# Using Measurement and Simulation for Understanding Distributed Development Processes in the Cloud

Ilaria Lunesu
University of Cagliari
Department of Electrical and Electronic Engineering
ilaria.lunesu@diee.unica.it

Jürgen Münch
Reutlingen University
Business Informatics & Herman Hollerith Center
j.muench@computer.org

Michele Marchesi
University of Cagliari
Department of Mathematics and Computer Science
marchesi@unica.it

Marco Kuhrmann
Clausthal University of Technology
Institute for Applied Software Systems Engineering
kuhrmann@acm.org

## ABSTRACT

Organizations increasingly develop software in a distributed manner. The Cloud provides an environment to create and maintain software-based products and services. Currently, it is widely unknown which software processes are suited for Cloud-based development and what their effects in specific contexts are. This paper presents a process simulation to study distributed development in the Cloud. We contribute a simulation model, which helps analyzing different project parameters and their impact on projects carried out in the Cloud. The simulator helps reproducing activities, developers, issues and events in the project, and it generates statistics, e.g., on throughput, total time, and lead and cycle time. The aim of this simulation model is thus to analyze the tradeoffs regarding throughput, total time, project size, and team size. Furthermore, the modified simulation model aims to help project managers select the most suitable planning alternative. Based on observed projects in Finland and Spain, we simulated a distributed project using artificial and real data. Particularly, we studied the variables project size, team size, throughput, and total project duration. A comparison of the real project data with the results obtained from the simulation shows the simulation producing results close to the real data, and we could successfully replicate a distributed software project. By improving the understanding of distributed development processes, our simulation model thus supports project managers in their decision-making.

## CCS CONCEPTS

• **Computing methodologies** → **Modeling and simulation**; • **Software and its engineering** → **Rapid application development**; **Agile software development**; **Collaboration in software development**; **Programming teams**;

## KEYWORDS

Software process development; distributed development; process simulation; simulation model; Cloud; Scrumban

**ACM Reference format:**
Ilaria Lunesu, Jürgen Münch, Michele Marchesi, and Marco Kuhrmann. 2017. Using Measurement and Simulation for Understanding Distributed Development Processes in the Cloud. In *Proceedings of 27th International Workshop on Software Measurement and 12th International Conference on Software Process and Product Measurement, Gothenburg, Sweden, October 25–27, 2017 (IWSM/Mensura '17),* 11 pages.
https://doi.org/10.1145/3143434.3143462


*IWSM/Mensura '17, October 25–27, 2017, Gothenburg, Sweden*

ACM ISBN 978-1-4503-4853-9/17/10…$15.00
https://doi.org/10.1145/3143434.3143462

## 1 INTRODUCTION

Being able to collaborate effectively has become a crucial factor of software development and maintenance. Organizations increasingly develop software in a distributed manner by appointing external developers and development teams, who collaboratively work at different sites utilizing a multitude of communication tools [4, 18]. Literature shows distributed software development being challenged by many factors, e.g., distance in language, culture, time and location, coordination of distributed (virtual) teams, and lack of trust among developers [9, 20]. Notably agile software development constitutes a challenge, as agile software development relies on a set of principles and values that put the people and close collaboration and interaction in the spotlight. Therefore, it is crucial to understand how agile methods "behave" in distributed software development, and how to tailor them. Adapting and deploying an agile method to a project spanning several sites bears some risk [14]. Therefore, a simulation-based approach grounded in statistical data from previous projects can help analyzing risks and evaluating different process variants, but also helps evaluating decisions and potential effects on a project. Moreover, a process simulation offers insights faster than a full case study. In particular, a *simulation model* can be modified and the results quickly provide indication whether or not modified parameters affect a project. For example, while it is hard to modify the team in a "real" project, in a simulation, modifying the parameter team size helps analyzing the impact, e.g., on work-in-progress (WIP) or lead/cycle time. Eventually, a process simulation is a tool to help project managers analyzing different actions, evaluating impact, and eventually selecting those actions best fitting a particular situation.

*Problem Statement.* Distributed software development is around for years, but still struggles with effectively adapting agile methods. In this context, the *Cloud* provides a highly flexible environment offering a variety of services. However, little is known which processes are used for distributed development in the Cloud, how these processes are used and customized, and how they might differ from other approaches.

*Objective.* We aim at better understanding the software process applied in distributed software development using the Cloud as development environment. Based on real project data, a simulation-based approach was chosen to improve the understanding of such processes and to support project managers to select and tailor software processes for Cloud-based distributed software development.

*Contribution.* This paper contributes a simulation model, which helps analyzing different project parameters and their impact on projects carried out in the Cloud. A Scrumban process was chosen to adapt a simulator, which was configured using project data[1] from a project executed in Finland and Spain. The event-driven simulator takes the number of issues, issue effort and priority in the backlog list as input. The simulator helps reproducing activities, developers, issues and events in the project, and it generates statistics, e.g., on throughput, total time, and lead and cycle time.

*Previously Published Material.* Prior to the work presented in this paper, we faced the difficulties of using simulation models to reproduce real case studies from China and India [2, 3, 5, 15]. We compared three process simulations with the original process, Kanban, and Scrum to study the methods' impact on performance, total time, and throughput.

*Outline.* The remainder of the paper is organized as follows: Section 2 provides an overview of related work. In Section 3, we describe the research design including research questions, simulation variables, and the specification and implementation of the simulation model. Section 4 presents the results from the different simulations. The paper is concluded in Section 5.

## 2 RELATED WORK

Distributed software development is affected by many aspects, such as demanding communication in a distributed setup, challenges related to coordination, and collaboration. These (and more) factors all influence the way in which software is defined, built, tested, and delivered. In response, distributed teams utilize specific tools to facilitate collaborative work [18]. However, different studies suggest the projects' processes being selected in a pragmatic rather than in a systematic manner [11, 22, 24], and studies also suggest agile methods stepping into the background when it comes to define proper tool support [7]. On the other hand, Global Software Engineering (GSE) as a discipline is maturing, as for instance Šmite et al. [25] show in their discussion of available empirical evidence in the field or Ebert et al. [6] who discuss the impact of GSE-related research on industry.

In the context of process simulation, Turner et al. [23] simulate the process performance of shared systems engineering services. They developed a specific Kanban-based process to support software development in rapid response environments, simulated this process using three modeling approaches (system dynamics, discrete events, and agents), and compared it to a simulated traditional process to determine if there were gains in effectiveness and value over time. Their overall goal was to study whether organizing projects as a Kanban-based scheduling system (KSS) results in better project performance. Instead of comparing Agile/Lean and traditional process models, Martin and Raffo [16] present a practically used hybrid software process with the purpose of evaluating potential process change. This model simulates discrete activities within the context of an environment described by a system dynamics model. However, discrete event models use simple building blocks and tend to be fairly basic, yet face the problems concerning the discretization of time and insufficient detail of parameters and variables. Walkeland et al. [27] lay the foundation for validating KSS with an agent-based simulation model. However, notably in GSE, process simulation is a promising route towards prediction and fast evaluation of process change, as several aspects can be analyzed quickly and without utilizing long-lasting and thus expensive case studies or limited student lab experiments.

One aspect crucial to GSE is the integration of distributed teams and the work alignment. A conceptual model of a process interface as described by Kuhrmann et al. [12] emphasizes the artifacts to be exchanged between project sites. However, there is an ongoing discussion about what are proper project artifacts after all—especially for agile methods (e.g., [12, 26]). A transition from Scrum to Scrumban is reported by Nikitina et al. [17], who explain the transition steps in detail and illustrate the transition with an action research case study. Although contributing to the body of knowledge, such case studies describe context-specific approaches and, therefore, transferring the outcomes to another context usually requires setting up a new case study. A process simulation as presented in the present paper helps improving decision-making processes by constructing a parameterized simulation model, which allows for modeling the intended process (or a set of alternatives), feeding the simulation with (empirical) data from past projects, calibrate the simulation, and eventually conclude a feasible solution; a procedure that was, so far, successfully applied to other field, e.g., risk management in distributed projects as presented by Lamersdorf et al. [13].

So far, in literature, few reports on using process simulation of agile methods in GSE using the Cloud as major development environment are available. Due to its *economies of scale*, Cloud computing has become the norm for consuming computing resources. While the potential for using the Cloud for GSE has been investigated in the literature, Alajrami et al. [1] go one step further and propose a Cloud-based software process enactment architecture which utilizes the Cloud elasticity, accessibility and availability to facilitate GSE, and to overcome some of the associated technical and communication challenges. Yara et al. [29] also present a Cloud-based platform that addresses core problems, e.g., computing capacity, bandwidth, storage, security, and outline a generic Cloud architecture and an initial implementation in the context of GSE. Nevertheless, even though companies have implemented GSE, they still face challenges in the development lifecycle. Hashmi et al. [8] provide a synopsis of Cloud computing, GSE challenges and,

[1] For seven weeks, six developers in Finnland and six in Spain, located at three sites (two in Spain and one in Finland) worked on a project developing a SmartGrid system.

**Table 1: Simulation input and output parameters and variables.**

| Input | | Output | |
|---|---|---|---|
| $I_1$ | Project size (total number of user stories) at time t, it is denoted by $N_F(t)$ | $O_1$ | Throughput |
| $I_2$ | Team size (number of developers), it is denoted by $N_D$ | $O_2$ | Total time |
| $I_3$ | Average effort | $O_3$ | Duration of simulation T |
| $I_4$ | Number of activities, it is denoted by $N_A$ | | |
| $I_5$ | WIP Limits in each activity (the maximum number of features that can be handled at any given time), it is denoted by $M_k$ for the $k^{\text{th}}$ activity | | |

the problem that "Cloud" denotes a process and a product alike. Therefore, Hashmi et al. [8] especially support our motivation to use Cloud technologies in GSE. This paper thus fills in gap in literature by contributing this facet, i.e., using the Cloud as development environment for GSE, to the body of knowledge.

## 3 RESEARCH DESIGN

This section presents the research design. The overall research questions are presented in Section 3.1. Section 3.2 describes the goals and requirements. The simulation model as such is specified in Section 3.3, and its implementation is presented in Section 3.4.

### 3.1 Research Questions

To study distributed software development in the Cloud using an extended simulation model, we formulate the following research questions:

**RQ 1:** How does the simulation model need to be calibrated, such that it reflects the particularities of the distributed project?

**RQ 2:** To what extent can the simulation model reproduce the data obtained in the real project?

**RQ 3:** How reliable is the simulation model[2]?

The first research question aims at extending a previously defined simulation model [3], such that it covers the particularities of distributed software development. For this, different elements of the model need to be adjusted, and several simulation runs need to be performed to tune the model. For each simulation run, only a single parameter varies (e.g., average effort of each user story, project size, and team size). Finally, the total time required and throughput values are examined to understand whether the variations are continuous or non-linear. For this, the following metrics are used: `throughput` and `total_time`, for a chosen value of one parameter and for fixed values of the other inputs, the simulator is run once until it stops (the end of the simulation) and the variation of the throughput (and total time) is examined.

Having the calibrated simulation model available, the second research question aims to study whether the simulation model can be used to reproduce the real project. In particular, results (i.e., `throughput` and `total_time`) are collected feeding the simulation model with artificial and real project data. Results are used to improve the simulation model and, eventually, a comparison is carried out using the metric `distance` (between curves) of released released user stories.

[2]In particular, if many simulation runs are performed using the same inputs: Does the model behave as expected?

**Table 2: Simulation-specific questions.**

| | Question |
|---|---|
| $Q_1$ | If the throughput is fixed, how can the other parameters be adjusted? |
| $Q_2$ | If the project size varies, but other parameters remain fixed, what is the effect on the throughput and on the total time required? |
| $Q_3$ | If the team size varies, but other parameters remain fixed, what is the effect on the throughput and on the total time required? |
| $Q_4$ | If the work-in-progress limit (i.e., the maximum number of user stories that can be handled at any given time) varies for different activities, how does the throughput change? |
| $Q_5$ | What is the relationship between the average effort for the user stories in the project and the total time required? |

The third research question studies the reliability of the simulation model. For this, several runs of the simulation model are performed using a list of artificial user stories. As metric, the `variation` (of average effort) is used to compare the variation in the calculated effort with the real effort from the project data.

### 3.2 Simulation Goals and Requirements

The overall goal of this study is to better understand distributed software development in a Cloud context. For this, an existing simulation model [3] is modified to better support decision-making processes concerning planning a distributed development process. The aim of this simulation model is thus to analyze the tradeoffs regarding throughput and total time on varying project size, team size, and other parameters. Furthermore, the modified simulation model aims to help project managers select the most suitable planning alternative. The overall simulation goals setting the scene for the simulation are therefore:

| | |
|---|---|
| Object | Simulation model of a distributed development process |
| Purpose | Support decisions for planning |
| Quality Focus | Throughput, total time, size of the project, and size of the team |
| View Point | Project Manager |
| Context | Software Factory Network |

The simulation model is purposed to answer the detailed questions collected in Table 2. For this, we define the input and output parameters/variables as summarized in Table 1. The simulation is performed instrumenting five scenarios, which are defined in Table 3.

**Table 3: Simulation scenarios.**

| | Scenario Description |
|---|---|
| $S_1$ | For a chosen value of the throughput or total time and for fixed values of the other inputs (project and team size), the simulator is run once until it stops and the total time required is examined. |
| $S_2$ | For a chosen value of the size of the project and for fixed values of the other inputs, the simulator is run once until it stops (the size of the project is reached) and throughput and total time are examined. |
| $S_3$ | The simulator is run for a chosen value of size of the team, and for the fixed values of the other inputs, the values of the throughput and total time are examined. |
| $S_4$ | For a chosen value of the WIP limits in each activity and for the fixed values of the other inputs, the values of the throughput and total time are examined. |
| $S_5$ | For a chosen number of simulation runs, and all parameters remain fixed, and the relation among average effort and total time is examined. |

## 3.3 Specification of the Simulation Model

In this section, we briefly introduce the Software Factory process, which serves as a blueprint for distributed development projects, and we analyze and explain the modifications required to use this process as input for the simulation model.

*3.3.1 The Software Factory Process Model.* In the Software Factory, Scrumban [10] is used to run the distributed software development projects. In general, a coach combined an agile process (Scrum) with a Kanban board, which visualizes the feature assignment in each process step.

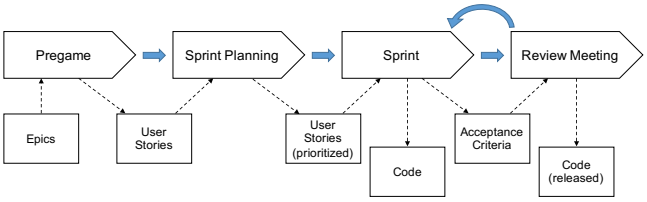


**Figure 1: Overview of the Scrumban process as used in the Software Factory. This overview illustrates the main steps in the process and the incoming/outgoing artifacts per process step. The thick opaque arrows show the control flow, and the dotted arrows show the product flow.**

The Scrumban model as shown in Figure 1 comprises the four steps *Pregame*, *Sprint Planning*, *Sprint*, and *Review Meeting*. In the reported setting, a single sprint takes two weeks. Apart from this, most of the well-known Scrum practices are applied, e.g., the *Product Owner* selects *User Stories*, *Developers* estimate the given stories, and *Daily Stand-up Meetings* are performed. To set up the simulation, we provide a formalization of the process model from Figure 1. Therefore, we need a detailed understanding of the process model and how specific practices are implemented. Table 4 provides a detailed description of the process steps and assigns inputs and outputs.

According to the general Scrum guideline [10, 19], the three roles *Scrum Master*, *Product Owner*, and *Team* are present in a software project. In the Software Factory, these roles are generally present and implemented. However, due to the distributed project setup, the team is spread across three project sites (one team per site). That is, the project is operated as a distributed project and, thus, the team faces several challenges of distributed projects, such as time loss due to long meetings caused by an inefficient Internet connection, due to the problems with communication tools, due to the dependencies among different user stories, and allocation of work among different sites at which the team members are located.

*3.3.2 Adaptation of the Simulation Model.* The presented simulation model is grounded in a previously developed model [3] for which the Software Factory process served as *calibration model*. We analyzed the practical application of the Software Factory process and compared it to the original simulation model to determine those parameters to be used for calibration. In particular, multi-site development and the resulting challenges for collaboration and communication had to be implemented in the simulation model. In particular, the following changes were made to the original simulation model to adequately reflect the Software Factory process:

- The *Pregame* activity was added to the simulation model
- *Rework* was added to the simulation model
- The simulation model was modified to better reflect productivity in distributed settings

The implementation of *Rework* in the simulation model allows for repeating those tasks that are not yet finished or that do not fulfill the *Acceptance Criteria*. Such tasks are scheduled for the next *Sprint* and continue previous activities (from *Review Meeting* to *Sprint*). The productivity-related modification was performed to better reflect the productivity in terms of the number of hours worked (per developer) and changes of the team size in different phases of the project. For instance, the modification covers changing team setups, such as on-boarding a team, e.g., the core team consists of six developers (begin, end) and in selected phases, another six developers joint the team.

## 3.4 Implementation of the Simulation Model

Figure 2 shows the final implementation on the simulation model as a UML class diagram, which shows the entities of the system and the relationships between the different actors. The classes `KanbanSimulator` and `KanbanSystem` represent the simulator's core system comprising all methods to create the simulation environment. The remaining classes, e.g., `Feature`, `Activity`, and `Developer`, reflect the process model entities to be simulated. The entity classes are complemented with some utility classes, e.g., `ActivityRecorder`, that help recording data for the simulation analysis. A more detailed explanation of the (original and unadjusted version of the) simulation model can be found in [2].

In the simulation presented in the paper at hand, the main actors are the developers of a distributed team working according to the process as shown in Figure 1, whereas each activity requires a certain set of skills. The most important events in the simulation model are: `FeatureCreation`, `FeatureToPull`, `StartDay`, and `FeatureWorkEnded`. These are used to set the scene for a simulation and to analyze the (potential) need for rework.

To run a simulation using the presented model, the following input is required: The main input is a list of user stories of which

**Table 4: Detailed description of the different process elements considered in the process simulation. Implementation of actual practices in the Software Factory are explained.**

| Process Activity | Input | Output |
|---|---|---|
| In the *Pregame*, *Epics* as input are divided into *User Stories*. The outcome of this meeting is the (initial) Backlog containing all *User Stories* to be prioritized in the *Sprint Planning* activity. In the Software Factory, the *Pregame* usually takes two days. | Epics | User Stories |
| Based on the Backlog, in the *Sprint Planning*, each *User Story* or task (in which some user stories are divided) is prioritized. | User Stories | User Stories (prioritized) |
| In the *Sprint*, the actual development activities (including analysis and coding tasks performed by the developers) are carried out. During the *Sprint*, daily meetings (10-15 minutes) are held in which the four basic Daily Scrum questions are asked and answered. Eventually, this activity produces the actual systems, i.e., the *Code* of the system, and a set of *Acceptance Criteria* (according to a "Definition of Done"; DoD), which are used in later analyses of the goal achievement. | User Stories (prioritized) | Code, Acceptance Criteria |
| In the *Review Meeting*, the team and the Product Owner verify the fulfillment of the *Acceptance Criteria* defined in the analysis steps of the Sprint. The product-centered *Review Meeting* is complemented by a more process-oriented retrospective. Depending on the review outcomes, some tasks might be subject to rework, i.e., certain tasks might be repeated, and those tasks are scheduled for the next *Sprint*. Tasks that are considered done eventually result in released *Code*. In the Software Factory, a *Review Meeting* takes about one hour. | Acceptance Criteria | Code (released) |

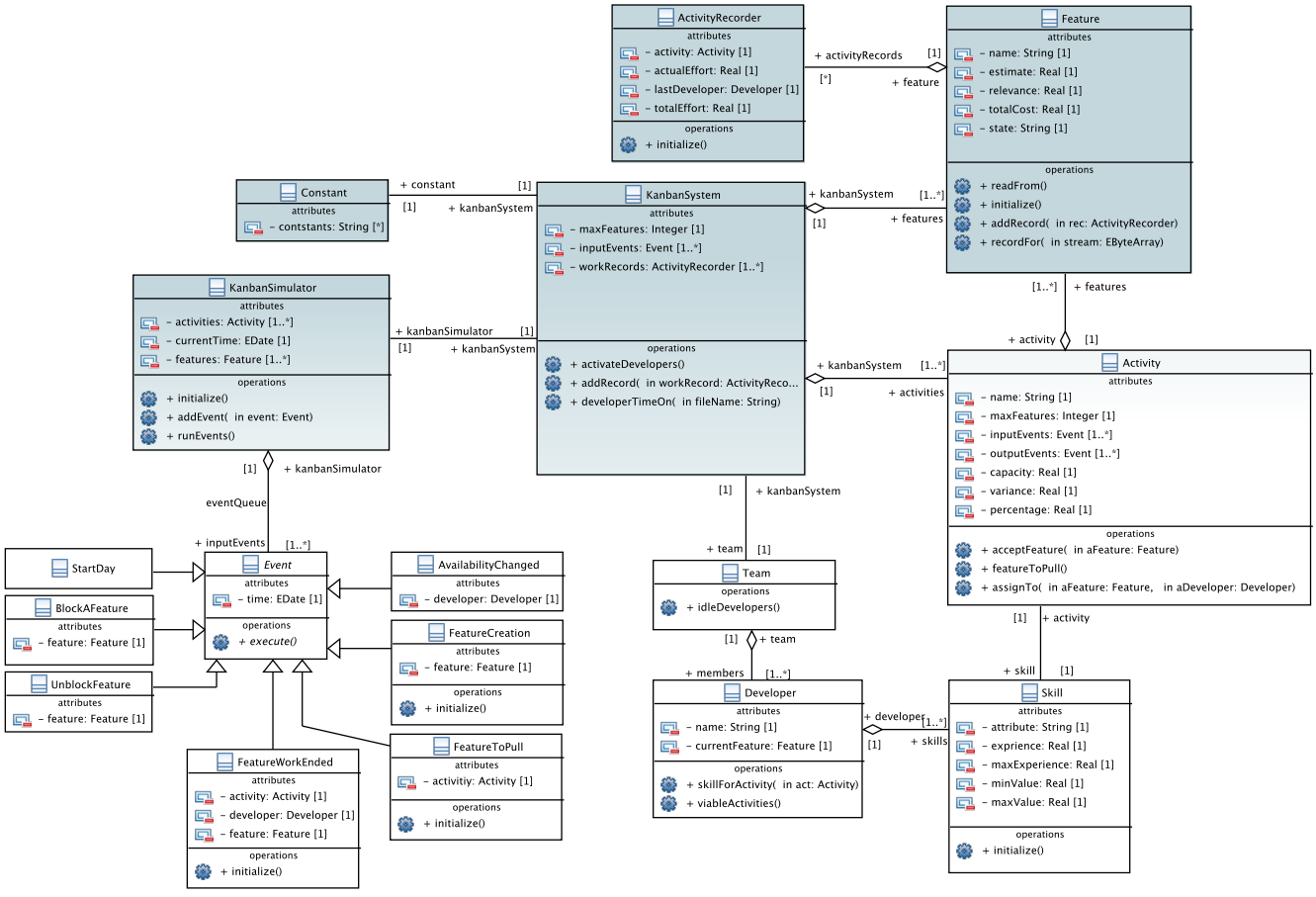


**Figure 2: UML class diagram of the simulation model.**

each is characterized by an identifier, a report date, an effort characterizing the amount of work required to complete a user story (in days), and a priority (as a numerical value; the higher the value the higher the user story's priority). Furthermore, a set of parameters related to the real process data, such as number of developers, developer skills, probability of rework, and work-in-progress (WIP) limits is required. Finally, a script initializes the process (the process variables), e.g., duration of meetings or sprint length. The script also runs the simulation, collects, and stores data to CSV files.

## 4 SIMULATION RESULTS

In this section, we present the simulation results. In Section 4.1, we describe the actual simulation setup. In Section 4.2, we present the outcomes of the simulation runs and a discussion. Finally, in Section 4.3, we critically discuss our findings regarding the threats to validity.

### 4.1 Simulation Setup

We observed a project[3] from April 23, 2012 till July 6, 2012 in which a team of six developers (divided into two groups) started working for 3 h/d in Spain. From May 14, 2012 until June 29, 2012, another team of six developers located in Helsinki joined the project and worked for 6 h/d. In these periods, we monitored the processes implemented and collected the raw project data, which has been analyzed and used to create[4] the input for the simulation model. Eventually, for the initial setup, we considered 25 user stories and tasks stored in the backlog, whereas we expect new user stories coming in after the last review meeting of an iteration, or at the beginning of a new iteration. Furthermore, we assume developers always available to work on and release upcoming user stories.

In this simulation we analyzed five sprints, and we performed simulation runs using real and artificial data. Real data has been collected directly from the aforementioned project, and artificial data has been collected by using an algorithm of the simulation model that takes the real data as input. After the data analysis, we calculated the average effort and standard deviation to identify the data distribution[5] and to obtain statistical values for incoming user stories. Having the data required, we built the list of user stories that serve as input for the simulator (Figure 3 shows the resulting user story setup used for the simulation).

### 4.2 Simulation Runs

In this section, we provide insights into the simulations and present and discuss the results. The different simulations address the questions (Table 2) and scenarios (Table 3) as introduced in Section 3.2.

[3]In this project in the context of a smart grid environment, the system to be implemented had to process and analyze a substantial quantity of data concerned with measurement of data consumption. Data was collected hourly, daily, and monthly. The teams were appointed to implement the different modules that compose the system for the processing data.

[4]To better reproduce rework on interconnected tasks, we have reduced the 64 issues (considering user stories and tasks in which some user stories have been divided) to 25 user stories. The throughput in the real project was almost 3 user stories per week with an average effort of 1.3 to 1.5 person days.

[5]For the data distribution, we assume a log-Normal distribution. For incoming user stories, however, the distribution is unknown. Therefore and in order to allow for replicating input data in different simulation runs, we use a linear interpolation method.

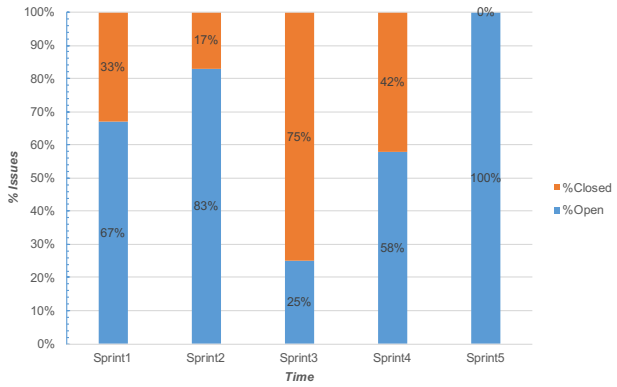


**Figure 3: Percentage of open and closed user stories as used for the different simulated sprints.**

The full mapping of simulations, questions, and scenarios is shown in Table 5.

**Table 5: Mapping of simulations to questions and scenarios.**

| $Q_1$ | $Q_2$ | $Q_3$ | $Q_4$ | $Q_5$ | Simulation | $S_1$ | $S_2$ | $S_3$ | $S_4$ | $S_5$ |
|---|---|---|---|---|---|---|---|---|---|---|
| ✗ | | | | | 1 | ✗ | | | | |
| | ✗ | | | ✗ | 2 | | ✗ | | | |
| | | ✗ | | | 3 | | | ✗ | | |
| | | | ✗ | | 4 | | | | ✗ | |
| | | | | ✗ | 5 | | | | | ✗ |
| | | | | | 6 | | | | | ✗ |

For example, question $Q_1$ is studied using simulation $S_1$, i.e., for a given throughput, what parameters can be varied. Similarly, simulation $S_2$ helps answering $Q_2$ and $Q_5$, i.e., how are total time and throughput vary in relation to varying project size.

*4.2.1 Simulation 1.* The first simulation studies variables that can be modified—and how they can be modified—if the variables `team_size` or `project_size` are fixed. In particular, the variables `throughput` and `total_time`, and how they can be modified are of interest. For $S_1$, the throughput is set and time required to complete the simulation is examined.

In the chosen team setup (12 developers, group 1 has six developers working 6 h/d and group 2 has six developers working 3 h/d), a project of 25 user stories, each with an effort of 1.3–1.5 person days, was chosen. The simulation showed that the team was able to work on a high number of incoming user stories. The overall performance was 5–6 user stories per day and, eventually, the team could work on about 500 user stories without inflationary growth of the backlog. However, communication issues and dependencies among user stories and/or tasks limited the performance, such that $S_1$ yielded in an average throughput of only 3 user stories per day.

*4.2.2 Simulation 2.* The second simulation studies $Q_2$, i.e., studying what effect a varying project size has (i.e., keeping the other parameters fixed) on the throughput and the total time required for

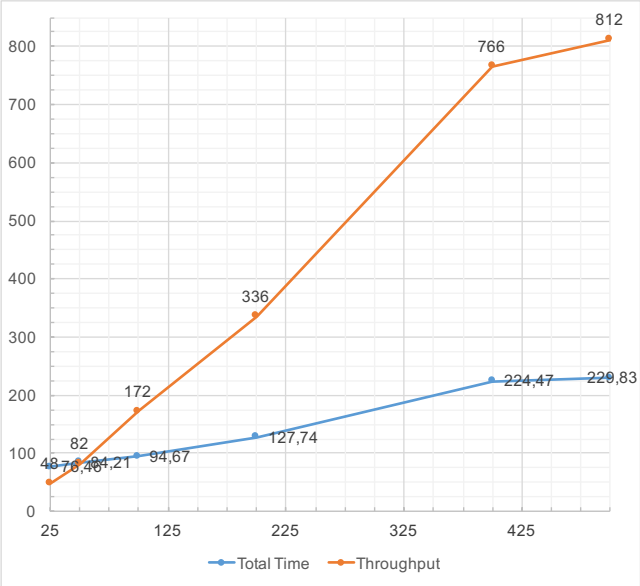


**Figure 4: Total time and throughput.**

a project. For a chosen value[6] of `project_size` and fixed values of other inputs, the simulator is run once until it stops (the size of the project is reached) and the values of the variables `throughput` and `total_time` are evaluated. The number of user stories (with an average effort of 1.3 person days) increases and we checked the differences in `throughput` and `total_time` as shown in Figure 4.

**Table 6: Variation in the size of the project and impact on throughput and total time required.**

| Size | Total Time | Throuput |
|---|---|---|
| 25 | 76.46 | 48 |
| 50 | 84.21 | 82 |
| 100 | 94.87 | 172 |
| 200 | 127.74 | 336 |
| 400 | 224.47 | 766 |
| 500 | 229.83 | 812 |

For a doubled project size the throughput increases in linearly, yet shows a little steep when the size of the project increases from 200 to 400 user stories; and then continues with a linear trend until 500 user stories[7]. Regarding the total time required, the trend is almost linear with slow rise and a steep when the projects size increases from 200 to 400 user stories—and then continues with a linear trend until 500 user stories. In Table 6 we tabulated the obtained throughput as the total user stories released during the project. Table 6 shows that for a project size of 25 user stories, the throughput is approx. 3–4 user stories per week (48 user stories, including assumed 20% of rework). For a project size of 50 user stories, the throughput is 7–8 user stories per week and so on. Hence, doubling the project size also doubles the throughput. In Table 6, we consider the throughput as the number of user stories released at the end of the project.

*4.2.3 Simulation 3.* In $S_3$, we study the relationship between `team_size`, `throughput` and `total_time` to answer $Q_3$, i.e., what the effect on the throughput and the total time required is if the team size varies, but other parameters are fixed[8]. The summary of the simulation results is shown in Table 7.

The team size was chosen, in order to study the impact on the throughput when other project parameters remained fixed. For this, we use the number of hours that each developer works. We assume that variations on `throughput` and `total_time` depend on the developers' skills, on the number of hours they work, and on the strategy used to assign them to the activities rather than the size of the team. We selected the cases of the whole team, and teams with six and three developers respectively. The results demonstrated that variations in `throughput` and `total_time` mostly depend on the skills of the developers and their assignment to the different activities. Table 7 shows that if three developers or six developers, that are skilled in all activities, work on the same number of user stories, they may obtain the same `throughput` and the same `total_time`. Instead, when the number of developers is not high enough to satisfy the effort required for an activity, `throughput` decreases and the `total_time` increases.

*4.2.4 Simulation 4.* In $S_4$, we study the relation between the use of WIP limits, `throughput` and `total_time` to answer $Q_4$, i.e., what the impact on the throughput is if the WIP limit for activities varies.

For a given WIP limit, it is possible to examine the resulting `throughput` and `total_time`, if other parameters remain fixed. It is required to perform many simulation runs to obtain WIP-limit values, which can yield optimal throughput in the minimum time required. For a team setup of 12 developers, we performed several simulation runs with different values for the WIP limit, and without limits[9]. We observed that for lower WIP-limit values, `throughput` decreases and the `total_time` increases—a bottleneck may exist. However, if we consider WIP limits of 6–8 or higher, results are the same as if there were no limits at all. This could be a result from the small number of user stories or the big team size and, thus, WIP limits are not useful (this also hampers generalizability). In a nutshell, for low WIP limits, we obtained a low `throughput` and a longer `total_time`, yet, higher WIP limits have not shown any effect on the `throughput`.

*4.2.5 Simulation 5.* In simulation $S_5$, we study the relation between `average_effort`, `throughput`, and `total_time` to answer $Q_5$, i.e., whether there is a relation between effort for user stories and the total time required for the project. For a given number of

[6]We assumed that a linear relation among `project_size`, and `throughput` and `total_time` required exists. Therefore, we re-ran the simulation with a stepwise increasing `project_size`, but kept the other parameters fixed.
[7]Which is almost the maximum number of user stories the team can work on without an exceeding growth of the backlog.

[8]We assume that no linear relation exists among `team_size`, `throughput`, and `total_time`. That is, if the number of developer skilled in testing goes to zero, throughput is blocked. If the number of blocked user stories grows, adding new tester does not increase the throughput, due to the bottleneck from the previous phase.
[9]For example, at first one may consider a WIP limit of 10–12, i.e., 10 in the first and last activity, and 12 in the second and third activity. WIP limits tested were also be 6–8 and 3–4.

**Table 7: Team size and throughput (project performances in relation to different team size and skill profiles; skills for activities: 1=analysis, 2=development, 3=testing, and 4=deployment).**

| Team Size | Skills | | | | Throughput | Backlog | Pregame | Sprint Planning | Sprint | Review Meeting | User Stories Released |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | | | | | | | |
| Whole Team | | – | | | 83.923 | 25 | 25 | 25 | 33 | 30 | 42 |
| 6 | | ✗ | ✗ | ✗ | 85.0125 | 25 | 25 | 25 | 32 | 28 | 34 |
| 6 | | ✗ | ✗ | ✗ | 98.0263 | 25 | 25 | 25 | 30 | 29 | 37 |
| 6 | | ✗ | ✗ | ✗ | 98.0263 | 25 | 25 | 25 | 30 | 29 | 37 |
| 6 | | ✗ | ✗ | ✗ | 84.3339 | 25 | 25 | 25 | 39 | 34 | 52 |
| 6 | ✗ | ✗ | ✗ | | 76.7853 | 25 | 25 | 25 | 26 | 20 | 0 |
| 6 | | ✗ | ✗ | | 77.4417 | 25 | 25 | 25 | 30 | 19 | 0 |
| 6 | | ✗ | ✗ | | 76.7853 | 25 | 25 | 25 | 26 | 20 | 0 |
| 6 | | ✗ | ✗ | | 96.651 | 25 | 25 | 25 | 41 | 31 | 45 |
| 6 | ✗ | ✗ | ✗ | ✗ | 96.0761 | 25 | 25 | 25 | 33 | 28 | 38 |
| 6 | | ✗ | ✗ | ✗ | 96.0761 | 25 | 25 | 25 | 33 | 28 | 38 |
| 6 | | ✗ | ✗ | ✗ | 99.2204 | 25 | 25 | 25 | 33 | 27 | 31 |
| 6 | ✗ | ✗ | ✗ | ✗ | 86.5942 | 25 | 25 | 25 | 37 | 32 | 46 |
| 6 | | ✗ | ✗ | ✗ | 89.6905 | 25 | 25 | 25 | 36 | 32 | 46 |
| 3 | ✗ | ✗ | ✗ | ✗ | 98.5971 | 25 | 25 | 25 | 39 | 34 | 52 |
| 3 | | ✗ | ✗ | ✗ | 98.5971 | 25 | 25 | 25 | 39 | 34 | 52 |
| 3 | | ✗ | ✗ | ✗ | 98.5971 | 25 | 25 | 25 | 39 | 34 | 52 |
| 3 | | ✗ | ✗ | | 125.745 | 25 | 25 | 25 | 38 | 32 | 46 |
| 3 | | ✗ | ✗ | ✗ | 79.3474 | 25 | 25 | 25 | 26 | 22 | 0 |
| 3 | | ✗ | ✗ | ✗ | 105.672 | 25 | 25 | 25 | 39 | 31 | 43 |
| 3 | | ✗ | ✗ | ✗ | 104.795 | 25 | 25 | 25 | 35 | 30 | 40 |
| 3 | | | ✗ | ✗ | 77.4448 | 25 | 25 | 25 | 33 | 22 | 0 |
| 3 | | ✗ | ✗ | ✗ | 78.2208 | 25 | 25 | 25 | 32 | 22 | 0 |
| 3 | | ✗ | ✗ | ✗ | 84.6146 | 25 | 25 | 25 | 37 | 29 | 39 |
| 3 | | ✗ | ✗ | ✗ | 95.5282 | 25 | 25 | 25 | 34 | 29 | 39 |
| 3 | | | ✗ | ✗ | 105.672 | 25 | 25 | 25 | 39 | 31 | 43 |
| 3 | ✗ | ✗ | ✗ | ✗ | 84.7019 | 25 | 25 | 25 | 32 | 28 | 34 |
| 3 | ✗ | ✗ | ✗ | ✗ | 84.7019 | 25 | 25 | 25 | 32 | 28 | 34 |
| 3 | | ✗ | ✗ | ✗ | 103.79 | 25 | 25 | 25 | 36 | 32 | 16 |

**Table 8: Correlation of effort and total time.**

| Variable | Average | Standard Deviation |
|---|---|---|
| Total Time | 77.26 | 2.62 |
| Effort | 1.297 | 0.203 |
| Corr (effort/time) | 0.0888 | |

**Table 9: Correlation of effort and throuput.**

| Variable | Average | Standard Deviation |
|---|---|---|
| Throughput | 44.19 | 6.02 |
| Effort | 1.297 | 0.203 |
| Corr (effort/thput) | -0.119 | |

simulation runs, all simulation parameters remain fixed. At the end of each simulation, values for `average_effort`, `throughput`, and `total_time` are examined to understand if variations, as expected, are continuous. Accordingly, two experiments have been conducted: one with real data, and another using artificial data.

The results obtained show that variations in the effort cause variations in `throughput` and `total_time`. Furthermore, the relation is almost continuous without major gaps. Performing many simulation runs, we found a low correlation between variations in `average_effort` and `total_time`.

Simulations performed using the real project data, did not show any variation for neither variable. Yet, simulations using the artificial data showed variations. In total, we performed 100 simulation runs and found a low correlation 0.0888 between the variation in `average_effort` and `total_time` (Table 8). Furthermore, we found a correlation of -0.119 between `average_effort` and `throughput` (Table 9). Hence, there is no direct relation between `average_effort` and `throughput`.

*4.2.6 Simulation 6.* In last simulation $S_6$, we study the relations between `average_effort`, `throughput`, and `total_time` with a particular focus on the question to what extent the simulation model can reproduce data from the real project.

The implemented software development model presented in Section 3.3.1 is used to allow for comparing the results (i.e., `throughput` and `total_time` required to finish the project) obtained from the simulations performed on real and artificial data. In particular, simulations were run using the list of user stories, parametrized with

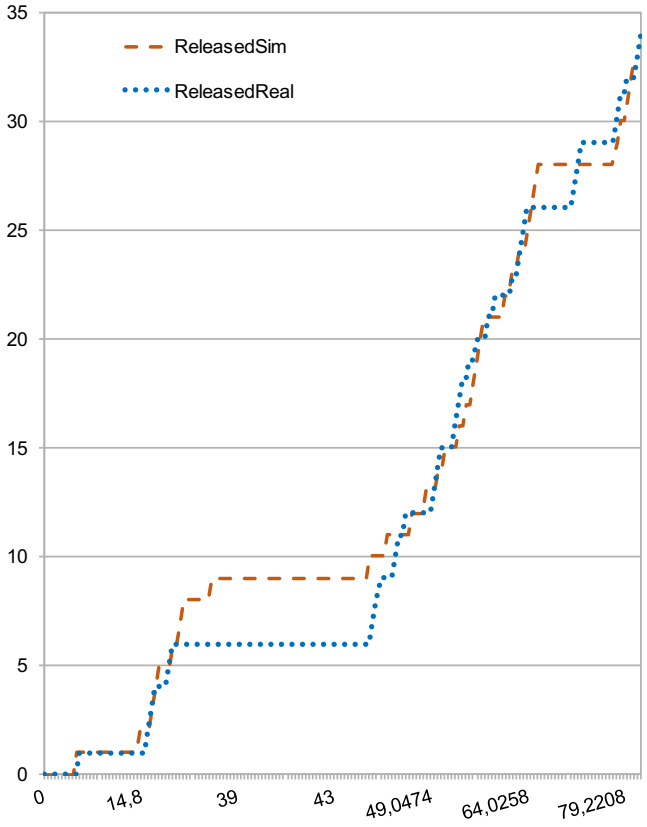


**Figure 5: Comparison between simulated and real case.**

values for the effort taken from the real project. The analysis was carried out on the number of released user stories, in particular by comparing the two performance curves shown in Figure 5. The curves represent the cumulative number of user stories released in the project and, in an optimal case, both curves should overlap. As Figure 5 shows, our experimental results suggest that the presented simulation model produces data that well match, which demonstrates the feasibility of the approach presented.

*4.2.7 Summary of the Simulation Results.* In this section, we briefly summarize our simulation results and answer the research questions (see Section 3.1). To support answering the research questions, in Table 5, we relate the different simulation scenarios Table 3 with the detailed simulation questions Table 2.

*RQ 1.* To answer the first research questions, we use the simulations $S_{1-5}$. The different outcomes presented in the previous paragraphs show the relationships between the three variables `throughput`, `total_time`, `average_effort`. The findings further show how the original simulation model [3] can be calibrated in order to reproduce distributed software development (processes). In particular, simulation $S_1$ showed that for a given throughput, `total_time` is the only parameter that can change (gives all other variables are immutable). Simulation $S_2$ showed a linear relationship between `throughput` and `total_time` for a varying `project_size`, whereas $S_3$ found no linear relationship if `time_size` is the subject of study. Simulation $S_4$ studied WIP limits and the impact on `throughput`, finding no effect on the throughput for higher WIP limits. Eventually, $S_5$ compared the simulation results obtained from real project data and artificial data.

*RQ 2.* The second research question aims at comparing simulation results with real project data (and experience). For this, simulation $S_6$ is used. The results are shown in Figure 5, which shows the distance of the two curves as a measure of accuracy. In summary, the adapted simulation model was found feasible to reproduce a real project.

*RQ 3.* The third research question aims to study the reliability of the simulation model. For this, simulation $S_5$ was used, and the simulation was run several 100 times. The outcomes show the simulation model reliably reproducing results with acceptable variations for `throughput`, `total_time`, and `average_effort` regardless of the input data, i.e., real or artificial data.

## 4.3 Threats to Validity

In the following, we discuss the threats to validity to be considered when applying the method presented in the paper at hand.

*Internal Validity.* According to Shadish et al. [21], an experiment may have unknown and hidden factors that could affect the results. In the presented case, information regarding teams and organization of work originated from the projects. Data used in the simulation model was extracted from systems used by the teams and personal observations, which might influence result quality. Although the model properly simulates skilled developers performing task sequences, still, the simulation model does not fully cover interactions among the developers thus introducing a threat regarding the inclusion of human factors in the simulation.

*Construct Validity.* Construct validity concerns the degree to which inferences are warranted from the observed phenomena to the constructs that these instances might represent [28]. A first threat to construct validity is that, although in this study we have carefully analyzed and preprocessed the Software Factory data, our results could be affected by the data quality (such as possible noisy data). Another threat related to construct validity is the fact that our work is centered on the study of how the process determines the efficiency of the development activity. However, there are many other human-related factors that could affect the efficiency and productivity of the team, e.g., considering (co-)workers, keeping the team motivated and satisfied, and so on. Just limiting the work-in-progress will not be effective if a team is troubled and dissatisfied. A simulation model can simulate a process, but it is very difficult to explicitly include human factors.

*External Validity.* If a study possesses external validity, its results will generalize to a larger population not considered in the experiment [21, 28]. In this study we only ran the simulation model on one development project. This project is small, and the number of subjects used in this study is small. This is a clear threat to external validity of our results. However, the simulation methods we proposed are evaluated on large software systems that experienced a long evolution. For a small or short-living system, the number of

development requests is often small thus making the simulation and statistical analysis unsuited.

## 5 CONCLUSION

In this paper, we presented a simulation process model able to reproduce the process followed in the Software Factory project. We demonstrated the calibration of the simulation model and its implementation. An existing simulation model was modified to reflect the Scrumban process as used in the Software Factory. We described the customization of the relevant parameters and aspects to implement the Software Factory process. Eventually, we performed a case study with (real-life) data gathered from Software Factory project.

The simulation results in the following major findings: Project teams face problems regarding communication and organization of distributed projects affecting the teams' productivity and/or increasing the time required to achieve the project goals. The results obtained from our simulation show the influence of decisions in the project planning activities, e.g., in assigning work, when a distributed development is considered for a project. Therefore, our simulation model can be used to model project setups of interest, to elaborate potential pitfalls, and to work out solutions to address those problems. This opportunity was especially shown by a comparative analysis of a simulated case and a real case. We could successfully model and reproduce the Scrumban process as used in the Software Factory, and our simulation generated results comparable to the real project data. Hence, the simulation model allows for modeling a distributed project, analyzing and predicting trends, and eventually selecting the most promising (according to the respective project goals) project configuration. Hence, project managers get a tool to early analyze project configurations, to better understand the development process and variations thereof, and, in future work, to apply the most suitable planning alternatives for the respective context.

Nevertheless, the presented model only partially addresses the (quantitative) relationship of different actions, which introduces some conceptual issues (e.g., human factors) in the model. Hence, the simulation capabilities of the model are limited to only those project aspects that can be sufficiently measured. Therefore, the results obtained in the presented simulation are limited for specific cases and can only serve as indication, but do not yet allow for generalization. However, such a generalization would be very helpful to have "standard" process customizations at disposal, which could be used to calibrate an organization- or project-specific simulation model.

Future work thus comprises gathering data from further Software Factory projects and from other industrial projects from different contexts. These steps will enhance the data bases and they will support the model's validation to improve the its reliability. Furthermore, the present model is expected to be extended to allow for simulating and reproducing further processes, i.e., to be generalized and then customized for application to further domains. Another aspect that is worth consideration is the improvement of the presented simulation model towards a prediction tool. So far, we could increase understanding of the relationships, e.g., project size and work-in-progress, and we could reproduce real project data, i.e., the model is primarily used as analysis tool. Therefore, given a sufficient data set as a basis and a sufficiently validated model, the approach presented in this paper could also serve as prediction tool to proactively improve the decision-making process of project managers. In this regard, an updated work-in-progress version of the simulator can directly access issue tracking systems such as Jira or Redmine. The new simulation tool collects data about the project such as, e.g., number and list of issues, estimated time and time spent for resolving issues, priority of issues, team size, and the process followed as a workflow (number of steps and connection among the steps). Furthermore, the new simulator can be quickly adapted for a particular project to reproduce and/or simulate the project providing the total time needed to finish the project and some statistics, e.g., concerning the number of issues per day, developer productivity, and so forth. Using Montecarlo simulations and variations of project parameters such as developer availability or error in effort estimation, the updated simulator also allows for risk analyses.

## REFERENCES


[1] Sami Alajrami, Barbara Gallina, and Alexander Romanovsky. 2016. *Enabling global software development via cloud-based software process enactment.* Technical Report TR-1494. Newcastle University.

[2] David Anderson, Giulio Concas, Maria Ilaria Lunesu, and Michele Marchesi. 2011. *Agile Processes in Software Engineering and Extreme Programming: 12th International Conference, XP 2011, Madrid, Spain, May 10-13, 2011. Proceedings.* Lecture Notes in Business Information Processing, Vol. 77. Springer Berlin Heidelberg, Berlin, Heidelberg, Chapter Studying Lean-Kanban Approach Using Software Process Simulation, 12–26. DOI: https://doi.org/10.1007/978-3-642-20677-1_2

[3] David J. Anderson, Giulio Concas, Maria Ilaria Lunesu, Michele Marchesi, and Hongyu Zhang. 2012. *Agile Processes in Software Engineering and Extreme Programming: 13th International Conference, XP 2012, Malmö, Sweden, May 21-25, 2012. Proceedings.* Lecture Notes in Business Information Processing, Vol. 111. Springer Berlin Heidelberg, Berlin, Heidelberg, Chapter A Comparative Study of Scrum and Kanban Approaches on a Real Case Study Using Simulation, 123–137. DOI: https://doi.org/10.1007/978-3-642-30350-0_9

[4] Christian Bird, Nachiappan Nagappan, Premkumar Devanbu, Harald Gall, and Brendan Murphy. 2009. Does Distributed Development Affect Software Quality? An Empirical Case Study of Windows Vista. In *International Conference on Software Engineering (ICSE).* IEEE Computer Society, Washington, DC, USA, 518–528. DOI: https://doi.org/10.1109/ICSE.2009.5070550

[5] Giulio Concas, Maria Ilaria Lunesu, Michele Marchesi, and Hongyu Zhang. 2013. Simulation of software maintenance process, with and without a work-in-process limit. *Journal of Software: Evolution and Process* 25, 12 (2013), 1225–1248. DOI: https://doi.org/10.1002/smr.1599

[6] C. Ebert, M. Kuhrmann, and R. Prikladnicki. 2016. Global Software Engineering: An Industry Perspective. *Software, IEEE* 33, 1 (Jan 2016), 105–108. DOI: https://doi.org/10.1109/MS.2016.27

[7] H. Femmer, M. Kuhrmann, J. Stimmer, and J. Junge. 2014. Experiences from the Design of an Artifact Model for Distributed Agile Project Management. In *International Conference on Global Software Engineering (ICGSE).* IEEE Computer Society, Washington, DC, USA, 1–5. DOI: https://doi.org/10.1109/ICGSE.2014.9

[8] Sajid Ibrahim Hashmi, Viktor Clerc, Maryam Razavian, Christina Manteli, Damian Aandrew Tamburri, Patrica Lago, Elisabetta Di Nitto, and Ita Richardson. 2011. Using the Cloud to Facilitate Global Software Development Challenges. In *International Conference on Global Software Engineering Workshops (ICGSE Workshops).* IEEE Computer Society, Washington, DC, USA, 70–77. DOI: https://doi.org/10.1109/ICGSE-W.2011.19

[9] J.D. Herbsleb and A. Mockus. 2003. An empirical study of speed and communication in globally distributed software development. *Software Engineering, IEEE Transactions on* 29, 6 (June 2003), 481–494. DOI: https://doi.org/10.1109/TSE.2003.1205177

[10] Henrik Kniberg and Mattias Skarin. 2010. *Kanban and Scrum-making the most of both.* lulu.com.

[11] Marco Kuhrmann, Philipp Diebold, Jürgen Münch, Paolo Tell, Vahid Garousi, Michael Felderer, Kitija Trektere, Fergal McCaffery, Christian R. Prause, Eckhart Hanser, and Oliver Linssen. 2017. Hybrid Software and System Development in Practice: Waterfall, Scrum, and Beyond. In *Proceedings of the International Confernce on Software System Process (ICSSP).* ACM, New York, NY, USA, 30–39.

[12] Marco Kuhrmann, Daniel Mendéz Fernández, and Matthias Gröber. 2013. Towards Artifact Models as Process Interfaces in Distributed Software Projects. In

*International Conference on Global Software Engineering (ICGSE)*. IEEE Computer Society, Washington, DC, USA, 11–20.

[13] Ansgar Lamersdorf, Jürgen Münch, Alicia Fernández del Viso Torre, Carlos Rebate Sánchez, Markus Heinz, and Dieter Rombach. 2012. A rule-based model for customized risk identification and evaluation of task assignment alternatives in distributed software development projects. *Journal of Software: Evolution and Process* 24, 6 (2012), 661–675. DOI: https://doi.org/10.1002/smr.576

[14] Pernille Lous, Marco Kuhrmann, and Paolo Tell. 2017. Is Scrum Fit for Global Software Engineering?. In *International Conference on Global Software Engineering (ICGSE)*. IEEE Press, Piscataway, NJ, USA, 1–10. DOI: https://doi.org/10.1109/ICGSE.2017.13

[15] Maria Ilaria Lunesu. 2013. *Process Software Simulation Model of Lean-Kanban Approach*. Ph.D. Dissertation. University of Cagliari.

[16] Robert Martin and David Raffo. 2001. Application of a hybrid process simulation model to a software development project. *Journal of Systems and Software* 59, 3 (2001), 237 – 246. DOI: https://doi.org/10.1016/S0164-1212(01)00065-6 Software Process Simulation Modeling.

[17] Natalja Nikitina, Mira Kajko-Mattsson, and Magnus Stråle. 2012. From Scrum to Scrumban: A Case Study of a Process Transition. In *International Conference on Software and System Process (ICSSP)*. IEEE Press, Piscataway, NJ, USA, 140–149. http://dl.acm.org/citation.cfm?id=2664360.2664382

[18] Javier Portillo-Rodríguez, Aurora Vizcaíno, Mario Piattini, and Sarah Beecham. 2012. Tools Used in Global Software Engineering: A Systematic Mapping Review. *Inf. Softw. Technol.* 54, 7 (July 2012), 663–685. DOI: https://doi.org/10.1016/j.infsof.2012.02.006

[19] Ken Schwaber and Mike Beedle. 2002. *Agile software development with Scrum*. Prentice Hall.

[20] Bikram Sengupta, Satish Chandra, and Vibha Sinha. 2006. A Research Agenda for Distributed Software Development. In *International Conference on Software Engineering (ICSE)*. ACM, New York, NY, USA, 731–740. DOI: https://doi.org/10.1145/1134285.1134402

[21] William R. Shadish, Thomas D. Cook, and Donald T. Campbell. 2001. *Experimental and Quasi-Experimental Designs for Generalized Causal Inference* (2 ed.). Cengage Learning, Boston, New York.

[22] Georgios Theocharis, Marco Kuhrmann, Jürgen Münch, and Philipp Diebold. 2015. Is Water-Scrum-Fall Reality? On the Use of Agile and Traditional Development Practices. In *International Conference on Product Focused Software Development and Process Improvement (Lecture Notes in Computer Science)*, Vol. 9459. Springer, Cham, 149–166.

[23] Richard Turner, Raymond Madachy, Dan Ingold, and Jo Ann Lane. 2012. Modeling Kanban Processes in Systems Engineering. In *Proceedings of the International Conference on Software and System Process (ICSSP)*. IEEE Press, Piscataway, NJ, USA, 23–27. http://dl.acm.org/citation.cfm?id=2664360.2664365

[24] Leo R. Vijayasarathy and Charles W. Butler. 2016. Choice of Software Development Methodologies: Do Organizational, Project, and Team Characteristics Matter? *IEEE Software* 33, 5 (Sept 2016), 86–94. DOI: https://doi.org/10.1109/MS.2015.26

[25] Darja Šmite, Claes Wohlin, Tony Gorschek, and Robert Feldt. 2010. Empirical Evidence in Global Software Engineering: A Systematic Review. *Empirical Softw. Engg.* 15, 1 (Feb. 2010), 91–118. DOI: https://doi.org/10.1007/s10664-009-9123-y

[26] Gerard Wagenaar, Remko Helms, Daniela Damian, and Sjaak Brinkkemper. 2015. *Product-Focused Software Process Improvement: 16th International Conference, PROFES 2015, Bolzano, Italy, December 2-4, 2015, Proceedings*. Lecture Notes in Computer Science, Vol. 9459. Springer International Publishing, Cham, Chapter Artefacts in Agile Software Development, 133–148. DOI: https://doi.org/10.1007/978-3-319-26844-6_10

[27] Wayne W. Wakeland, Robert H. Martin, and David Raffo. 2004. Using design of experiments, sensitivity analysis, and hybrid simulation to evaluate changes to a software development process: a case study. *Software Process: Improvement and Practice* 9, 2 (2004), 107–119. DOI: https://doi.org/10.1002/spip.200

[28] Claes Wohlin, Per Runeson, Martin Höst, Magnus C. Ohlsson, Björn Regnell, and Anders Wesslén. 2012. *Experimentation in Software Engineering*. Springer-Verlag, Berlin Heidelberg.

[29] Pavan Yara, Ramaseshan Ramachandran, Gayathri Balasubramanian, Karthik Muthuswamy, and Divya Chandrasekar. 2009. *Global Software Development with Cloud Platforms*. Lecture Notes in Business Information Processing, Vol. 35. Springer Berlin Heidelberg, Berlin, Heidelberg, 81–95. DOI: https://doi.org/10.1007/978-3-642-02987-5_10