ABSTRACT

Process simulation has become a powerful technology in support of software project management and process improvement over the past decades. This research, inspired by the Impact project, intends to investigate the technology transition of software process simulation to the use in industrial settings, and further identify the best practices to release its full potential in software practice. We collected the reported applications of process simulation in software industry, and identified its wide adoption in the organizations delivering various software intensive systems. This paper, as an initial report of the research, briefs a historical perspective of the impact upon practice based on the documented evidence, and also elaborates the research-practice transition by examining one detailed case study. It is shown that research has a significant impact on practice in this area. The analysis of impact trace also reveals that the success of software process simulation in practice highly relies on the association with other software process techniques or practices and the close collaboration between researchers and practitioners.

Keywords
process simulation, software process, impact analysis

1. INTRODUCTION

Software Process Simulation (SPS), since its pioneering work in 1980s, has been applied in many software development/maintenance projects with varying process scales and organization settings over the past decades. It leverages planning, managing, controlling, improving software processes, and provides software practitioners (professionals and managers) powerful tools and recognizable benefits. There are now hundreds of studies exclusively on process simulation for Software Engineering (SE); several SE conferences/workshops series encouraging the contributions in this area; many universities including process simulation in their software engineering (process/project) curriculum.

The research reported in this paper aims to reflect how process simulation technologies and modeling methodologies are applied in software engineering practice, and further to identify the best practices in order to encourage more industrial adoptions and maximize the potential benefits to software organizations. To achieve these, we start with identifying industrial application cases on the basis of a systematic review and its extension, find the SPS research origins by tracing them backward, and elaborate how impact took place along with one sample thread using a case study.

This study has been conducted under the Impact project. The aim of the project is to provide a scholarly study of the impact that software engineering research (both academic and industrial) has had upon practice [25]. Unlike typical software engineering research, the Impact project is more akin to the discipline of history of science and technology [9] and sheds light on the influence of research on practice to increase the level of reflection in many ways. The Impact project has investigated completely formal (e.g., configuration management [10]) and semi-formal (e.g., inspections [36]) SE practices. However, the impact of SPS research is very difficult to quantify. Anecdotal evidence exists for the successful applications of process simulation in software companies. An indicator might be how widely the SPS industrial success cases exist. Compared to the existing reports under the project, we document the SPS industrial cases and research impact in a relatively qualitative form.

2. RESEARCH METHOD

According to Merriam-Webster Dictionary, impact is “the force of impression of one thing on another”; it “may be used to imply contact between two things, at least one of which is impelled toward the other”. It implies that impact can be directional. There may exist bi-directional impact between research and practice. To be specific in SPS area, as illustrated in Figure 1, process simulation research introduces and provides effective simulation techniques, methods, and tools, which combined with the detail of other SE technologies, for constructing process simulation models. These models can be further adopted in software practice in certain organizational context. In the other direction, with support
of empirical research methods, the data and experience collected from the adoption can also be used as feedback to continuously impact SPS research [46]. The research reported in this paper concentrates on the impact of process simulation research on software engineering practice.

We designed and followed a three-step approach (shown in Figure 2) to researching the impact: 1) impact evidence search, 2) impact trace analysis, and 3) impact case study. The three steps investigate the impact at different zooms, where one step is built upon its preceding steps.

2.1 Impact Evidence

We decided to concentrate on the public literature (scholarly papers, books, dissertations and technical reports) for searching impact evidence. Although this strategy may miss some gray-literature or unpublished work, an entire search of those relevant but unpublished work is somehow impossible. In addition, we believe that impact can be gauged more by the extent to which SPS techniques and tools are applied than by the quantity. Hence we conclude that basing this report on documented evidence would be reasonable.

Unlike the existing reports for Impact project that were built on either the researchers’ knowledge or the search of limited venues, the evidence of this research is solidly based on a Systematic Literature Review (SLR) and its extension. In 2008 and 2010, we reported a two-stage SLR on software process simulation [45, 47], which aimed to assess how SPS has evolved over the decade (1998-2007). In particular the SLR addresses a series research questions as enhancement and update to Kellner, Madachy, and Raffo’s (KMR) landmark paper [18], such as whether the reasons for SPS, the simulation techniques (tools), application domains, and model scopes have changed [45]. The SLR identified 156 relevant studies to SPS research and practice, which are further grouped into four categories: A) software process simulation models and simulators; B) process simulation techniques, methods, and tools; C) applications, guidelines for SPS adoption in practice; D) experience reports and empirical studies. The detailed study search and selection criteria are available in [45, 47].

We base the impact evidence on the Class C and D studies identified by the SLR. In order to draw a historical perspective of SPS, an extension of the SLR’s time span is necessary. We applied the automated search elaborated in [47] to capture the relevant studies reported before 1998 and after 2007 (to present). To avoid missing important studies, the work by some important researchers in this area (identified by the SLR) were also checked manually in the entire time span. The relevant studies were reported in the following publication types: scholarly papers (research and experience papers published by journals, conferences, and workshops), PhD theses, books, and technical reports.

We note that the nature of this research is considerably different from our SLR. The Impact project aims to reveal how SE research influenced industrial practices, rather than emphasize academic research only. However, the citations of academic and industrial publications are among the strongest form of evidence that we can collect for tracing the impact. The SLR and its extension provide a ‘gold standard’ of the published studies as candidate evidence.

As this paper does not investigate the impact from industry to research, we excluded a number of studies that only address the calibration and validation of process simulation models using specific industrial data and experience, for example, some large-scale research projects funded by industry collaboration (e.g., FEAST project [40]), and the research based on specific experience or data from industry (if there is no explicit impact back to industry).

2.2 Impact Trace

Two approaches to defining the impact of research on practice were discussed in [36]: Top-down by identifying all companies that adopt process simulation, or Bottom-up by identifying example application cases and tracing the impact of research for them. The main difficulty of the top-down approach is that it requires a representative survey on the use of process simulation in industry. Selecting a representative sample requires a complete listing of software companies with their characteristics, which we are not able to access. On the other hand, the bottom-up approach is able to draw some individual impact trace but may lose an overview of the area. Therefore, we decided to analyze the impact by following a hybrid approach, i.e., starting at a number of reported industrial application cases identified by the SLR and its extension, and then tracing them back to research origins. This approach allows us to verify which forms of SPS had been adopted in the software companies as well as acquire more associated detail. However, we did not intentionally pick up success cases only (like [36]) because we believe even the failures may provide insightful lessons for both the researchers and practitioners in this area.

2.3 Impact Exemplar

The impact trace derived from the identification and analysis of the studies related to SPS industrial applications and research origins from public literature. Though a number of SPS applications were identified as the impact evidence, the public literature may be not able to provide rich interesting detail of the impact about how process simulation research was transferred to certain software companies and adopted successfully. Accordingly, an in-depth investigation of the impact needs to zoom into some continuous applications with sustainable success in certain software organization(s), which means process simulation has been applied.

Figure 1: Bi-directional impact

Figure 2: Research method
over a period of time with expected effects.

One impact exemplar was selected from the impact threads identified in the previous step for the in-depth case study. However, the selection was not on a random basis, because we had to consider the continuity of the application, the time effectiveness of the case(s), and the possibility of data access. After the exemplar being selected and confirmed, we collected the impact detail using a series of interview-styled email communications due to the geographic distance and time difference between us.

3. SOFTWARE PROCESS SIMULATION

This section sets the background of SPS, briefs the major techniques for process simulation, and summarizes the state-of-the-practice of SPS based on the SLR and its extension.

3.1 Software Process and Process Modeling

Software (engineering) process is composed of a set of logically ordered tasks or activities in order to deliver or to maintain a software product [8]. These activities involve the development of software from scratch, and enhancement of existing systems by configuring and integrating off-the-shelf software or system components. Software processes are important in software engineering practice not only because they impose consistency and structure on a set of development activities, but for enabling us to capture our experiences and pass them along to others.

Osterweil identifies two complementary types of software process research, which can be characterized as macro-process, focused on phenomenological observations of external behaviors of processes, and micro-process, focused on the internal details and workings of processes [27].

Software process modeling is an active aspect of software engineering research and practice and has grown since the early 1980s. A software process model consists of a set of elements or entities and their relationships with a well-defined goal for representing and analyzing the processes utilized in developing software. Given the large diversity of software processes, modeling software process is a complicated task. Depending on the modeling purposes and approaches, software process models may vary as static or dynamic, qualitative or quantitative, black-box or white-box, descriptive or prescriptive, continuous or discrete, and so on [44]. Unlike static models, dynamic (simulation) process models are executable. They not only employ the concepts of static models but go beyond them by describing behavior of elements or changes in the complex interactions and by allowing modelers to observe process performance over time.

3.2 Process Simulation in a Nutshell

The diversity and complexity of software processes and the richness of process related questions determine the different capabilities of simulation paradigms needed. Overall, 15 simulation techniques were found in the SPS studies. Among them, System Dynamics and Discrete-Event Simulation are the most widely used techniques in SPS, which can be classified into two conventional simulation paradigms respectively, i.e., continuous and discrete simulation, and conform to macro-process and micro-process research in SE [27].

3.2.1 Continuous Simulation

System Dynamics (SD) was invented by Forrester and his colleagues in the mid-1950s and early 1960s [11, 12], and became a major technique of continuous simulation. SD is used for modeling complex dynamic systems with continuous changes. SD is capable to “deal with the time-dependent behavior of managed systems with the aim of describing the system and understanding, through qualitative and quantitative models, how information feedback governs its behavior, and designing robust information feedback structures and control policies through simulation and optimization” [7].

These models are formulated using continuous quantities interconnected in cause-effect relationships and feedback loops. The quantities are expressed as levels (or stocks), which are the current values of variables that have resulted from the accumulated difference between inflows and outflows and represent a dynamic system’s current state, and rates (or flows), which are considered as control variables that represent the activity in a dynamic system and determine the levels. The directed information links between rates and levels form the feedback loops. The levels are affected by the flow rates, which may be affected by the levels as well.

SD models are quantitative. The basic mathematical representation of an SD model is a system of coupled, nonlinear, first-order differential equations,

\[ x'(t) = f(x(t), p) \]

where \( x \) is a vector of levels, \( p \) is a set of parameters, and \( f \) is a nonlinear vector valued function. State variables of the modeled systems are represented by the levels.

Since Abdel-Hamid and Madnick’s (AHM) seminal work [3], SPS research and practice during the past three decades have successfully adopted and continuously relied heavily on SD. The power of modeling a software development process using SD lies in its ability to take into account a number of product and process factors that affect reliability, cycle time, and cost to determine the global impact of their interactions.

Though qualitative analysis is one important step during SD modeling, SD models are solely used for quantitative study. Qualitative Simulation (QSIM), the qualitative form of continuous simulation, models the real world systems at a coarser level of abstraction where a qualitative differential equation represents a large set of possible ordinary differential equations [19]. Thus QSIM is able to cope with a lack of precise knowledge with fewer assumptions.

3.2.2 Discrete Simulation

Discrete-Event Simulation (DES) or state-based simulation is concerned with the modeling of discrete system that can be represented by a series of events or states. The state variables of such a system change only at discrete set of points in time [4]. The simulation describes each discrete event or state, moving from one to the next in the right order as time progresses. One typical discrete system is a queuing system that is described by its calling population. The state of the system is the number of units inside and the status of server (busy or idle). An event is a set of circumstances that causes an instantaneous change in the system.

The measurement of time in a simulation corresponds to appropriate units of time in the real system. The simulation clock indicates the time of the next event to be performed. A simulation, starting at time zero, performs all events in the order in which they occur, and runs until either there are no more events to perform, or the clock time exceeds the given duration, or some interrupt is triggered.

In software development, developers are represented as
servers in a DES model. They perform development activities, e.g., coding, inspection, testing, and reworking, in given delay. The common entities are the artifacts in development processes, such as documents, functions or modules, defects.

Process Programming was encouraged by Osterweil’s view that “software processes are software too” [26]. He suggested that software processes were themselves a form of software and that there would be considerable benefits that would derive from basing a discipline of software process development on the more traditional discipline of application software development such as programming. In the late 1980s Ada was first used as a process programming language in TRW, which was later extended by APPL/A. JIL [39], a successor of APPL/A, emphasizes process steps and programming control with exception handling. Little-JIL [41] is a subset of JIL with visualization support. Process is modeled as a tree of steps, whose leaves represent the units of work - steps. Little-JIL uses non-leaf steps to capture step ordering. Juliette, a Little-JIL environment, provides Java-based common characteristics of conventional programming languages [6]. Process models implemented in Little-JIL can be simulated based on state transition, and the results of modeled process can be predicted.

These two conventional and popular simulation paradigms are different from each other in nature, such as view of process (e.g., macro- or micro-process), loop structure (open or closed), time span ('step' or 'event'), precision, and deterministic [44]. In the real world, “few systems are wholly discrete or continuous, but since one type of change predominates for most systems, it will usually be possible to classify a system as being either discrete or continuous” [20]. The selection of continuous or discrete paradigm is determined by the nature of problem and the purpose of simulation. Since the late 1990s, hybrid process simulation has become an active attempt to combine the advantages of the both.

3.2.3 Emerging Approaches for Process Simulation

Apart from the continuous and discrete paradigms, some emerging simulation techniques are also observed in SPS studies, e.g., Agent-Based Simulation (ABS), Role-Playing Game (RPG), Cognitive Map, Dynamic System Theory.

An Agent-Based Simulation model is regarded as a Multi-Agent System (MAS), which is a system composed of multiple interacting intelligent agents. The agents in a MAS have several important characteristics [42]: autonomy, localization, decentralization, and adaptability. In an ABS, the overall behavior of the system is an emergent property of the individual, independent interactions of the agents. It emerges as a promising approach to simulating agile and open source software development involving complex and random behaviors of the developers (agents).

A Role-Playing Game is different from the simulation models developed using the other modeling approaches. The 'story' or 'scenarios' have to be predefined prior to simulation. However, the results of simulation highly depend on the interactions, the player’s decisions, on the fly. Often RPG simulator (like a flight simulator) requires the support of other simulation paradigm(s), such as SD. Plus, it enriches the model’s graphic user interface and allows a simulator to support game-like training and learning at cognitive level.

Nevertheless, these emerging approaches for SPS still remain for laboratory research and education purposes. Their industrial applications were rarely observed in the literature.

3.3 State-of-the-Practice

In [18], KMR provided the state of SPS research by answering the fundamental questions: why? what? and how? We brief an up-to-date state of SPS arena by revisiting them.

3.3.1 Why simulation?

Compared to the other methods of modeling and analyzing software processes, SPS offers several advantages, like: 1) dynamic simulations model the structure and behavior of software process in varying levels of degree using flows and/or discrete events, which is able to address the dynamic and human factors of development processes; 2) feedback loops enable a dynamic simulation of the nonlinear behaviors exhibited in a software process; 3) though a simulation uses calculations, it does not require an underlying complicated mathematical model, and avoids factor independence required in regression analysis [5].

By examining the relevant studies, ten purposes for SPS can be identified and further grouped at three application levels. The cognitive level contains the purposes of 1) understanding, 2) communication, 3) process investigation, 4) training and learning. On the tactical and strategic levels the purposes are similar but differ in scope and impact: 5) prediction and planning, 6) control and operational management, 7) risk management, 8) process improvement, 9) technology adoption, 10) trade-off analysis and optimization.

3.3.2 What to simulate?

Problem domain identifies the problem(s) in software engineering that the simulation model investigates. It further determines the model’s structure, input parameters, and output variables. We found 21 problem domains studied by the simulation models identified in the review [47]. The top 10 include 1) generic development, 2) software evolution, 3) software process improvement, 4) requirements engineering process, 5) incremental and concurrent development, 6) verification and validation, 7) open-source development, 8) global development, 9) agile development, and 10) software maintenance. Upon these topics, model scopes range from single phase, project, to product evolution.

Output variables reflect the research interests of software process. Out of 15 output variables found in the SPS models, time (duration), effort (cost), quality, (software) size, and resource are the most interesting indicators.

3.3.3 How to simulate?

Process simulation would not happen without the support of computer-aided simulation environments (compilers, engines, or workbenches). In addition to the techniques for process simulation (cf. Section 3.2), 20+ simulation tools have been used to support SPS. Vensim and Extend, which provide continuous and/or discrete simulation capability, were most observed in the impact evidence.

Hybrid process simulation has become an increasingly active research area in software process simulation since the late 1990s. Hybrid approach employs more than one simulation technique in developing a simulation model. By combining discrete and continuous simulations, for example, hybrid process simulation is capable of addressing both the micro-level and macro-level process dynamics, and breaks through the limitations of applying any single simulation method. However, integrating these two approaches faces the issues of compatibility, interoperatability and synchronization when
executing simulation. Most of the hybrid process simulations are based on the combination of SD and DES.

Process simulation is often time-consuming and requires (sometimes PhD level) expertise and experience, which may resist its industrial application. In order to ease model development, model composition for reuse has gained considerable attention in SPS community in the recent years.

4. TRACING INDUSTRY IMPACT

By examining the relevant studies through the SLR and its extension, 32 industrial application cases were identified as the impact evidence. These cases reveal that two types of process simulation have been applied in industrial settings during the past decades: system dynamics and discrete-event (state-based) simulation. This section respectively describes their impact by tracing the application cases back to the research origins. Figure 3 shows the overall SPS impact trace. The impact evidence (red blocks) and corresponding research origin (blue or gray blocks) are located chronologically and grouped in terms of their organizations. Note that given the limited space this paper only describes some of the important SPS application cases we identified.

4.1 System Dynamics

Since 1951, Forrester joined Massachusetts Institute of Technology (MIT), and spent his entire career there. Meanwhile, a number of system dynamics models in varying scales were developed in MIT to solve the problems in many disciplines, particularly management. In 1982, the first SD model [2] (an early version of EXAMPLE model) addressing software process issues appeared. The model was developed by Abdel-Hamid during his PhD at MIT, Sloan School of Management. In 1984, he finished his PhD thesis on Software Project Dynamics [1]. He continued his research on SD modeling of software process/project, and published the most cited book in SPS community [3] with Madnick (his adviser) in 1991. The book summarizes his PhD work and the follow-up research. The SD-based model (EXAMPLE) captures the managerial aspects of a waterfall software life-cycle and is regarded as the starting point for other subsequent SPS models of either the entire development process or parts of it. Note that Boehm’s COCOMO model [5] was used to compute the initial project plan in the EXAMPLE.

Base upon Software Project Dynamics, in 1994 Madachy completed his PhD at University of Southern California (USC) with a dynamic simulation model of inspection-based software life-cycles process to support quantitative process evaluation and risk assessment [22]. His SD model serves to examine the effects of inspection practices on cost, schedule, and quality throughout the project life-cycle by interrelating flows of tasks, errors and personnel in each development phase. When being applied and tested in Litton Data Systems, the model matched the actual project data very well [23]. Litton also created small-scale SD models for planning and training purposes at multi-project or departmental level including domain learning, product-line, reuse processes and resource contention among projects. It is reported that “insights provided by the models have supported decision-making at different levels and helped galvanize process improvement efforts...by examining the models and simulated behavior, managers share a process vision and can discuss issues against the common models...understand the key factors in complex scenarios” [21].

In 1998, the first International Workshop on Software Process Simulation Modeling (PROSIM) was held. The event fostered SPS research and practice. Most of the published SPS application studies in 1999 (Figure 3) were originally reported in PROSIM’98. In this workshop and its special issue, Kellner, Madachy and Raffo intended to answer some fundamental questions about SPS research [18], which became another most-cited reference in SPS community in the past decade. The numbers of both research and application studies of SPS dramatically increased after 1998. In the following years, the PROSIM series workshop became one of the major forums of software process research.

Since 1990s, the research and applications of SPS emerged and increased in Europe. Pfahl was involved in some important industry collaborations during his PhD research at Fraunhofer Institute for Experimental Software Engineering (ISEE). He designed an SD-based framework for Integrated Measurement, Modeling, and Simulation (IMMoS) that supports both strategic and project management in software organizations by providing guidance on developing and using quantitative process simulation models as a source for learning and improvement [28]. IMMoS was evaluated by two case studies of industrial projects (PSIM and RESIM) and one controlled experiment (GENSIM).

The two industrial cases are the applications of SD-based process simulation in different divisions in Siemens. PSIM is an SD modeling project that was conducted inside a large software development department of Siemens Private Networks Group in 1994 and 1995, when Pfahl still worked with Siemens. From this project the SD model Project SIMulator (PSIM) resulted. The goals of PSIM were related to the planning and control of software development projects, and further supporting continuous process improvement [29]. Although the PSIM model was only a prototype that had not been used on a regular basis, it generated many new insights, especially on methodological and organizational aspects of SD modeling in software industry.

The other project developed the Requirements Engineering SIMulator (RESIM) for Siemens Corporate Technology. The purpose of RESIM was to investigate the impact of unstable software requirements on project duration and effort, and to analyze the required effort for stabilizing requirements and optimizing cost effectiveness [30]. The model results have provided a deeper understanding of the procedures for capturing and changing requirements as well as a broader view of the requirement process within software development. With a calibration of the model parameters and functions to empirical data, the model is able to be “used for precise estimates in the sense of a predictive model”. Experts at Siemens felt that “building the SD model was a useful experience, and that similar models can help them in future process improvement projects”.

4.2 Discrete Simulation

The impact trace (Figure 3) shows that discrete process simulation originated the research on (static) software process modeling since the late 1980s. Process models provide the operational guidance regarding the critical sequence of process steps, information flows, and organizational responsibilities [8], and the capability to check the integrity of the process. In 1990, the Sixth International Software Process Workshop (ISPW-6) Process Example [17] was designed to “incorporate examples of 18 important process aspects and
issues” and “contain a diversity of process aspects encountered in real-world software processes”. It had been used to benchmark a number of process modeling approaches.

The ISPW-6 process example was used as the baseline process for the process trade-off analysis in Raffo’s PhD thesis [34]. He developed a discrete (state-based) simulation model using Task Element Decomposition (TED) method based on Markov chain to quantitatively evaluate the performance of alternative software processes and process changes in terms of quality, cost and schedule.

After his PhD at Carnegie Mellon University, Raffo moved to Portland State University (PSU) and continued his research on SPS. Based on his PhD outcome Raffo et al. developed quantitative process analysis tools for Northrop Grumman’s Surveillance and Battle Management Systems. These tools employ stochastic SPS models and provide a mechanism for quantitative analysis of the current process as well as the proposed process change alternatives prior to implementation [35]. The use of SPS is “a key part of the company’s strategy for achieving a higher process capability and moving to CMM Levels 4 and 5”.

Martin and Raffo later constructed a hybrid simulation model of Northrop Grumman’s process. With reference to AHM’s EXAMPLE model [3] and other earlier models, they employed SD to model project environment, and used DES to model detailed process activities [24]. By applying hybrid modeling approach, the simulation model can be easily extended at project level to examine changes to policies and at process level by changing process steps and influencing the behavior of items in the process.

With the growth of software engineering research, an increasing number of software technologies have been invented. Raffo et al. developed generalized (discrete) process simulation model for evaluating the impact of IV&V activities [32] and later an automated requirements analysis tool [31] for the development process in NASA.

In Arizona State University, Rus developed an SD model to support software project planning focus on software reliability during her PhD [37], and then transformed it into an alternative DES model to support a large army development project, by modeling more detail of production phase. Crusader management believes “there is a great potential benefit for using the discrete event modeling approach not only for reliability, but for many other aspects of project management, including cost and schedule” [38].

Similar to the cases in software companies, many other enterprises in modern society continually seeks to get their processes done ‘faster, better, and cheaper’. In pursuit of
these goals, the Lending Industry XML Initiative (LIXI), a leading e-Business industry standardization body that services the lending industry in Australia, consulted National ICT Australia (NICTA) in investigating ways in which process specifications and simulations might be useful for defining standard business processes across the whole lending industry [48]. It is expected that such models can be effective as the basis for improving the speed and quality of real estate loan transactions, and reducing cost. The processes are also expected to be analyzed, simulated, and eventually mapped to implementations using software technologies. NICTA employed Little-JIL [41] for these purposes. It is noted that “preliminary work indicates that Little-JIL’s strong semantic basis renders processes defined in the language amenable to analysis and simulation by powerful finite state verification tools”. The use of such tools “offers the clearest path toward effective use of process definitions and simulations to support making LIXI processes faster, better, and cheaper”.

5. CASE STUDY

Following our research method, we carefully examined the abovementioned impact trace, then selected and invited Dr. Dan Houston to be the interviewee in this case study. The content of this impact exemplar is based on Houston and his colleagues’ relevant publications and a series of interview-styled email communications with him. Figure 4 depicts a detailed impact trace of this case study.

5.1 Research Profile

Prior to his PhD, Dan Houston worked for Honeywell as a software developer. In returning to graduate school, Houston wanted to study software engineering because “it seemed the fastest growing field of engineering”. Through the course of his master’s studies at Arizona State University (ASU), Houston wanted to take all SE courses he could get. He realized that “software engineering had a long way to go to become an engineering discipline, in no small part due to the difficulties of producing large software products that performed as expected within budget and schedule”.

When he talked to Prof. James Collofello, who oversaw the software engineering curriculum at ASU and advised a number of graduate students who wrote theses and dissertations on SPS, Houston learned his interest in applying simulation to software development process. Collofello’s interest was first sparked when Ed Yourdon came and spoke about his book, “Rise and Resurrection of the American Programmer” [43], which has a chapter on system dynamics as applied to SPS. As Collofello knew well the problems that “software projects founder upon and that they are so hard to manage due to lack of understanding of what actually goes on in a project”, he showed his enthusiasm for the subject. At one time he had about 7 students working on SPS. The students who were employed used their workplace connections to either conduct research or deploy a model being adopted in practice, e.g., Houston and Doug Sycamore (who produced a model that was used in training at Motorola).

When Houston came across Abdel-Hamid’s work [3], he recognized the power of applying simulation to software processes. Having studied simulation in industrial engineering courses, Houston was keen to apply simulation to software engineering, so seized the opportunity to work with James Collofello and his simulation instructor, Gerald Mackulak, who co-chaired his dissertation committee. In 2000, he graduated and joined the ranks of PROSIM contributors with papers on evaluating software development risk factors with simulation and on statistical analysis of SPS model.

After graduation, Houston continued working at Honeywell, moving into software process work through the Six Sigma program. He joined a team that adapted Design for Six Sigma for software engineers and led the development of that curriculum. Meanwhile, Houston maintained his interest in process simulation, and realized that “SPS remained a research topic and that its widespread usefulness had yet to be demonstrated”. His opportunity to apply SPS full-time came with The Aerospace Corporation’s interest in using system dynamics modeling to address the complex problems encountered in acquiring large, software-intensive systems. The following subsections elaborate on his industrial experiences in applying results of SPS research.

5.2 Process Simulation in Honeywell

Prior to 1999, Honeywell was primarily an automated controls company with three major businesses: avionics, process automation, and home & building controls. The products of these businesses had become increasingly software-intensive, requiring a range of software types that include real-time data analysis and control, data collection and storage, and system configuration tools. Many of the Honeywell software development organizations have process-oriented individuals and groups, e.g., Software Engineering Process Groups (SEPGs). These groups often focus their efforts on process definition and compliance, and qualitative process studies.

In 1999, Honeywell merged with AlliedSignal, a Six Sigma company, and the merged company retained the Honeywell name. The Six Sigma process improvement methodology was deployed throughout the legacy Honeywell businesses. Where process improvement work had previously been the specialization of a few, with Six Sigma it became the responsibility of all individual contributors to seek opportunities for and participate in analytically-based process improvement projects. Process mapping and statistical analysis were introduced to encourage the collection of data and quantitative analysis as a basis for process improvements.

The ‘democratization’ of software process work and the introduction of process analysis techniques created a climate of acceptance for software process simulation, though its usage was left to individual initiative. Houston provided two examples of industrial applications from this period. In the first case [13], two sets of software functionality for configuring a control system were simulated and compared for productivity savings. The study relied on business process maps and data collected through surveys of users. The simulation results were combined with upgrade investment estimates to calculate return on investment and justify an upgrade.

In the second example [14], simulating software problem report flow provided a view into the dynamics of a backlogged integration process and revealed a basic development problem. The simulation was especially helpful when alternative process 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.

5.3 Process Simulation in Aerospace

The Aerospace Corporation (Aerospace) operates a non-
profit, federally funded research and development center sponsored by the United States Air Force. Though not a software developer, Aerospace provides scientific and engineering support for launch, space, and related ground systems.

Since the late 1980s, the amount of software development for space systems has grown dramatically, requiring more software development expertise at Aerospace: “The growth of software size and complexity has been accompanied by increasing difficulties in managing software development projects. These difficulties have manifest themselves in quality problems as well as in contributions to program budget and schedule overruns.” In an effort to address these problems, Aerospace personnel began looking for new tools in the early 2000s. Through independent research and development projects, they began acquiring knowledge of SD modeling applied to software development projects and software-intensive systems acquisition programs. Slowly developing a corporate capability, Aerospace continued investing in this expertise through contracts with SD modeling experts and eventually recruited Houston as an SPS expert.

Much of the employed SPS work was funded by the corporation with the intent of growing this capability to the point that it would become funded by the services provided to program offices. The SPS work has reached that point, providing process modeling and simulation services by approximately two full-time equivalent personnel. In order to promote the work, Houston have taught a course in statistics for software engineering, with emphasis on simulation input analysis and output analysis.

By 2010, the modelers have produced 7 models founded either by research funds or program offices. One of the research-funded models is Dynamic COQUALMO [15], a defectivity profiling model based on COQUALMO. It has been used retrospectively by calibrating it to an organization’s actual profile of defects, then looking at the effects of alternative management decisions on software quality.

Most of the program-funded models are variations of a discrete event test-and-fix model. These were commissioned by program offices that posed the question, “How long will testing take?” Houston and Lieu describe such a model and its usefulness in analyzing test-and-fix processes [16]. Not only have they been able to forecast duration of testing periods, but they have also provided guidance to test managers regarding staffing, when to increase the capacity of test facilities, the order of processing test cases, and when to start successive testing stages.

6. LESSONS LEARNED

In software process simulation practice, potential customers are usually unfamiliar with using simulation as a process management tool. Consequently, they often do not understand well how simulation results can be used to support their process management decisions. However, after years of facing challenging product development project and program management problems, some see a model of process of the type they manage and instantly react with excitement as they envision possibilities for gaining better understanding of their process dynamics. Others react mostly with skepticism at the ability to obtain the data necessary to produce an accurate simulation. Still others have had experience commissioning models that take a long time to produce and end up with outdated results. These reactions represent barriers to putting software process modeling into practice, and we offer the following lessons in addressing these barriers.

Visionary clients and eager modelers can easily embrace a scope of work too large for practical modeling. One technique for managing scope is to ask a client to pose a single question to be answered by a modeling project. The modeling will provide more information than an answer to the modeling question, but the question focuses the modeling effort so that unnecessary work is avoided. For example, a model designed to answer a duration question must represent activity durations, but does not have to represent productivity explicitly, a requirement that can present a considerable data collection and/or measurement challenge.

Process modelers must also be cautious in applying simulation to every process. Some process dynamics are simple enough that they do not require the resources to produce a simulation model. On the other hand, a simple process may have dynamics that cannot be represented well in static tools such as a project estimation or management tool. A test-and-fix cycle is such a process dynamic: it is very difficult to represent in static tools but it can be represented in a simple simulation model. One of the challenges for process modelers is discovering the process dynamics in which simulation is very useful and sometimes indispensable.

To meet the challenge of those who think of simulation as requiring accurate data and who fear ‘garbage in, garbage out’, process modelers must demonstrate the value of a dia-
logue using a model with uncertain inputs. The value comes in two forms. One is simply having a tool in which process owners can identify their process steps and access parameter values. Such a trial objectifies a conversation, steering it away from opinions and toward a shared representation.

The second form of value is statistical analysis obtained from simulation that tells a process owner more than he previously knew about his process. A model should be analyzed for its sensitivity to factors and analyzed with designed experiments that demonstrate factor interactions. It should also be subjected to alternative scenarios to answer customers’ questions.

Finally, practical models are usually needed in very short time frames relative to what is required for research models. Some suggestions are provided for facilitating model production in short time frames.

- One is using small models to capture the essential dynamic pattern of a process. Madachy and Tarbet describe their use of five software process simulation models, four of them very small models [21].

- A second means of facilitating short turnarounds in modeling is to build models from a library of components that represent commonly used elements in software processes. Raffo et al. have demonstrated the utility of this approach with generic process blocks [33].

- Another alternative is use of a generic, but highly configurable model for a given process level, such as the project level. Among a number of existing software project simulators, AHM’s model [3] remains a good example of a generic and configurable project model.

7. CONCLUSION

The Impact project is tracing industrial success cases to research origins and identifying best practices for successful technology transfer. Based on the evidence, trace, and exemplar case in this study, we found for software process simulation the following factors usually critical to adoption success in industry:

- Process simulation has to be applied with other software process techniques and practices to ensure the effects. For example, many success cases identified in this study happened in the software organizations with CMM(I) Level 3 or higher certification.

- The Development of an initial process simulation model may be expensive. However, in the long-term, a configurable model structure and regular model maintenance or update turn out to be more cost-effective.

- SPS model development requires both knowledge and experience. The collaboration between researchers and practitioners is important (if not indispensable) to produce a high quality process simulation model that is able to realistically capture the problem.

Given the research method and our effort on this study, we applied impact analysis on the documented evidence. We are fully aware that our results are based on the examination of a limited number of cases. We encourage the community to provide comments and additions to our findings.

8. REFERENCES


[17] M. I. Kellner, P. H. Feiler, A. Finkelstein,


