Hybrid modeling and simulation for trustworthy software process management: a stakeholder-oriented approach

Xu Bai\textsuperscript{1}, LiGuo Huang\textsuperscript{1,*,1}, He Zhang\textsuperscript{2} and Supannika Koolmanojwong\textsuperscript{3}

\textsuperscript{1}Department of Computer Science and Engineering, Southern Methodist University, Dallas, TX 75205, U.S.A.
\textsuperscript{2}National ICT Australia, University of New South Wales, Sydney, Australia
\textsuperscript{3}Center for Systems and Software Engineering, University of Southern California, Los Angeles, CA 90089, U.S.A.

SUMMARY

Process Management Model (PMM) and Process Simulation Model (PSM) are the critical infrastructural components of the Trustworthy Process Management Framework (TPMF), which involves a large and heterogeneous group of stakeholders in process modeling and simulation to improve process trustworthiness. Process Modeling Stakeholders (PMS) have different levels of dependency on various process modeling and simulation techniques. They may also possess different perspectives or concerns in modeling. To support trustworthy process management, this paper integrates the stakeholder-oriented approach and hybrid simulation technique into software process modeling at three levels of abstraction (i.e., activity, sub-process and system). The hybrid process simulation combines micro-level discrete process models with the macro-level continuous process models to capture process dynamics. In particular, the stakeholder-oriented approach addresses the various perspectives of PMS during process modeling and simulation. Finally, a case study with a realistic process model demonstrates that this approach incrementally integrates stakeholders’ modeling concerns through hybrid simulation, which is difficult to achieve using discrete or continuous modeling/simulation techniques independently. Copyright © 2010 John Wiley & Sons, Ltd.

1. INTRODUCTION

Software trustworthiness is defined as ‘degree of confidence that the software satisfies its requirements’ [1]. ISO 9126 provides a comprehensive list of quality attributes to be taken into account when defining a set of attributes for measuring the trustworthiness of software systems. The recent trend of global software development enables large-scale software systems to be developed in a geographically distributed environment, which imposes a number of challenges on software trustworthiness [2]. In distributed large-scale software development settings, project success-critical stakeholders are much more concerned about the product trustworthiness than in traditional greenhouse development [3]. Therefore, effectively managing a large and heterogeneous group of stakeholders’ trustworthiness concerns in development process becomes a success-critical factor in assuring the delivery of a trustworthy software system. Process Trustworthiness is thus considered as an overall scale to measure process capability in developing trustworthy software [4].
The initial architecture of Trustworthy Process Management Framework (TPMF) is proposed in [4], which incorporates the Software Trustworthy Model, Process Trustworthy Model and Trustworthy Measurement Model. It is also supported by four infrastructural components, which are Requirement Management Model, Process Management Model (PMM), Risk Management Model and Process Simulation Model (PSM). The TPMF targets at better supporting the measurement and assurance of software trustworthiness through the measurement and assessment of process trustworthiness of the development process. However, these infrastructural components are still undergoing development and a number of challenges are emerging during implementation attempts.

A critical challenge is how we can engineer a trustworthy process and automatically monitor its execution to support various types of analyses (e.g., change impact analysis, risk pattern analysis, process strategy optimization) given certain process constraints and trustworthiness objectives [5]. In order to investigate the trustworthiness of a software process, a formal software process model needs to be built. Existing process modeling and simulation techniques provide a variety of mechanisms of computer-based process formalization. Many of these techniques have been proposed or discussed in the literature [6]. However, there is no ‘one-size-fits-all’ solution yet in selecting and applying these techniques in practice. All these techniques provide different process modeling capabilities in terms of their design intuitive and semantic properties. Furthermore, a diversity of Process Modeling Stakeholders (PMS) [7, 8], e.g., PMs, process engineers (PEs) and tool providers, is either involved in or affect process modeling activities. They may possess different (sometimes conflicting) perspectives and concerns of process modeling and simulation. For example, a PE may be constrained by the modeling capabilities of tools when building a descriptive process model (e.g., the artifact generated by process modeling tools), whereas a tool provider may choose to implement selected process modeling and simulation techniques to maintain its specific application scenarios instead of trying to build an all-purpose solution.

However, PMS’s roles in process modeling and simulation activities are seldom identified, as most of them are not directly involved in extracting the descriptive process models from real processes, but affecting the modeling and simulation activities indirectly by constraints, such as legislation or standards. In practice, process modeling and simulation approaches have to be selected based upon their capabilities to address PMS’s concerns at different phases and abstraction levels of software development and maintenance.

Based on the above observations, our primary research objectives are to (1) identify levels of dependencies of PMSs on process modeling and simulation techniques; (2) propose a stakeholder-oriented interpretation of software process trustworthiness; and (3) effectively combine stakeholder-oriented perspective and hybrid process modeling techniques in support of the TPMF [4].

In order to achieve these objectives, we have followed a three-step methodology to conduct our research. First, a comprehensive literature review was performed on existing process modeling and simulation techniques [6, 9]. Second, we harvested our discussion results and proposals on PMSs from the Workshop of Modeling Systems and Software Engineering Process [7] in 2008, and then we undertook a questionnaire-based survey to investigate their levels of dependency on process modeling and simulation techniques. Finally, we applied hybrid modeling and simulation approach, which combine the advantages of both discrete and continuous modeling approaches as well as minimize their limitations, as our vehicle to investigate the stakeholder-dependent properties of process trustworthiness. With respect to the findings derived from our literature review and questionnaire-based survey, we develop a stakeholder-oriented approach to model software process by incrementally integrating discrete and continuous modeling techniques for hybrid simulation. The case study shows that our stakeholder-oriented approach can efficiently adjust the process model attributes in response to the changes and dynamic stakeholder win-win conditions, accordingly to improve the process trustworthiness.

The remainder of this paper is organized as follows: Section 2 reviews the concepts of process trustworthiness and Process Modeling Stakeholders (PMS) and provides a stakeholder-oriented vision of process trustworthiness. Section 3 focuses on process modeling and simulation techniques and addresses the stakeholders’ dependencies on them. Section 4 elaborates our stakeholder-oriented process modeling approach for trustworthy process management through hybrid process simulation. Section 5 describes a case study to demonstrate our approach and then analyzes the
results. Section 6 presents the related works and discusses the limitations of our approach at this stage. Section 7 summarizes our research and proposes the future work.

2. STAKEHOLDER-ORIENTED SOFTWARE PROCESS TRUSTWORTHINESS

This section revisits the definition of Process Trustworthiness and differentiates the PMS with project and process stakeholders. Then we present the understanding of Software Process Trustworthiness from PMS’s viewpoints.

2.1. Definition of process trustworthiness

Process Trustworthiness is defined as ‘the degree of confidence that the software process produces expected trustworthy work products that satisfy their requirements’ [4]. Process trustworthiness plays as a confidence level indicator for all process stakeholders. The higher level of confidence (trustworthiness) that process stakeholders have on process implies better assurance on the delivery of trustworthy software products. Different PMSs may have varying concerns and perspectives on software processes. Each PMS depends on a particular subset of process trustworthiness attributes (such as efficiency, usability, scalability, evolvability, etc.).

2.2. Project, process and PMS

As one emphasis of our paper is a discussion of the relationship between PMS and Process Trustworthiness, this subsection first differentiates project stakeholders, process stakeholders and PMSs.

Project Stakeholders are individuals and organizations that are actively involved in a project, or whose interests may be affected as a result of project execution or project completion. For example, the project manager and end users are project stakeholders [10].

Process Stakeholders are any group or individual who can affect or is affected by the software development and/or maintenance process as a generative system [8], following which the projects are implemented. Their responsibilities are to design and implement the process, to enforce the process enactment and to improve the process. For example, PM and PE are both process stakeholders.

PMS are individuals and organizations that are involved in or affected by the course of activities relating to software process modeling and simulation [7, 8].

There exists overlapping between the roles of Process Stakeholders and PMS. Process Stakeholder may be PMS who use process modeling and simulation techniques to facilitate their decision making and achieve their process trustworthiness during actual process execution. For example, process engineer is a process stakeholder as well as PMS. In this paper, the scope of the stakeholder involved in our approach is limited to PMS.

2.3. PMS’s vision of process trustworthiness

Real-world software process modeling and simulation activities involve a heterogeneous group of stakeholders with various perspectives and different (sometimes even conflicting) goals on process trustworthiness. Ideally, one would like to have a single trustworthiness metric by which the development process could be driven, and by which the contributions of each process modeling and simulation technique could be ranked. In practice, however, such a one-size-fits-all metric is difficult to achieve.

Different systems may have distinct success-critical stakeholders involved in process modeling and simulation activities. And these stakeholders may possess their different perspectives and goals on process modeling and account for heterogeneous modeling and simulation techniques in different ways.

Thus, a critical step in understanding the nature of software process trustworthiness is to identify the success-critical stakeholder classes specifically for process modeling activities, and to
characterize the relative strengths of their dependencies on various process modeling and simulation goals [7]. This involves answering four main questions as follows.

(1) What are the primary attributes of software process trustworthiness that PMS depends on?
(2) What are PMS’s dependency patterns on these process attributes?
(3) What are the main categories of process modeling and simulation techniques in support of PMS’s research and practice?
(4) For each PMS, what is the relative strength of their dependency on each category of process modeling and simulation techniques?

3. STAKEHOLDER-ORIENTED PROCESS MODELING AND SIMULATION

In this section, we first review the related software process modeling and simulation techniques, and then discuss PMS in depth with their roles and goals. Finally, we extract PMS’s dependencies on process modeling and simulation techniques from our survey results.

3.1. Software process modeling and simulation techniques

Software process modeling and simulation techniques provide various ways to formalize, visualize and automatically execute the process, which help us measure and improve the trustworthiness of software processes. Based on the analogy that the control structures of software processes are similar to those of programming languages, Osterweil originated his well-known statement ‘Software processes are software too’ [11]. Correspondingly, Process Modeling Languages (PMLs) were defined to implement the process modeling and simulation and describe the process and capture the essential elements and/or properties of process (e.g., activities, resources, metrics) by linguistic abstraction in the workshop [7] in 2008. A subset of PMLs support certain process simulation techniques. Most of the latest PMLs, including Little-Jil, Petri Nets and System Dynamics (SD), support process visualization, process formalization and process simulation.

3.1.1. Software PMLs. The Workshop of Modeling Systems and Software Engineering Process [7] discussed discrete and continuous PMLs based on their modeling perspectives [9]. We did a comprehensive literature review [9] and analysis of widely used PMLs including Little-JIL [12, 13], Petri Nets and its variants [14], SD [15], etc., and we found that the modeling capabilities of discrete and continuous PMLs are complementary at different levels of abstraction and in addressing different stakeholders’ perspectives. The stakeholders’ win-win conditions play as the criteria of trustworthy software processes. The following software process modeling and simulation techniques can be categorized into these two common classes, respectively.

Discrete PMLs: E3 [16] provides an early object-oriented approach to process modeling, which uses pre-defined classes and relations denoted by graphical symbols. DYNAMITE is formally defined in PROGRESS, which is an executable specification language that uses UML in early implementation [17].

Melmac [18], Slang [19] and Object Petri-Net (OPN) [14] extended the traditional Petri Nets to represent the software process. Melmac represents the general structure of software process using different process views that define the data flow and constraints between software development activities. In Slang, the process is modeled as a set of hierarchical-structured activity nets. In particular, OPN supports the separation of concerns among different process modeling perspectives using the object-oriented approach.

APPL/A [20] extends Ada with features that satisfy the object management needs of process programming to allow the execution of process programs. An APT translator translates APPL/A code into Ada code, which can then be compiled and executed. JIL [21, 22], a successor of APPL/A, emphasizes on process steps and programming control with exception handling.

Little-JIL [12, 13] is a subset of JIL with visualization support. Software process is modeled as a tree of steps, whose leaves represent the units of work: steps. The structure of step tree represents
the way in which the work will be coordinated [23]. Little-JIL uses non-leaf steps to capture step ordering. Little-JIL also employs late-binding techniques in resource management.

The Eclipse Process Framework (EPF) [24] by IBM provides a customizable software process engineering framework. In EPF, users can create their own software development process by structuring it in one specific way using a predefined schema, which is an evolution of the SPEM 1.1 OMG specification referred to as the Unified Method Architecture (UMA).

Continuous PMLs: SD is an approach to model the behaviors of complex systems over time, e.g., the defect generation flow in a software development process. It has been widely applied to business and software process modeling to deal with internal feedback loops and time delays that affect the behaviors of the entire process at the system level [25, 26]. SD provides a systematic view of software process, which can capture the process dynamics and interactions among process (or project) attributes thus to probe process improvement solution by conducting sensitivity analysis on these attributes.

3.1.2. Software process simulation techniques. There have been a number of software process simulation techniques proposed during the last two decades. The systematic review [6] undertaken in 2008 revisited and updated the state-of-the-art of Software Process Simulation Modeling (SPSM) research. This research summarizes the 10 years of (1998–2007) research progress on SPSM. Totally 10 process simulation paradigms are identified in this review. Their results show that SD (SD, 49%) and Discrete Event simulation (DES, 31%) are the most widely used techniques out of the other paradigms in SPSM.

Discrete process simulation: Process models implemented with Little-JIL and OPN can be simulated based on state transition, and the outputs of modeled process can be predicted [27]. DES interprets process as a series of entities flowing through event sequences and captures the effects of variation in the entities on activities and the dependencies among activities [28]. Simulating the discrete process model can also help validate and verify constraints between the discrete events, especially constraints in resource allocation. The limitation of discrete model is that the changes inside each event (activity) cannot be captured since the activity is the minimum process advancing unit. The process attributes varying during the entire process life cycle can only be approximated at each event advance, which is considered problematic [28].

Continuous process simulation: SD was first applied in software process simulation in the late 80s [29] and then has been developed and enhanced as the major continuous simulation technique for software process research in [30–32]. In comparison to discrete process simulation, continuous approach is ideal to address the dynamic factors in process. However, it does not easily represent the individual activities in process and the constraints among these activities. For example, the process work flow structure is difficult to be modeled by continuous models. In practice, the continuous simulation model is frequently used in sensitivity analysis, as the quality of model outputs highly relies on the quality of model inputs.

Hybrid process simulation: Hybrid process simulation has become an increasingly active research area in software process simulation in the last decade. Hybrid modeling employs more than one simulation technique in developing a PSM. By combining discrete and continuous process simulations, for example, hybrid process simulation is capable of addressing both the micro-level and macro-level [33] process dynamics, and break through the limitations of applying any single simulation method. However, integrating these two simulation approaches faces the issues of compatibility, interoperability and synchronization when executing simulation.

3.2. Software PMS and process modeling artifacts

PMS was initially summarized and discussed in the Workshop of Modeling Systems and Software Engineering Process [7], including Process Performer (PP), PE, PM, Customer (CU), Educator
(ED), Tool provider (TP), Researcher (RS), Regulator (RG), Standardizer (SD) and Domain Specific Stakeholder (DS). A set of goals of process modeling and simulation were also identified in this workshop. Table I shows each PMS’s dependency on the related process modeling and simulation goals. The bullet in the cell indicates a PMS’s dependency on a process modeling and simulation goal. The roles of each PMS are elaborated as follows.

- The Process Performer (PP) is the one who performs actual process tasks/activities and generates process deliverables. He/she is assigned tasks by the PM, tries to complete assigned tasks within schedule, provides feedback to the PM when an issue arises and makes adjustment to current tasks as directed. The product and process documents are examples of process deliverables.
- The Process Engineer’s (PE) might be the most active role involved in process modeling and simulation. Their responsibilities include designing the process, choosing appropriate tools, building process models, conducting process model validation and verification, simulating process execution and optimizing process model. Process descriptive model is generated by PE using process modeling tools.
- The responsibilities of the PM are to assign activities to process performers, monitor process enactment, get feedback from process performers, make changes to the assigned activities when necessary and advance the process upon satisfying activity completion.
- The Customer (CU) relies on the process model to achieve the anticipated software product. They provide modeling constraints to the PE and evaluate the process simulation results.
- The Educator (ED) helps people learn software process theories, process modeling and simulation techniques and the usage of process modeling tools.
- The Tool Provider (TP) develops software tools for process modeling and simulation (e.g., SD and its associated software packages, e.g., Vensim [34] and iThink [35]).
The Researcher (RS) studies the software process, proposes process reference (prescriptive) models (e.g., Spiral Model, ICM) and develops process modeling and simulation techniques. His/Her roles may be overlapped with those of Tool Providers and Educators.

The Standardizer (SD) is an organization or person who stipulates or updates the standards and frameworks of software processes, such as ISO 12207 and CMMI.

The Regulator (RG) provides legal enforcement of certain subsets of process standards. For example, the Health Insurance Portability and Accountability Act (HIPAA) has been enacted to enforce the security and privacy-related standards for electric health-care records.

The Domain Specific Stakeholder (DS) is the one who works together with other process stakeholders to provide domain expertise. However, he/she may not be the expert on process modeling and simulation.

Process Modeling Artifacts are either final or intermediate work products produced and used during a project. They capture project information including Process Reference Models, Process Descriptive Models, Process Standards, Process Regulations, Process Modeling Tools and Process Deliverables. They may take various forms or representations.

Figure 1 illustrates the PMS, process artifacts and their relationships. PMS and process artifacts are denoted by *figurines* and *rectangles*, respectively. Each solid arrow indicates that a process artifact is produced by a corresponding PMS. Each double-lined arrow represents that a process artifact is used by a corresponding PMS.

### 3.3. Stakeholders’ dependencies on PMLs

Based on their roles and activities in process modeling and simulation, PMS may have different levels of dependencies on discrete and continuous PMLs. Table II [9] shows various levels of dependencies of PMS on discrete and continuous PMLs derived from our questionnaire-based survey [36], which collects the qualitative assessment from the population, including software process researchers, practitioners and educators. We use ‘very low’, ‘low’, ‘medium’, ‘high’ or ‘situation dependent’ to indicate various levels of dependencies.

Different PMSs may have concerns on either *micro-level* or *macro-level* of process [33]. For example, process performers are assigned tasks/activities by the PM, perform the activities toward
Table II. Levels of dependencies of process modeling stakeholder (PMS) on discrete and continuous PMLs.

<table>
<thead>
<tr>
<th>Stakeholder classes</th>
<th>Discrete PMLs</th>
<th>Continuous PMLs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Process Performer (PP)</td>
<td>High</td>
<td>Low</td>
</tr>
<tr>
<td>Process Engineer (PE)</td>
<td>High</td>
<td>High</td>
</tr>
<tr>
<td>Process Manager (PM)</td>
<td>High</td>
<td>High</td>
</tr>
<tr>
<td>Customer (CU)</td>
<td>Situation dependent</td>
<td>Situation dependent</td>
</tr>
<tr>
<td>Educator (ED)</td>
<td>High</td>
<td>Medium</td>
</tr>
<tr>
<td>Tool Provider (TP)</td>
<td>Medium</td>
<td>Situation dependent</td>
</tr>
<tr>
<td>Researcher (RS)</td>
<td>High</td>
<td>High</td>
</tr>
<tr>
<td>Regulator (RG)</td>
<td>Very low</td>
<td>Situation dependent</td>
</tr>
<tr>
<td>Standardizer (SD)</td>
<td>Situation dependent</td>
<td>Situation dependent</td>
</tr>
<tr>
<td>Domain Specific (DS)</td>
<td>Situation dependent</td>
<td>Situation dependent</td>
</tr>
<tr>
<td>Stakeholder</td>
<td>Situation dependent</td>
<td>Situation dependent</td>
</tr>
</tbody>
</table>

the deadline, provide feedback to the PM and adjust the activities if necessary. They pay more attentions on the micro-level process activities and count more upon discrete PMLs that describe activity-related attributes. However, they might be less concerned with the process attributes on macro-level. PEs play active roles in process modeling and simulation. Their responsibilities include designing the process, choosing the PML, building and verifying the process model, simulating process execution and optimizing the process. Thus they may rely on both discrete and continuous PMLs in order to achieve their goals (e.g., optimizing resource allocation, minimizing system defect density) at both micro-level process activities and macro-level view of the entire process.

4. A STAKEHOLDER-ORIENTED HYBRID MODELING APPROACH FOR TRUSTWORTHY PROCESS MANAGEMENT

PSM and PMM are two critical infrastructural components of TPMF [4]. In order to support the change impact analysis and investigation of process optimization strategies based on the trustworthiness objectives from various stakeholders’ perspectives, we propose a stakeholder-oriented approach to develop the process model using hybrid process simulation techniques. The optimized process model can provide runtime instantiation and management support for PMM. By implementing and supporting these two critical components (PSM and PMM) of TPMF, our approach demonstrates pertinent implementation toward trustworthy process management.

4.1. Stakeholder-oriented hybrid process modeling

Examples of PMS’s perspectives include workforce modeling, earned value evaluation, software evolution, software reuse, quality and defects, requirement volatility [15] and other process-specific perspectives. Based on the behavior analysis of PMS’s roles in process modeling activities [9], Table III summarizes PMSs and their process modeling perspectives. The acronyms of PMS are reused from Section 3.2. The perspectives are classified into three categories: People, Process/Product and Project. The bullet indicates the PMS and their perspectives in continuous process modeling.

To integrate PMS’s concerns into a process model, we adopt a hybrid approach to construct the process model at multiple levels of abstraction based on PMS’s perspectives. There exist two broad approaches to construct a hybrid process model [37]: vertical integration, which primarily implements discrete modeling at the lower activity level, then continuously calculates the process attributes and incorporates the feedback loops at the system level [38]; and horizontal integration, in which the sub-processes or phases within a large-scale and/or complex software process may be modeled using different approaches respectively and sequentially, and the data flow has to be...
A STAKEHOLDER-ORIENTED APPROACH

Table III. Continuous process modeling perspectives and their associated stakeholder classes.

<table>
<thead>
<tr>
<th>Continuous process modeling perspectives</th>
<th>Associated stakeholder classes</th>
</tr>
</thead>
<tbody>
<tr>
<td>People</td>
<td>PP</td>
</tr>
<tr>
<td>Workforce modeling</td>
<td></td>
</tr>
<tr>
<td>Exhaustion and burnout</td>
<td></td>
</tr>
<tr>
<td>Learning</td>
<td></td>
</tr>
<tr>
<td>Team composition</td>
<td></td>
</tr>
<tr>
<td>Process/Product</td>
<td>PP</td>
</tr>
<tr>
<td>Inspection</td>
<td></td>
</tr>
<tr>
<td>Software evolution</td>
<td></td>
</tr>
<tr>
<td>Software reuse</td>
<td></td>
</tr>
<tr>
<td>COTS</td>
<td></td>
</tr>
<tr>
<td>Software architecting</td>
<td></td>
</tr>
<tr>
<td>Quality and defect</td>
<td></td>
</tr>
<tr>
<td>Requirement volatility</td>
<td></td>
</tr>
<tr>
<td>Process improvement</td>
<td></td>
</tr>
<tr>
<td>Project</td>
<td>PP</td>
</tr>
<tr>
<td>Integrated project modeling</td>
<td></td>
</tr>
<tr>
<td>Business case analysis</td>
<td></td>
</tr>
<tr>
<td>Personnel resource allocation</td>
<td></td>
</tr>
<tr>
<td>Staffing</td>
<td></td>
</tr>
<tr>
<td>Earned value</td>
<td></td>
</tr>
</tbody>
</table>

Figure 2. Overview of incremental process modeling with hybrid simulation from different stakeholders’ perspectives.

converted at the interface between them for process phase (state) transition [39]. However, no research has been done to combine the vertical and horizontal integration in order to address the PMS’s concerns at different levels of abstraction in process modeling and simulation.

Figure 2 illustrates the overview of our stakeholder-oriented modeling and simulation approach. The horizontal axis represents the process time line and the vertical layers represent different levels of abstraction with respect to different PMSs’ perspectives. The dependencies among activities are captured by discrete process models. And continuous models are built upon stakeholder’s perspectives at the relevant abstraction levels.

4.2. Hybrid modeling based on stakeholders’ perspectives

The primary objective of our stakeholder-oriented process modeling is to effectively integrate various stakeholders’ concerns on process trustworthiness using hybrid modeling and simulation. Our approach aims to improve the process adaptability to internal (process dynamics) or external (environmental) changes by incrementally integrating the affected process attributes into the process model at multiple levels of abstraction in terms of PMS’ perspectives. The hybrid approach uses the continuous PML to model changes in the affected process attributes, and provide feedbacks to adjust the corresponding attributes in the discrete process models [39].

At the activity level, we model each process activity with both discrete and continuous PMLs, and the discrete and continuous simulations synchronously. Note that, as activities in discrete model are treated as events with starting and ending points, the attributes’ values of discrete model and continuous model are simulated and compared at the end of the activities. By comparing these simulated values across the discrete and continuous models built upon each activity, we can adjust the corresponding process attributes in the discrete model toward a desired or optimized process performance.

At the sub-process level, where PMS’s perspectives may cover several activities, the hybrid simulation enables the feedback loop within the involved process activities in the sub-process, so that the related process attributes across the activities can be adjusted. Also, the results can be used as inputs to the subsequent phases.

At the system level, the entire process performance can also be measured and optimized using a similar approach.

Overall, our hybrid modeling approach attempts to combine the horizontal and vertical [39] integration approaches to address process trustworthiness from different stakeholders’ perspectives. It improves the quality of process model by incrementally and timely adjusting affected process attributes and quickly adapting to environment changes at three abstraction levels for achieving process optimization (as shown in Figure 2).

4.3. Hybrid simulation based on stakeholders’ perspectives

At the activity level, multiple continuous models can be constructed for an individual activity with respect to various stakeholders’ perspectives. For example, stakeholder A (e.g., PM) in Figure 2 concerns with a set of process attribute \( PA_a \) (e.g., Schedule, Earned Value, etc.) for activity \( a \). Each process attribute can be simulated by a different continuous component. Both discrete and continuous models are simulated synchronously with respect to activity \( a \) in order to verify or optimize \( PA_a \). Based on the continuous model simulation results, if a certain process attribute (e.g., Schedule) in \( PA_a \) cannot satisfy stakeholder A’s constraint defined in the discrete model based, a set of related process attributes (e.g., Productivity, Budget) will be adjusted accordingly and fed into the discrete and continuous models to redo the hybrid simulation. This step reiterates until stakeholder A’s schedule constraint is satisfied. Such a local optimization improves the process model by verifying or optimizing process attributes for each activity.

At the sub-process level, a stakeholder with his/her modeling perspectives (see Table III) may concern with a set of process attributes with respect to several activities across phases. For instance, stakeholder C (e.g., PE) is interested in the process attribute set \( PA_c \) (including Schedule, Defects generated, etc.) with respect to a set of activities in phases 2 and 3 as shown in Figure 2. A sub-process level hybrid simulation can be performed on the combined phases 2 and 3 by developing the continuous models simulating C’s perspectives and discrete model covering all the activities in these two phases. The dependency relationships among activities from the discrete process model are captured in the continuous model.

At the system level, hybrid simulation is similar to the sub-process level simulation except that it is based on the perspectives of stakeholders whose concern is about the entire process. This is illustrated as stakeholder A’s (e.g., PM) system level perspective (e.g., Overall schedule, Residual defect number, etc.) in Figure 2.
4.4. Information interchange

In order to integrate the discrete and continuous process models based on a particular modeling/simulation perspective, we need to identify and model the information flow between them. We define $\text{Attribute}_D$ as process attributes modeled by the discrete model and $\text{Attribute}_C$ as process attributes modeled by the continuous model. $\text{Attribute}_H$ is defined as the process attributes that a specific PMS is interested in and can be modeled in both discrete and continuous process model. Then we have

$$\text{Attribute}_H \subseteq \text{Attribute}_D \cap \text{Attribute}_C$$

where $\text{Attribute}_H$ can be either numeric or constraint attributes. These attributes may be represented in different forms in discrete and continuous models. For instance, a numeric attribute can be a discrete value in discrete models while being a continuous distribution in continuous model.

In discrete models, the numeric attributes of a process activity usually have fixed values during the execution of the activity, as discrete modeling treats each activity as the smallest advancing unit. For example, the manpower allocation or total workload is assigned a fixed value for a process activity in discrete models. In continuous models, a process attribute can be modeled as a discrete value or statistical distribution. For example, Productivity is usually modeled as a continuous distribution over time. The constraint attributes define the modeling scope of a PMS’s perspective, such as design review, defect detection, etc.

To implement hybrid process model, first we need to identify those attributes that are semantically equivalent but in different forms in both discrete and continuous models. We define $\text{Attribute}_I$ as attributes having the identical forms in both discrete and continuous model. Attributes having distinct forms in either model are defined as $\text{Attribute}_D$ in the discrete model and $\text{Attribute}_C$ in the continuous model, respectively. Thus, we have

$$\text{Attribute}_H = \text{Attribute}_I \cup \text{Attribute}_D \cup \text{Attribute}_C$$

Let $R_D$ and $R_C$ represent the relationships among these attributes in discrete model $D$ and continuous models $C$, respectively. We have the hybrid simulation equations in the following forms for model $C$ and $D$, where $E_D$ is the discrete simulation equation and $E_C$ is the continuous simulation equation. Thus,

$$E_D = R_D(t, \text{Attribute}_I, \text{Attribute}_D)$$

and

$$E_C = R_C(t, \text{Attribute}_I, \text{Attribute}_C)$$

where $E_D$ and $E_C$ are simulated synchronously on time $t$.

5. CASE STUDY

Table IV summarizes the most supported drivers for simulation study of software process [37], in which the rightmost column indicates the percentage of studies including the leftmost output in the recent 10 years’ studies. The case study is designed for modeling and simulating process attributes including schedule, effort, quality, plan and productivity process attributes, which are top prioritized drivers in Table IV, as well as the critical attributes to assure process trustworthiness.

5.1. The process models and baseline project

The ISPW-6 software development process is a standard software process modeling example, where the core is a relatively confined portion of the software change process, focusing on the designing, coding, unit testing and management of a localized change to a software system [40]. We use ISPW-6 to illustrate how to use our stakeholder-oriented approach to incrementally model
Table IV. Summary of simulation outputs.

<table>
<thead>
<tr>
<th>Output</th>
<th>Description</th>
<th>Per cent</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time</td>
<td>Project schedule or elapsed time</td>
<td>35.7</td>
</tr>
<tr>
<td>Effort</td>
<td>Effort or cost</td>
<td>28.6</td>
</tr>
<tr>
<td>Quality</td>
<td>Product quality or defect level</td>
<td>19.6</td>
</tr>
<tr>
<td>Size</td>
<td>Requirement size or functionality</td>
<td>19.6</td>
</tr>
<tr>
<td>Resource</td>
<td>Resource or staffing level</td>
<td>12.5</td>
</tr>
<tr>
<td>Plan</td>
<td>Project or development plan</td>
<td>5.4</td>
</tr>
<tr>
<td>ROI</td>
<td>Return on investment or cost/benefit analysis</td>
<td>3.6</td>
</tr>
<tr>
<td>Productivity</td>
<td>Development productivity</td>
<td>1.8</td>
</tr>
<tr>
<td>Market share</td>
<td>Product market share</td>
<td>1.8</td>
</tr>
<tr>
<td>Index</td>
<td>Nominal index</td>
<td>1.8</td>
</tr>
<tr>
<td>Behavior</td>
<td>Behavior patterns</td>
<td>1.8</td>
</tr>
<tr>
<td>Flow</td>
<td>Process flow</td>
<td>1.8</td>
</tr>
</tbody>
</table>

Table V. Baseline project and project change information.

<table>
<thead>
<tr>
<th>Project baseline information attributes</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project Size</td>
<td>90,667 LOC</td>
</tr>
<tr>
<td>Increment 1</td>
<td>22,667 LOC</td>
</tr>
<tr>
<td>Increment 2</td>
<td>32,000 LOC</td>
</tr>
<tr>
<td>Increment 3</td>
<td>32,000 LOC</td>
</tr>
<tr>
<td>Schedule</td>
<td>250 Days</td>
</tr>
<tr>
<td>Team Size</td>
<td>15 Engineers</td>
</tr>
<tr>
<td>Estimated Budget</td>
<td>3750 Man-Days</td>
</tr>
<tr>
<td>Nominal Development Productivity</td>
<td>40 LOC/Man-Day</td>
</tr>
<tr>
<td>Nominal Defect Generation Rate</td>
<td>33 errors/KLOC</td>
</tr>
<tr>
<td>Nominal Defect Regeneration Rate</td>
<td>4:1</td>
</tr>
<tr>
<td>Nominal Defect Detection Rate</td>
<td>0.84</td>
</tr>
<tr>
<td>Nominal Review Productivity</td>
<td>220–1100 LOC/Man-Day</td>
</tr>
<tr>
<td>Nominal Test Productivity</td>
<td>40 LOC/Man-Day</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Project change information attributes</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Estimated Change Size</td>
<td>5000 LOC</td>
</tr>
<tr>
<td>Estimate Change Budget</td>
<td>250 Man-Day</td>
</tr>
</tbody>
</table>

In the process by integrating the discrete and continuous PMLs in the hybrid simulation. In the case study, we focus on schedule and defect dynamics of the software process, which are two important attributes of process and (software) product trustworthiness. The empirical data used for this case are shown in Table V [32]. The baseline project core has 80,000 lines of code (LOC) in COBOL and 10,667 LOC as the increment overheads. The project was scheduled for one calendar year with 15 full-time engineers. A moderate change occurred late in the development phase due to the requirement volatility, an additional 5000 LOC workload needs to be modeled to verify the schedule and quality constraints. Table VI shows the planned schedule and 10 engineers are allocated to the rework activities.

In this scenario, we are going to integrate three PMSs’ (PP, PM and PE) concerns using our approach. The PP needs to follow the schedule assigned by the PM. The PM wants to verify and maintain the planned deadlines for each activity and the entire process. And the PE tries to estimate the residual defect density in the delivered product to ensure overall system quality is satisfied. The results from Table II indicate that the PP is highly dependent on the discrete PMLs to obtain a detailed activity information but not much on the continuous PMLs, whereas the PM and the PE require both discrete and continuous PMLs to satisfy the process and product trustworthiness at sub-process level (e.g., schedule and defect dynamics). In Table VI, the project manager’s role
A STAKEHOLDER-ORIENTED APPROACH

Table VI. Planned schedule for change.

<table>
<thead>
<tr>
<th>Activity</th>
<th>Man power</th>
<th>Days</th>
</tr>
</thead>
<tbody>
<tr>
<td>Schedule and Assign Activities</td>
<td>1 Project Manager</td>
<td>1</td>
</tr>
<tr>
<td>Modify Design</td>
<td>6 Design Engineers</td>
<td>8</td>
</tr>
<tr>
<td>Review Design</td>
<td>4(1 DE, 1 QA, 2 Other)</td>
<td>1</td>
</tr>
<tr>
<td>Modify Code</td>
<td>6 Design Engineers</td>
<td>10</td>
</tr>
<tr>
<td>Modify Test Plans</td>
<td>4 QA Engineers</td>
<td>3</td>
</tr>
<tr>
<td>Modify Unit Test Package</td>
<td>4 QA Engineers</td>
<td>8</td>
</tr>
<tr>
<td>Test Unit</td>
<td>10 (6 DE, 4 QA)</td>
<td>10</td>
</tr>
</tbody>
</table>

Figure 3. Discrete model (a), and continuous models in earned value (b), in defect dynamics (c) of ISPW-6.

may include PP, PM and PE, who needs to enact the process, manage the process and try to improve the process. On the other hand, both design engineer and QA engineer are assumed as only acting as PP in this case.

We used Little-JIL to present the discrete model of the ISPW-6 process [40] (as shown in Figure 3(a)). Each leaf in this tree-like model represents the corresponding activity defined in ISPW-6. The sequencing badges in parent steps indicated whether their children steps can be executed in sequence or in parallel. Little-JIL enforces the critical path interdependencies among activities and we applied SD for continuous modeling. Figure 3(b) shows the SD sub-model (built with iThink [35]) for each activity. It helps the PM to verify the planned deadlines from his/her earned value perspective after the change occurred. Figure 3(c) shows the PE’s perspective on defect dynamics, which is at the sub-process level covering Modify design, Review design, Modify code, Modify unit test plan and Test unit activities of the process.
5.2. Deadline verification and adjustment

The PM intends to verify if each activity and the entire process can meet the planned schedule in Table VI and makes adjustment when necessary. Both discrete and continuous PMLs are used to verify and maintain the deadline of this process from the earned value perspective in Table III.

Activity level hybrid simulation: Little-JIL is used to model dependencies among activities and the planned process attributes. SD is used to model the cumulative workload along the time line for each activity. We input the identical process attributes, i.e., planned manpower and work duration, to both the discrete process model and the continuous process model. The productivity in discrete process model is given in Table V. Figure 5(a) shows the productivity distribution over time from the SD process model. We simulate both models for each activity to compare the cumulative workload at the deadline $t$. Figure 5(b) and (c) show that the simulated Modify design and Modify test plan activities (the solid line) are behind the planned schedule (the dash line).

Thus, the PM decides to adjust the planned process attributes in the discrete process model. In this case, as the Modify design activity is on the process critical path (shown in Figure 4), its deadline cannot be extended. Since its manpower allocation is fixed, increasing the designers’ productivity becomes the only option. For Modify test plan activity, as it is not on the critical path, we can either extend its deadline within its slack days or increase productivity by postponing the activity close to the ‘hard’ deadline on the critical path. Assuming that we choose the latter solution due to limited budget, the cumulative workload of Modify test plan is recalculated during hybrid simulation, as if the Modify test plan were postponed up to 6 days while keeping its planned work duration. Figure 5(d) shows that the adjusted cumulative workload (the solid line) satisfies the planned cumulative workload (the dash line) when the planned schedule is postponed by 4 days.

There is no change on the process critical path in the above scenario. However, extending the activity deadline is the only option to complete the planned workload, the process critical path might be changed due to the interdependencies among activities. For example, if the deadline of Modify test plan activity has to be extended, the Modify unit test package activity also has to be postponed. Thus, the adjustment at activity level alone cannot guarantee this sub-process to meet its overall deadline. In this condition, we also need to simulate at the sub-process level to satisfy the PM’s needs.

Sub-process level hybrid simulation: To verify the overall deadline, the discrete process simulation provides the manpower allocation and activity dependencies to the continuous process simulation. The continuous process model is used to calculate the cumulative work along the time line for the entire sub-process. Based on our simulation result, the planned deadline of this process segment can still be met after the suggested changes by the activity-level hybrid simulation.

5.3. Defect dynamics: generation, detection and density control

The PE investigates the quality factors of the ISPW-6 process to verify whether the residual defect density is within the acceptable range, while the original deadline can be maintained. His/Her concern is from defect dynamics perspective as shown in Table III. Modify design activity and
Modify code activity generate defects, whereas Design review activity and Test unit activity detect defects.

Sub-process level hybrid simulation: The number of residual defects from Design review activity is amplified 4 times when they escape into Modify code activity due to defect regeneration effect. The review productivity (220-1100 LOC/Man-Day), test productivity (40 LOC/Man-Day), defect generation rate (33 defects/KLOC) and defect detection rate (0.84) are obtained from Table V. Using these process attributes and time as input to sub-process level SD model as shown in Figure 3(c), we get Figure 5(e) that shows the defect distribution (the solid line) along the time line, and there are 14 residual defects (the dash line) at the 30th day deadline. The calculated residual defect density is 2.8 defects/KLOC. Assuming that the acceptable residual defect density is 3 defects/KLOC, the simulation result verified that the quality of product is acceptable while the planned deadline is met. However, if the acceptable residual defect density is 2.5 defects/KLOC, the quality requirement is not satisfied based on the simulation result. In this case, adjustments to process attributes at the activity level is needed to assure the required quality.

Activity level hybrid simulation: To decrease the number of residual defects as well as to maintain the planned deadline, we can either reallocate manpower resource to increase defect detection rate in Design review activity and Test unit activity or decrease defect generation rate in Modify design activity and Modify code activity. As the manpower in this project is fully used in Modify design, Modify code, Modify test plan and Test unit, we could only adjust resource allocation in the Design review team to increase the defect detection rate. To satisfy the quality requirement of 2.5 defects/KLOC, additional 0.325 defects have to be detected in Design review, which account for additional 1.5 escaped defects at the deadline. By simulating the hybrid model of Design review activity with planned defect detection rate and review productivity, the number of detected defects is calculated as 53.22 defects shown in Figure 5(f) by the solid line at the deadline. Thus, the adjusted Design review activity should detect 53.545 defects (the dashed line) to meet the overall quality requirement. Feeding this into the hybrid process model for the Design review activity at the activity level, we found that the defect detection rate should be increased to 0.845. Using
this as the criteria to reallocate the resources, the project manager can make sure that the overall deadline and quality requirement are both satisfied.

5.4. Summary

In our ISPW-6 case study, we interpret and integrate stake-holders’ (PP, PM and PE) perspectives during a small software change process in terms of the hybrid process simulation. Our stakeholder-oriented hybrid simulation approach provides an integrated method to find, evaluate and cross-validate possible decision-making activities with varying concerns during software process, which increases the confidence levels of process stakeholders by generating evidences from our simulation results. Overall, this approach instantiates the Process Simulation Model (PSM) and Process Management Model (PMM) in TPMF to support the trustworthiness objectives from various stakeholders.

6. RELATED WORKS AND DISCUSSION

6.1. Hybrid software process models for simulation

Martin and Raffo analyzed the manpower utilization using hybrid simulation model [38], which presents how workforce levels vary on the basis of the changes in a set of factors. Their hybrid simulation model was developed to maintain the consistency between the discrete activities and the change of continuous portions. This hybrid model allows investigation of the effects of discrete resource changes on continuously varying productivity, and the influences of discrete activities on continuously varying defect rates.

Rus et al. [41] and Lakey [42] modeled the software process as a set of discrete activities. To achieve continuous modeling and incorporate the feedback loops of SD, they divided the inputs by five, and iterated the activity five times. This solution improves the continuity of calculating the process factors and performances, such as implementation size, defect level and workforce. It also makes it possible to explicitly analyze the performance of a single discrete process, which is regarded as a difficult task in pure continuous simulation, e.g., SD [38]. However, such an approximation of continuous models is coarse, and the time advance might be different between discrete phases.

Choi et al. [43] proposed their hybrid modeling approach by constraining the time advance of discrete-event model to be a small enough constant-time, which enables them to model differential equation system with discrete-event modeling. Therefore, their solution is an enhancement to Rus and Lakey’s approach with the refined time advance, which presents the feedback mechanism of SD more effectively in a discrete-event framework.

Different from the above vertical hybrid modeling solutions, Zhang et al. [39] modeled and simulated a test-and-fix process in incremental software development by implementing a horizontal integration structure. This approach ensures the integrity of PSM, while allowing the process knowledge at different levels of abstraction among the phases.

Table VII summarizes the characteristics of these hybrid modeling solutions. These solutions employ continuous and discrete modeling at different levels, i.e., modeling portions of continuous model with discrete approach, or using finer time advance for modeling discrete activities to observe the continuously varied process performance.

6.2. Stakeholder/value-based software process simulation

In [44], product/process tradeoffs were assessed for economic business case analysis via process simulation based on various stakeholders’ value functions, opposing market factors and business constraints. In [14], OPN was used to model the separations of concerns for different stakeholders’ value propositions in value-based software quality achievement process.
A STAKEHOLDER-ORIENTED APPROACH

Table VII. Hybrid software process models for simulation.

<table>
<thead>
<tr>
<th>Authors</th>
<th>Model framework</th>
<th>Module/Component</th>
<th>Package</th>
</tr>
</thead>
<tbody>
<tr>
<td>Martin and Raffo</td>
<td>Continuous</td>
<td>Discrete</td>
<td>Extend</td>
</tr>
<tr>
<td>Rus et al. and Lakey</td>
<td>Discrete</td>
<td>Continuous</td>
<td>Extend</td>
</tr>
<tr>
<td>Choi et al.</td>
<td>Discrete</td>
<td>Continuous</td>
<td>DEVSim++</td>
</tr>
<tr>
<td>Zhang et al.</td>
<td>Discrete</td>
<td>Continuous/discrete</td>
<td>Extend</td>
</tr>
</tbody>
</table>

Biffl et al. [10] applied process simulation techniques to complex project situations in order to account for business value loss due to the combined materialization of typical project risks including time and budget overruns or lack of product quality.

6.3. Limitations

Compared to a real-world complex software process model, the application through the ISPW-6 case study is relatively small at the current stage, which only involves a limited number of stakeholder’s perspectives.

Our current approach is not fully automatic yet, which still requires manual intervention to identify stakeholders’ perspectives in hybrid process modeling and simulation without intelligent process modeling knowledge base built from the previous experience.

In addition, the accuracy of continuous model simulation highly relies on the model inputs. In some cases where these model inputs are inaccurate or unavailable, the continuous model simulation part can only be used as sensitivity analyzer of certain process attributes.

7. CONCLUSIONS AND FUTURE WORK

In this paper, we summarize the existing hybrid process simulation approaches and extend the capability of hybrid simulation to address PMSs’ perspectives as well as inevitable changes in process evolution. We develop a stakeholder-oriented hybrid process modeling approach to model and simulate a software process by integrating discrete and continuous methods. This hybrid process modeling approach is capable of supporting the trustworthy software process management with emphasis on stakeholders’ concerns. In the case study, we applied this approach for validating project deadline, adjusting schedule and monitoring residual defects successfully at both activity and sub-process levels. The example application covers the top prioritized process drivers and the hybrid simulation model provides powerful analysis capability especially in addressing the different perspectives of stakeholders during the software development process.

The future research can be carried out in two directions: (1) to extend this hybrid modeling scheme to address other essential process issues or subject areas, such as software reuse and evolution; (2) to enhance the automation of this hybrid modeling and simulation approach by automatically identifying the interchanging information flow and applying AI approaches to build knowledge/experience base in support of decision-making during hybrid process modeling simulation.

ACKNOWLEDGEMENTS

He Zhang’s research is supported by NICTA, which is funded by the Australian Government as represented by the Department of Broadband, Communications and the Digital Economy and the Australian Research Council through the ICT Centre of Excellence program, and also supported, in part, by the Science Foundation Ireland grant 03/CE2/1303 1 to Lero—the Irish Software Engineering Research Centre (www.lero.ie).
REFERENCES


A STAKEHOLDER-ORIENTED APPROACH

34. Systems V. Vensim for ventana systems. Available at: http://www.vensim.com/ [15 November 2008].

AUTHORS’ BIOGRAPHIES

**Xu Bai** is currently a PhD Candidate under the direction of Dr. LiGuo Huang in the Computer Science and Engineering Department of Southern Methodist University, TX, U.S.A. He received his MS degrees in Advanced Computer Science from University of Manchester, UK in 2004 and received his Bachelor’s degree from the Tsinghua University, Beijing, P.R.China in 2002. His current research interests include Software Engineering, Software Process and especially in Software Process Modeling, Simulation and Automation. He has been asked to serve as a reviewer for journal and conference articles in software engineering, including JSME, ICSP and ESEM. He has also served as research/teaching assistant in the Lyle School of Engineering, Southern Methodist University, TX, U.S.A. He is a member of ACM and IEEE.

**LiGuo Huang** is Assistant Professor of Computer Science and Engineering Department at the Southern Methodist University (SMU). She received both her PhD (2006) and MS from the Computer Science Department and Center for Systems and Software Engineering at the University of Southern California (USC). Her current research interests include process modeling and improvement, mining systems and software engineering repository, software quality assurance, value-based systems and software engineering, and software metrics. She has been intensively involved in research on value-based integration of systems and software engineering and has published related papers in IEEE Computer and IEEE Software. She is currently a member of the IEEE.
He Zhang is a researcher with National ICT Australia (NICTA) and a conjoint lecturer with University of New South Wales (UNSW). Prior to joining NICTA, he was a research fellow at Lero, The Irish Software Engineering Research Centre. He joined academia after 7 years experience in the industry, developing software systems in the areas of aerospace, manufacturing, and complex data management. He received his PhD in Software Engineering from UNSW, with doctoral research that applied simulation modeling to study software engineering processes. He also holds an M.PM. from the University of Sydney, a MSc from the Chinese Academy of Astronautics, and a BEng from Nanjing University of Aeronautics and Astronautics in China. His research focuses on software process modeling, analysis and simulation, evidence-based software engineering, empirical software engineering, service-oriented architecture and software project management. He is a member of ACM SIGSOFT.

Supannika Koolmanojwong is currently a PhD student in the Computer Science Department at the University of Southern California. She is a member of the Center for Systems and Software Engineering. Her advisor is Professor Barry Boehm. Her primary research area focuses on software process modeling and software process improvement especially in rapid-fielding software development, Architected Agile, NDI-Intensive, and Net-Centric Services areas. Prior to this, she was a lecturer at Faculty of Science and Technology, Assumption University, Thailand and a RUP/OpenUp Content Developer at IBM Software Group.