The Journal of Systems and Software 82 (2009) 1568–1577

Contents lists available at ScienceDirect

The Journal of Systems and Software

journal homepage: www.elsevier.com/locate/jss

Understanding the effects of requirements volatility in software engineering by using analytical modeling and software process simulation

Susan Ferreira [a,*], James Collofello [b], Dan Shunk [c], Gerald Mackulak [c]

[a] Industrial and Manufacturing Systems Engineering Department, The University of Texas at Arlington, Arlington, TX 76019, USA
[b] Computer Science and Engineering Department, Arizona State University, Tempe, AZ 76019, USA
[c] Industrial Engineering Department, Arizona State University, Tempe, AZ 76019, USA

ARTICLE INFO

Article history:
Available online 19 March 2009

Keywords:
Requirements volatility
Software process modeling
Requirements engineering risk

ABSTRACT

This paper introduces an executable system dynamics simulation model developed to help project managers comprehend the complex impacts related to requirements volatility on a software development project. The simulator extends previous research and adds research results from an empirical survey, including over 50 new parameters derived from the associated survey data, to a base model. The paper discusses detailed results from two cases that show significant cost, schedule, and quality impacts as a result of requirements volatility. The simulator can be used as an effective tool to demonstrate the complex set of factor relationships and effects related to requirements volatility.

1. Requirements volatility introduction

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 (1994) 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 Boehm, 1991; Curtis et al., 1988; Houston, 2000; Jones, 1994; Känsälä, 1997; Moynihan, 1997; Ropponen, 1999; Ropponen and Lyytinen, 2000; Schmidt et al., 2001; The Standish Group, 1995; Tirwana and Keil, 2006).

Changes to a set of requirements can occur at multiple points during the development process (Kotonya and Sommerville, 1998). 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 (Reifer, 2000). Kotonya and Sommerville (1998) 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. The term "requirements engineering" refers to the processes required to generate and maintain the software requirements throughout the duration of the project.

Concern for the effects of requirements volatility is not usually associated with the front end of the process, for example, during requirements definition. Volatility during the requirements definition phase is expected because this is when requirements are being created. However, once the design process begins, the impact of requirements change is progressively greater due to the additional investment in time and effort as the project continues to generate artifacts and complete required tasks. Additions or modifications may need to be made to previously generated or in process project artifacts and additional time investment or scrapped effort can result. Due to the additional unplanned effort, severe consequences can potentially occur, including significant cost and schedule overruns, and at times, cancelled projects. The impact of changing requirements during later phases of a project and approaches for assessing the impacts of these changes has been well documented (Yau et al., 1978, 1986, 1988; Yau and Kishimoto, 1987; Yau and Liu, 1988). In an agile software development environment, changes are welcomed throughout the development process (Beck et al., 2001). The target of this article is not agile type projects but more traditional development type projects.

* Corresponding author. Tel.: +1 817 272 1332; fax: +1 817 272 3406.
E-mail addresses: ferreira@uta.edu (S. Ferreira), collofello@asu.edu (J. Collofello), dan.shunk@asu.edu (D. Shunk), mackulak@asu.edu (G. Mackulak).

0164-1212/$ - see front matter © 2009 Elsevier Inc. All rights reserved.
doi:10.1016/j.jss.2009.03.014

2. The need for a requirements volatility assessment tool

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 that considered the factors involved and the integrated quantitative effects of requirements volatility on factors 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 (Jones, 1998, 1994; Lane, 1998; Nidumolu, 1996; Zowghi et al., 2000; Zowghi and Nurmuliani, 2002), interviews and case studies (Javed et al., 2004; Loconsole and Börstler, 2005, 2007; Nurmuliani et al., 2004; Zowghi and Nurmuliani, 1998), regression analysis (Stark et al., 1999), reliability growth model (Malaiya and Denton, 1999), analytic hierarchy process analysis (Finnie et al., 1993), and simulation models (Houston, 2000; Lin and Levary, 1989; Lin et al., 1997; Madachy et al., 2007; Madachy and Tarbet, 2000; Pfahl and Lebsanft, 2000; Smith et al., 1993).

The existing simulation models discussed in the literature were developed and tailored for one organization, have a limited view of requirements volatility or requirements engineering, or do not include requirements engineering processes considered in concert with the rest of the lifecycle or other critical project factors. A paucity of the literature exists on process modeling and simulation work performed in requirements engineering, an area now receiving more focused attention because of the impact that it has on the rest of the systems and software engineering lifecycle.

The limited research and relative importance of requirements volatility as a risk and the relatively sparse level of requirements engineering process modeling led the researchers to more analysis and examination of these areas. A system dynamics process model simulator, the Software Project Management Simulator (SPMS) that includes data which is stochastically based on industry survey data distributions, was then developed as part of a doctoral dissertation (Ferreira, 2002). 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. SPMS presents a more comprehensive and detailed view of the researched areas than previous models. The development of the SPMS simulator and associated results developed in this paper are discussed in this journal article.

3. Requirements volatility tool development process

This section of the paper briefly discusses the research method used to develop the simulation model. Key research questions addressed in the initial research study include: (1) Which software factors are affected by requirements volatility? (2) How can these factors and the uncertainty associated to these factors be modeled? and (3) What is the project management impact of requirements volatility? Fig. 1 provides a summary view of the processes used during the research effort. Starting with the figure's top left and top right sides and flowing down, the figure illustrates that two efforts (one per figure side) were initiated concurrently and these efforts flowed into the development of the software process simulator discussed in this paper.

A rigorous review of the requirements engineering and requirements volatility related literature was performed. 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 the simulator development. Relevant factors and associated relationships were identified based on analysis of the literature review material and discussions with software engineering experts. Further analysis of the captured information led to the

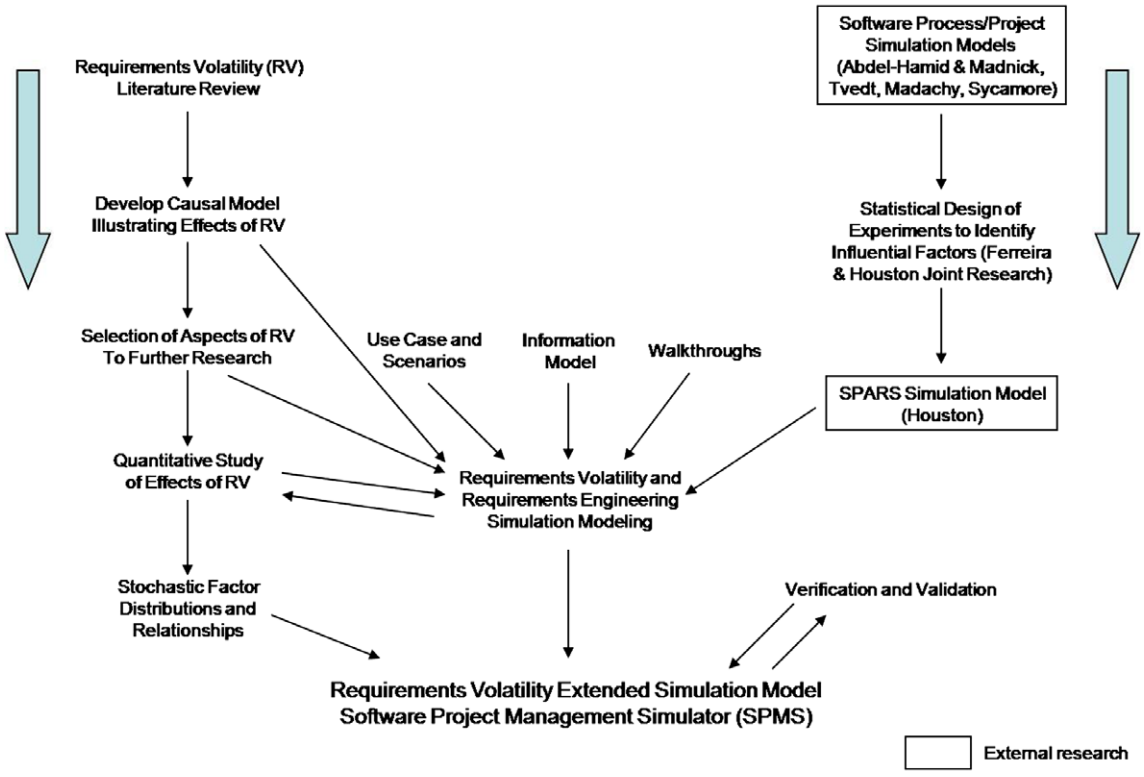

Fig. 1. Research method.

development of a causal model (Fig. 2). The causal model is important because it illustrates the cause and effect relationships between software development factors related to requirements volatility and includes hypothesized relationships between factors. The causal model was iteratively developed based on fundamental factor relationships (for example, job size and overall project effort), researcher industry and academic experience, and hypothesized relationships. The figure highlights (blue shading) factors and associated relationships (blue lines) that were further explored during the research effort. This causal model is expected to evolve over time as more relationships are explored and understood.

One of the proposed solutions to address the identified research questions was to develop a software process simulator. A simulator was selected because it provides a tool for software project managers and researchers to perform "what if" analyses, and enables users to examine the risk of various levels of requirements volatility and determine project outcomes. A simulator can represent the complexity of relationships between large quantities of interrelated factors and effectively illustrates the effects and impact of requirements volatility. A simulator with a graphical icon format was chosen to represent project factors and relationships. The previously developed causal model was used to develop the simulator. A subset of the previously generated causal model relationships and factors were selected and modeled. A subset of the causal model relationships and factors was chosen because follow-on work needed to be limited in length to allow key factor data to be collected or these areas were already modeled in the pre-existing simulator that was extended to create the new simulator, SPMS. Concepts and constructs from all of the generated process and information models contributed to the creation of the simulator's workflows and other model sectors.

Joint research was performed with Daniel Houston to characterize four deterministic software process simulators using statistical design of experiments (Houston et al., 2001). Results of this experimentation work fed into Dan Houston's development of the Software Process Actualized Risk Simulator (SPARS) (Houston, 2000). The SPMS research simulation model evolved from Houston's SPARS model. The SPARS model also represents an adaptation, as it reuses or modifies large portions from Abdel-Hamid and Madnick's model (1991) and Tvedt's model (1996) and then extends the consolidated model to incorporate effects of a number of risk factors. The SPARS model was selected because of its comprehensive software project management scope and updates for more modern development practices. Reusing components from SPARS facilitated the development of the SPMS model in that common constructs did not need to be recreated and the model used previously validated simulator components. As part of the research effort, the SPARS model was modified to eliminate some unnecessary factors and extended to create the research model, SPMS.

As the initial simulator sector designs were generated, walkthroughs were conducted with an initial set of individuals who were familiar with the research. 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 group. These reviewers 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. The model was then ready to include quantitative data.

Many of the model variables required data that was not available in the literature. In order to populate these model parameters, a survey was developed and administered to collect the needed data. The Project Management Institute's 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. Although 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. Three hundred twelve software project managers and other software development personnel submitted responses for the survey.

Survey results indicated that 78% of the respondents experienced some level of requirements volatility on their project. The survey findings highlight that requirements volatility can increase the job size dramatically, extend the project duration, cause major rework, and affect other project variables such as morale, schedule pressure, productivity, and requirements error generation. Fig. 3 shows an example of requirements volatility related data from the survey. The histogram in the figure depicts how requirements change affected the job size as a percent of the original job size. In the vast majority of cases, requirements volatility increased the job size. However, one can see that there were also cases where there was no change or a net decrease in the job size. Survey respondents had an average of 32.4% requirements volatility related job size increases. Other captured volatility effects included significant increases in project rework and reduced team morale. The survey also captured effects from schedule pressure. As the resource effort increases due to requirements volatility (to address job size additions and rework), schedule pressure also increases. The survey data showed increases in requirements error generation as the schedule pressure increases. These effects cause consequences leading to impacts on key project management indicators such as cost, schedule, and quality. More details on the survey findings showing primary and secondary effects of requirements volatility are addressed in Ferreira (2002).

Statistical analysis of the survey responses allowed the generation of stochastic distributions for many of the simulation model's requirements volatility and requirements engineering factors and relationships. The model's stochastic inputs are primarily generated using either empirical discrete distributions derived from analyzing histograms of the data or are generated using the inverse transform method (Abramowitz and Stegun, 1964). The selection of the type of distribution depended on the survey analysis results. Using these stochastic inputs, the simulation model's 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.

Once the derived stochastic factor distributions and relationships were added to the simulator, verification and validation (V&V) exercises were performed. Model validation determines whether a model is a "useful or reasonable representation of the system" (Pritsker et al., 1997). Verification checks that the simulation model runs as intended. Matko et al. (1992) indicate that modeling work is not an exact science since the real system is never completely known. Given this situation, the validation and verification effort primarily focused on building confidence in the model as a reasonable representation of the system and in its usefulness in the provision of results. The overall approach for verification and validation included tests of the structure and behavior of the model. This strategy follows a framework of guidelines presented by Richardson and Pugh (1981) that builds confidence in the model and its results. The tests focus on suitability and consistency checks. The model verification and validation activities were performed by the model developer, and various software process and project experts.

Additional verification work was performed in order to understand differences between the SPMS and SPARS model when the

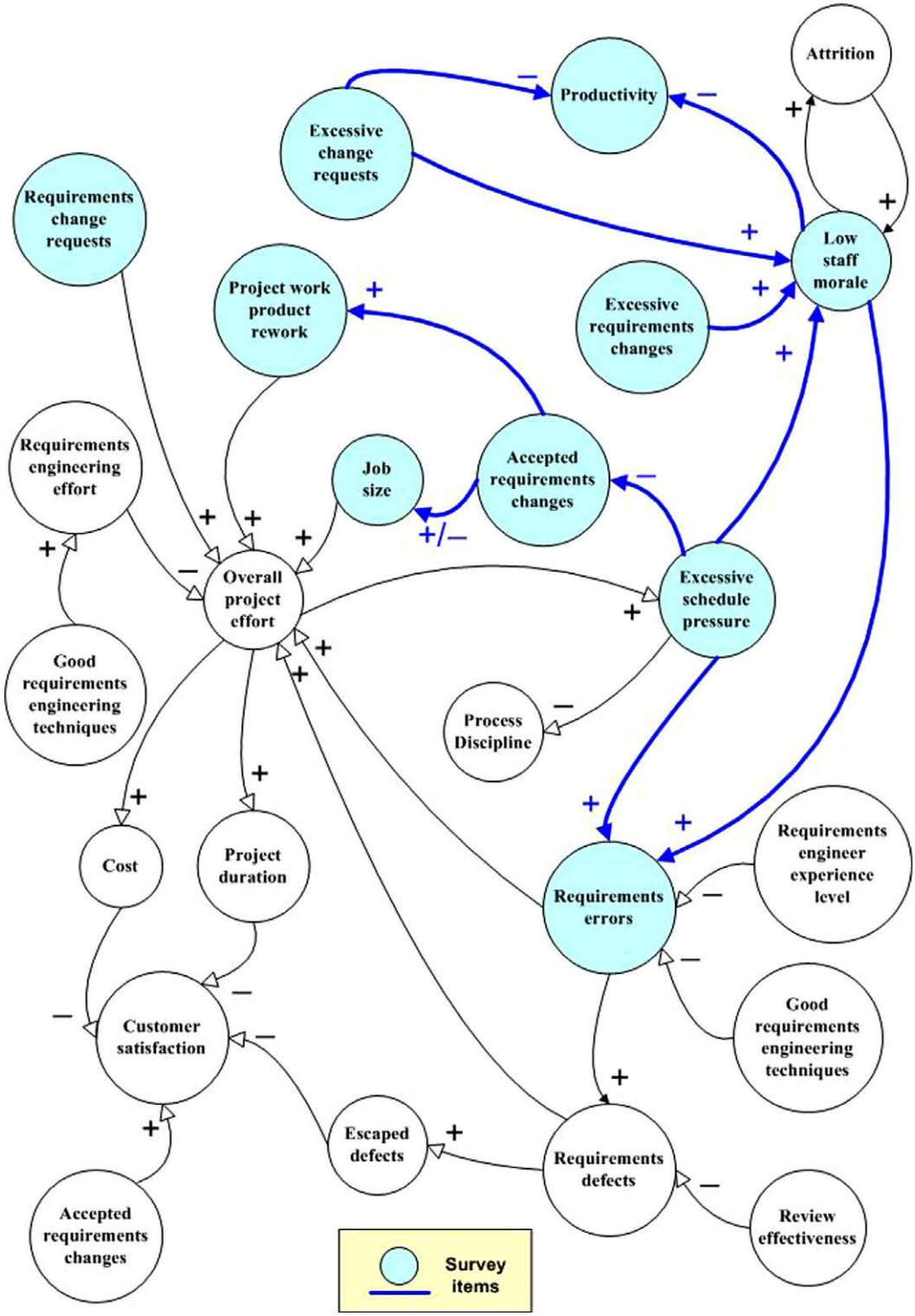

Fig. 2. Causal model.

risk factors are not actualized. Differences between the models in this case were relatively small. More information about this verification work is discussed in Ferreira et al. (2003).

4. Assessment tool capability and use

SPMS illustrates researched effects of requirements volatility and includes requirements engineering extensions to the SPARS software project management model. The SPARS model was modified to eliminate some unnecessary factors and extended to create the research model. Major additions to the base model include the researched 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

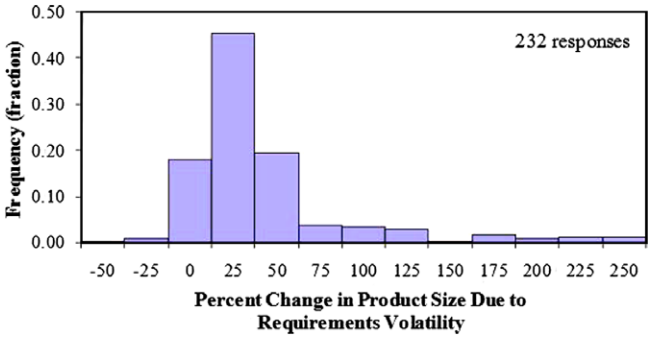

Fig. 3. Distribution of requirements volatility related percent change in job size.

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. Survey findings showing the impact of requirements volatility on increasing software project job size (this occurs a majority of the time), increased rework, and lowered staff morale are represented in the model using stochastic relationships derived during the 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 model contributions include the addition of a requirements engineering staff type and requirements engineering support activities which were added to pre-existing simulator 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. (1999). The typical model audience is expected to be software development project managers or researchers 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

Table 1
Model classification.

Purpose	Planning, understanding, process improvement
Scope	Medium to large size project, short to long duration, one product/project team
Model approach	Continuous, mixed mode (stochastic and deterministic variables), iThink™ simulation tool

Table 2
Model sector descriptions.

Sector name	Description
Change request work flow	Stochastic change request entry over 10 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 10 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
Requirements change additions	A support sector that calculates the amount of product to add to the product cycle for requirements volatility additions based on survey data
Requirements change rework	A support sector that calculates the amount of product to pull from the development and test work for requirements volatility related rework based on survey data
Requirements change reductions	A support sector that calculates the amount of product to pull from development and testing work flow activities for requirements related reductions. Includes policy choice that defines where to remove product
Planned staffing	Entry and summation of requirements engineering, developer, and tester planned staffing profile information
Requirements engineer effort allocation	Allocation of requirements engineer effort to requirements engineering activities based on activity priority
Developer and tester effort allocation	Allocation of developer and tester effort to project activities based on activity priority
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
Attrition and replacement	Attrition (including attrition due to low morale) and replacement calculations for requirements engineers and the grouped set of developers and testers
Planning	Calculation of requirements engineer and the grouped developer and tester work force levels needed based on willingness to hire
Control	Calculation of effort perceived still needed to complete project, effort shortages, schedule pressure, and assumed requirements engineer, developer, tester productivity. Calculations to determine start of development and testing
Adjustment of job effort	Adjustment of job effort based on requirements volatility related additions, rework, and reductions as well as from underestimation
Productivity inputs	Productivity inputs and calculation of project activity productivity based on productivity multipliers
Productivity	Work rate modification due to effort remaining, effort required, and staff exhaustion. Generation of two staff type productivity multipliers (including learning, communication overhead, staff experience, and morale)
Progress measure	Calculation of requirements engineering and development and testing progress. Modification of currently perceived job size based on requirements volatility and discovered work due to underestimation
Senior management commitment	Adjustment of staffing and schedule multipliers based on senior management commitment. This is the only sector that was not modified from the original SPARS model

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 20 sectors. The sectors are organized into convenient and logical groupings of related factors. Table 2 provides an overview 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 sector excerpted from the model is shown in Fig. 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 (sector not shown).

The requirements work flow sector encompasses a normal product work flow. Requirements are generated and reviewed during the normal product work flow. The normal requirements work flow begins at the initiation of the requirements engineering phase, at the To_Be_Worked stock. The To_Be_Worked stock is initially populated with the estimated starting job size. The work then flows through the Generated_Reqts, Reqts_Awaiting_Review, Reviewed_Reqts stocks, and then into the development process as an inflow. The requirements generation activities are aggregated, encompassing requirements elicitation, analysis, negotiation, and initial requirements management. Once reviewed, the requirements product is dispositioned and defective product containing requirements errors is removed from the normal work flow and becomes part of the requirements error rework work flow, to be reworked before flowing into the development and test activities. Requirements defects caught post the requirements phase come back into the requirements defect rework work flow to be reworked. The reworked product then flows back into the development and test activities. Additions to job size due to volatility

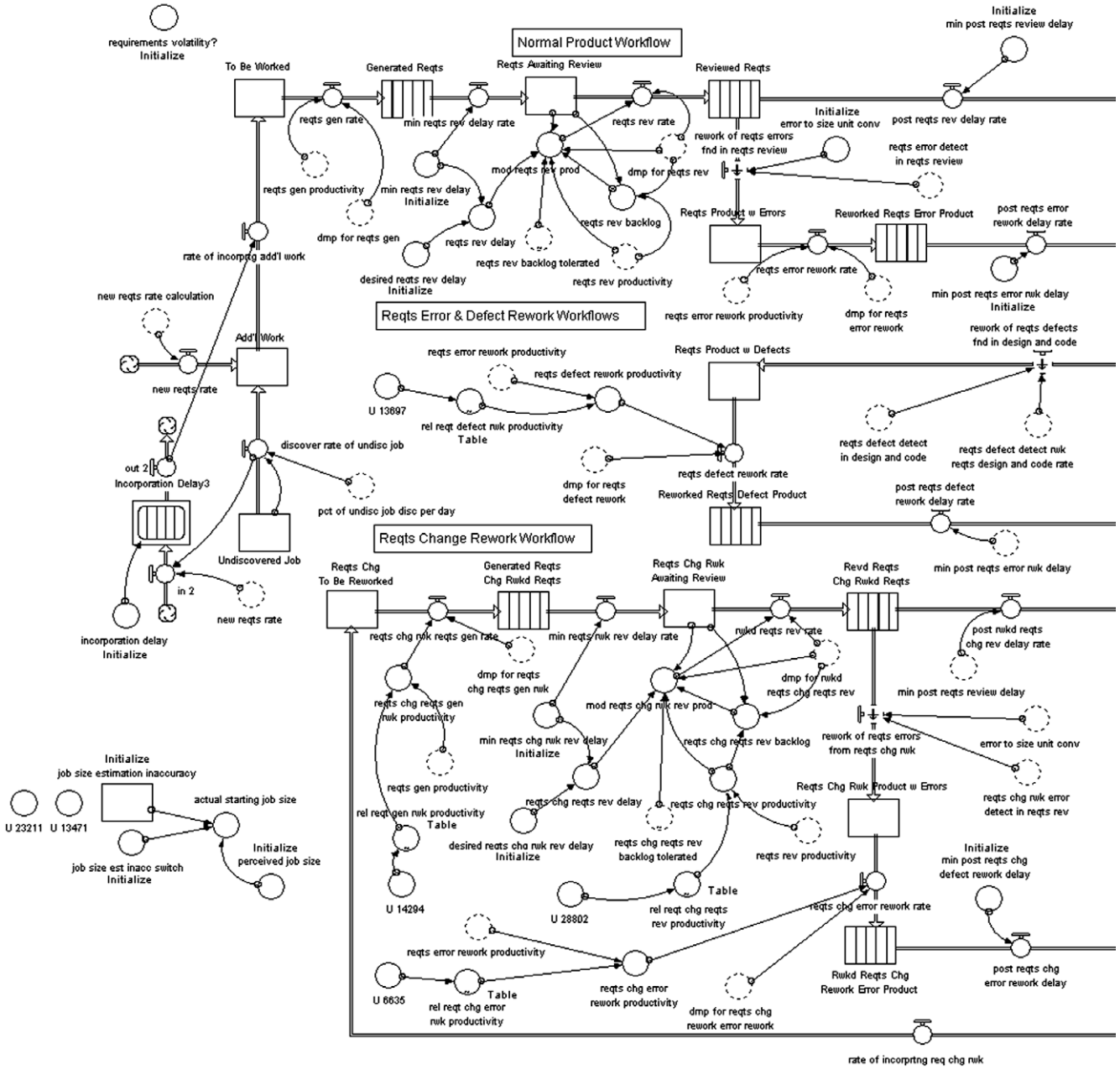

Fig. 4. SPMS simulator requirements engineering work flow sector.

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 work flow. Rework due to requirements volatility is drawn from the development and test work flows and is worked though the requirements change rework work flow. This work flow contains separate requirements error related activities.

The model allows the user to enter detailed inputs related to their project. Other data is automatically extracted from the stochastic distributions derived from the survey data. Table 3 provides a listing of a subset of the new inputs added to the model to provide a flavor for the types of data required from the user. These inputs can be input per run or a set of runs. Table 4 identifies the new survey distributions added to the model. Data from the survey is extracted from stochastic distributions with timing as defined in Table 4. "1 Selection per run" means that a value is pulled from the stochastic distribution one time per run and used throughout the run. While the data listed in Tables 3 and 4 do not include all the new or modified factors in the model it does provide a perspective of the type of data that the user can enter and use.

Table 3
New factor user inputs (subset only).

Factor	Input quantity
Quantity of experienced requirements engineers (REs), per interval	10 (1 for each of 10 intervals)
Quantity of new requirements engineers, per interval	10 (1 for each of 10 intervals)
Quantity of experienced personnel allocated to training new reqts engineers (%)	1
RE transfer in day (days)	1
Experienced organization RE transfer quantity	1
Time delay to transfer REs off project (days)	1
Max quantity of new RE hires per experienced staff (staff/staff)	1
Time for organization experienced REs (but not project experienced) to be productive (days)	1
Project day that person in org is scheduled to come onto project (day)	1
Quantity of RE staff experienced in the organization who are to be transferred on project (staff)	1
Requirements review adjustment policy – adjusts review effort (boolean switch)	1 Selection
Reworked requirements review adjustment policy (boolean switch)	1 Selection
Requirement error bad fix fraction (%)	1
Requirements defect bad fix fraction (%)	1
Nominal change request analysis productivity (function points/person-day)	1
Nominal requirements generation productivity (function points/person-day)	1
Nominal requirements review productivity (function points/person-day)	1
Requirements volatility (boolean switch)	1 Selection

5. Assessment tool results

The simulator was used to run two cases for the purpose of comparing them. Each case was setup for 100 runs apiece. The set of 100 runs for each case was selected for convenience as the modeling tool allows additional runs, if desired. The two cases are as follows: (1) A baseline case without the requirements volatility actualized [baseline] and (2) a case with the requirements volatility related factors actualized [reqts volatility]. The data in the square brackets corresponds to the case identifier in later figures. Actualizing the requirements volatility risk for the second case allows the model to represent the stochastic researched effects of requirement volatility. These include survey-based effects related to requirements additions, changes, and modifications as well as entering change requests. Also included, among the other effects, are morale, and related schedule pressure effects on morale. The initial job size was estimated to be 400 function points. The original schedule estimate (planned duration) was defined to be 408 days.

Figs. 5–8 depict the differences, at the completion of the projects, between the baseline runs (no requirements volatility considered) and runs with the other case where the requirements volatility risk is actualized for various summary outputs. The model allows additional detailed outputs, if desired. Box plots were used to represent the data because the simulation results were positively skewed given the tendency for higher project size, cost, duration, and released defects (among other outputs) 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 (Ott, 1988). With the exception of the final project size (Fig. 5), 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.

Table 4
New survey data distributions.

Factor	Selection frequency
Quantity of requirements change requests, per interval (intervals 1–10)	10 Selections per run (1 selection for each interval)
Change request time span with requirements volatility (ratio of duration)	1 Selection per run
Determination of change request acceptance/deferral due to schedule pressure (ratio)	1 Selection per run
Relative requirements defect rework productivity (ratio)	1 Selection per run
Relative requirements generation rework productivity (ratio)	1 Selection per run
Relative requirements change error rework productivity (ratio)	1 Selection per run
Relative requirements change requirement review productivity (ratio)	1 Selection per run
Relative design and code requirements defect rework productivity (ratio)	1 Selection per run
Percentage of perceived job size increased due to requirements volatility, per interval (%)	10 Selections per run (1 selection for each interval)
Percentage of perceived job size reworked due to requirements volatility, per interval (%)	10 Selections per run (1 selection for each interval)
Percentage of perceived job size reduced due to requirements volatility, per interval (%)	10 Selections per run (1 selection for each interval)
Requirements review effectiveness (%)	1 Selection per run
Design and code requirement defect detection effectiveness (%)	1 Selection per run
Design and code review requirement defect detection effectiveness (%)	1 Selection per run
Test requirement defect detection effectiveness (%)	1 Selection per run
Reworked requirements review effectiveness (%)	1 Selection per run
Reworked requirements design and code defect detect effectiveness (%)	1 Selection per run
Requirements error multiplier for schedule pressure (ratio)	1 Selection per run
Requirements error multiplier for morale (ratio)	1 Selection per run

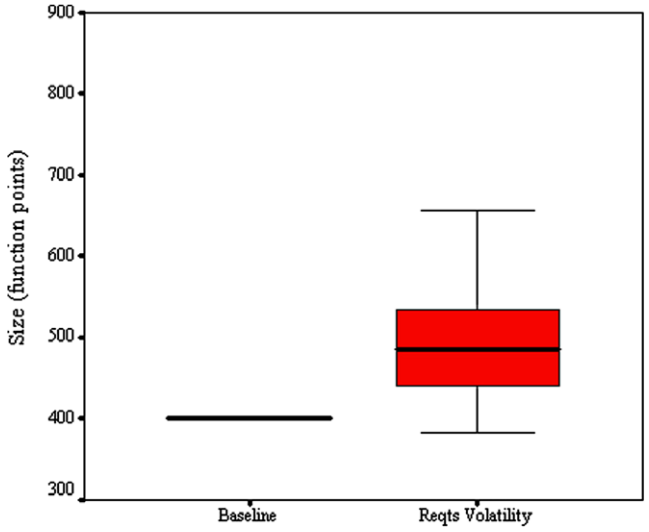

Fig. 5. Project size box plots.

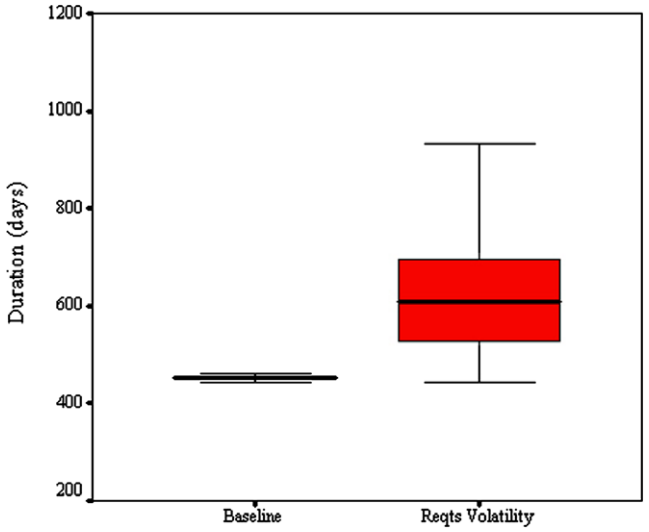

Fig. 7. Project duration box plots.

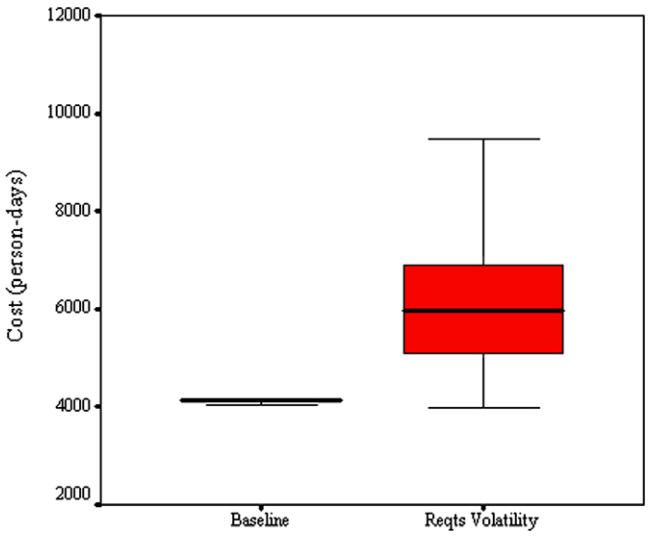

Fig. 6. Project cost box plots.

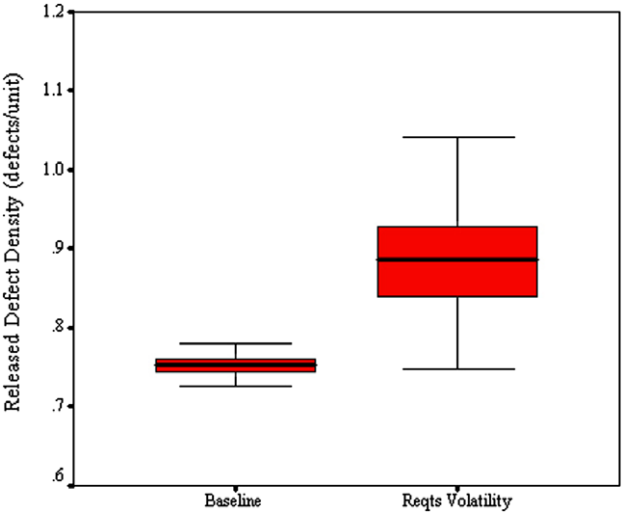

Fig. 8. Project released defect density box plots.

Fig. 5 illustrates the difference in project size between the baseline case and the requirements volatility case. The baseline size is 400 function points. The average size for the requirements volatility case is 499.9 function points with a standard deviation of 82.0, and a range from 382.0 to 788.3 function points over the 100 runs. The runs that have values below the baseline 400 function points (e.g. 382.0) can be explained by the fact that requirements changes do not always add additional project scope, but can be made to remove scope or unnecessary requirements. However, 93 of the 100 runs had a final project size greater than the base case of 400 functions points with relatively large scope added in the course of the project. This scope addition has a significant negative ripple effect on the project cost (effort in person-days) and project duration. Table 5 contains the detailed statistics for each case including average, median, standard deviation calculations and the minimum and maximum run values.

Fig. 6 presents the results for cost. Cost is considered to be human effort consumed during the project in person-days. This cost is equivalent to effort and does not include other project costs. The effort or cost presented in the figure encompasses the cumulative effort for the requirements engineering through test activities.

Table 5
Case study simulation results.

Output/Case	Average	Median	Std. Dev.	Minimum	Maximum
Size (function points)					
Baseline	400	400	0	400	400
Reqts volatility	499.9	485.7	82.0	382.0	788.3
Cost (effort in person days)					
Baseline	4107.6	4119.0	35.9	3969.5	4155.5
Reqts volatility	6208.2	5968.9	1398.1	3988.5	11196.1
Duration (days)					
Baseline	452.4	452.0	4.3	442.0	461.0
Reqts volatility	634.0	608.5	132.8	444.0	1101.0
Released defect density (defects per function point)					
Baseline	0.753	0.753	0.013	0.726	0.792
Reqts volatility	0.894	0.886	0.075	0.747	1.170

The detailed statistics are presented in Table 5. As mentioned earlier, the baseline results do show a small level of variability because some of the model parameters are represented stochastically. The average for the requirements volatility case is over 2000 person-days more than the average for the baseline case.

Considering that the baseline cost is an average of 4107.6 person days, this is a very large number for a program manager to justify as it represents more than a 50% increase in resource costs. The cost range for the requirements volatility case is very large as well. The requirements volatility case has a maximum of 11,196.1 person-days. This represents a very significant difference from the baseline maximum.

Fig. 7 displays the results for project duration in days. The duration encompasses the requirements engineering through test activities for the project. The detailed statistics for the two cases are shown in Table 5. It takes an average of 452.4 days to complete the project for the baseline case. The average for the requirements volatility case is significantly higher at an average of 634.0 days. As in the case of the cost, the range for the requirements volatility case has a wide span.

Fig. 8 presents the project released defect density box plots. The data represents the defects released post the test process and is in units of defects per function point. The quantity of released defects is also significantly higher for the requirements volatility case. The baseline case has an average of 0.753 defects per function point. The average for the requirements volatility case is 0.894. The range span for this case is also relatively large.

For the case that include the risk of requirements volatility, the box plots for each parameter have a wide span. The wide span in outcomes indicates a large potential for unpredictable results as well as considerable impact on project results. A comparison of the model results for projects with no requirements volatility and those with requirements volatility show significant differences in the box plots. One can clearly see the potential impact of requirements volatility on key project management parameters such as cost (effort), schedule, and quality. Differences between the baseline and requirements volatility case results can be attributed to the addition (or reduction) in project scope caused by requirements volatility, cause and effect relationships between factors, and the stochastic representation of a number of the factors.

6. Summary and conclusions

Key research questions were addressed as part of the research. These questions include: (1) Which software factors are affected by requirements volatility? (2) How can these factors and the uncertainty associated to these factors be modeled? and (3) What is the project management impact of requirements volatility? A causal model was developed and used to evaluate which software factors are affected by requirements volatility. A survey was administered to collect information for a subset of the factors identified in the causal model. This allowed the researchers to verify the relationships and quantify the level of the relationships. A simulator was chosen as the means to model how the factors relate to other project factors and to represent the uncertainty associated to the factors using stochastic distributions derived from survey data. The SPMS model assists in understanding and evaluating the program management 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. The factor relationships represented in the model represent a significant contribution. This work expands our understanding of the requirements engineering process and the effects of requirements volatility. 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 is a significant contribution.

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 parameters integrated with a very comprehensive development and test related 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.

Survey data used in the simulator captures information on factors and relationships not previously available from a wide population in the software industry and allows modeling variables stochastically given the large quantity of survey responses. Many of the model variables were modeled stochastically to allow the user to assess project outcomes probabilistically. The model results quantitatively illustrate the potential for significant cost, schedule, and quality impacts related to requirements volatility.

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.

7. Future research

Additional research in the area of 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 and that can represent different facets of understanding in this under-represented yet critical research area. 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 affected 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.

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 (Houston et al., 2001). One of the benefits of a simpler model includes a shortened training and learning ramp-up time. Core concepts that make the most difference in results can be emphasized. Since the model is less complex it becomes easier to understand and may perhaps be more popular given the reduced time to understand and then 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 and fix problems.

Agile processes which welcome requirements changes present another valuable area to study. As these processes continue to mature and more quantitative results are available, simulating different types of agile processes, (e.g. XP, Scrum, etc.) would be of interest to determine the project management ramifications related to cost, schedule, and quality as compared to more traditional approaches.

References

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

Abramowitz, M., Stegun, I.A. (Eds.), 1964. Handbook of Mathematical Functions, Applied Mathematics Series 55. National Bureau of Standards, Washington, DC.

Beck, K., Beedle, M., van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., Highsmith, J., Hunt, A., Jeffries, R., Kern, J., Marick, B., Martin, R., Mellor, S., Schwaber, K., Sutherland, J., Thomas, D., 2001. Principles Behind the Agile Manifesto. Retrieved 11.6.2008 from: <http://www.agilemanifesto.org/principles.html>.

Boehm, B.W., 1991. Software risk management: principles and practices. IEEE Software 8 (1), 32–41.

Costello, R.J., 1994. Metrics for Requirements Engineering. Master of Science Thesis, California State University, Long Beach.

Curtis, B., Krasner, H., Iscoe, N., 1988. A field study of the software design process for large systems. Communications of the ACM 31 (11), 1268–1287.

Ferreira, S., 2002. Measuring the Effects of Requirements Volatility on Software Development Projects, Ph.D. Dissertation, Arizona State University.

Ferreira, S., Collofello, J., Shunk, D., Mackulak, G., Wolfe, P., 2003. Utilization of Process Modeling and Simulation in Understanding the Effects of Requirements Volatility in Software Development. In: Proceedings of the 2003 Process Simulation Workshop (ProSim 2003).

Finnie, G.R., Witting, G.E., Petkov, D.I., 1993. Prioritizing software development productivity factors using the analytic hierarchy process. Journal of Systems and Software 22 (2), 129–139.

Houston, D.X., 2000. A Software Project Simulation Model for Risk Management, Ph.D. Dissertation, Arizona State University.

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), 259–270.

Javed, T., Maqsood, M., Durrani, Q.S., 2004. A study to investigate the impact of requirements instability on software defects. ACM SIGSOFT Software Engineering Notes 29 (3), 7.

Jones, C., 1994. Assessment and Control of Software Risks. PTR Prentice-Hall, Inc., Englewood Cliffs, NJ.

Jones, C., 1998. Estimating Software Costs. McGraw-Hill, New York.

Känsälä, K., 1997. Integrating risk assessment with cost estimation. IEEE Software 14 (3), 61–67.

Kellner, M.I., Madachy, R., Raffo, D.M., 1999. Software process modeling: why? what? how? Journal of Systems and Software 46 (2–3), 91–105.

Kotonya, G., Sommerville, I., 1998. Requirements Engineering: Processes and Techniques. John Wiley and Sons, Ltd..

Lane, M.S., 1998. Enhancing software development productivity in Australian firms. In: Proceedings of the Ninth Australasian Conference on Information Systems (ACIS '98), vol. 1, pp. 337–349.

Lin, C.Y., Abdel-Hamid, T., Sherif, J.S., 1997. Software-engineering process simulation model (SEPS). Journal of Systems and Software 38 (3), 263–277.

Lin, C.Y., Levary, R.R., 1989. Computer aided software development process design. IEEE Transactions on Software Engineering 15 (9), 1025–1037.

Loconsole, A., Börstler, J., 2005. An industrial case study on requirements volatility measures. In: Proceedings of the 12th Asia-Pacific Software Engineering Conference (APSEC '05), 8 p.

Loconsole, A., Börstler, J., 2007. Are size measures better than expert judgment? An industrial case study on requirements volatility. In: Proceedings of the 14th Asia-Pacific Software Engineering Conference (APSEC '07), pp. 238–245.

Madachy, R., Tarbet, D., 2000. Initial experiences in software process modeling. Software Quality Professional 2 (3), 1–13.

Madachy, R., Boehm, B., Lane, J., 2007. Assessing hybrid incremental processes for SISOS development. Software Process: Improvement and Practice 12 (5), 461–473.

Malaiya, Y.K., Denton, J., 1999. Requirements volatility and defect density. In: Proceedings of the 10th International Symposium on Software Reliability Engineering, pp. 285–294.

Matko, D., Zupancic, B., Karba, R., 1992. Simulation and Modeling of Continuous Systems: A Case Study Approach. Prentice-Hall International Ltd., Great Britain.

Moynihan, T., 1997. How experienced project managers assess risk. IEEE Software 14 (3), 35–41.

Nidumolu, S.R., 1996. Standardization, requirements uncertainty and software project performance. Information and Management 31 (3), 135–150.

Nurmuliani, N., Zowghi, D., Fowell, S., 2004. Analysis of requirements volatility during software development life cycle. In: Proceedings of the 2004 Australian Software Engineering Conference (ASWEC '04), pp. 28–37.

Ott, L., 1988. An Introduction to Statistical Methods and Data Analysis. PWS-KENT Publishing Company.

Pfahl, D., Lebsanft, K., 2000. Using simulation to analyze the impact of software requirements volatility on project performance. Information and Software Technology 42 (14), 1001–1008.

Pritsker, A. Alan B., O'Reilly, Jean J., LaVal, David K., 1997. Simulation with Visual SLAM and AweSim. System Publishing Company, West Lafayette, IN.

Reifer, D.J., 2000. Requirements management: the search for Nirvana. IEEE Software 17 (3), 45–47.

Richardson, George, P., Alexander, L., Pugh III, 1981. Introduction to System Dynamics Modeling with DYNAMO. The MIT Press, Cambridge, MA.

Ropponen, J., 1999. Risk assessment and management practices in software development. Chapter 8 in Beyond the IT Productivity Paradox. John Wiley and Sons. pp. 247–266.

Ropponen, J., Lyytinen, K., 2000. Components of software development risk: how to address them? A project manager survey. IEEE Transactions on Software Engineering 26 (2), 98–112.

Schmidt, R., Lyytinen, K., Keil, M., Culle, P., 2001. Identifying software project risks: an international Delphi study. Journal of Management Information Systems 17 (4), 5–36.

Smith, B.J., Nguyen, N., Vidale, R.F., 1993. Death of a software manager: how to avoid career suicide through dynamic software process modeling. American Programmer 6 (5), 10–17.

The Standish Group, 1995. The Chaos Report. Obtained from <http://www.standishgroup.com/chaos.html>.

Stark, G.E., Oman, P., Skillicorn, A., Ameele, A., 1999. An examination of the effects of requirements changes on software maintenance releases. Journal of Software Maintenance: Research and Practice 11 (5), 293–309.

Tirwana, A., Keil, M., 2006. Functionality risk in information systems development: an empirical investigation. IEEE Transactions on Engineering Management 53 (3), 412–425.

Tvedt, J.D., 1996. An Extensible Model for Evaluating the Impact of Process Improvements on Software Development Cycle Time. Ph.D. Dissertation, Arizona State University.

Yau, S.S., Collofello, J.S., MacGregor, T., 1978. Ripple effect analysis of software maintenance. In: Proceedings of the IEEE Computer Society's Second International Computer Software and Applications Conference (COMPSAC '78), pp. 60–65.

Yau, S.S., Kishimoto, Z., 1987. A method for revalidating modified programs in the maintenance phase. In: Proceedings of 11th IEEE International Computer Software and Applications Conference (COMPSAC '87), pp. 272–277.

Yau, S.S., Liu, C.S., 1988. An approach to software requirement specifications. In: Proceedings of 12th International Computer Software and Applications Conference (COMPSAC '88), pp. 83–88.

Yau, S.S., Nicholl, R.A., Tsai, J.P., 1986. An evolution model for software maintenance. In: Proceedings of 10th IEEE International Computer Software and Applications Conference (COMPSAC '86), pp. 440–446.

Yau, S.S., Nicholl, R.A., Tsai, J.P., Liu, S.S., 1988. An integrated life-cycle model for software maintenance. IEEE Transactions on Software Engineering 14 (8), 1128–1144.

Zowghi, D., Nurmuliani, 1998. Investigating requirements volatility during software development: research in progress. In: Proceeding of the Third Australian Conference on Requirements Engineering (ACRE98), pp. 38–48.

Zowghi, D., Nurmuliani, 2002. A study of the impact of requirements volatility on software project performance. In: Proceedings of the Ninth Asia-Pacific Software Engineering Conference (ASPEC '02), pp. 3–11.

Zowghi, D., Offen, R., Nurmuliani, 2000. The impact of requirements volatility on the software development lifecycle. In: Proceedings of the International Conference on Software Theory and Practice (IFIP World Computer Congress), pp. 19–27.

Susan Ferreira is an Assistant Professor and the founding Director of the Systems Engineering Research Center (SERC) at The University of Texas, Arlington (UTA). Before joining UTA, Dr. Ferreira worked as a systems engineer in the Defense industry on complex software intensive systems. Her industry background includes work for Lockheed Martin, General Dynamics/Motorola, and Northrop Corporation. Her teaching and research interests are related to systems engineering (SE) and include requirements engineering, SE process modeling and simulation, lean SE, SE return on investment, SE cost estimation, and system of systems engineering.

James Collofello is currently Associate Dean for the Engineering School and Professor of Computer Science and Engineering at Arizona State University. He received his Ph.D. in Computer Science from Northwestern University. His teaching and research interests lie in the software engineering area with an emphasis on software project management, software quality assurance and software process modeling. In addition to his academic activities, he has also been involved in applied research projects, training and consulting with many large corporations over the last 25 years.

Dan Shunk is the Avnet Professor of Supply Network Integration in Industrial Engineering at Arizona State University. He is currently pursuing research into collaborative commerce, global new product development, model-based enterprises and global supply network integration. He won a Fulbright Award in 2002-2003, the 1996 SME International Award for Education, the 1991 and 1999 I&MSE Faculty of the Year award, the 1989 SME Region VII Educator of the Year award, chaired AutoFact in 1985, and won the 1982 SME Outstanding Young Engineer award. Dr. Shunk studied at Purdue where he received his Ph.D. in Industrial Engineering in 1976.

Gerald Mackulak is an Associate Professor of Engineering in the Department of Industrial, Systems and Operations Engineering at Arizona State University. He is a graduate of Purdue University receiving his B.Sc., M.Sc., and Ph.D. degrees in the area of Industrial Engineering. His primary area of research is simulation methodology with a focus on model abstraction, execution speed and output analysis. He has authored over 75 articles, served as an associate editor for two simulation journals and continues to guide research in simulation efficiency.