Stochastic Performance Analysis of Global Software Development Teams

RICARDO M. CZEKSTER, UNISC - University of Santa Cruz do Sul
PAULO FERNANDES, LUCELENE LOPES, AFONSO SALES, and ALAN R. SANTOS, PUCRS University
THAIS WEBBER, UNISC - University of Santa Cruz do Sul

Measuring productivity in globally distributed projects is crucial to improve team performance. These measures often display information on whether a given project is moving forward or starts to demonstrate undesired behaviors. In this paper we are interested in showing how analytical models could deliver insights for the behavior of specific distributed software collaboration projects. We present a model for distributed projects using stochastic automata networks (SAN) formalism to estimate, for instance, the required level of coordination for specific project configurations. We focus our attention on the level of interaction among project participants and its close relation with team's productivity. The models are parameterized for different scenarios and solved using numerical methods to obtain exact solutions. We vary the team's expertise and support levels to measure the impact on the overall project performance. As results, we present our derived productivity index for all scenarios and we state implications found in order to analyze popular preconceptions in GSD area, confirming some, and refusing others. Finally, we foresee ways to extend the models to represent more intricate behaviors and communication patterns that are usually present in globally distributed software projects.

Categories and Subject Descriptors: K.6.3 [**Software Management**]: Software Development; D.2.9 [**Management**]: Programming Teams; D.4.8 [**Performance**]: Modeling and Prediction; D.4.8 [**Performance**]: Stochastic Analysis

General Terms: Management, Performance, Experimentation

Additional Key Words and Phrases: Global software development, analytical modeling, stochastic automata networks, performance analysis

ACM Reference Format:
Ricardo M. Czekster, Paulo Fernandes, Lucelene Lopes, Afonso Sales, Alan R. Santos, and Thais Webber. 2016. Stochastic performance analysis of global software development teams. ACM Trans. Softw. Eng. Methodol. 25, 3, Article 26 (August 2016), 32 pages.
DOI: http://dx.doi.org/10.1145/2955093

1. INTRODUCTION

Software development has crossed geographical boundaries as companies adopt new ways to deliver high-quality products in a timely manner. Global Software Development

The order of authors is alphabetical, not by order of rank. Paulo Fernandes is funded by CNPq-Brazil (307602/2013-3 and 459725/2014-9). Lucelene Lopes receives grant from FAPERGS/CAPES-Brazil (Docfix SPI n.2843-25.51/12-3). Afonso Sales is funded by CNPq-Brazil (470096/2013-6).
Authors' addresses: R. M. Czekster and T. Webber, Computer Science Department, UNISC - University of Santa Cruz do Sul; emails: ricardoc@unisc.br, thaiscs@unisc.br; P. Fernandes, L. Lopes, A. Sales, and A. R. Santos, Computer Science Department, PUCRS University; emails: paulo.fernandes@pucrs.br, lucelene.lopes@pucrs.br, afonso.sales@pucrs.br, alan.santos@pucrs.br.

DOI: http://dx.doi.org/10.1145/2955093

(GSD) has been intensively chosen around the world as a valid alternative to balance costs and productivity [Ebert 2007]. Effective individuals' interactions and a team's knowledge domain play an important role in GSD projects, within single-site and multisite contexts. According to Sangwan et al. [2006], these aspects are a frequent concern in distributed software projects and their effects are directly related to a team's productivity.

In order to reduce the need for constant communications and external support in GSD, research results suggest dividing a project into self-contained units, loosely coupled, to maximize work periods [Herbsleb and Mockus 2003; Sangwan et al. 2006], taking into account levels of teams' expertise. Another issue that has elicited a considerable amount of discussions lately is the team leader's availability. Its importance should not be neglected since it affects the whole project organization and impacts on the way that projects evolve, hopefully within budget and time constraints due to deadlines. Resource availability has been the subject of much interesting research [Setamanit et al. 2006]; however, there is a lack of quantitative analysis to uncover their influence in several scenarios for GSD projects.

The literature on quantitative analysis and formal models is rich, and several authors offer quite different points of view. For instance, Haake et al. [2010] have presented a model of collaborative work to support context-based adaptation using the notion of states in a graph, whereas Lamersdorf et al. [2011b] have presented an analysis on the effect of distribution and time zones on communication in distributed projects. Abrahamsson et al. [2007] offer an incremental model for estimating development effort during project execution, without imposing overhead on the project, allowing early and more frequent feedback to developers and managers. Browning and Ramasesh [2007] discuss the application of activity-based models to decompose a development process into a network of tasks in order to evaluate ways for reduction in process time. Earlier, Moser et al. [1997] applied this modeling practice to evaluate global product development considering distance in coordination features, and further, using simulation for predicting behavior [Moser et al. 1998].

Other authors have discussed risks and mixed statistical evaluations applied to GSD realities. For instance, Lamersdorf et al. [2011a] address customizable models for project evaluation mainly focusing on the assessment of risks related to specific work distributions. Sfetsos et al. [2006] have presented a statistical evaluation on the application of Extreme Programming (XP) practices in software companies using a sample survey technique consisting of questionnaires and interviews.

There is a trend in GSD research to combine and relate formal models to empirical studies. In Broy et al. [2007], service-oriented software systems are described using interacting components, proposing a formal model of services. Syed-Abdullah et al. [2006] have discussed an empirical study investigating whether an agile methodology has any distinct effect on the overall well-being of the software developers. Quantitative and qualitative methods were utilized, including participative observations, focusing on group interviews, close-ended questionnaires, and simple statistical tests. Lin et al. [1997] propose a framework for software-engineering processes simulation called SEPS, in which a simulation approach is used to measure the trade-off of cost, schedule, and functionality in a planning tool for testing the implications of different configurations on project success. It is important to note that simulation approaches often produce samples for later statistical analysis, which is an approximative technique to estimate entities behavior and scenario outcome [Stewart 2009]. Unlike simulation, analytical modeling provides a more reliable way of calculating quantitative indices about scenarios, since models are translated to a state-transition system whose related matrix representation and corresponding linear equations can be solved by numerical

methods. Therefore, the results obtained from analytical models are exact indices calculated from a set of parameters, which is an uncontested advantage over simulation techniques that need huge amounts of samples to achieve a satisfactory precision.

The general approach adopted in this work is directed toward analytical modeling using real global software project data as input parameters to extract performance indicators. The main idea is to enable a comprehensive analysis by alternating selected parameters to inspect productivity gains and losses and point out process improvements that could directly impact the project course until completion. One important advantage of such modeling approaches over sample analysis, especially when those samples comes from actual recorded data, is the possibility of generating abundant data with small and large variations about specific situations. Analysis of real data always refers to a specific situation from which measures were taken. It is rare to have data from two realities similar enough to pinpoint specific variations. For example, it is difficult to determine two similar projects, one with four junior developers, and another with two experienced developers, in order to foresee the different behavior of these two realities. Using analytical or simulation models, experiments can be conducted on small and large variations by change of input parameters.

Analytical models have advantages over straightforward event-driven simulations, since they are able to provide exact indices calculated from a discrete set of parameters, using numerical methods[1] such as iterative approaches [Stewart 1994], for example, the Power method, Arnoldi, and GMRES [Philippe et al. 1992]. This process is different from running simulations, in which the sampling process is dependent of the amount of collected samples to provide a valid statistical estimation considering Markovian behavior.

The use of analytical modeling in software-engineering contexts was successfully employed in the past to provide quantitative performance measures [Whittaker and Thomason 1994; Walton and Poore 2000; Farina et al. 2002; Bertolini et al. 2004; Antoniol et al. 2004; Bertolino and Mirandola 2004] for several distinct realities. The literature on this matter is wide and there are different ways to obtain performance estimations, such as (i) monitoring – providing empirical results from analysis of team behaviors and interactions [Mockus 2009; Mockus and Herbsleb 2002]; (ii) simulation– analysis of the evolution and intercommunication of software-development processes in order to help project managers grasp the related impacts in a global context as well as in relation to a team's productivity [Avritzer and Lima 2009; Setamanit et al. 2007, 2006; Kellner et al. 1999]; and (iii) modeling – identification of entities and relations considering spatial and temporal boundaries in distributed projects [Cummings et al. 2009; Avritzer and Lima 2009].

The impact of coordination or cultural diversity on team performance are out of the scope of this article. However, we are aware that these aspects are quite relevant in respect to GSD projects, and further works may include them in our current model. Nevertheless, we experiment on our colocated and globally distributed project models with various scenarios, and the results are analyzed showing the trade-off of choosing different team sizes and compositions.

In this sense, the goal of our article is to present findings in the form of implications found from a large number of model results in order to confirm some preconceptions of GSD area, as well as to question others. In a previous work [Fernandes et al. 2011] an

[1]Large analytical models sometimes need to be solved by simulation approaches. In this case, specialized algorithms and properties are studied as alternatives for achieving approximate indices [Fernandes et al. 2008]. However, the majority of models are solved by iterative methods [Stewart 1994], providing exact indices.

analytical model for software-development teams was proposed and validated with a real case scenario in which our previsions deliver an error inferior to 2%, predicting the overall execution time of a globally distributed project with 16 members working for 22 months. In our current work, we perform several experiments on this generic model in order to analyze the impact of different variations in the performance of a software-development team. We also compare our findings with previous statements from the literature in order to present implications found. This approach illustrates the benefits of analytical modeling, since all conducted experiments were performed by input parameter changes. The observation of real case scenarios for all our experiments would be impracticable; nevertheless, our sets of parameters are based on average behaviors from real project record data. Thus, our experiments reflect possible combinations of real data.

Section 2 contextualizes our work, stating ideas perceived by the GSD community that are discussed in this article. Section 3 presents a systematic mapping study about GSD and Stochastic Modeling (SM) in order to contextualize our contribution among related works from the literature. Section 4 introduces SAN formalism, presenting previously developed models to describe colocated (single-site) projects. In contrast, Section 5 describes in detail a SAN model for a globally distributed project. Sections 6 and 7 present the main contribution of this article, with analysis of the obtained performance indices for the proposed models and the implications found from these results. Section 8 presents our conclusions and discusses future works directed to the extension of models to capture advanced characteristics such as cultural issues and diverse communication problems.

2. BACKGROUND

Related works concerning stochastic models and simulation are developed for the specification of the dynamics of software projects [Padberg 2002] and the usage of analytical models to interpret a team's productivity variability [Avritzer and Lima 2009]. Considering the fact that performance analysis of geographically dispersed teams is emerging [Swigger et al. 2009; Bass et al. 2007; Sangwan et al. 2006; Herbsleb et al. 2005; Herbsleb and Mockus 2003], advances are still needed for the quantitative evaluation of such systems using stochastic modeling as a valuable tool to derive performance indices.

To represent a software project as an analytical model, one can choose from several available structured stochastic formalisms based on Markov chains [Stewart 1994] such as Stochastic Petri Nets [Ajmone-Marsan et al. 1995], Process Algebras [Hillston 1996] or Stochastic Automata Networks (SANs) [Plateau 1985; Brenner et al. 2005]. The present work focuses on the SAN, since it has been successfully used to represent GSD projects [Fernandes et al. 2011]. The SAN is a powerful modeling formalism that works with an underlying Markov chain, providing a high-level description (abstraction) of any given reality. Its basic idea is to represent a system by a set of modules with an *independent behavior* and *occasional interdependencies*. Furthermore, the SAN is a suitable formalism for modeling globally distributed projects due to the fact that development teams can be smoothly abstracted in a modular way. Basically, a module is described by a *stochastic automaton* depicted by a *state-transition diagram*, in which the transitions are labelled with probabilistic and timing information. A SAN model has a set of *events* that triggers state changes on one or more automata. Each event has an estimated frequency, which indicates how often this event occurs per time unit.

The general solution of any analytical model is the numerical computation of its *steady-state probabilities* [Sales 2012; Brenner et al. 2007; Fernandes et al. 2008] enabling the extraction of selected measures of interest, that is, performance indices

regarding the system under evaluation. Previous works in the context of software engineering described SAN models for the analysis of the impact of external dependencies in dealing with teams' local issues [Fernandes et al. 2011], showing that leaders' availability and expertise have a considerable effect on team members' productivity. However, it is important to consider also the members' skills to deal with their own tasks, depending on team size. Such quantitative scenario evaluations are crucial to develop techniques for allocating teams in different locations, deciding how each resource will behave within each environment. Organizations have developed methods throughout the years for capturing detailed knowledge about projects using modeling as a tool to evaluate their work processes, the relationship between activities and requirements, and the interdependency among entities in order to provide better project coordination [Levitt et al. 1999]. Furthermore, at the level of software design, some specific modeling techniques can be used to derive task dependencies and constraints in order to improve the design process itself, identifying its difficult aspects [Smith and Eppinger 1997].

Both industry and academia have a special interest in modeling and predicting the behavior of software-development processes, teams compositions and evaluation, that is, estimating performance indices in scenarios according to different sets of parameters (e.g., different skills, experience levels, and availability for collaboration). In GSD teams, the participants spend large amounts of their time interacting and communicating, and it is well known that, despite best efforts at communicating among dispersed teams, GSD brings more challenges than single-site development [Bass et al. 2007; Herbsleb and Mockus 2003]. In a broader perspective, product-development companies are engaged in complex development processes, since their task force is globally distributed and they face new challenges related to communication and interactions [Tripathy and Eppinger 2011].

Those are the main reasons why it is important to quantify project scenarios and to use the gained information to enhance decision making, avoiding improper utilization of valuable resources. This article aims to exemplify the usefulness of analytical modeling applied to investigate interaction patterns in colocated and geographically distributed projects. We focus our attention on the impact of centralized control mechanisms, a problem that usually surfaces as a major source of communication difficulties in distributed projects. This communication metric may be related to teams with geographic distance and different time zones, but for our study's purposes, all these characteristics will be summarized by the central team's availability.

3. SYSTEMATIC MAPPING STUDY

The systematic mapping study (SMS) of the literature allows the categorization of research types and results. Specifically, we decided to follow the steps defined by Petersen et al. [2008], which consists of defining the research question, conduct search, paper screening, key wording using abstracts, and data extraction (mapping).

3.1. SMS Protocol

The specific SMS for this article was defined over the intersection between GSD and SM. We thus came up with this research question:

"*What has been employed to evaluate the performance of globally distributed software development teams?*"

We believe that, by answering this question, we will be able to contextualize our article's contribution, as well as to compare our findings with other similar approaches.

The conduct search starts by identifying the relevant literature. Specifically, we choose to apply three distinct search strings:

(1) Global Software Development *AND* Analytical Modeling
(2) Global Software Development *AND* Stochastic Automata Networks
(3) Global Software Development *AND* Performance Analysis

Those search strings were applied to some important digital libraries, such as ACM[2], IEEE[3], Scopus[4], Wiley,[5] and Science Direct[6], delivering 62 papers. In order to avoid duplicated studies, we have selected 9 papers out of the 62, excluding the those

- published in nonacademic venues;
- written in other languages than English;
- considered as tutorials, short papers, or lectures;
- mentioning "performance analysis," "stochastic automata networks," or "analytical modeling" only in the abstract or introduction, but not actually making any further considerations in the central contributions.

As part of the SMS process [Petersen et al. 2008], during the screening of papers, we have applied exclusion and inclusion criteria. In this context, it is important to note that from the nine primary studies found, four were related to our research group. Consequently, we decided to apply the inclusion criteria of studies based on analysis of primary studies references, thus including 14 additional studies. Two of those 14 were not part of the digital libraries that we have used and 10 were using different keywords that did not match our research strings. The selected studies of this process are presented as follows:

Primary Found:

- Fernandes et al. [2011]: This study is the basis of our GSD modeling work using single-site and multiple-site comparisons.
- Urdangarin et al. [2008]: This is a related work that started mapping the central team configuration for GSD teams and reported that the existence of a central entity is beneficial for GSD projects.
- Cataldo et al. [2007]: This research pointed out that tools may not be sufficient to improve GSD communication issues. The point is that our work can help to predict those issues prior to occurrence.
- Poikolainen and Paananen [2007]: This study reports how difficult it is to define measures for GSD performance evaluation. This gap can be reduced using performance evaluation.
- Czekster et al. [2011b]: We have created a model to handle a specific type of GSD project named *follow the sun*, which has been used as the basis for our work.
- Matusse et al. [2012]: This work presents a literature review, showing modeling studies that have been used as part of our reference background.
- Avritzer et al. [2010]: In this related work, the authors reported that the success of a GSD project is based on team configuration and communication, factors that we cover in our model.
- Celik et al. [2010]: This related work also proposes a model to help GSD team allocation; the difference from our study is that the authors proposed a model using social network analysis.

[2]http://dl.acm.org.
[3]http://ieeexplore.ieee.org/Xplore/home.jsp.
[4]http://www.scopus.com.
[5]http://onlinelibrary.wiley.com.
[6]http://www.sciencedirect.com.

Inclusion Criteria:

- Jalote and Jain [2004]: This related work covers GSD team allocation, specifically at the task level using CPM analysis.
- Hossain et al. [2009]: This paper presents a literature review of Scrum in the GSD environment. Of specific interest for our study, it draws some conclusions concerning the benefits brought by a qualified and proactive central team.
- Padberg [2002]: This related work presents a simulation model using a theoretical project in order to evaluate schedule performance.
- Casey and Richardson [2006]: This study presents a qualitative analysis of two organizations in terms of communication and management issues. Of specific interest for our study are their considerations regarding technical ability and knowledge transfer between the central team and members.
- Setamanit et al. [2007]: This study presents a model to evaluate allocation issues and schedule impact. The difference from our work is that our model is stochastic and has an approach evaluating the availability of a central team.
- Sooraj and Mohapatra [2008]: This related work presents an approach for a 24h software-development process showing the interaction between two sites. It is different from our model, which can be used for N sites.
- Taweel and Brereton [2006]: This related work demonstrates GSD challenges that can be applied to modeling as well as providing a graphical model.
- Ramasubbu et al. [2011]: This related work describes experience supporting conclusions concerning configuration of globally distributed software-development teams.
- Dafoulas et al. [2009]: This related work presents a base model for GSD configurations and reports that simulation is a feasible tool to evaluate GSD projects.
- Ebert and De Neve [2001]: This work presents general statements concerning GSD decisions, in contrast with the traditional single-site solution.
- Houston et al. [2001]: This work presents a simulation tool created by the authors in order to evaluate risk management activities, a different view from the scope of our model.
- Laurent et al. [2010]: This related work presents a visual notation for requirements of the management process on GSD, presenting the challenges in this context, and also mentions important aspects covered by our model, such as roles, communication, and shared resources.
- Raffo and Setamanit [2005]: The authors report in this study that simulation models can be used for GSD projects, confirming the findings of our research group work.
- Setamanit et al. [2006]: The authors presented a GSD model that can be used as support for project planning, especially aspects such as communication frequency and trust. The scope of our model does not cover the trust aspect.

After finishing data extraction, we started the SMS keywording process, assigning at least one category to each found study as presented in Table I. This SMS resulted in 11 categories related to GSD modeling and performance evaluation. The most commonly found variable was communication (6), followed by time-zone differences (5), allocation strategies (4), and schedule performance (2). The remaining variables were found once.

3.2. SMS Final Remarks

This literature mapping enabled us to select 19 different studies about GSD aspects using performance-evaluation models. However, excluding those developed by our research group, all other works are based on specific simulation tools. Therefore, we believe our line of work is original by the actual proposal of a formal model that can be solved and analyzed by generic performance evaluation tools, for example, Ciardo et al. [2006], Brenner et al. [2007], and Sales [2012].

Table I. Study Categories

Study	Categories
Urdangarin et al. [2008]	Communication
Cataldo et al. [2007]	Communication
Poikolainen and Paananen [2007]	Communication
Czekster et al. [2011b]	Time zone difference, task allocation
Matusse et al. [2012]	Metrics
Fernandes et al. [2011]	Communication, workload
Avritzer et al. [2010]	Communication
Celik et al. [2010]	Resource assignment
Jalote and Jain [2004]	Resource assignment
Hossain et al. [2009]	Communication, roles
Padberg [2002]	Schedule performance
Casey and Richardson [2006]	Communication, task allocation
Setamanit et al. [2007]	Schedule performance
Sooraj and Mohapatra [2008]	Time zone difference
Ramasubbu et al. [2011]	Resource assignment, schedule performance
Taweel and Brereton [2006]	Time zone difference, task allocation
Dafoulas et al. [2009]	Task complexity
Ebert and De Neve [2001]	Communication
Houston et al. [2001]	Risk management
Laurent et al. [2010]	Requirements management
Raffo and Setamanit [2005]	Time zone difference
Setamanit et al. [2006]	Communication, effort

4. SAN FORMALISM AND SINGLE-SITE TEAM MODEL

Analytical modeling formalisms are usually employed to describe real systems in a state-based approach. An example of a well-known modeling formalism is *Markov chains* [Stewart 1994], which is applied to several domains such as bioinformatics, economics, chemistry, and engineering, to name a few. Such models use simple primitives, such as states and labelled transitions, to represent system evolution and operational semantics. In the context of software engineering, analytical models can be widely used to estimate costs, needed effort, resource utilization, and the amount of time it would take to build a specific software-based system or product, or even to study the dynamics of software projects and a team's productivity. Modeling software development teams, often geographically dispersed, is an important task since it can determine the success or failure probabilities of a project. Stochastic models [Bernardo and Hillston 2007] can focus, for example, on important factors such as communication and coordination issues among teams.

The SAN is a high-level structured formalism to represent structured Markov chains [Plateau 1985; Brenner et al. 2005]. There is a wide scope of SAN applications mainly focused on performance evaluation of parallel and distributed systems [Assunção et al. 2013; Brenner et al. 2009; Chanin et al. 2006; Baldo et al. 2005; Bertolini et al. 2004; Farina et al. 2002; Fernandes et al. 2013b; Santos et al. 2015] that can profit from SAN primitives, such as synchronizing events among entities and functional dependencies to model complex interactions. Moreover, SAN is a formalism well fitted to model global software development environments in a modular and efficient fashion. The basic idea is to observe discrete states related to a scenario as well as the dynamics represented by a set of events with associated timed information (rates). After parameterization of the model events, one can obtain the numerical solution, that is, the steady-state probabilities of being in each state. The numerical analysis

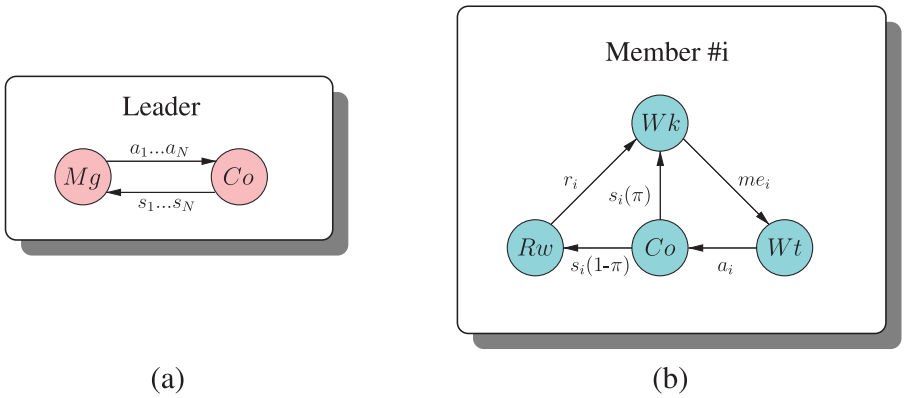


Fig. 1. Leader and Member entities (single-site context).

of SAN models can be performed by dedicated software packages such as *Performance Evaluation of Parallel Systems* (PEPS) [Brenner et al. 2007], *SAN Lite-Solver* [Sales 2012], or *GTAexpress* [Czekster et al. 2009], which implement numerical solutions using iterative [Czekster et al. 2010a, 2011a], symbolic [Fernandes et al. 2013a] and simulation [Czekster et al. 2010b] techniques.

4.1. Single-Site Team Model Automata and Events

This section presents a model for a development team for which members are located in a single-site context. This model was initially presented in a previous work [Fernandes et al. 2011] and is presented here to introduce the SAN formalism, as well as to serve as a comparison paradigm to the multisite (and globally distributed) team model proposed in Section 5.

As any SAN model, the single-site team model is composed of several automata, each representing an individual, but not independent, entity. The aim in modeling such entities is to represent discrete states in which team components can be during a full workday. Moreover, one can express the interactions of these members using different events at various rates, composing several scenarios of evaluation. The states of automata are changed due to local events, which change one automaton state at a time, or synchronizing events, which possibly change more than one automaton at the same time.

In our model, one automaton describes the management staff, typically the project manager, but also the supplier and R&D managers can be represented. The other automata describe the developers, designers, architects, or even Q&A experts, since these members work under leader supervision, that is, reacting to leader demands or requesting leader assistance. The automaton called *Leader* represents the management staff. Each automaton *Member #i* represents the ith team member of the team ($i = 1, \ldots, N$).

Figure 1(a) illustrates the automaton *Leader*, which has two different states: *Mg* (Management) and *Co* (Collaboration). For all purposes, these two states represent that the leader is available to collaborate with members while in the *Co* state, or unavailable while in *Mg*. Practically, being in those states enables events for transitions, depending on automata *Member #i* synchronizations. The events called a_i (with the leader availability rate) and s_i (with the leader support rate) are related, respectively, to the time spent in management activities and the period of time that each team member collaborates with the leader. Actually, event a_i represents the start and event s_i represents the end of communication between the leader and the ith member.

Figure 1(b) generically depicts the automata describing a team member. Each automaton *Member #i* has four states: *Wk*, meaning the member working in an assigned task; *Wt*, meaning the member waiting for collaboration; *Co*, meaning the member collaborating with the leader in a meeting, chat, or other channel; and *Rw*, meaning the member reworking an issue. Events a_i and s_i presented in the *Member #i* automaton synchronize the change of state of the ith member with the leader. These two events are related to the leader's availability and support, considering that there is a probability π of the leader to solve the issues during the collaboration and the complementary probability $(1 - \pi)$ to keep the issue unsolved and force the member to rework. Local events me_i and r_i, on the contrary, are independent of the leader behavior. These events are related to the member's expertise itself and capacity to rework in a given issue, that is, the time spent working (*Wk* state) until the member needs to collaborate (*Co* state) and the time spent reworking (*Rw* state) an unsolved issue.

The development team interaction pattern used for this single-site context model, that is, the relations between leader and members, is focused on synchronizing interactions in fixed periods of time. With this perspective, we consider a team composed of single members to a maximum size of N members attached to one leader[7] to collaborate. The development team behavior states that members work on their assigned tasks, engaging in local cooperation with other members and interactions with the leader. In this abstraction, the leader has a holistic project view and the capability to reassign tasks, to consider minor decisions, and to carry out new (small) developments according to project demands (encapsulated in the *Mg* state of the leader automaton).

4.2. Single-Site Team Model Event Rates and Numerical Results

For simplification in this model example, the members and issues to study are homogeneous, meaning that every member has the same level of expertise and time spent in rework, but the model is flexible to consider distinct rates for each member. According to estimations based on empirical experience acquired in academic and business projects, permanence times are assigned to every state in the model, that is, rates correspond to frequencies at every connection among states.

Specifically, for all rates used in this article, we estimate event rates according to average values observed from a database with record data of nearly 300 real projects conducted in a world-class software-development company. Such data is protected by a nondisclosure agreement; therefore, we limit the information about it to the average values assumed for the events of the models in this article.

Despite the origin of estimated event rates, a project manager willing to use our model can set specific rates according to its own project assuming mean behaviors for each participant's model rates. For instance, according to our project's data, an inexperienced member works an average of 90min before encountering a problem that requires assistance from the leader. Such a value was a round value from 92min average time between impediments for junior developers found in our 300 projects' recorded data. An expert member, on the contrary, may work an entire day (eight hours, or 480min) before needing assistance from the leader.

Therefore, assuming inexperienced members, the rate of event *me* that leads from the *Wk* state to the *Wt* state will be equal to $\frac{480}{90} = 5.3333$, that is, the event will occur, on average, 5.3333 times per working day. On the contrary, assuming expert members, the rate of event *me* will be 1, that is, once a working day ($\frac{480}{480}$).

Table II shows the estimated rates for the events in the single-site model, assuming as reference an 8h workday. It is important to remember that the numerical values

[7]Although using a flat structure with one leader and N undistinguishable members, this model could be extended to several leaders and members split in subsets without any loss of generality.

Table II. Event Estimated Rates for the Single-Site Team Model Example

Event	*Estimated rates for an 8h workday*
me	**(I)nexperienced:** Member demands cooperation, on average, after *90min* of work, that is, a member with a lack of required skills (or low expertise) to accomplish its own responsibilities autonomously.
	(E)xpert: Member requires cooperation, on average, *once a day*, that is, an expert member presenting a high expertise level related to its role.
s	**(L)ow:** Leader requires, on average, *90min* for solving issues brought by members.
	(H)igh: Leader requires, on average, *30min* for solving issues brought by members.
a	**(A)vailable:** Leader cooperates with a member after *30min* of management time.
	(B)usy: Leader presents low availability due to management duties, that is, the leader cooperates only *once a day*.
r	**(R)ework:** Member requires, on average, *120min* to review/correct its tasks.

needed to estimate each event rate was based on managers' records from real projects. Nevertheless, it was our choice for this experiment to model a software-development team with one leader and N members, where N varies from 2 to 9. From these initial values assigned to each event, one can assume different combinations of rates in order to analyze more effectively their impact on the model dynamics considering various team formations.

Therefore, combining the estimated event rates presented in Table II, we assemble the eight possible scenarios[8] [Fernandes et al. 2011] called: ILA, ILB, IHA, IHB, ELA, ELB, EHA, and EHB. For instance, the ILA scenario represents a team with *inexperienced* members (I), a leader that provides *low* support (L) – that is, an inexperienced leader in solving issues demanded by members – but often *available* to cooperate with the members (A). We choose to vary parameters of the three major aspects that our model captures, looking for results that promote a better understanding of the relations among those variables. It is important to keep in mind that event estimated rates are empirical values and those estimations are very dependent on the modeler's knowledge and available data from previous project observations.

Note that different inputs (event rates) can generate different outputs (performance indices or steady-state probabilities) even for small scenarios. For example, Figure 2 shows the results for three scenarios in the single-site development context, for which the impact on team members' productivity is considered according to the assumed members' expertise and leader's availability. The productivity (y-axis) is computed as the average probability to have the members in the working (Wk) state. These results distinctly point out that, for small-sized teams, it is better to have an available leader that provides high support to an inexperienced team (ILA scenario), than a busy leader providing support to an experienced team (EHB and ELB scenarios). On the contrary, as team size increases, it is possible to disregard available leaders only if the members are experienced, independent of the level of support given by the leader (EHB and ELB scenarios). This result is emphasized in Figure 2 by the significant decrease of productivity of IHA in contrast to the less pronounced degradation of performance presented for ELB scenarios, and even less for EHB scenarios.

[8]The terminology "scenarios" instead of "experiments" is intended to call the reader's attention to the fact that our results are of a different nature than those often found in the GSD literature. As mentioned in Section 3, related works are based on simulation methods, while our work performs a numerical solution of formal methods, thus delivering exact measures.

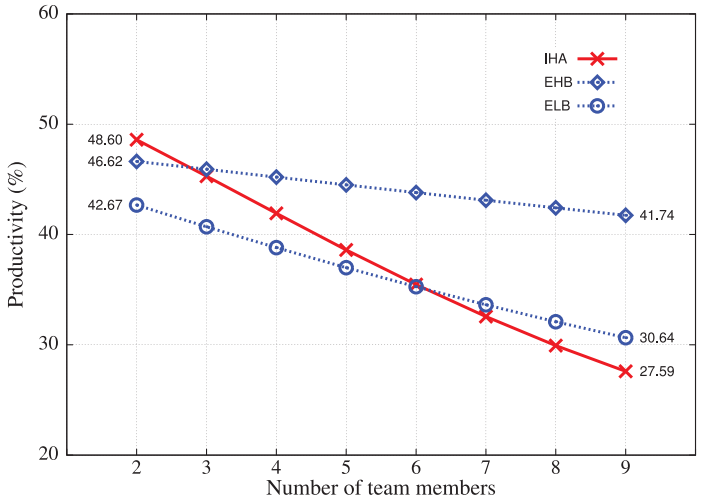


Fig. 2. Impact on the productivity considering members' expertise and leader's availability.

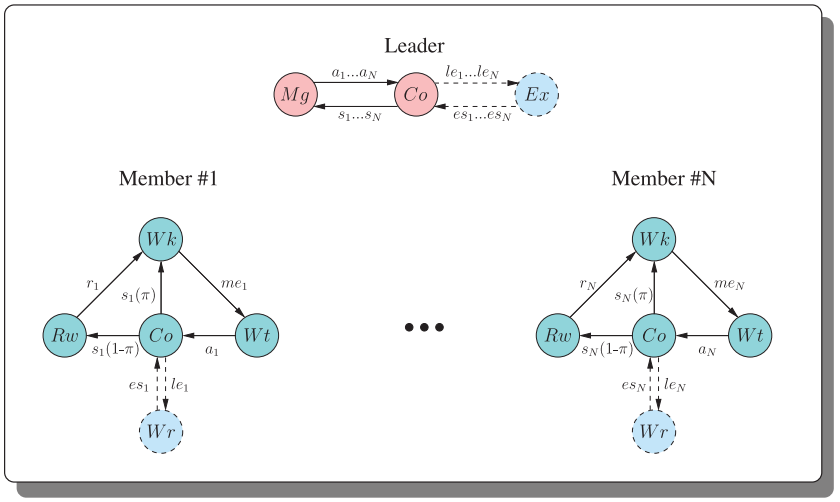


Fig. 3. Example of a software development team with off-site support.

4.3. Single-Site Team Model Extended Version with Off-site Support

The use of formal models and analytical evaluations can generate new ideas regarding how to compose teams and interaction patterns to deal with different team sizes and project demands. Recent research parameterized the single-site model and extended it to a multisite context [Fernandes et al. 2011], in which a centralized external resource is needed to solve more complex questions, also forcing synchronizations among members and their leader to accomplish assigned tasks. In order to model this new form of interaction, new states and synchronizing events with their respective rates were added.

Figure 3 shows a multisite configuration with additional states for the leader and for team members, and the interplay between them. The leader is modeled with an extra state *Ex*, which indicates that the leader is interacting with an off-site central

Table III. Event Estimated Rates for Multisite Model in Figure 3

Event	*Estimated rates for an 8h workday*
a	**(A)vailable:** Leader cooperates with a member after *30min*.
	(B)usy: Leader presents low availability due to management duties, that is, the leader cooperates only *once a day*.
le	**(I)nexperienced:** Leader demands cooperation with central team, on average, *four times a day*.
	(E)xpert: Leader requires cooperation with central team, on average, *once a week*.
es	**(L)ow:** Central team requires, on average, *one day* for responding to issues demanded by leaders.
	(H)igh: Central team requires, on average, *30min* for responding to issues demanded by leaders.
me	Member demands cooperation, on average, *twice a day*.
r	Member requires, on average, *2h* to review/correct its tasks.
s	Leader requires, on average, *1h* for solving issues demanded by members.

team, becoming unavailable to collaborate with the members. In a similar way, state *Wr* was added for members' automata, indicating that the members wait for the leader to resume collaboration. State *Wr* forces a synchronization between the leader and members while the leader is communicating or cooperating with the central team. In Figure 3, we represent the leader and members dependability of the central team by the dashed transitions and states (*Ex* and *Wr*).

In this extended example, the external collaboration with the off-site central team is directly influenced by the level of expertise of the leader (event *le*) in addition to factors such as distance from the central team, different time zones, and cultural and language diversities [Herbsleb and Moitra 2001] represented by synchronizing event *es*. The other events remain the same as those of the single-site model for simplification purposes. Table III redefines the rates parameterization for this new example.

Using the event estimated rates for the off-site model presented in Table III, eight possible scenarios were defined as follows: AIL, AIH, AEL, AEH, BIL, BIH, BEL, and BEH. These scenarios were parameterized with average values to the team's expertise and experience, and level of support provided by the leader, which refers to the events *me*, *r*, and *s*, respectively. As explained in Section 4.2, all estimations for scenarios in this article come from average values computed from data records of more than 300 actual projects.

The performance indices obtained from these models bring some questions about the need of broad leader availability in teams with different levels of expertise and specific characteristics as well as external support availability to communicate. For example, in Figure 4(a), we observe the team members' productivity on highly available external support scenarios (AIH and BEH), in order to inspect some relevant decisions such as *which type of leader is better: an available and inexperienced leader or a busy and expert one?* In this case, our results show that it is better to have an available leader even if the leader is inexperienced, since the leader provides high external support, compensating for the lack of experience. This fact is corroborated by the numerical results for all team sizes. However, the variation of productivity is considerably greater for AIH scenarios than BEH scenarios, that is, the decrease of team members' productivity is emphasized much more in AIH than BEH scenarios.

Additionally, these results allow us to answer this question: *Is it also better to have an available and inexperienced leader than a busy and expert one in low external support*

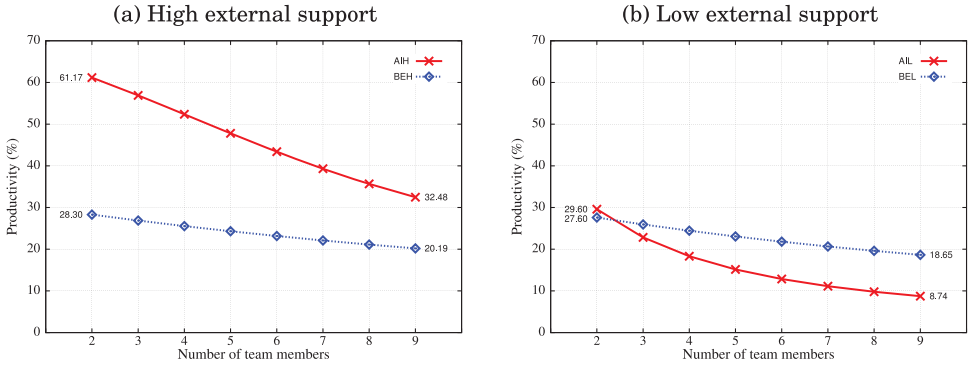


Fig. 4. Team members' productivity on high and low external support scenarios.

scenarios? Observing the initial productivity values, that is, the results for very small teams (just two members), the answer is yes. On the contrary, as team size increases, the team members' productivity is better overall having an expert leader, despite being busy. In this case, the leader experience improves the members' productivity, as indicated by the AIL and BEL curves crossing shown in Figure 4(b).

5. GLOBALLY DISTRIBUTED TEAM MODEL

This section proposes a model of a software development team with N participants that interact (communicate) among themselves and with a central team to solve issues and collaborate. The interactions are now dependent on time zones, availability, levels of support, and so on. The model copes with this information declaring events associated to specific transitions. Moreover, in comparison with earlier modeling experiments, we model more frequent collaborations and new states for abstractions related to face-to-face communications among participants and self-learning in some situations triggered by the level of expertise and a lack of leader availability.

5.1. Globally Distributed Team Model Automata and Events

Figure 5 depicts the development team interaction pattern in a globally distributed project. We model the central team as two automata representing the states in which the central team could be in terms of activities and availability. The first automaton (*Availability*) has two states: A (central team *available* to cooperate) and U (central team *unavailable* for a given reason, for example, time zones and other meetings). The second automaton (*Activities*) also has two states: M (central team performing *management* activities in general, according to the specific modeled scenario) and C (central team effectively cooperating with a participant).

We model the participants of a software development team with different states as follows: W means that the participant is *working*, completing its tasks, or collaborating with other members; S means that the participant is *seeking for a specific solution*, information, documentation, sources of data, or even learning some technical issue by its own; C means that the participant is collaborating with the central team due to solve technical questions. Figure 5 presents the stochastic automata network for this proposed scenario. Note that more development teams could be attached to a central team, including more instances of synchronizing events s and co in automaton *Activities*.

The local behavior of a team describes that, when members are actually working, they can stop for a while seeking a solution on their own, or preferably move to cooperate with the central team, returning to the working state after that. The central team

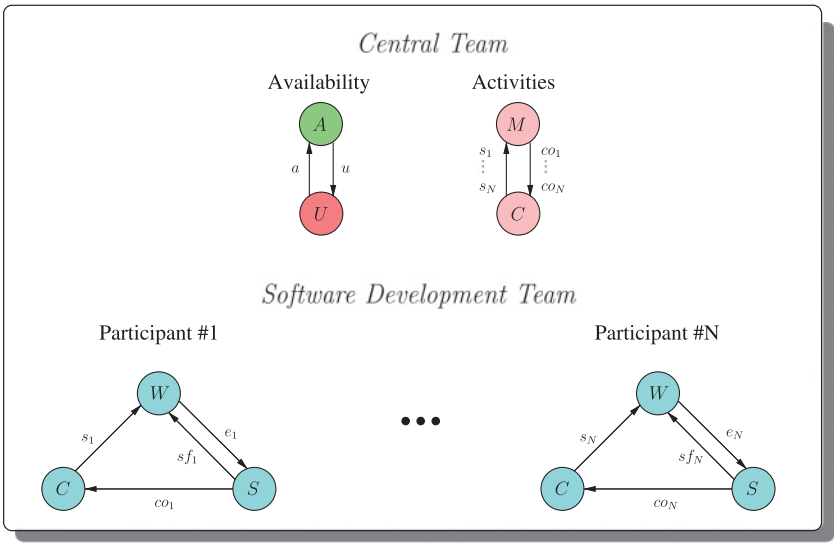


Fig. 5. Global software development team model.

is also managing or cooperating with the participants, if automaton *Availability* is in state *A*.

It is worth noting that our abstraction is powerful enough to consider the global context as being represented by the addition of new states and events, allowing the model to simply capture new characteristics in comparison to the earlier models presented. However, our modeling choice was to envelop all details in each task in the states, transition, events, and rates. For instance, eventual breaks, extra working hours, absence of participants, and people replacements are all transparent to the model, since when we consider an average of 480min as a working day, we do not take into account if the participant is continuously working 480min or it is a composition of smaller portions of time.

5.2. Globally Distributed Team Model Event Rates

It is important to note that events *s* and *co* in Figure 5 are fully synchronized, that is, as soon as the central team is available, a participant requesting collaboration can immediately start. This abstraction is represented by a functional rate involving the verification of central team availability each time a participant wants to collaborate while seeking a solution. The collaboration state has an average fixed time in which the central team and a participant remain in the state, and the given participant returns to the working state when finished with the meeting/collaboration. Table IV presents the events depicted in the model of Figure 5, which correspond to the new activities performed by the central team and participants in the global context.

We proceed to the explanation of the events and their associations with some estimated rates (Table V). These rates take into account the configuration of a remote team in a global context in order to map the participants' behavior for different case studies. As explained in Section 4.2, all estimations for scenarios in this article came from average values computed from data records of more than 300 actual projects. It is important to stress that a project manager using our model is able to model one's own team assuming specific productivity levels for each participant. The rates assumed in Table V may be used as reference since they represent a sample of GSD data. However, our model is flexible enough to consider particularities of any specific project.

Table IV. Description of the Events in Figure 5

Event	*Description*
a	**Availability:** This event indicates the moment when the central team becomes available to cooperate with participants.
u	**Unavailability:** This event indicates the moment when the central team becomes unavailable to cooperate with the participants.
co_i	**Collaboration:** This event starts the cooperation of the ith participant with the central team.
e_i	**Participant's expertise:** This event represents the ith participant starting to solve issues without cooperating with the central team.
s_i	**External support:** This event represents the ith participant resuming work after the end of the cooperation with the central team.
sf_i	**Solution found:** This event represents the ith participant resuming work after finding a solution by himself or herself.

Table V. Event Estimated Rates for the Model in Figure 5

Event	*Estimated rates for an 8h workday*
$a \mid u$	**(A)vailable:** Central team is available to cooperate with participants, on average, during *420min per* workday, that is, 7 of the 8 hours of a workday. Note that events a and u present complementary rates for an 8h workday, for example, 420min actually collaborating (available), and 60mins unavailable (busy).
	(B)usy: Central team presents low availability due to time spent in management duties, or very small time-zone overlap, that is, central team is available only *60min per* workday, for example, software development team in India and the central team in the United States.
e_i	**(I)nexperienced:** Participant works, on average, *1h* before seeking cooperation or finding one's own solution.
	(E)xperienced: Participant works for a long period, approximately *7h,* without requiring any external support or starts looking for a solution on one's own.
s_i	**(L)ow:** Central team takes, on average, *60min* for responding to participant issues. Low support is often characterized by communication issues such as language, available channels, time zones and cultural diversity, but also the central team's expertise.
	(H)igh: Central team requires, on average, *30min* for responding to participant issues.
co_i	This event occurs immediately when the central team is available. Hence, a functional rate verifies this condition in automaton *Availability*.
sf_i	This event assumes that the participant finds a solution by independently in *1h*.

Using the estimated rates for the model presented in Table V, we consider eight possible scenarios combining the defined rates: AIL, AIH, AEL, AEH, BIL, BIH, BEL, and BEH. For example, the AEL scenario represents a team in which the central team is frequently *Available* to cooperate with the participants (7h of 8h); the participants are *Experienced* (working for 7h before seeking cooperation or one's own solution), and receiving *Low* quality support from the central team (issues handled in 1h).

6. PERFORMANCE ANALYSIS OF TEAMS

In this article, a team's productivity is evaluated as the probability of a participant being in the working state (W) since the workday time is split in collaboration (C), seeking solution (S), and working states. In our model, team participants generate output and contribute to project completion in a prompt manner, that is, task

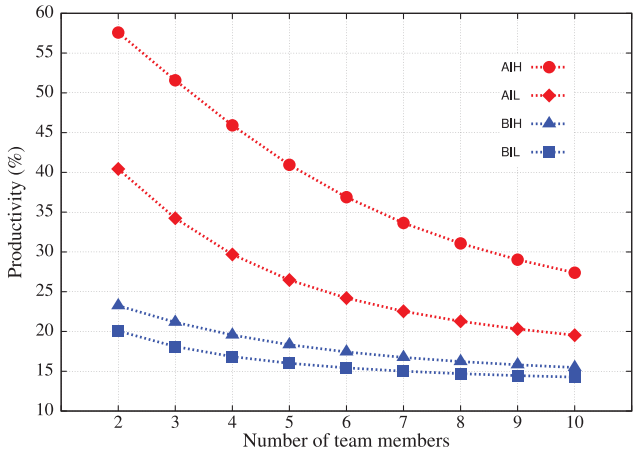


Fig. 6. Productivity trend results (%) for teams with inexperienced participants.

assignments and completion are encapsulated in the working state. The time spent on the *W* state reflects the amount of useful work. Therefore, we consider the team productivity percentage the average probability of being in the *W* state for all participants. The following figures and tables demonstrate the main results obtained from the GSD team model, varying the team size from two to ten participants, since specialists of the domain suggest keeping the number of participants within this limit [Sangwan et al. 2006].

6.1. Analysis of Inexperienced Participant Scenarios

Figure 6 presents the productivity results for four scenarios with inexperienced participants. In this figure, the exact values obtained from the model's numerical solution are plotted in percentage.

As expected, the worst-case scenario is configured by an inexperienced participant coping with a busy central team providing low-quality support when collaborating, that is, the BIL scenario. Obviously, this scenario presents the worst productivity because the participants are often seeking solutions themselves and collaborating with the central team rather than actually working on their tasks.

Given the worst-case scenario, one alternative to increase productivity is to increase the availability of the central team, since changing all inexperienced participants for more experienced ones is probably more difficult. Therefore, we compare the productivity achieved by BIL and AIL scenarios, that is, assuming the central team more available to cooperate. For small-team configurations, the productivity gain is very large (from 20% to 40% with two participants), but as the number of participants increases, the productivity gain drops considerably (from 14% to 19% with ten participants).

Another alternative to improve productivity is to increase the quality of the support, that is, change from the BIL to the BIH scenarios. However, this improvement is negligible (3% gain with two participants and 1% gain with ten participants). Nevertheless, combining the two improvements, that is, improving the availability and support quality of the central team (AIH scenario), the productivity gains are much more impressive (57% productivity with two participants and 27% with ten participants).

The availability and support quality parameters are changed to determine some aspects considering inexperienced participants in terms of communication. As seen in the results of Figure 6, the productivity increases in the scenarios with more available external resources (AIL and AIH scenarios) and, as expected, the best productivity is

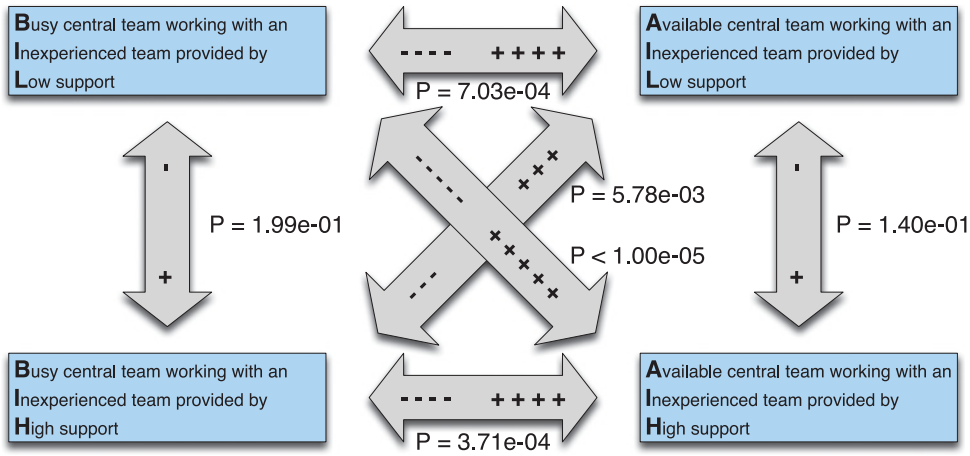


Fig. 7. Teams with inexperienced participants productivity flow.

achieved in the one with high-quality external support to the participants, that is, the AIH scenario.

In addition to the visual observation of the scenario productivity depicted in Figure 6, we include in Figure 7 the computation of the P value using t-test over the values obtained comparing each pair of scenario sets. Specifically, we considered productivity percentage for each class of scenarios (each curve in Figure 6) as a group, and the productivity of scenarios with a different number of participants (vertical section for all curves in Figure 6) as an independent sample of the scenario class. For example, we applied a t-test between AIH and AIL scenarios considering individually the red curves in Figure 6 as sets of 9 samples of AIH and AIL, respectively. Therefore, the resulting P value of 0.140 was considered the statistical relevance of the difference between AIH and AIL, and in Figure 7 the arrow between AIH and AIL indicates with one minus and one plus signs the relevance of the difference. Following the same reasoning, when a smaller P value, such as the 0.000371 encountered between AIH and BIH, indicates a much more significant difference, then the arrow between AIH and BIH is indicated with four minus and four plus signs. When the computed P value was below 10^{-5}, as between the AIH and BIL scenarios, we adopted five minus and five plus signs, indicating a statistically clear difference in favor of AIH over BIL scenarios.

It is important to mention that the choice of t-test instead of other measures of statistical comparison was based on the fact that ordinal data comparisons, such as Friedman and Wilcoxon signed-rams tests, delivered obvious results. After all, the groups of samples are clearly ordered, for example, the pairwise comparison of BIH and BIL scenarios has always the same rank, with BIH values superior to BIH values. For the same reason, other statistical relevance tests, such as ANOVA, delivered the obvious conclusion of hierarchy among methods. Further details about the use of statistical relevance computation methods may be found in Demšar [2006].

6.2. Analysis of Experienced Participant Scenarios

Now, observing more experienced participants, we analyze the impact of support quality and availability provided by the central team on productivity. Figure 8 presents the numerical productivity achieved for the scenarios composed of experienced teams, that is, BEL, BEH, AEL, and AEH, according to the exact values obtained from the model's numerical solution. They were obtained with a 10^{-10} floating-point precision from the model's numerical solution. Specifically, comparing AEL and BEL scenarios

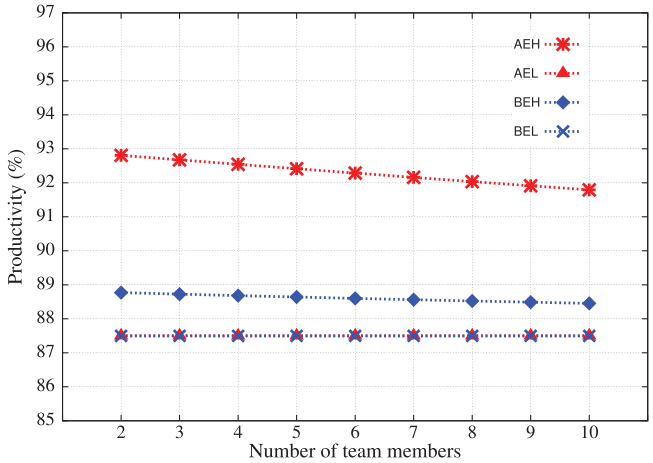


Fig. 8. Productivity trend results (%) for teams with experienced participants.

productivity, there were slight differences found in 10^{-6} decimal places in favor of AEL. However, we would like to stress the fact that AEL and BEL have different results below 10^{-5}, which means a negligible difference of less than 0.001%. Therefore, it is possible to affirm that the availability of a low-support quality central team is not an issue for a project with experienced team participants.

The first clear observation from Figure 8 results is that the case with experienced participants shows that they are far more productive than the results obtained for the inexperienced participants (Figure 6). Nevertheless, an interesting result emerged from these comparisons. For instance, one could verify that a more experienced central team, even less available to others, is more important to have in a project than a less experienced one with greater availability, mainly because such project leaders solve issues faster, releasing others to resume working. From a project perspective, the experience to solve and decide the most important issues is much more important than availability, which is a remarkable insight, mapped from a numerically computed index.

Analogous to the results of inexperienced-participant scenarios, Figure 9 depicts the productivity flow for the scenarios with experienced participants. Note that for the scenarios in which the central team provides low-quality support (BEL and AEL), productivity does not seem to change, regardless of the central team's availability. On the contrary, for the BEH and AEH scenarios, that is, the central team delivering high-quality support, the increase in the central team's availability represents a considerable increase in productivity.

It is important to note that the computation of P value using t-test over the sets of scenarios delivered statistical relevance between all cases with high and low support that were very significant (P values under 10^{-5}), despite the visual impression of rather different behaviors given by the curves in Figure 8.

It is important to stress that the model is evaluated using numerical methods. Hence, the presented results are exactly computed and more statistically relevant than results obtained from simulations or real case observations, which often depend on a sampling process. Therefore, assuming that the designer could provide reliable parameters, reliable probabilistic predictions can be achieved for different scenarios.

For instance, the practical equality found for AEL and BEL scenarios regardless of the number of participants is justified by the fact that we assume equally distributed

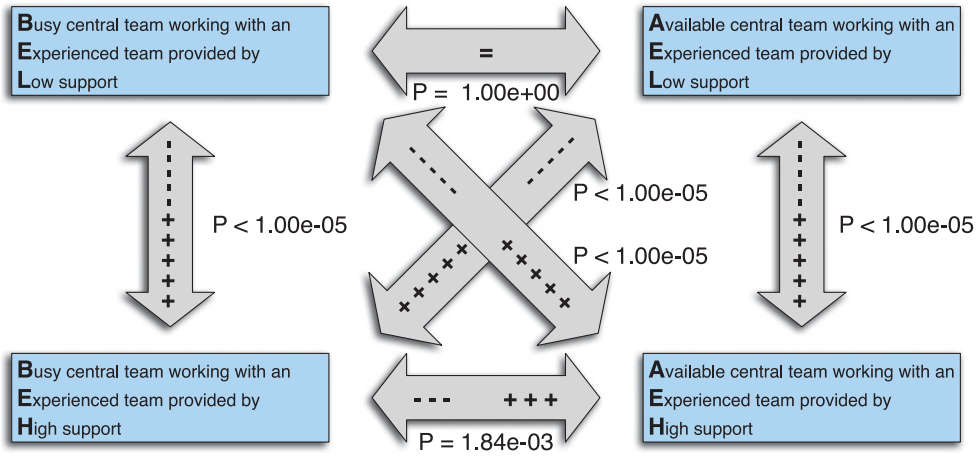


Fig. 9. Teams with experienced participants productivity flow.

tasks among participants, without explicit task dependency indicated, that is, the only dependency found in the model is related to the communication dependence of each participant with a central team or leader. In all activities performed except collaborating, participants are considered to be independent in this model. We know for a fact that further improvements in the models will consider more intricate dependency interactions and more detailed definitions of tasks in quantity, time to completion, and even a more detailed view of productivity needs to be explored. However, even in this situation, which could be improved, it is possible to discover new and counterintuitive relations. For instance, with experienced participants on teams, the degree of availability of a central team providing low support (BEL=AEL) simply does not matter.

An interesting fact numerically shown was the situation in which one has to choose between an available and low-support central team (AEL or AIL) and a busy, but high-support central team (BEH or BIH). In such cases, a different decision must be made according to the experience of the development teams. In the case of experienced teams, the quality of support is more important (BEH > AEL), but for inexperienced teams, the availability is more important (BIH > AIL).

6.3. Analysis of a Very-Low-Quality Support Scenario

Let us consider different scenarios, in which the central team provides very-low-quality support, that is, the central team takes 2 hours to answer requests from participants. Once again, let us observe this situation under a large availability (7h of 8h of a workday, noted as "A") and busy (1h of 8h a workday, noted as "B") situations for the central team. Additionally, let us consider the cases of experienced (7h of work before requesting support or seeking own solution, noted as "E") and inexperienced (1h of work before requesting support or seeking own solution, noted as "I") for the development team's participants.

Figure 10 presents the productivity percentage for these very-low-quality support scenarios labeled AEvL, BEvL, AIvL, and BIvL. These results are quite different from the previous ones since, in these configurations, the central team does not really help the participants, but sometimes actually hinders the productivity of experienced participants, which represents another counterintuitive finding.

Considering the central team availability and experienced-participants scenario (AEvL) from Figure 10, the numerical results show that, as the number of participants increases, the productivity increases. In fact, for this scenario, the experienced

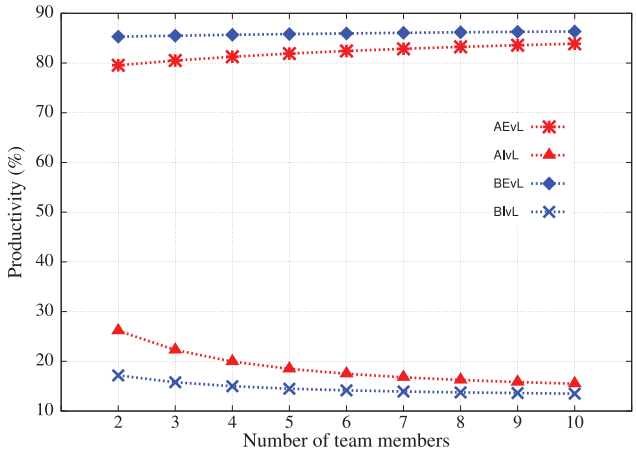


Fig. 10. Productivity trend results (%) for teams with very-low-quality support from central team.

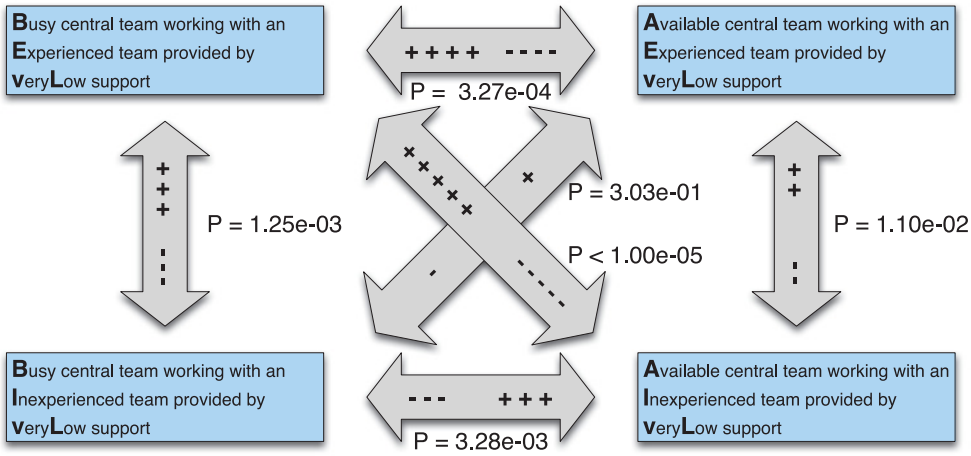


Fig. 11. Very-low-quality support from central team productivity flow.

participants find solutions to their problems faster by themselves versus with collaboration of the central team, as seen by the higher productivity of the analogous BEvL scenario. On the contrary, with inexperienced participants (AIvL and BIvL), the productivity decreases as the number of participants increases. The participants left by themselves are even worse than with the low-quality support provided by central team, as attested by the lower productivity of the BIvL scenario, that is, for inexperienced teams, low-quality support is feeble, but nonetheless, still a help.

Observing the P values computed using t-test, as indicated in Figure 11, we observe that the statistical relevance does not follow the intuition about the differences between the curves presented in Figure 10. In fact, the pure observation of the curves is numerically represented by the average of the values. For example, the impression that scenarios AEvL are much more productive than scenarios AIvL is because the average productivity of AEvl is 82.13%, while productivity of AIvL is 18.74%. However, the P value delivered by t-test between AEvL and AIvL is 1.10e-02, that is, the statistical

relevance of the difference between AEvL and AIvL scenarios is not as statistically significant as the difference between AEvL and BEvL (3.27e-04).

6.4. Analysis of the Variation of Participants' Experience

It is a known fact that the participant's expertise is an important parameter to increase productivity [Sangwan et al. 2006]. In Figure 12(a), we present a set of results for scenarios in which an available central team provides low support to the participants, varying the average time in which participants work before seeking cooperation or finding a solution by themselves. For these scenarios, we vary the working time from 60min to 420min *per* workday, which is related to the participants' experience. This parameter variation corresponds to considering participants seeking help from the central team once an hour, that is, working productively for 60min, to once a workday, that is, working 7h (420min). Note that it is assumed that an experienced participant will split the 8h workday into 7h doing productive work and 1h contacting the central team or trying to solve a problem independently.

In order to highlight the increase (+) or decrease (−) in terms of productivity, we have calculated a percentage index called Δ that relates the configuration with 2 and 10 participants. Δ is computed as the ratio between how much productivity was lost from two to ten participants and the maximum productivity (which is always with 2 participants). For example, considering the AIL scenario, which is the first x-axis value in Figure 12(a), the probability for the working state (W) measure for $N = 2$ participants is 40.44%, while for $N = 10$ participants, it decreases to 19.53%. Therefore, $\Delta = 51.70\% = \frac{40.44\% - 19.53\%}{40.44\%}$.

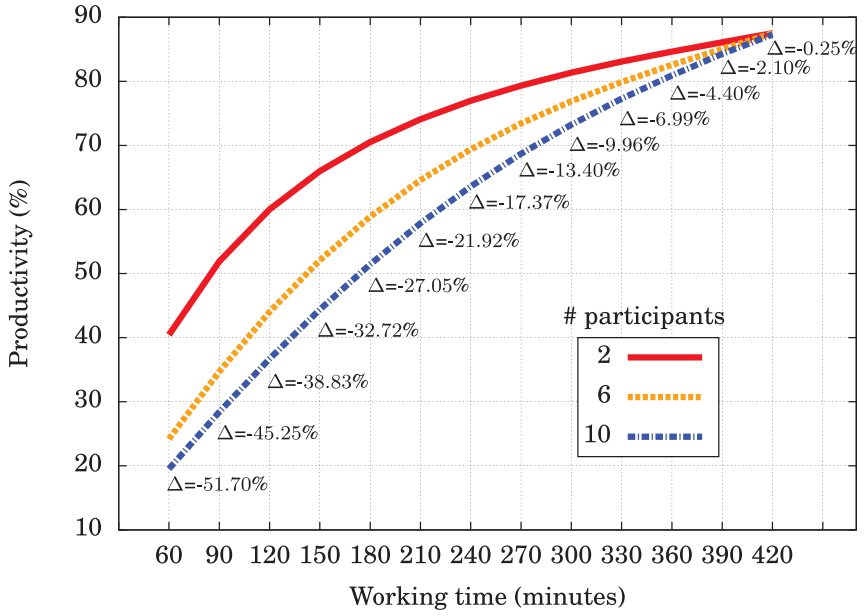
Our aim is to analyze the relation between participants' experience and the central team's availability. In other words, the Δ index quantifies the visual impression that as the participants' experience grows (from 60min to 420min of uninterrupted work), the productivity decrease is less relevant. Numerically, Δ decreases from 51.70% with inexperienced participants (AIL scenario) to 0.25% with experienced participants (AEL scenario). Therefore, it is possible to affirm that the importance of an available and low-quality support central team decreases very quickly as the participants' experience increases.

Analyzing the same situation for scenarios with a busy central team and high-quality support (Figure 12(b)) we observe similar results with a smaller difference between two and ten participants. Note that the Δ index variation here from BIH to BEH scenarios is smaller than the variation in AIL to AEL scenarios (Figure 12(a)). In that case, it is possible to affirm that the importance of a busy and high-quality support central team was not as big as for the A_L scenarios for inexperienced participants (Δ for A_L = 51.7% is bigger than Δ for B_H = 33.39%). However, as the participants' experience increases, the central team importance will clearly decrease, but not as low as for the A_L scenarios (Δ for A_L = 0.25% is smaller than Δ for B_H = 0.36%).

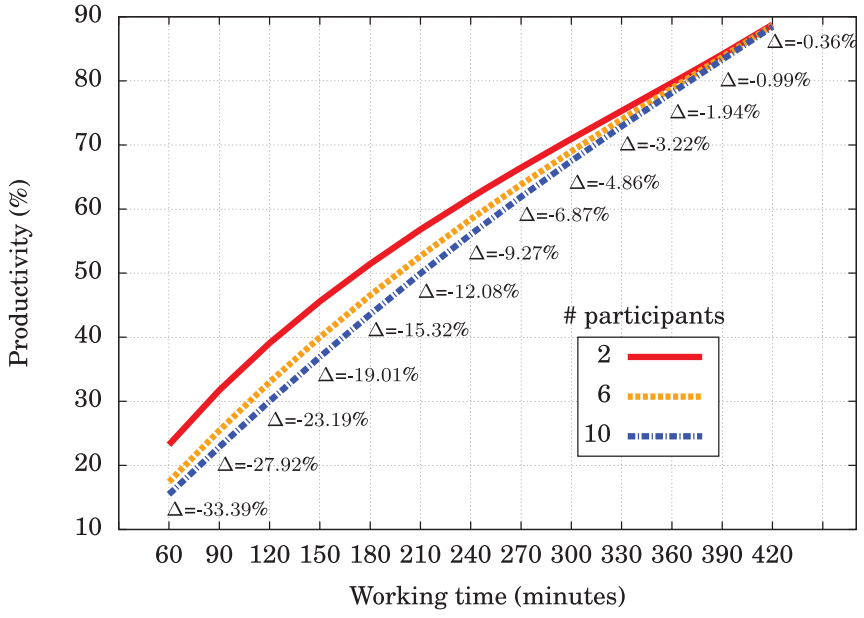
It is important to stress that the use of statistical significance tests such as Friedman and ANOVA are not applicable for this analysis, since the curves depicted in Figure 12 are clearly stratified, that is, the productivity results for 2 participants are always superior to 6 participants, and those are always superior to 10 participants. Consequently, any paired test based on ranking will deliver the obvious conclusion of the ranking order among scenarios.

7. IMPLICATIONS FOUND

We now turn our attention to some implications found for the case under study, considering analysis provided by the numerical results obtained from the models.



(a) A_L scenarios;



(b) B_H scenarios;

Fig. 12. Team's productivity analysis, varying the level of participants' experience.

I1: *For teams with experienced participants, the availability of the central team is of little importance if it provides good-quality support, or irrelevant if it provides low-quality support.*

Observing the results from Figure 8, we have noticed that the results from AEL and BEL scenarios were practically the same. In fact, experienced teams are sufficiently capable to solve their own issues quickly, without the need of constant interactions with the central team. Only the high-quality support from the central team is able to improve productivity, as could be seen in the BEH scenario, and even clearly for the ideal AEH scenario. It is important to note that this finding is counterintuitive for many researchers and practitioners in the GSD area, since they usually assume that central team availability is always beneficial. For example, the observation made by Casey and Richardson [2006] states that, in successful GSD environments, the central team must provide an excellent level of management and knowledge transfer. Analogously, Jalote and Jain [2004] state that communication and coordination difficulties have clear negative effects on project schedule, but our findings indicate that this is not the case for teams with experienced participants.

I2: *According to productivity, teams with experienced participants scale better than teams with inexperienced participants. Moreover, the models show numerically that only teams with an available central team and inexperienced participants tend to not scale very well in comparison to others.*

We observe that the productivity index for scenarios BEL, BEH, AEL, and AEH (Figure 8) remains practically the same as the number of participants increases. However, the teams with experienced participants are not the only cases that scale well. Scenarios BIL and BIH (Figure 6) scale almost as well as the scenarios with experienced participants. These findings are perfectly in accordance with Ebert and De Neve [2001], who state that experienced senior developers enhance productivity, since the engineering costs to detect and correct defects is particularly high, especially for projects without an intensive central team support.

I3: *Teams with experienced participants with high-quality external support provided by the central team have better productivity if the external staff is often available.*

We obtained the best productivity results for AEH scenarios. However, the numerical increase of productivity from the scenarios with low availability (BEH) is quite small (around 3%), as seen in Figure 8. These numbers probably indicate that a highly available central team is not really necessary, especially considering the usual high costs to make a high-quality central team available often. These findings were verified for Agile projects globally [Hossain et al. 2009], but it is natural to expect similar behavior for other approaches.

I4: *Sometimes teams with experienced participants also need central team high-quality support and availability to improve their productivity even more. However, the drawback of this kind of configuration requires additional project costs and a percentage of extra effort from the central team.*

For teams with inexperienced participants, high-quality support and availability is very important, as seen in Figure 6, in which the AIH scenario practically doubles the productivity of the BIL scenario. However, the gains achieved by the AEH scenario compared to the BEL scenario represent a 5% increase in an already highly productive situation (Figure 8), that is, in this case, the extra costs needed must be balanced to become a benefit in the context of the project. These findings are compatible with the

notion of importance of a qualified central team stated in previous works [Jalote and Jain 2004].

I5: *A central team highly available and providing high-quality support does not mean that teams' productivity will significantly change.*

Even though a good and available central team could increase productivity, as seen in Figure 6, the key factor to productivity seems to be the experience of the team's participants. In fact, a central team highly available and providing high-quality support matters only when you have inexperienced participants. Even for these situations, when the central team is both available and highly qualified, the performance increase will be insignificant, as can be clearly seen for the AEH scenario in Section 6.2 results. This conclusion is somewhat surprising compared with the findings in previous works. For instance, Raffo and Setamanit [2005] states that a central team managing strategic factors significantly impacts a GSD project's performance. However, our findings indicate that the importance of the central team is linked to the low level of experience of participants.

I6: *A central team providing very-low-quality support may hinder the productivity of teams with experienced participants.*

As seen in Figure 10, the central team providing very-low-quality support may be seen as an adversity to experienced teams. The higher productivity of BEvL scenarios in comparison with AEvL shows that availability of the central team hinders the team's performance, since the central team provides very-low-quality support. This result is counterintuitive, since it could be expected that a low-quality central team would be neutral to teams with experienced participants. Curiously, when the participants' experience decreases, that is, scenarios AIvL and BIvL, even a low-quality-support central team represents some help. This conclusion was, at the authors' best knowledge, a novelty in the domain, since previous works [Ebert and De Neve 2001; Ramasubbu et al. 2011] express opinions about the benefits of experienced participants in a team, but they do not mention the fact that a low-quality-support central team may actually do harm. In that case, it could be preferable to have inexperienced participants rather than experienced ones, if the central team is not expected to deliver good support.

7.1. Discussion on Model Application

Many software development projects, regardless of chosen methodology, require project managers that are responsible for a team of programmers/testers. Moreover, results from other researchers point out that teams should gather professionals with valued contributions to the project, avoiding communication issues and other problems due to eventual mandatory synchronizations. In this article, we stress that practitioners could benefit from our modeling template as a basis to map their own reality with their own parameters, then compute the numerical solution and analysis in hopes of determining interesting trade-offs for their own projects.

As mentioned during the explanation of models in Sections 4 and 5, the choice of parameters is a crucial aspect for the performance prediction. Our results presented so far assume parameters according to a sample of 300 GSD projects, but it may be interesting for a specific project modeling the choice of specific event rates. The proposed solution mechanism works regardless of the parameters; therefore, stakeholders using our methods may use their own time or frequency information to set their own event rates. Professionals adopting our technique for analysis could also vary a set of parameters and investigate trade-offs and relations, in a "what if" setting. All of

this is possible with our analytical modeling proposal, which analyzes parallel and synchronized behavior within development teams.

As stated earlier, it will be sufficient to parametrize the model with data pertaining to each participant's behavior and execute the numerical solution mechanism to obtain interesting performance measures to guide project decisions, such as for adequate capacity: for example, increase the number of programmers or testers or, for instance, decrease project managers' availability. It is possible to obtain model parameters from several approaches, such as application logs, questionnaires, and even project managers' educated guesses. It is also possible to benefit from systematic reuse of our own model to fine-tune event-rates estimation.

8. CONCLUSION

The use of a formal modeling approach to describe a GSD project is not a simple task. Nevertheless, there are clear advantages in doing so. First, it allows a very thorough reflection about the players, roles, and actual interactions in a GSD project. Another important advantage is the scientific credibility achieved by the performance predictions. In fact, the conclusions obtained are free from possible misinterpretations, since all numerical conclusions are based on solid probabilistic analysis over a given set of input parameters.

In this article, we present two variations to model development teams in a global context, and we analytically solve a large set of scenarios to investigate the productivity index. For our analysis, it is important to note that basically three parameters play a direct influence in a team's productivity: the central team's availability, quality of external support, and the participants' experience level. The results provided valid insights regarding the analyzed examples, but we believe that this article's main contribution is the use of a formal stochastic modeling approach to describe and predict the productivity[9] of a globally distributed development project. With that in mind, we welcome the readers interested in more details about our models to ask us to run variations in terms of different parameters, as well as to disclose the SAN models to those interested in solving the models by themselves[10].

It is our belief that future works could enhance the models including a large number of significant parameters. Especially for globally distributed teams, it is possible to imagine the inclusion of features such as organization behavior, team dynamics, learning curves, and so on. Despite that, even in the current version, the presented models results are already insightful. According to our conclusions, it is possible to foresee the impact of some basic decisions while the team assembling is still being planned. Furthermore, model parameters can be easily changed to produce a wide variety of results, which is unthinkable using the traditional observation of real cases. For example, it is unrealistic to believe that some project manager could observe four different projects to work with experienced and inexperienced participants, with low- and high-quality support central teams in order to take decisions to assemble the team for a specific project. The presented models, on the contrary, were good enough to offer a panoramic view of a wide range of the chosen parameters and even draw some conclusions stated as implications found from our scenarios.

It is important to note that the use of the term "scenarios" instead of "experiments" has a purpose in this article. We chose such terminology to stress the fact that our

[9]The productivity index computed for our models was abstracted from the probability of the working state (*Wk* in single-site and off-site support models or *W* in the GSD model) in teams' participants (Member or Participant) automata (see Section 4).

[10]To request details and running variations or to receive SAN models described in this article, please send a message to paulo.fernandes@pucrs.br.

results are based on a numerical solution of a formally defined stochastic model; therefore, it is distinct from those usually found in the GSD literature. However, this nomenclature does not imply a value judgment, since we do not believe that results found here are better or worse than those obtained from experimental observations. Both options have their advantages, such as the higher statistical relevance of numerically computed results, and the avoidance of misunderstandings on experiments' observations.

Unfortunately, there is also the major disadvantage of a formal approach, which is the high-level abstraction that keeps the input parameters somewhat distant from the daily routine of software-engineering projects. It is rather easy to feed the model with numerically inaccurate input parameters, mostly due to *a priori* misconceptions. For example, the modeler may assume a same average time to an expert project manager to solve a developer request, but that estimation may be based only on the project manager skill, ignoring the developer skill level. Modeling a project with developers having very different skill levels may be much more accurate assuming different rates to synchronize the leader with different members. However, this problem far from invalidates the applied formal approach because informal approaches also may suffer from these incorrect assumptions. Despite the fact that formal approaches are more prone to this kind of error, in informal approaches these errors are usually harder to detect.

Markovian formalisms, usually in their structured form, are widely used for stochastic modeling of myriad real-life problems, since they are reliable and the state-transition paradigm is quite intuitive. Usually, the parameterization of models is a little harder, but it is also the key point in prediction effectiveness.

Therefore, in this article, we present a model considering different degrees of availability, levels of support, and participants' expertise. More complex situations, such as cultural differences and coordination issues, can be added to the proposed model to improve applicability in real GSD projects. Such improvements will probably require a few changes in the states and events, but a much more complex estimation of event rates. However, even with the current level of abstraction, it was possible to obtain relevant performance indices to set up development teams, multisite projects, and resource allocation decisions.

An interesting aspect worth considering is the team building process, in which optimal team configuration is to be sought to improve financial considerations. The model presented here can help this overall process since it sufficiently observes the reality, then performs a sensitivity parameter analysis to discover more efficient team settings. Nevertheless, more experiments may be required before such future work. For instance, in our article, we performed a sensitivity analysis of team sizes to some extent, from 2 to 10 participants. A wider range, or the sensitivity analysis of other aspects, such as the quality of central team support, could provide interesting new insights.

Other possible future work is to enhance the formal model, considering more subtle information about the modeled project. There are different factors that affect teams' productivity such as perceived schedule versus actual schedule, increased interactions due to project milestones and task misconceptions or changes in requirements, and even concentration on the analysis of accomplished project tasks. The measure of teams' productivity can be also influenced by the developers' morale, as they must constantly feel that the project is smoothly progressing [Sangwan et al. 2006]. Most, if not all, of these project characteristics could be included in the model, but such future work would demand many assumptions that involve complex human behaviors.

A more tangible future line of study consists of extending the model to consider other forms of project organization. The proposed model presents a high centralized communication pattern, which can be very risky for some projects. Nevertheless, depending on the size of the team, as well as the levels of expertise and support provided, these risks

can be attenuated or aggravated. The proposed model could be extended to take into account the complexity of tasks, the duration of the project, or even specific phases.

The main contribution of this article is to propose a valid modeling exercise to formally describe software-development projects and extract meaningful performance indices. Some insights on the analyzed models are quite reasonable, since they confirm to intuitive expectations, but also demystify some preconceived assumptions. The best results achieved with our stochastic modeling approach were to express performance predictions with numerical values in a research area that is usually more driven by qualitative aspects based on implications found from previous experiences. Much more research on formal modeling of GSD projects remains to be done, and we believe that SAN formalism may play an important role as a tool.

REFERENCES

P. Abrahamsson, R. Moser, W. Pedrycz, A. Sillitti, and G. Succi. 2007. Effort prediction in iterative software development processes – Incremental versus global prediction models. In *Proceedings of the 1st International Symposium on Empirical Software Engineering and Measurement (ESEM'07)*. IEEE Computer Society, Madrid, Spain, 344–353.

M. Ajmone-Marsan, G. Balbo, G. Conte, S. Donatelli, and G. Franceschinis. 1995. *Modelling with Generalized Stochastic Petri Nets*. John Wiley & Sons, New Era Estate, West Sussex, UK.

G. Antoniol, A. Cimitile, G. A. Di Lucca, and M. Di Penta. 2004. Assessing staffing needs for a software maintenance project through queuing simulation. *IEEE Transactions on Software Engineering* 30, 1, 43–58. DOI: http://dx.doi.org/10.1109/TSE.2004.1265735

Joaquim Assunção, Luciana Espindola, Paulo Fernandes, Maria Pivel, and Afonso Sales. 2013. A structured stochastic model for prediction of geological stratal stacking patterns. *Electronic Notes in Theoretical Computer Science* 296, 27–42. DOI: http://dx.doi.org/10.1016/j.entcs.2013.07.003

A. Avritzer and A. Lima. 2009. An empirical approach for the assessment of scheduling risk in a large globally distributed industrial software project. In *Proceedings of the International Conference on Global Software Engineering (ICGSE'09)*. IEEE Computer Society, Piscataway, NJ, 341–346.

Alberto Avritzer, Daniel Paulish, Yuanfang Cai, and Kanwarpreet Sethi. 2010. Coordination implications of software architecture in a global software development project. *Journal of Systems and Software* 83, 10, 1881–1895. DOI: http://dx.doi.org/10.1016/j.jss.2010.05.070

L. Baldo, L. Brenner, L. G. Fernandes, P. Fernandes, and A. Sales. 2005. Performance models for master/slave parallel programs. *Electronic Notes in Theoretical Computer Science (ENTCS)* 128, 4, 101–121.

M. Bass, J. D. Herbsleb, and C. Lescher. 2007. Collaboration in global software projects at Siemens: An experience report. In *Proceedings of the 2nd IEEE International Conference on Global Software Engineering (ICGSE'07)*. IEEE Computer Society, Piscataway, NJ, 33–39.

M. Bernardo and J. Hillston. 2007. Formal methods for performance evaluation. In *7th International School on Formal Methods for the Design of Computer, Communication, and Software Systems (SFM'07)*, Lecture Notes in Computer Science, Vol. 4486. Springer, Berlin.

C. Bertolini, A. G. Farina, P. Fernandes, and F. M. Oliveira. 2004. Test case generation using stochastic automata networks: Quantitative analysis. In *Proceedings of the 2nd International Conference on Software Engineering and Formal Methods (SEFM'04)*. IEEE Computer Society, Washington, DC, Beijing, China, 251–260.

Antonia Bertolino and Raffaela Mirandola. 2004. CB-SPE tool: Putting component-based performance engineering into practice. In *Component-Based Software Engineering*, Ivica Crnkovic, Judith A. Stafford, Heinz W. Schmidt, and Kurt Wallnau (Eds.). Lecture Notes in Computer Science, Vol. 3054. Springer, Berlin, 233–248. DOI: http://dx.doi.org/10.1007/978-3-540-24774-6_21

L. Brenner, P. Fernandes, J.-M. Fourneau, and B. Plateau. 2009. Modelling Grid5000 point availability with SAN. *Electronic Notes in Theoretical Computer Science (ENTCS)* 232, 165–178.

L. Brenner, P. Fernandes, B. Plateau, and I. Sbeity. 2007. PEPS2007 - Stochastic automata networks software tool. In *Proceedings of the 4th International Conference on Quantitative Evaluation of SysTems (QEST'07)*. IEEE Press, Edinburgh, UK, 163–164.

L. Brenner, P. Fernandes, and A. Sales. 2005. The need for and the advantages of generalized tensor algebra for Kronecker structured representations. *International Journal of Simulation: Systems, Science & Technology* 6, 3–4, 52–60.

T. R. Browning and R. V. Ramasesh. 2007. A survey of activity network-based process models for managing product development projects. *Production and Operations Management* 16, 2, 217–240.

M. Broy, I. H. Krüger, and M. Meisinger. 2007. A formal model of services. *ACM Transactions on Software Engineering and Methodology* 16, 1, 1–40.

Valentine Casey and Ita Richardson. 2006. Uncovering the reality within virtual software teams. In *Proceedings of the International Workshop on Global Software Development for the Practitioner (GSD'06)*. ACM, New York, NY, 66–72. DOI:http://dx.doi.org/10.1145/1138506.1138523

Marcelo Cataldo, Matthew Bass, James D. Herbsleb, and Len Bass. 2007. On coordination mechanisms in global software development. In *Proceedings of the International Conference on Global Software Engineering (ICGSE'07)*. IEEE Computer Society, Washington, DC, 71–80. DOI:http://dx.doi.org/10.1109/ICGSE.2007.33

N. Celik, Hui Xi, Dong Xu, and Young-Jun Son. 2010. Simulation-based workforce assignment considering position in a social network. In *Proceedings of the 2010 Winter Simulation Conference (WSC)*. IEEE Computer Society, Washington, DC, 3228–3240. DOI:http://dx.doi.org/10.1109/WSC.2010.5679015

R. Chanin, M. Corrêa, P. Fernandes, A. Sales, R. Scheer, and A. F. Zorzo. 2006. Analytical modeling for operating system schedulers on NUMA systems. *Electronic Notes in Theoretical Computer Science (ENTCS)* 151, 3, 131–149.

G. Ciardo, R. L. Jones, III, A. S. Miner, and R. I. Siminiceanu. 2006. Logic and stochastic modeling with SMART. *Performance Evaluation* 63, 6, 578–608. DOI:http://dx.doi.org/10.1016/j.peva.2005.06.001

J. N. Cummings, J. A. Espinosa, and C. K. Pickering. 2009. Crossing spatial and temporal boundaries in globally distributed projects: A relational model of coordination delay. *Information Systems Research* 20, 3, 420–439.

R. M. Czekster, P. Fernandes, R. Prikladnicki, A. Sales, A. R. Santos, and T. Webber. 2011b. Follow-the-sun methodology in a stochastic modeling perspective. In *6th IEEE International Conference on Global Software Engineering (ICGSE'11): Methods and Tools for Project/Architecture/Risk Management in Globally Distributed Software Development Projects (PARIS)*. IEEE Computer Society, Washington, DC, 54–59. DOI:http://dx.doi.org/10.1109/ICGSE-W.2011.26

R. M. Czekster, P. Fernandes, A. Sales, D. Taschetto, and T. Webber. 2010b. Simulation of Markovian models using bootstrap method. In *Proceedings of the 2010 Summer Simulation Multiconference*. Society for Computer Simulation International, San Diego, CA, Ottawa, Canada, 564–569.

R. M. Czekster, P. Fernandes, A. Sales, and T. Webber. 2010a. Restructuring tensor products to enhance the numerical solution of structured Markov chains. In *Proceedings of the 6th International Conference on the Numerical Solution of Markov Chains (NSMC'10)*. NSMC, Williamsburg, VA, 36–39.

R. M. Czekster, P. Fernandes, and T. Webber. 2009. GTA express - A software package to handle Kronecker descriptors. In *Proceedings of the 6th International Conference on Quantitative Evaluation of SysTems (QEST'09)*. IEEE Press, Budapest, Hungary, 281–282.

R. M. Czekster, P. Fernandes, and T. Webber. 2011a. Efficient vector-descriptor product exploiting time-memory trade-offs. *SIGMETRICS Performance Evaluation Review* 39, 3, 2–9. DOI:http://dx.doi.org/10.1145/2160803.2160805

G. A. Dafoulas, K. Swigger, R. Brazile, F. N. Alpaslan, V. L. Cabrera, and F. C. Serce. 2009. Global teams: Futuristic models of collaborative work for today's software development industry. In *42nd Hawaii International Conference on System Sciences (HICSS'09)*. IEEE Computer Society, Washington, DC, 1–10. DOI:http://dx.doi.org/10.1109/HICSS.2009.231

Janez Demšar. 2006. Statistical comparisons of classifiers over multiple data sets. *The Journal of Machine Learning Research* 7, 1–30.

Christof Ebert. 2007. Optimizing supplier management in global software engineering. In *Proceedings of the International Conference on Global Software Engineering (ICGSE'07)*. IEEE Computer Society, Washington, DC, 177–185.

C. Ebert and P. De Neve. 2001. Surviving global software development. *IEEE Software* 18, 2, 62–69. DOI:http://dx.doi.org/10.1109/52.914748

A. G. Farina, P. Fernandes, and F. M. Oliveira. 2002. Representing software usage models with stochastic automata networks. In *Proceedings of the 14th International Conference on Software Engineering and Knowledge Engineering (SEKE'02)*. ACM, New York, NY, Ischia, Italy, 401–407.

P. Fernandes, L. Lopes, and S. Yeralan. 2013a. Symbolic solution of Kronecker-based structured Markovian models. In *IEEE 21st International Symposium on Modelling, Analysis and Simulation of Computer and Telecommunication Systems*. IEEE, San Francisco, CA, 409–413. DOI:http://dx.doi.org/10.1109/MASCOTS.2013.62

P. Fernandes, M. E. J. O'Kelly, C. T. Papadopoulos, and A. Sales. 2013b. Analysis of exponential reliable production lines using Kronecker descriptors. *International Journal of Production Research* 51, 14, 4240–4257. DOI:http://dx.doi.org/10.1080/00207543.2012.754550

P. Fernandes, A. Sales, A. R. Santos, and T. Webber. 2011. Performance evaluation of software development teams: A practical case study. *Electronic Notes in Theoretical Computer Science (ENTCS)* 275, 73–92.

P. Fernandes, J.-M. Vincent, and T. Webber. 2008. Perfect simulation of stochastic automata networks. In *Proceedings of 15th International Conference on Analytical and Stochastic Modelling Techniques and Applications (ASMTA'08),* Lecture Notes in Computer Science, Vol. 5055. Springer, Berlin, 249–263.

J. M. Haake, T. Hussein, B. Joop, S. Lukosch, D. Veiel, and J. Ziegler. 2010. Modeling and exploiting context for adaptive collaboration. *International Journal of Cooperative Information Systems (IJCIS)* 19, 1–2, 71–120.

J. D. Herbsleb and A. Mockus. 2003. An empirical study of speed and communication in globally distributed software development. *IEEE Transactions on Software Engineering* 29, 6, 481–494.

J. D. Herbsleb and D. Moitra. 2001. Global software development. *IEEE Software* 18, 2, 16–20.

J. D. Herbsleb, D. J. Paulish, and M. Bass. 2005. Global software development at Siemens: Experience from nine projects. In *Proceedings of the 27th International Conference on Software Engineering (ICSE'05)*. ACM, New York, NY, St. Louis, MO, 524–533.

J. Hillston. 1996. *A Compositional Approach to Performance Modelling*. Cambridge University Press, New York, NY.

E. Hossain, M. A. Babar, and Hye young Paik. 2009. Using scrum in global software development: A systematic literature review. In *4th IEEE International Conference on Global Software Engineering (ICGSE'09)*. IEEE Computer Society, Washington, DC, 175–184. DOI:http://dx.doi.org/10.1109/ICGSE.2009.25

Dan X. Houston, Gerald T. Mackulak, and James S. Collofello. 2001. Stochastic simulation of risk factor potential effects for software development risk management. *Journal of Systems and Software* 59, 3, 247–257. DOI:http://dx.doi.org/10.1016/S0164-1212(01)00066-8

P. Jalote and G. Jain. 2004. Assigning tasks in a 24-hour software development model. In *11th Asia-Pacific Software Engineering Conference.* IEEE Computer Society, Washington, DC, 309–315. DOI:http://dx.doi.org/10.1109/APSEC.2004.33

M. I. Kellner, R. J. Madachy, and D. M. Raffo. 1999. Software process simulation modeling: Why? What? How? *The Journal of Systems & Software* 46, 2–3, 91–105.

A. Lamersdorf, J. Munch, A. Fernandez del Viso Torre, and C. R. Rebate Sanchez. 2011a. A risk-driven model for work allocation in global software development projects. In *Proceedings of the International Conference on Global Software Engineering (ICGSE'11)*. IEEE Computer Society, Helsinki, Finland, 15–24.

A. Lamersdorf, J. Munch, A. Fernandez del Viso Torre, and C. R. Rebate Sanchez. 2011b. How do distribution and time zones affect software development? A case study on communication. In *Proceedings of the International Conference on Global Software Engineering (ICGSE'11)*. IEEE Computer Society, Helsinki, Finland.

P. Laurent, P. Mader, J. Cleland-Huang, and A. Steele. 2010. A taxonomy and visual notation for modeling globally distributed requirements engineering projects. In *5th IEEE International Conference on Global Software Engineering (ICGSE'10)*. IEEE Computer Society, Washington, DC, 35–44. DOI:http://dx.doi.org/10.1109/ICGSE.2010.55

R. E. Levitt, J. Thomsen, T. R. Christiansen, J. C. Kunz, Y. Jin, and C. Nass. 1999. Simulating project work processes and organizations: Toward a micro-contingency theory of organizational design. *Management Science* 45, 11, 1479–1495.

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

E. A. Matusse, E. H. M. Huzita, and T. F. C. Tait. 2012. Metrics and indicators to assist in the distribution of process steps for distributed software development: A systematic review. In *XXXVIII Conferencia Latinoamericana En Informatica (CLEI'12)*. IEEE Computer Society, Washington, DC, 1–10. DOI:http://dx.doi.org/10.1109/CLEI.2012.6427205

A. Mockus. 2009. Succession: Measuring transfer of code and developer productivity. In *Proceedings of the 31st International Conference on Software Engineering (ICSE'09)*. IEEE Computer Society, Washington, DC, 67–77.

A. Mockus and J. D. Herbsleb. 2002. Expertise browser: A quantitative approach to identifying expertise. In *Proceedings of the 24th International Conference on Software Engineering (ICSE'02)*. ACM, Orlando, FL, 503–512.

B. Moser, F. Kimura, and H. Suzuki. 1998. Simulation of distributed product development with diverse coordination behavior. In *Proceedings of the 31st International Seminar on Manufacturing Systems*. CIRP, Paris, France, Berkeley, California.

B. Moser, K. Mori, H. Suzuki, and F. Kimmura. 1997. Global product development based on activity models with coordination distance features. In *Proceedings of the 29th International Seminar on Manufacturing Systems*. CIRP, Paris, France, Osaka, Japan, 161–166.

F. Padberg. 2002. A discrete simulation model for assessing software project scheduling policies. *Software Process: Improvement and Practice* 7, 3–4, 127–139.

Kai Petersen, Robert Feldt, Shahid Mujtaba, and Michael Mattsson. 2008. Systematic mapping studies in software engineering. In *Proceedings of the 12th International Conference on Evaluation and Assessment in Software Engineering (EASE'08)*. British Computer Society, Swinton, UK, 68–77. http://dl.acm.org/citation.cfm?id=2227115.2227123

Bernard Philippe, Youcef Saad, and William J. Stewart. 1992. Numerical methods in Markov chain modeling. *Operations Research* 40, 6, 1156–1179.

B. Plateau. 1985. On the stochastic structure of parallelism and synchronization models for distributed algorithms. *ACM SIGMETRICS Performance Evaluation Review* 13, 2, 147–154.

T. Poikolainen and J. Paananen. 2007. Performance criteria in inter-organizational global software development projects. In *2nd IEEE International Conference on Global Software Engineering (ICGSE'07)*. IEEE Computer Society, Washington, DC, 60–70. DOI:http://dx.doi.org/10.1109/ICGSE.2007.35

D. Raffo and S. Setamanit. 2005. A simulation model for global software development project. In *Proceedings of the 6th International Workshop on Software Process Simulation and Modeling (ProSim'05)*. Fraunhofer IRB Verlag, Stuttgart, Germany, 1–7. Retrieved July 20, 2016 from http://www.sba.pdx.edu/faculty/davidr/draccess/WEB/publications/JOURNAL/ProSim?05-GSD.pdf.

N. Ramasubbu, M. Cataldo, R. K. Balan, and J. D. Herbsleb. 2011. Configuring global software teams: A multi-company analysis of project productivity, quality, and profits. In *33rd International Conference on Software Engineering (ICSE'11)*. ACM, New York, NY, 261–270. DOI:http://dx.doi.org/10.1145/1985793.1985830

A. Sales. 2012. SAN lite-solver: A user-friendly software tool to solve SAN models. In *Spring Simulation Multi-conference (SpringSim'12): SCS/ACM Theory of Modeling and Simulation: DEVS Integrative M&S Symposium (DEVS'12)*. SCS/ACM, Orlando, FL, 44:9–16.

R. Sangwan, N. Mullick, M. Bass, D. J. Paulish, and J. Kazmeier. 2006. *Global Software Development Handbook*. Auerbach Publications, New York, NY.

Alan R. Santos, Afonso Sales, and Paulo Fernandes. 2015. Using {SAN} formalism to evaluate follow-the-sun project scenarios. *Journal of Systems and Software* 100, 182–194. DOI:http://dx.doi.org/ 10.1016/j.jss.2014.10.046

S. Setamanit, W. Wakeland, and D. M. Raffo. 2006. Planning and improving global software development process using simulation. In *Proceedings of the 2006 International Workshop on Global Software Development for the Practitioner (GSD'06)*. ACM, New York, NY, Shanghai, China, 8–14.

S. Setamanit, W. Wakeland, and D. M. Raffo. 2007. Using simulation to evaluate global software development task allocation strategies: Research sections. *Software Process: Improvement and Practice* 12, 5, 491–503.

P. Sfetsos, L. Angelis, and I. Stamelos. 2006. Investigating the extreme programming system—An empirical study. *Empirical Software Engineering* 11, 2, 269–301.

R. P. Smith and S. D. Eppinger. 1997. Identifying controlling features of engineering design iteration. *Management Science* 43, 3, 276–293.

P. Sooraj and Pratap K. J. Mohapatra. 2008. Modeling the 24h software development process. *Strategic Outsourcing: An International Journal* 1, 2, 122–141. DOI:http://dx.doi.org/10.1108/17538290810897147

W. J. Stewart. 1994. *Introduction to the Numerical Solution of Markov Chains*. Princeton University Press, Princeton, NJ.

W. J. Stewart. 2009. *Probability, Markov Chains, Queues, and Simulation*. Princeton University Press, Princeton, NJ.

K. Swigger, F. C. Serce, F. N. Alpaslan, R. Brazile, G. Dafoulas, and V. Lopez. 2009. A comparison of team performance measures for global software development student teams. In *Proceedings of the 4th International Conference on Global Software Engineering (ICGSE'09)*. IEEE Computer Society, Los Alamitos, CA, 267–274.

S. Syed-Abdullah, M. Holcombe, and M. Gheorge. 2006. The impact of an agile methodology on the well being of development teams. *Empirical Software Engineering* 11, 143–167. Issue 1.

A. Taweel and P. Brereton. 2006. Modelling software development across time zones. *Information and Software Technology* 48, 1, 1–11.

A. Tripathy and S. D. Eppinger. 2011. Organizing global product development for complex engineered systems. *IEEE Transactions on Engineering Management* 58, 3, 510–529.

Roger Urdangarin, Paulo Fernandes, Alberto Avritzer, and Daniel Paulish. 2008. Experiences with agile practices in the global studio project. In *IEEE International Conference on Global Software Engineering (ICGSE'08)*. IEEE Computer Society, Washington, DC, 77–86.

G. H. Walton and J. H. Poore. 2000. Generating transition probabilities to support model-based software testing. *Software: Practice and Experience* 30, 10, 1095–1106.

James A. Whittaker and Michael G. Thomason. 1994. A Markov chain model for statistical software testing. *IEEE Transactions on Software Engineering* 20, 10, 812–824.

Received April 2015; revised April 2016; accepted June 2016