulation in the software industry.

[Plekhanova 1999] $\boxed{O}$

"A capability-based approach to software process modelling"

This unpublished paper presents a capability-based approach to stochastic software process modeling represented by a project schedule simulation, which is considered a combination of discrete event simulation and system dynamics modeling. The feedback path can be the reworking of a task or any sequence of project activities (i.e., step, phase, period, or stage). Since the number of loops is a random variable relevant to project resource capabilities, an integration of process simulation and system dynamics approach is used for stochastic modeling. A capability-based approach provides detailed process analyses, resource utilization, and how resources (people or team) fit a project not only at each discrete point in time, but also for each feedback path. Monitoring of performance feedback provides the determination of

an optimal schedule. This ensures a simulation of a sequence of a random number of optimal schedules, which represents the evolution of the schedule, with feedback effect over time.

[Powell et al. 1999] $\boxed{\text{J}}$

"Strategies for lifecycle concurrency and iteration: A system dynamics approach"

This paper documents the work in progress of a detailed model for investigating life-cycle concurrency and iteration to improve process performance. Based on the incremental life-cycle practices at Rolls Royce plc, the process is modeled that accounts for simultaneous, concurrent development between projects/phases, where time to market and business performance are key drivers. It also models the staged delivery approach and serves to evaluate combined strategies of concurrence and iteration to achieve improved cycle-time performance, increased product quality, and potentially lower overall development costs. Also see [Powell 2001].

[Powell 2001] $\boxed{\text{A}}$

"Right on Time: Measuring, Modelling and Managing Time-Constrained Software Development"

This Ph.D. dissertation describes a technique to aid the management of time-constrained development with system dynamics modeling. The need to compress development timescales influences both the software process and the way it is managed. Conventional approaches to measurement, modeling, and management treat the development process as being linear, sequential, and static; but the processes used to achieve timescale compression in industry are iterative, concurrent, and dynamic. The Process Engineering Language (PEL) captures important elements of a process that are obscured by conventional approaches to data collection. These fine-grained measures are essential to observations of industrial process behavior within Rolls Royce and BAE Systems, and the isolation of three specific problems of time-constrained development: overloading, bow-waves, and the multiplier effect. In response, a new modeling technique, called Capacity-Based Scheduling (CBS), is proposed to control risk across a portfolio of time-constrained projects. Finally, industrial data from Rolls Royce is used to evaluate the validity and utility of the modelling approach and propose new strategies for the management of time-constrained software development.

[Raffo, Setamanit 2005] $\boxed{\text{C}}$

"A simulation model for global software development projects"

Global software development is becoming a dominant paradigm in the software industry. Conducting development projects in multiple countries offers many potential benefits including significantly reduced cost and better response times. However, it also poses some difficulties and challenges in managing this kind of project. Software process simulation modeling can be used to support, enrich, and evaluate global development theories, to facilitate understanding, and to support corporate decisions. The discrete-event paradigm and system dynamic paradigm compliment each other and together enable the construction of models that capture both the dynamic nature of project variables and the complex sequences of discrete activities that take place. The authors argue that the ideal model for representing global projects would have to effectively support both system dynamics equations and discrete-event logic.

[Ramesh, Abdel-Hamid 2003] $\boxed{\text{BC}}$

"Integrating genetic algorithms with system dynamics to optimize quality assurance effort allocation"

This chapter describes the integrated use of genetic programming and system dynamics modeling. Genetic programming is a machine learning technique that uses an evolutionary optimization algorithm. It imitates genetic algorithms, which use mutation and replication to produce algorithms that represent the "survival of the fittest." An application is shown for optimizing quality assurance allocation.

[Ramil et al. 2005] $\boxed{\text{BC}}$

"Simulation process modelling for managing software evolution"

Software that is regularly used for real-world problem solving or addressing a real-world application must be continually adapted and enhanced to maintain its fitness in an ever changing real world, its applications, and application domains. This adaptation and enhancement activities are termed *progressive.* As progressive activity is undertaken, the complexity (e.g., functional, structural) of the evolving system is likely to increase unless work, termed *antiregressive,* is also undertaken in order to control and even reduce complexity. However, with progressive and antiregressive work naturally competing for the same pool of resources, management will benefit from means to estimate the amount of work and resources to be applied to each of the two types. After providing a necessary background, this chapter describes a systems dynamics model that can serve as a basis of a tool to support decision making regarding the optimal personnel allocation over the system lifetime. The model is provided as an example of the use of process modeling in order to plan and manage long-term software evolution (see [Lehman 1996] etc.). This model is highlighted in Chapter 5.

[Rodriguez et al. 2006] $\boxed{\text{J}}$

"E-Learning in project management using simulation models: a case study based on the replication of an experiment"

Current e-learning systems are increasing in importance in higher education, but e-learning applications do not achieve the level of interactivity that current learning theories advocate. This paper discusses the enhancement of e-learning systems based on replicated experiments with system dynamics models described in [Pfahl et al. 2004b].

[Roehling, Collofello 2000] $\boxed{\text{J}}$

"System dynamics modeling applied to software outsourcing decision support"

To illustrate some of the dynamics, potential benefits, and potential drawbacks of software outsourcing, this paper describes a simulation model software practitioners can leverage for useful insight and decision support. Practical benefits and a rationale for applying the model to the software outsourcing problem are provided. The model's applicability and usefulness is demonstrated with snapshots of simulation results, which are analyzed and discussed.

[Rothman 1996] $\boxed{\text{C}}$

"Applying systems thinking to the issues of software product development"

This paper discusses issues of software product development, such as meeting the schedule, implementing desired functionality, and removing a sufficient number of

defects. A systems thinking perspective is stressed in choosing the product's goal, planning the project, being aware of changing conditions, and changing the goals that will cause the schedule to slip.

[Rubin et al. 1994] $\boxed{J}$

"With the SEI as my copilot: Using software process flight simulation to predict the impact of improvements in process maturity"

In this article, four distinct stages in the model construction life cycle of a process flight simulator are given. These are: establishing the scope of the simulation, constructing a model of the process being explored, turning the model into executable form by specifying the underlying mathematical measures and relationships, and turning the model prototype into a flight simulator by defining the key aspects of the "cockpit" or interface.

[Rubin et al. 1995] $\boxed{J}$

"Software process flight simulation: Dynamic modeling tools and metrics"

Managers can get help in understanding the dynamics of new processes and technologies before they are introduced with process flight simulation tools. The authors show how these exploratory tools can be used for information systems (IS) project management, simple modeling of the software process, and modeling the dynamics of the IS organization in transition.

[Ruiz et al. 2001] $\boxed{J}$

"A simplified model of software project dynamics"

In this work, several dynamic models were developed in order to simulate the behavior of software projects. From the comparison made between one of the best known empirical estimation models and dynamic estimation models, they analyzed the existing problems in dynamic models in order to make dynamic estimations at the early stages of software projects, when little information is available. The results are obtained from a reduced dynamic model developed to estimate and analyze the behavior of projects in the early phases, in which there is not much information regarding the project. The modeling approach followed to obtain this simplified model has been determined by the simplification of Abdel-Hamid and Madnick's model using the works of Eberlein about understanding and simplification of models.

[Ruiz et al. 2002], [Ruiz et al. 2004] $\boxed{C}$ $\boxed{J}$

An integrated framework for simulation-based software process improvement

These papers present a doubly integrated dynamic framework for CMM-based software process improvement that combines traditional static models with dynamic simulation. The aim is to support a qualitative and quantitative assessment for software process improvement and decision making to achieve a higher software development process capability according to the CMM. First, it is based on the systematic integration of dynamic modules to build dynamic models simulate each maturity level proposed in the reference model. As a consequence, a hierarchical set of dynamic models is developed following the same hierarchy of levels suggested in the CMM. Second, the dynamic models of the framework are integrated with different static techniques commonly used in planning, control, and process evaluation. The papers describe the reasons found to follow this approach, the integration process of models and techniques, and the implementation of the framework, and show an ex-

ample of how it can be used in a software process improvement concerning the cost of software quality.

[Rus et al. 1999] $\boxed{J}$

"Software process simulation for reliability management"

This paper describes the use of a process simulator to support software project planning and management. The modeling approach here focuses on software reliability, but is just as applicable to other software quality factors, as well as to cost and schedule factors. The process simulator was developed as a part of a decision support system in [Rus 1998] for assisting project managers in planning or tailoring the software development process, in a quality driven manner. The original simulator was developed using the system dynamics approach. As the model evolved by applying it to a real software development project, a need arose to incorporate the concepts of discrete event modeling. The system dynamics model and discrete event models each have unique characteristics that make them more applicable in specific situations. The continuous model can be used for project planning and for predicting the effect of management and reliability engineering decisions. It can also be used as a training tool for project managers. The discrete event implementation is more detailed and therefore more applicable to project tracking and control. In this paper, the structure of the system dynamics model is presented. The use of the discrete event model to construct a software reliability prediction model for an Army project, Crusader, is described in detail.

[Rus, Collofello 1998] $\boxed{C}$

"Software process simulation for reliability strategy assessment"

This paper describes the use of a process simulator for evaluating different quality-driven software reliability strategies. As part of a decision support system for project managers, different strategies can be compared via simulation for a project to choose from and tailor. The system also includes a rule-based fuzzy logic expert system component for suggesting alternative processes. Product and project factors that affect software reliability are modeled with other related project parameters. The effect of different software reliability practices impacting different factors is shown, as well as the impact on cost and schedule. This led to the dissertation in [Rus 1998].

[Rus 1998] $\boxed{A}$

"Modeling the Impact on Cost and Schedule of Software Quality Engineering Practices"

This Ph.D. dissertation investigated how to design the process for developing software with given quality requirements, in order to achieve product requirements while maintaining the balance between software quality, project cost, and delivery time. The solution proposed is to develop a framework for customizing a project by selecting software reliability engineering practices for achieving specified software reliability requirements, and modeling the resulting process in order to assess, by simulation, the impact of these practices on project cost, schedule, and software quality throughout the development life cycle. The framework is implemented by a decision support system (DSS) for software reliability engineering strategy selection and assessment. This dissertation describes the framework and presents the methodology used for developing it, as well as framework implementation and the results

of using the fuzzy expert system for strategy selection and the system dynamics process simulator for practices effect evaluation.

[Smith et al. 1993] J

"Death of a software manager: How to avoid career suicide through dynamic process modeling"

Software managers commit career suicide each year because they do not alter their mental models that often stifle consideration of alternatives until they are exposed or fail dramatically. To improve software development both at an individual and corporate level, the authors feel that our mental models must expand to include new experiences and alternatives, and we need a way to collect and share experiences.

[Stallinger 2000] J

"Software process simulation to support ISO/IEC 15504 based software process improvement"

A generalized system dynamics model of *"a set of improving software processes"* was developed to support SPI action planning at a tactical level. The basic intention is to determine the impact of a set of scheduled improvement actions on the strategic target variables of SPI (time to market, cost, quality, etc.). The approach integrates the two main "mental models" behind ISO/IEC 15504: the process model described textually as a network of processes and work products, and the model of maturing single processes describing the evolution of process capability through a series of process attributes. The development of the model is oriented toward organizations with lower capability and medium-sized development projects. The results of preceding software process assessments are used as a major source for model initialization. The feasibility of the approach is demonstrated by providing results from prototype applications to improvement projects, and insights and lessons learned on how to build such a model are described.

[Stallinger, Gruenbacher 2001] J

"System dynamics modelling and simulation of collaborative requirements engineering"

Requirements engineering is a success-critical activity in the software development process. Within this context, this paper presents selected aspects of the system dynamics modeling and simulation of the EasyWinWin requirements negotiation methodology. EasyWinWin approaches requirements definition as a learning rather than a gathering activity and aims at fostering stakeholder cooperation and involvement. It is based on the WinWin requirements negotiation model and provides a set of collaborative techniques and tools to enable stakeholder interaction. The major goal behind the modeling and simulation effort is to assess the issues associated with the social and behavioral aspects of the EasyWinWin process and to explore how these issues affect the overall outcome of the process. This paper introduces the EasyWinWin process, presents the simulation model, and provides details and results of the requirements elicitation component of the model. It discusses model validation and calibration, and summarizes the conclusions and insights obtained.

[Sycamore 1995] A

"Improving Software Project Management through System Dynamics Modeling"

This M.S. dissertation introduces a tool that gives accurate foresight into the dynamics of a system based upon intuitive managerial decisions for an incremental devel-

opment process. The relationships among various components of a software system are described through equations that represent causal influences rather than statistical correlation.

[Taweponsomkiat 1996] $\boxed{A}$

"Report for re-engineering of concurrent incremental software development model"

A description of the process of reengineering a concurrent incremental software development model implemented in Ithink to the Extend simulation language is given. A rationale for the need for reengineering and a description of this process is also provided.

[Tvedt, Collofello 1995] $\boxed{C}$

"Evaluating the effectiveness of process improvements on software development cycle time via system dynamics modeling"

The objective is to provide decision makers with a model that will enable the prediction of the impact a set of process improvements will have on their cycle time. A description is given of the initial results of developing such a model and applying it to assess the impact of software inspections. This work pre-dated the dissertation in [Tvedt 1996].

[Tvedt 1995] $\boxed{A}$

"A system dynamics model of the software inspection process"

The author describes the modeling of software inspections used in predicting their impact on software development cycle time. Along with an overview of the software inspection process, a software inspection model is presented in this paper. Also see [Tvedt 1996].

[Tvedt 1996] $\boxed{A}$

"A Modular Model for Predicting the Impact of Process Improvements on Software Development Time"

This Ph.D. dissertation describes a model for concurrent incremental software development and applies it to assess the impact of software inspections. Reducing software development cycle time without sacrificing quality is crucial to the continued success of most software development organizations. Software companies are investing time and money in reengineering processes incorporating improvements aimed at reducing their cycle time. Unfortunately, the impact of process improvements on the cycle time of complex software processes is not well understood. The objective of this research was to provide a model that will enable researchers and software project managers to evaluate the impact a set of process improvements will have on their software development cycle time. The model enables researchers and software project managers to gain insight and perform controlled experiments to answer "what if" type questions, such as, "What kind of cycle time reduction can I expect to see if I implement inspections?" or "How much time should I spend on inspections?"

[Twaites et al. 2006] $\boxed{C}$

"Modeling inspections to evaluate prioritization as a method to mitigate the effects of accelerated schedules"

Inspections have long been accepted as one of the most efficient ways of detecting many types of product defects. Given sufficient time, trained reviewers are quite good at detecting defects. Unfortunately, there is often insufficient time in which to

perform a rigorous review. This paper uses a system dynamics model to assess the effect of prioritizing the items under inspection when presented with an accelerated schedule.

[Waeselynck, Pfahl 1994] $\boxed{J}$

"System dynamics applied to the modelling of software projects"

A summary is presented of the major results of a study performed to review worldwide activities in the field of system dynamics modeling and simulation applied to software development projects. The five potential uses of this method for software development projects are the following: research, training, policy investigation, postmortem analysis, and monitoring ongoing projects.

[Wakeland et al. 2004] $\boxed{J}$

"Using design of experiments, sensitivity analysis, and hybrid simulation to evaluate changes to a software development process: A case study"

Hybrid simulation models combine the high-level project issues of system dynamics models with the process detail of discrete event simulation models. Hybrid models not only capture the best of both of these simulation paradigms, but they also are able to address issues the other simulation paradigms are only able to address alone. This article describes a structured approach that applies design of experiments (DOE) and broad-range sensitivity analysis (BRSA) to a hybrid system dynamics and discrete event simulation model of a software development process. DOE is used to analyze the interaction effects. The sensitivity of the model to parameter changes over a broad range of plausible values is used to analyze the nonlinear aspects of the model. The end result is a deeper insight into the conditions under which the process change will succeed, and improved recommendations for process change design and implementation. In this particular study, significant interactions and nonlinearities were revealed, supporting the hypothesis that consideration of these complex effects is essential for insightful interpretation of model results and effective decision making.

[Wernick, Lehman 1999] $\boxed{J}$

"Software process dynamic modeling for FEAST/1"

This article presents a high-level system dynamics model of software evolution developed for the FEAST project (see also [Lehman 1996] etc. on software evolution). The model investigates the role and effect of feedback in the global software process. It tests the FEAST hypothesis that the dynamics of real-world software evolution processes leads to a degree of process autonomy. It is shown that the model behavior closely reflects data from an actual defense project that involved successive releases with strict deadlines, and supports the contention that external feedback significantly influences progress. It demonstrates that a simple top-down model can reproduce real-world trends, and that some aspects of the process can be ignored when considering long-term software product evolution.

[Wernick, Hall 2002] $\boxed{J}$

"Simulating global software evolution processes by combining simple models: An initial study"

A number of studies of the long-term evolution of commercial software products over many releases have established a consistent pattern for growth in the size of the systems examined. This shows a trend toward a progressive slowdown in growth

over time. The work presented here is a step in developing a simulation of the combined effects of the causes of this trend. A number of simple system dynamics simulations, each capturing the causal structure and behavior of one of a number of causes are presented. These models are then combined into a single simulation model reflecting the effect of a combination of these causes. Possible causes considered here are the reduction over time in both the conceptual space, which can be searched for new uses of a system and the design spaces constraining developers' ability to modify their systems; the effects of the degradation of the structure of a software system as is it is evolved in unanticipated directions; and the reducing coverage of developers' knowledge of the structure of the system. The combined model incorporates all of these causal mechanisms and introduces a relative weighting for each. The long-term aim is to produce a simulation embodying those possible causes which are found to be supported by actual evidence.

[Williford, Chang 1999] $\boxed{\text{J}}$

"Modeling the FedEx IT division: A system dynamics approach to strategic IT planning"

This paper describes a model for long-term IT strategic planning at Federal Express. The macro-level model predicts staffing, training, and infrastructure needs over a five-year period. It was developed using actual metrics, expert judgement, and business predictions. Strategic factors that could be adjusted included process improvement programs, investment in tools and training, recruiting, and architecture strategies. The submodels include balancing of permanent employees with contract workers, and included staff retraining associated with the transition from mainframe to distributed systems. An infrastructure rollout model estimated server purchases to support computing workloads as they migrated, and incorporated Moore's Law to predict changing hardware costs. The model outputs were judged to be reasonable by participating management, and served to strengthen senior management's commitment to process improvement and, particularly, human resource strategies.

Table B-1. Special journals with ProSim and SPW/ProSim papers

International Workshop on Software Process Simulation and Modeling (ProSim) Year and Location	Journal Issue with Revised Papers
ProSim 1998 Portland, OR, USA	*Journal of Systems and Software,* Vol. 46, Issue 2-3, Elsevier, 1999
ProSim 1999 Portland, OR, USA	*Software Process Improvement and Practice,* Vol. 5, Issue 2-3, John Wiley & Sons, Ltd., 2000
ProSim 2000 London, England	*Journal of Systems and Software,* Vol. 59, Issue 3, Elsevier, 2001
ProSim 2001 Portland, OR, USA	*Software Process Improvement and Practice,* Vol. 7, Issue 3-4, John Wiley & Sons, Ltd., 2002
ProSim 2003 Portland, OR, USA	*Software Process Improvement and Practice,* Vol. 9, Issue 2, John Wiley & Sons, Ltd., 2004
ProSim 2004 Edinburgh, Scotland	*Software Process Improvement and Practice,* Vol. 10, Issue 3, John Wiley & Sons, Ltd., 2005
ProSim 2005 St. Louis, MO, USA	*Software Process Improvement and Practice,* Vol. 11, Issue 4, John Wiley & Sons, Ltd., 2006
SPW/ProSim 2006 Shanghai, China (first joint conference with International Software Process Workshop)	*Software Process Improvement and Practice,* Vol. 12, Issue 5, John Wiley & Sons, Ltd., 2007 (to be published)

Table B-2. Additional special journals on system dynamics for software processes

Journal Issue
American Programmer, Yourdon E (ed.), Cutter Information Group, New York, May 1993