

Utilization of Process Modeling and Simulation in Understanding the Effects of Requirements Volatility in Software Development

Susan Ferreira
*Industrial Engineering
Dept.
Arizona State University
ferreira@fastq.com*

James Collofello
*Computer Science and
Engineering Dept.
Arizona State University
collofello@asu.edu*

Dan Shunk
*Industrial Engineering
Dept.
Arizona State University
dan.shunk@asu.edu*

Gerald Mackulak
*Industrial Engineering Dept.
Arizona State University
mackulak@asu.edu*

Philip Wolfe
*Industrial Engineering Dept.
Arizona State University
wolfe@asu.edu*

Abstract

Requirements volatility is a common software project risk that can have severe consequences resulting in cost and schedule overruns and, at times, cancelled projects. This paper introduces an executable system dynamics simulation model developed to help project managers comprehend the effects of requirements volatility. Requirements volatility and its effects were studied using various analysis and modeling techniques. The requirements volatility study results were used to design major simulator components that were integrated into a previously developed and validated simulator, 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.

1.1 Requirements Volatility

Requirements volatility refers to growth or changes in requirements during a project's development lifecycle. There are multiple aliases commonly associated with or related to the phenomenon of requirements volatility. These terms include requirements change, requirements creep, scope creep,

requirements instability, and requirements churn among others. Costello [3] provides a relatively detailed set of metrics for requirements volatility. Other simple metrics for requirements volatility define it as the number of additions, deletions, and modifications made to the requirements set per time unit of interest (per week, month, phase, etc.). Requirements volatility, in its various forms, surfaces as a frequent and high impact risk in numerous empirical studies performed to identify risk factors or to understand variables leading to a project's success or failure (examples include [26], [7], [24], [25], [12], [19], [28], [9], [2], and [4]).

Requirements changes can occur at multiple points during the development process [14]. These changes can take place "while the requirements are being elicited, analyzed and validated and after the system has gone into service". Past philosophy dictated that requirements had to be firm by the completion of the requirements phase and that requirements should not change after this time. This view is now understood to be unrealistic [23]. Kotonya and Sommerville [14] discuss that requirements change is unavoidable. They also indicate that requirements changes do not necessarily imply that poor requirements engineering practice was utilized as requirements changes could be the result of a combination of factors.

1.2 When Is Requirements Volatility a Problem?

Volatility during the requirements phase is expected because this is when requirements are being created.

Concern for the effects of requirements volatility is not usually associated with the front end of the process, during requirements definition. However, once the design process begins, the impact of requirements change is progressively greater due to the investment in time and effort as the project continues to generate artifacts and complete required tasks.

1.3 Simulator Introduction

The effects of requirements volatility have been discussed in the literature for some time. However, little empirical research has been carried out on the topic of requirements volatility until recently that considered the factors involved and quantitative effects of requirements volatility on factors that are related to key project management indicators (cost, schedule, and quality). A relatively small number of studies consider requirements volatility and its associated effects, especially in a manner integrated with other software project management factors. These existing studies primarily fall into a few major research method categories: survey or software assessment based research ([32], [15], [10], [9]), interviews [31], regression analysis [29], and simulation models ([22], [7], [18], [17], [27], [16]). Finnie et al. [6] used an analytic hierarchy process analysis to identify that requirements volatility effects cost and duration of projects. Nidumolu [20] discusses dimensions of requirements uncertainty.

The existing simulation models discussed in the literature were developed for one organization or have a limited view of requirements volatility or requirements engineering, or do not include the requirements engineering process in concert with the rest of the lifecycle or other critical project factors. The literature also indicated that very sparse process modeling and simulation work had been performed in requirements engineering, an area recently receiving more focused attention because of the impact that it has on the rest of the systems and software engineering lifecycle. The term “requirements engineering” refers to the processes required to generate and maintain the software requirements throughout the duration of the project.

The limited research and relative importance of requirements volatility as a risk and the relatively sparse level of requirements engineering process modeling led to more analysis and examination of these areas. Further research study then led to developing a system dynamics process model simulator stochastically based on survey data results. The software project management simulator (SPMS)

illustrates a software business model that considers the effects of requirements volatility on a software project’s key management parameters: cost, schedule, and quality. The simulator presents a more comprehensive and detailed view of the researched areas than previous models.

1.4 Simulator Development

A rigorous review of the requirements engineering and requirements volatility related literature was performed. The intent was to uncover any relevant requirements volatility effects and process related knowledge and understand the research area’s current software business models. Various process and information models were created to represent and assist in analysis and synthesis of the knowledge gained during the literature review.

Requirements engineering process models and workflows, an information model, and a causal model were developed prior to simulator development. A basic information model (IDEF1X) covering part of the model’s context was developed to explore a set of data relationships in the requirements and change request area so that these areas could be more fully understood prior to modeling. The identification of relevant factor relationships was developed into a causal model based on a fusion of literature review material and discussions with software engineering experts. Figure 1 presents this causal model.

A use case and associated user scenarios to aid in focusing the model development work and to later provide to others as a training vehicle were also developed. These scenarios also helped to identify additional elements to add to the model and provided a check of parameter coverage. Concepts and constructs from each of the analyses and models contributed to the creation of the simulator’s workflows and other model sectors.

As the initial simulator sector designs were generated, walkthroughs were held with a limited set of research committee members. Once an initial version of the model containing the key constructs was completed, a secondary walkthrough of the model was held with four reviewers outside of the research committee. These included representatives from industry and academia that were currently performing research in requirements engineering and/or software process modeling and simulation. Following the model walkthroughs, the simulator was modified to incorporate reviewer comments and suggestions and was then ready to include quantitative data.

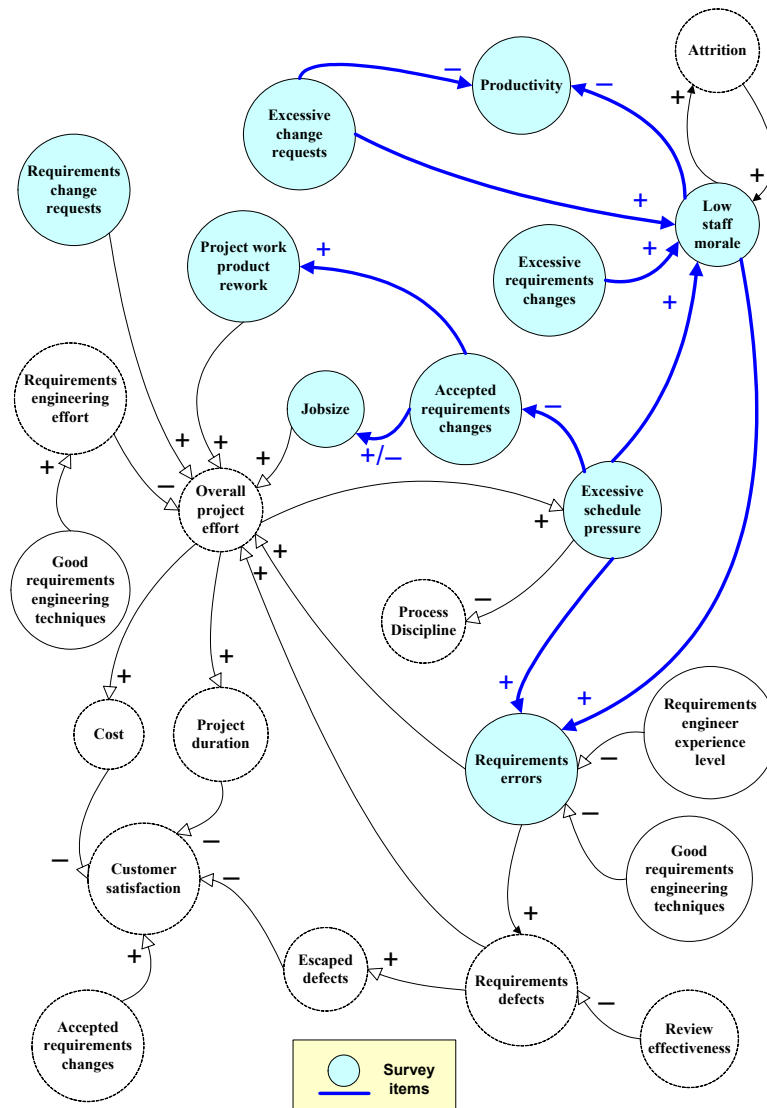


Figure 1. Causal Model

Many of the model variables required data that was not available in the literature or through other means. In order to populate these model parameters, a survey was developed and performed. The survey explored only a subset of the previous causal model relationships and factors. This subset of the causal model relationships and factors were chosen because the survey needed to be limited in length to allow key factor data to be collected or these areas were already modeled in the base simulator.

The Project Management Institute Information Systems Specific Interest Group (PMI-ISSIG) sponsored the survey by providing the host site for the web-based survey and sending notifications about the survey to its members. The survey was online from August 23 through September 28, 2001. While PMI-

ISSIG was the primary target population for the survey, one mailing was sent to individual Software Engineering Institute (SEI) Software Process Improvement Network (SPIN) group contacts within the United States and to individual professional contacts. 312 software project managers and other software development personnel submitted responses for the survey. 87% of survey respondents were project managers or leaders for their projects.

The survey showed that 78% of the respondents experienced some level of requirements volatility on their project. The survey showed that requirements volatility can increase the job size dramatically, cause major rework, and affect other project variables such as morale, schedule pressure, productivity, and requirements error generation.

The summary results showing how requirements change affected the job size as a percent of the original job size are compiled in the histogram shown in Figure 2. In the vast majority of cases, requirements volatility increased the product size. Survey respondents had an

average of 32.4% requirements volatility related job size increases. Figure 3 shows the individual net requirements volatility averages over the ten equally spaced intervals in which the data was collected.

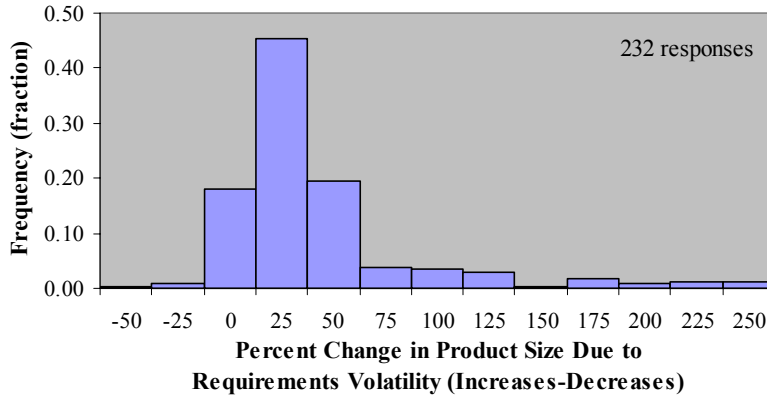


Figure 2. Distribution of Requirements Volatility Percent Change in Product Size

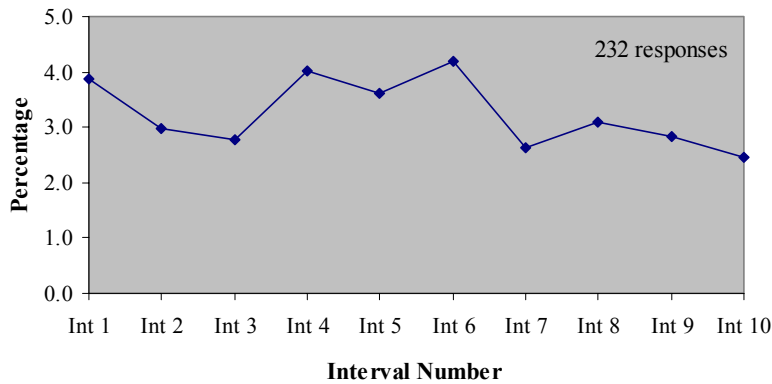


Figure 3. Requirements Volatility Job Size Change Averages, Per Time Interval

Other captured volatility effects included significant increases in project rework and reduced team morale. For those reporting low morale, a majority of respondents reported that low morale had detrimental effects on increasing requirements error generation and decreasing requirements engineering productivity. The survey also captured effects from schedule pressure. As the resource effort increases due to requirements volatility (to address job size additions and rework), with no other changes, schedule pressure also increases. The survey data showed increases in requirements error generation as the schedule pressure increases. More details on the survey findings showing primary and secondary effects of requirements

volatility are addressed in Ferreira [5]. These effects cause consequences leading to impacts on key project management indicators such as cost, schedule, and quality.

Statistical analysis of the survey responses generated stochastic distributions for many of the requirements volatility effects and requirements engineering factors in the model. The model's stochastic inputs are primarily generated using either empirical discrete distributions or are generated using the inverse transform method. The selection of the type of distribution depended on the survey results. The random variates use a random number as an input to generate the desired distribution. Based on the sampling frequency, the distributions are sampled once

per run or are sampled continuously throughout a run to be drawn as necessary.

1.5 Simulation Model

SPMS illustrates researched effects of requirements volatility and includes requirements engineering extensions to a base software project management model. This research simulation model evolved from Houston's SPARS model [7]. The SPARS model also represents an adaptation, as it reuses or modifies large portions from Abdel-Hamid and Madnick's model [1] and Tvedt's model [30] and then extends the consolidated model to incorporate effects of a number of risk factors. The SPARS model was selected due to its comprehensive software project management scope and updates for more modern development practices. Reusing components from SPARS facilitated the development of the model in that common constructs did not need to be recreated and the incorporated components came from previous and validated models. As part of the research effort, the SPARS model was modified to eliminate some unnecessary factors and extended to create the research model.

Major additions to the model include the research results for the effects of requirements volatility and significant extensions to add and support the requirements engineering portions of the software development lifecycle. SPMS encompasses the requirements engineering through test phases of the software development lifecycle. Requirements volatility starts and is stochastically simulated during the project's development and test phases. The model workflows also cover the entry of change requests,

change request analysis and review activities, and their disposition (accepted, rejected/deferred).

SPMS demonstrates causal model effects of requirements volatility. The survey findings showing the impact of requirements volatility on increasing software project job size (majority of the time), increased rework, and lowered staff morale are represented in the model using stochastic relationships derived during survey data analysis. Over 50 new parameters that used distributions, derived from the survey data, were added to the model. In addition to these survey drawn distributions, a significant number of distributions were reused from other sources, or parameters were modeled using single point inputs that could be changed by a user. The effects of lowered morale on requirements engineering productivity and requirements error generation are represented in the model. Schedule pressure effects on requirements error generation were also studied as part of the survey data analysis and were added to pre-existing schedule pressure effects in the model. Among other survey data used in the simulator, the model includes requirements defect detection effectiveness for various software development activities or milestones and relative work rates of requirements volatility related activities compared to their normal work rate. Other changes to the model include the addition of a requirements engineering staff type and requirements engineering support activities in addition to development and test personnel and activities.

Table 1 presents a view of the model's classification, according to the characterization framework from Kellner et al.[13].

Table 1. Model Classification

Purpose	Planning, Understanding, Process Improvement
Scope	Medium to large size project, short to long duration, one product/project team
Input/Output Variables	Key input variables include change requests, change request acceptance rates, requirements volatility rates, staffing profiles for requirements engineers, developers and testers, productivity rates, requirements engineering and software development quality assurance effectiveness, and software development policy decisions. Output variables include those illustrating effort (cost), duration (schedule), and defects (quality).
Model Approach	Continuous, mixed mode (stochastic and deterministic variables), iThink™ simulation tool

The typical model audience is expected to be a software development project manager or researcher seeking to gain an understanding of requirements engineering and requirements volatility and its effects integrated with other software project risks. The model is relatively complex, given its purpose, and assumes a sophisticated user that is educated on the use of

simulation models and software development project management. The model is practically based, relying on a significant and proven foundation of software project management and simulation research.

The model is segmented into twenty (20) sectors. Table 2 provides an overview of a small subset of the model sectors with a brief description of each in order

to give the reader an introduction to the model's scope. An example of one model sector is shown in Figure 4. The view illustrates the requirements engineering work flow sector. The connections on the right of the sector flow into or out of the development and test work flow sector. The requirements work flow sector encompasses a normal product work flow. Product with errors is removed and reworked before flowing into the development and test activities. Requirements defects caught in later phases, post the requirements phase come back into the sector to be reworked. The

reworked product then flows back into the development and test activities. Additions in job size due to volatility and underestimation flow into the normal work flow through the course of the project. The lower half of the sector includes the requirements change rework flows and contain separate requirements error related activities. Rework due to requirements volatility is drawn from the development and test work flows and is worked through the requirements change rework work flow.

Table 2. Model Sector Descriptions

Sector Name	Description
Change Request Work Flow	Stochastic change request entry over ten intervals. Incoming change request (CR) analysis, change request control board (CCB) review, and disposition.
Requirements Work Flow	Requirements generation and review process including requirements error and defect rework. Stochastic entry of requirements volatility related project scope changes and rework over ten intervals.
Development and Test Work Flow	Work product flow through the development lifecycle, from design and code through testing and rework of design and code errors. Work is pulled from development and test activities for requirements volatility related rework and reduction and for rework of requirements defects.
Planned Staffing	Entry and summation of requirements engineering, developer, and tester planned staffing profile information.
Requirements Quality Management	Generation and detection of requirements errors and defects.
Development and Test Quality Management	Generation and detection of design and code errors and defects.
Actual Staffing	Entry and exit of staff based on planned staffing profiles, assimilation of new staff, attrition, and replacements. Entry of contract personnel and organizationally experienced personnel handled separately for the different staff groups.

1.6 Simulator Results

In order to show simulator results, the model was populated with a SPARS base case and the results were compared to assess if differences existed. This comparison was performed because the SPMS model modified and extended the SPARS model, so the results for a model run without the risk factors actualized should be very similar. The SPARS base case produced 64 thousand lines of code (KLOC) in 383 days with a cost of 3606 person-days. SPARS was calibrated for an average full-time equivalent staff of 10 people that were scheduled for 348 days. SPMS includes the requirements engineering phase. Therefore, in order to more equitably compare results for the two models, the effort for requirements engineering was removed from the total results. In addition, many of the input factors, for example, requirements review effectiveness, were modeled

stochastically, using the survey data. So the SPMS base case needs to be represented stochastically. However, the stochastic set of results don't allow for an exact comparison with a single base point identified from SPARS. Central values representing the stochastic results were selected to perform the data comparisons. Median values were used for comparative purposes instead of the average values because the distribution of responses is positively skewed and the median is a better gauge of the data's center in this case. For the same size product and staffing, the SPMS base case had median values of 3870 days for project duration and 384 person-days for effort. The cost (effort) is within 7% of SPARS while the duration value has a very small difference (1 day).

There are a number of reasons that can account for the differences in results. SPARS uses size-units in KLOC which were directly converted from the previous Abdel-Hamid and Madnick model, which used delivered source instruction (DSI) for its size-units. The SPMS simulator uses functions points.

days. Box plots were used to represent the data because the simulation results were positively skewed given the tendency for higher cost and duration with the actualization of the requirements volatility risk. The box plots provide a graphical display of the center and the variation of the data, allowing one to see the symmetry of the data set [21]. The baseline results show a small level of variability because some of the model parameters (e.g. requirements engineering process factors) were modeled stochastically. Therefore, even when the requirements volatility risk is not actualized, some variability in results will appear. The box plots for both parameters, cost and duration, illustrating the survey data for the risk of requirements volatility, have a wide span. The wide span in outcomes for both the cost and duration box plots indicates a very large potential for unpredictable results as well as considerable impact on project results.

In addition to simulating the effects of requirements volatility, SPMS offers the ability to simulate various project scenario combinations starting with the requirements engineering portion of the lifecycle. The model offers a robust set of parameters that capture various facets of the project including requirements errors and defects, requirements defect density, defect containment effectiveness for various milestones, and other factors integrated with a very comprehensive development and test set of factors and relationships. Each of the model distributions can be easily calibrated and tailored to a specific organization's environment. All the other factors are also setup to be readily modifiable to reflect an organization's historical data.

1.7 Summary and Conclusions

The model discussed here was developed to assist in evaluating the impact of requirements volatility. This simulator can help an interested user to better understand the requirements engineering process and the impact of requirements volatility via its process-oriented workflows and comprehensive scope. This work expands our understanding of the requirements engineering process and the effects of requirements volatility because of the rigorous research to both understand and leverage the previous foundation of knowledge in requirements engineering and requirements volatility and the thorough nature of the survey and analysis of this valuable data to assess factor relationships and populate the simulator with empirical data.

Valuable quantitative data was captured and was used to populate the simulation model. Survey data used in the simulator captured information on factors and relationships not previously available from a wide population in the software industry and allowed modeling variables stochastically due to the quantity of responses. The collection, analysis, and usage of results represent a valuable commodity. Many of the model variables were modeled stochastically in order to allow the user to assess project outcomes in a probabilistic sense.

Software project managers can use this tool to better understand the effects of requirements volatility and this risk in concert with other common risks. The simulator can be used to assess potential courses of action to address the risk of requirements volatility.

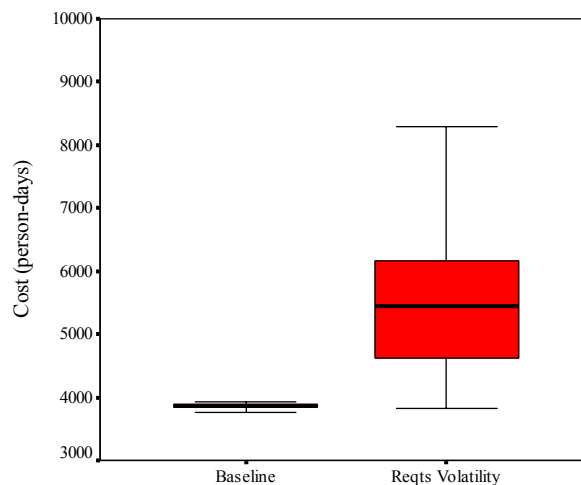


Figure 5. Project Cost Box Plots

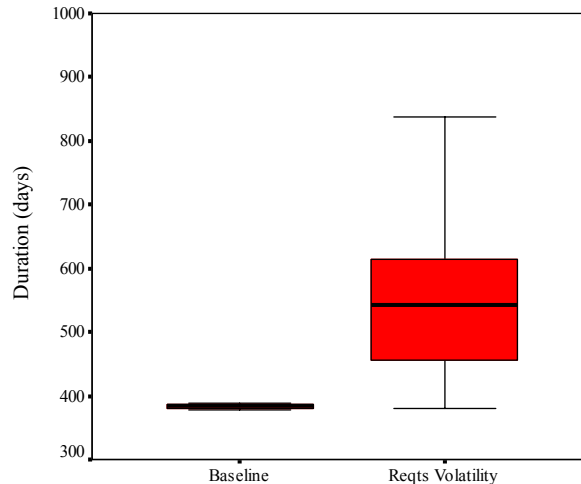


Figure 6. Project Duration Box Plots

1.8 Future Research

Additional research in the areas of both the impact of requirements volatility and requirements engineering, as it continues to evolve, is expected to provide a continuous stream of new ideas, perspectives, and information that can allow for the development of richer models or that can represent different facets of understanding in these under-represented yet critical research areas. More work needs to be done to model the impact of requirements engineering related processes and policies so that project managers and software development personnel are more aware of the impact of the decisions they make during this phase and how the rest of the lifecycle may be effected by their choices. This work can lead to the further identification of best practices and generate insights valuable to managing software development projects in the future.

Although not expressly discussed relative to the simulator in this paper, additional research captured during the survey discusses survey findings that show significant correlations between process factors and requirements volatility that could be used by software development managers to proactively address and attempt to mitigate the risk of requirements volatility [5]. These process factors could be modeled as policy changes showing resulting effects on requirements volatility levels entering as model inputs.

As this and other simulators that incorporate requirements engineering processes and relationships continue to evolve, additional experimentation with the models may prove valuable in identifying common factors and relationships. The research model presented in this paper is relatively large and complex.

Sensitivity analysis can assist in the identification of influential factors in this and other models. The results of further experimentation may be used to “prune” insignificant factors from the model so that a more efficient model can result [8]. One of the benefits of a reduced model includes a shortened training and learning ramp-up time due to a simpler model. Core concepts that make the most difference can be emphasized. Since the model is less complex it becomes easier to understand and perhaps more popular in its use due to the reduced time to populate the model with organizational and project specific data. Maintenance time may also be reduced because the model is smaller and it is easier to find problems.

Additional work is planned to further validate the current model. Data for multiple projects that experienced requirements volatility will be used to verify the model replicates project experience and expectations. This additional validation work may also assist in identifying other constructs to add to the model in future iterations.

1.9 References

- [1] Abdel-Hamid, Tarek and Stuart Madnick. 1991. *Software Project Dynamics: An Integrated Approach*. Englewood Cliffs, New Jersey: Prentice-Hall.
- [2] Boehm, Barry W. 1991. Software Risk Management: Principles and Practices. *IEEE Software* 8, 1 (January): 32-41.
- [3] Costello, Rita Jean. 1994. *Metrics for Requirements Engineering*. Master of Science Thesis, California State University, Long Beach.
- [4] Curtis, Bill, Herb Krasner, and Neil Iscoe. 1988. A Field Study of the Software Design Process for Large

- Systems. *Communications of the ACM* 31, 11 (November): 1268-1287.
- [5] Ferreira, Susan. 2002. Measuring the Effects of Requirements Volatility on Software Development Projects, Ph.D. Dissertation, Arizona State University.
- [6] Finnie, Gavin R. 1993. Prioritizing Software Development Productivity Factors Using the Analytic Hierarchy Process. *Journal of Systems and Software* 22, 2 (August): 129-139.
- [7] Houston, Dan X. 2000. *A Software Project Simulation Model for Risk Management*, Ph.D. Dissertation, Arizona State University.
- [8] Houston, D.X., Ferreira, S., Collofello, J.S., Montgomery, D.C., Mackulak, G.T., Shunk, D.L. 2001. Behavioral Characterization: Finding and Using the Influential Factors in Software Process Simulation Models. *Journal of Systems and Software* 59, 3 (December): 259-270.
- [9] Jones, Capers. 1994. *Assessment and Control of Software Risks*. Englewood Cliffs, New Jersey: PTR Prentice Hall, Inc.
- [10] Jones, Capers. 1998. *Estimating Software Costs*. New York, New York: McGraw Hill.
- [11] Jones, Capers. 2001. Conflict and Litigation Between Software Clients and Developers, Version 10.
- [12] Käsälä, Kari. 1997. Integrating Risk Assessment with Cost Estimation. *IEEE Software* 14, 3 (May/June): 61-67.
- [13] Kellner, Mark I., Ray Madachy, and David M. Raffo. 1999. Software Process Modeling: Why? What? How?. *Journal of Systems and Software*. 46, 2-3 (April): 91-105.
- [14] Kotonya, Gerald and Ian Sommerville. 1998. *Requirements Engineering: Processes and Techniques*, John Wiley & Sons, Ltd.
- [15] Lane, Michael S. 1998. Enhancing Software Development Productivity in Australian Firms. *Proceedings of the 9th Australasian Conference on Information Systems (ACIS '98)*, Volume 1. 337-349. September 29-October 2, 1998. Sydney, Australia.
- [16] Lin Chi Y. and Reuven R. Levary. 1989. Computer Aided Software Development Process Design. *IEEE Transactions on Software Engineering* 15, 9 (September): 1025-1037.
- [17] Lin, C.Y., T. Abdel-Hamid, and J.S. Sherif. 1997. Software-Engineering Process Simulation Model (SEPS). *Journal of Systems and Software*. 38, 3 (September): 263-277.
- [18] Madachy, Ray and Denton Tarbet. 2000. Initial Experiences in Software Process Modeling. *Software Quality Professional*. 2, 3 (June): 1-13. (Obtained from http://sqp.asq.org/vol2_issue3/sqp_v2i3_toc.html).
- [19] Moynihan, Tony. 1997. How Experience Project Managers Assess Risk. *IEEE Software* 14, 3 (May/June): 35-41.
- [20] Nidumolu, Sarma R. 1996. Standardization, Requirements Uncertainty and Software Project Performance. *Information and Management*. 31, 3 (December): 135-150.
- [21] Ott, Lyman. 1988. *An Introduction to Statistical Methods and Data Analysis*. PWS-KENT Publishing Company.
- [22] Pfahl, D. and K. Lebsanft. 2000. Using Simulation to Analyze the Impact of Software Requirements Volatility on Project Performance. *Information and Software Technology*. 42, 14 (November): 1001-1008.
- [23] Reifer, Donald J. 2000. Requirements Management: The Search for Nirvana. *IEEE Software* 17, 3 (May/June): 45-47.
- [24] Ropponen, Janne. 1999. Risk Assessment and Management Practices in Software Development. Chapter 8 in *Beyond the IT Productivity Paradox*. John Wiley & Sons. 247-266.
- [25] Ropponen, Janne and Kalle Lyytinen. 2000. Components of Software Development Risk: How to Address Them? A Project Manager Survey. *IEEE Transactions on Software Engineering*. 26, 2 (February): 98-112.
- [26] Schmidt, Roy, Kalle Lyytinen, Mark Keil, and Paul Culle. 2001. Identifying Software Project Risks: An International Delphi Study. *Journal of Management Information Systems* 17, 4 (Spring): 5-36.
- [27] Smith, Bradley J., Nghia Nguyen, and Richard F. Vidale. 1993. Death of a Software Manager: How to Avoid Career Suicide Through Dynamic Software Process Modeling. *American Programmer* May: 10-17.
- [28] The Standish Group. 1995. The Chaos Report. (Obtained from <http://www.standishgroup.com/chaos.html>).
- [29] Stark, George E., Paul Oman, Alan Skillicorn, and Alan Ameele. 1999. An Examination of the Effects of Requirements Changes on Software Maintenance Releases. *Journal of Software Maintenance: Research and Practice*. 11, 5 (September/October): 293-309.
- [30] Tvedt, John D. 1996. An Extensible Model for Evaluating the Impact of Process Improvements on Software Development Cycle Time. Ph.D. Dissertation, Arizona State University.
- [31] Zowghi, D. and Nurmiliani. 1998. Investigating requirements volatility during software development: Research in progress. *Proceeding of the 3rd Australian Conference on Requirements Engineering (ACRE98)*, Geelong, Australia. 38-48.
- [32] Zowghi, D., Ray Offen, and Nurmiliani. 2000. The Impact of Requirements Volatility on the Software Development Lifecycle. *Proceedings of the International Conference on Software Theory and Practice (IFIP World Computer Congress)*, Beijing, China, August 2000.