Pontifícia Universidade Católica do Rio Grande do Sul
Faculdade de Informática
Programa de Pós-Graduação em Ciência da Computação

Performance analysis of global software development teams using a structured stochastic modeling formalism

Ricardo M. Czekster, Paulo Fernandes, Afonso Sales, Thais Webber

Relatório Técnico N^o 061

Porto Alegre, Outubro de 2010

Abstract

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 be derived in order to represent a distributed software collaboration project. We present a model comprising a distributed reality and we show that it is possible, for instance, to determine the level of coordination required due to project requirements. We focus our attention to the level of interaction among project participants and its close relation with team's productivity. The models are parametrized for different sets of member where each member reacts according to the demands of a central team. We change the resource availability levels, team's expertise and support levels to see each metric impact on the overall project performance. Finally, we present our derived performance indices for all scenarios ending our analysis with a comprehensive discussion on how to extend the models to represent more intricate behaviors and communication patterns that are usually present in globally distributed software projects.

1 Introduction

Effective local interactions and teams domain knowledge within single-site and multi-site contexts play an important role in software development projects. It is an established fact that its effects are directly related to team's productivity. To reduce the need of constant communications and external support in globally distributed projects, research results suggest dividing a project in self contained units, loosely coupled, to maximize work periods [19, 28], taking into account teams expertise levels. Another issue that has brought a considerable amount of concern during the years is the team leader's availability. Its importance should not be neglected since it affects the whole project organization and it impacts the way projects evolve, hopefully within budget and time constraints due to deadlines. Resource availability have been the subject for many interesting researches in past years, however, there is a lack of quantitative analysis efforts to uncover their influence in several interaction scenarios for Global Software Development (GSD) projects.

The use of analytical modeling in software engineering contexts was successfully used in the past to provide quantitative performance measures [34, 33, 17, 7] 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 [24, 25]; (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 teams productivity [2, 30, 29, 23]; and (iii) modeling - identification of entities and relations considering spatial and temporal boundaries in distributed projects [12, 2].

Related works concerning stochastic models and simulation are developed towards to the specification of the dynamics of software projects [26], and the usage of analytical models to interpret team's productivity variability [2]. Considering the fact that the performance analysis of geographically dispersed teams is emerging [32, 4, 28, 21, 19], advances are still needed for the quantitative evaluation of such systems using stochastic modeling as a valuable tool to derive operational indices.

To represent a software project one can choose from several available stochastic formalisms such as Stochastic Petri Nets [1], Process Algebras [22] or Stochastic Automata Networks (SAN) [27, 10]. The present work will focus on SAN, since it has been successfully used to represent GSD projects [14]. SAN is a powerful modeling formalism that works with an underlined 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, 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*, where the transitions are labeled with probabilistic and timing information. A SAN model has a set of *events* which triggers changes of the state of one or more automata. Each event has an estimated duration, which indicates how often this event occurs.

Solving an analytical model numerically, one can obtain its *steady-state probabilities* [9, 18] and, hence, it is possible to extract measures of interest, *i.e.*, performance indices about the system under evaluation. Previous works in the context of software engineering developed SAN models for the analysis of the impact of external dependencies in solving teams local issues [14] showing that leader's availability and expertise have a considerable effect on team members' productivity. However, it is important to consider also the members' skills to deal with its own tasks, depending on the team size. Such quantitative scenario evaluations are crucial to develop techniques for setting teams in different locations deciding how each resource will behave within each context.

Both industry and academia have a special interest in modeling and predicting the behavior of software development processes, teams compositions and evaluation, *i.e.*, 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 sites, GSD brings more challenges than single-site development [4, 19]. Those are the main reasons why it is important to quantify project scenarios configurations and to use the gained information to enhance decision making and to avoid improper utilization of valuable resources.

This paper demonstrates the usefulness of analytical modeling applied to the investigation of interaction patterns in co-located and geographically distributed projects. We focus our attention on the team interactions as well as on the analysis of the impact of centralized control mechanisms, a problem that usually surfaces as a major source of communication difficulties in distributed projects. This communication metric is also related to teams geographic distance, different timezones, and central team's availability.

It is important to notice that the impact of coordination or cultural diversity are out of the scope of this paper. However, we are aware that such metrics are quite relevant in respect to GSD projects and further works may include such aspects in our current model of globally distributed project. Nevertheless, our co-located and globally distributed project models are experimented with various scenarios and the results are analyzed showing the trade-offs of choosing different team sizes and compositions.

Section 2 introduces the SAN formalism presenting a previously developed model [14] describing a co-located project. In contrast, Section 3 describes an original model of a globally distributed project and its mathematical solution. Section 4 presents the main contribution of our paper with the analysis of the obtained performance indices for the proposed models. Our final considerations (Section 5) discuss future works directed to the extension of models in order to capture advanced characteristics such as cultural issues and diverse communication problems.

2 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* [31], which is applied to several domains as bioinformatics, economics, chemistry and engineering, to name a few. Such models use simple primitives such as states and labeled transitions to represent system evolution and operational semantics. In the context of software engineering analytical models can be widely used to estimate costs, how much effort, how many resources, and how much time it will take to build a specific software-based system or product, or even to study the dynamics of software projects and teams productivity. Modeling software development teams, often geographically dispersed, is an important task since it can determine the success or failure of a project. Stochastic models [6] can focus, for example, on important factors such as communication and coordination issues among teams.

Stochastic Automata Networks (SAN) is a high-level structured formalism to represent structured Markov chains [27, 10]. There is a wide scope of SAN applications mainly focused on performance evaluation of parallel and distributed systems [8, 11, 3, 7, 17], that can profit of 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 parameterizing the events of a model, one can obtain the model numerical solution as the steady-state probabilities of being in each state. The numerical analysis of SAN models can be performed by dedicated software packages as *Performance Evaluation of Parallel Systems* (PEPS) [5, 9], or *GTAexpress* [16] which implement numerical solutions using advanced iterative [15] and simulation [13] techniques.

2.1 Single Site Team Model Automata and Events

This section presents a model for a development team where members are located in a single site context. This model was initially presented in a previous work [14] and it is presented here to introduce the SAN formalism, as well as serve as a comparison paradigm to the multi site (and globally distributed) team model proposed in Section 3.

As any SAN model, the single site team model is composed by several automata, each of them representing an individual, but not independent, entity. In our model, some automata describe the managing staff people, typically the project manager, but also 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 a leader supervision, *i.e.*, reacting to leader demands or requesting leader assistance. The automata called *Leader #j* represent each of the *j*-th leaders of the team. The automata called *Member #i* represent the N work members of the team ($i = 1 \dots N$).

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 time, or synchronizing events, which possibly change more than one automaton at the same time.

Automata Leader #j (Figure 1) have two different states: *Mg* (Management) and *Co* (Collaboration), enabling events for transitions between both states depending on automata *Member #i*. 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 each team member. Actually, event a_i represents the start and event s_i represents the end of communication between the leader and the *i*-th member.

Figure 2 generically depicts the automata describing a team member. Each automaton (*Member #i*) presents four states: *Wk* meaning the member working

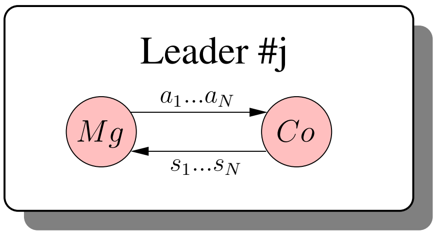


Figure 1: Leader entity (single-site context) [14]

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 *Member #i* automaton synchronize the change of state of the *i*-th 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 $(1 - \pi)$ of 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 expertise itself and capacity to rework in a given issue, *i.e.*, the time spent working (*Wk* state) until the member needs to collaborate (*Co* state) and the time spent reworking (*Rw* state) an unsolved issue.

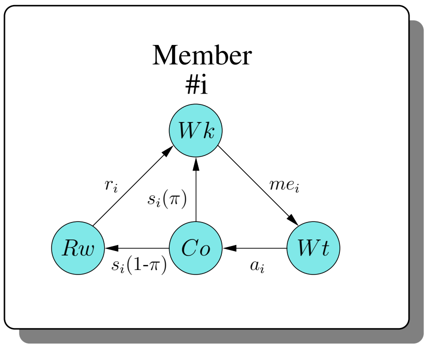


Figure 2: Developer entity (single-site context) [14]

The development team interaction pattern used for this single-site context model, *i.e.*, the relations between leader and members, is focused in synchronizing interactions in fixed periods of time. On this perspective, we consider a team composed of single members to a maximum size of N members attached to one leader[1] to collaborate (Figure 3). The development team behavior states that members work on their assigned tasks, performing local cooperation with other members, and interactions with the leader. In this abstraction, the leader has an 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.

2.2 Single Site Team Model Event Rates and Numerical Results

For simplification in this first model example, the members and issues to work are homogeneous, meaning every member has the same level of expertise and time spent in rework. According to estimations based on empirical experience, permanence times are assigned to every state in the model, *i.e.*, rates correspondent to frequencies at every connection among states. Table 1 shows the estimated rates for the events in the single-site model of Figure 3, assuming as reference an *eight-hour workday*. From these initial values assigned to each event, one can assume different combinations of rates in order to analyze more effectively the impact of them in the model dynamics considering various teams formations.

Therefore, combining the estimated event rates presented in Table 1, we assemble the eight possible scenarios [14] named as follows: ILA, ILB, IHA, IHB, ELA, ELB, EHA, and EHB. For instance, ILA scenario represents a team with *inexperienced* members (I), a leader that provides a *low* support (L) - *i.e.*, an inexperienced leader in solving issues demanded by members - but often *available* to cooperate with the members (A). The rest of the scenarios follows the same letter codification as presented in Table 1, *e.g.*, EHB represents a configuration with Expert members, High quality leader support, and a Busy leader. We choose to vary parameters of the three major aspects that our model captures, looking for results that promote a better understanding on the relations among those variables behaviors. It is important to keep in mind that event estimated rates are empirical values and those estimations are very dependable on the modeler's experience or available data from previous project observations.

Note that different inputs (event rates) can generate different outputs (performance indices or steady-state probabilities) even in this small scenario. For example, Figure 4 shows the results for three scenarios in the single-site development

[1]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.

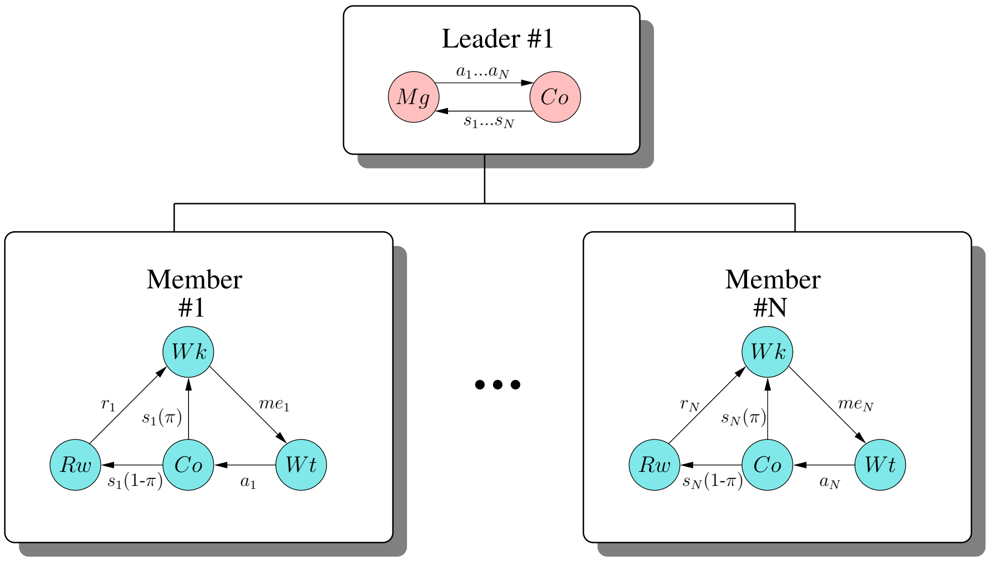


Figure 3: Example of a software development team in a single-site configuration

Table 1: Event estimated rates for the model example - Figure 3

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

context (Figure 3), where the impact on team members productivity is considered according to 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 points 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, independently of the level of support given by the leader (EHB and ELB scenarios). This result is emphasized in Figure 4 by the significant decrease of productivity of IHA in contrast to the less pronounced degradation of performance presented in ELB and EHB scenarios.

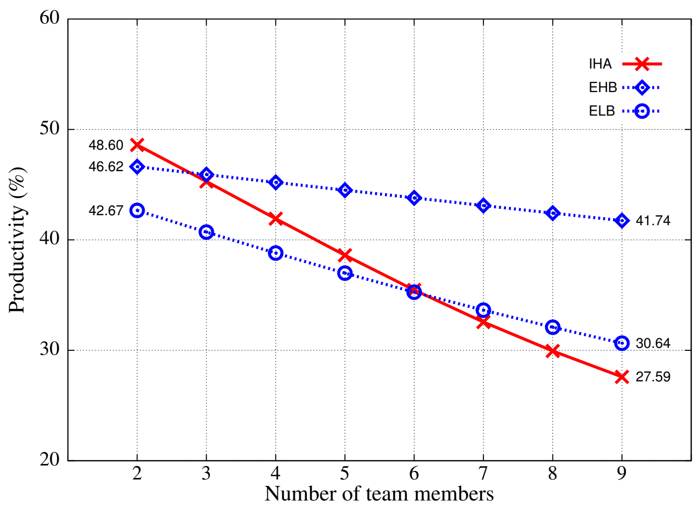


Figure 4: Impact on the productivity considering members' expertise and leader's availability

2.3 Extended Version of the Single Site Team Model Including a Off-site Coordination

The use of formal models and analytical evaluations can bring new ideas of how to compose teams and interaction patterns to deal with different team sizes and project demands. Recent researches parametrized the single site model and also extended it to a multi-site context [14], where a centralized external resource is needed to solve more complex questions, also forcing synchronizations among members and 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 5 shows a multi-site 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 team, becoming unavailable to collaborate with the members. In a similar way, state *Wr* was added for members automata, indicating 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 5, we represent the leader and members dependability of the central team by the dashed transitions and states (*Ex* and *Wr*).

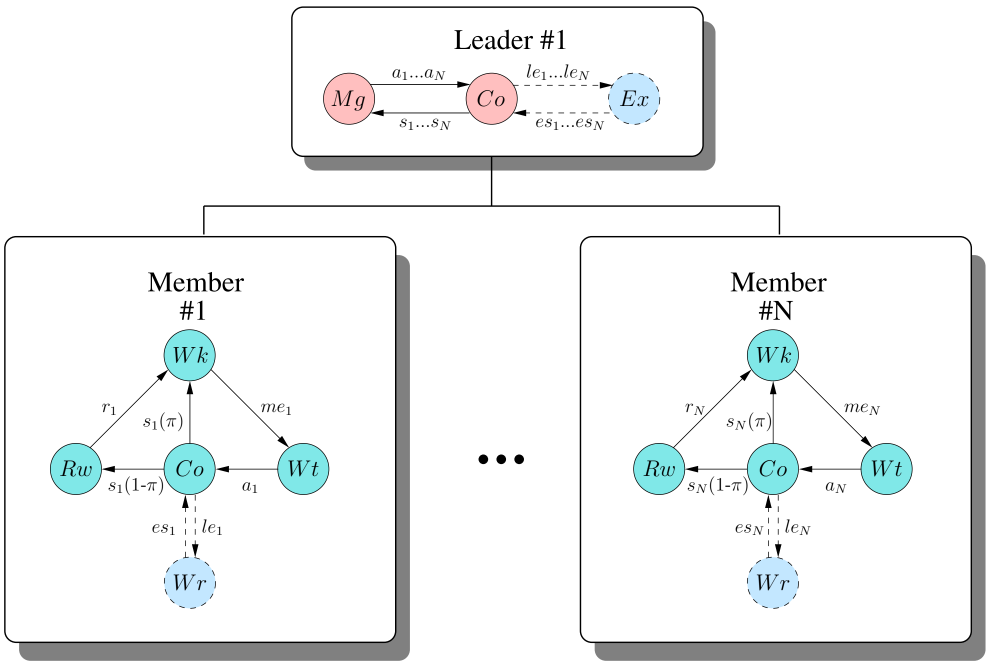


Figure 5: Example of a software development team with a off-site coordination

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, cultural and language diversities [20] represented by synchronizing event es. The other events remain the same from the single-site model for simplification purposes. Table 2 redefines the rates parametrization for this new example.

Table 2: Event estimated rates for multi-site model - Figure 5

Event	*Estimated rates for an eight-hour workday*
a	**(A)vailable:** leader cooperates with a member after *30 minutes*.
	(B)usy: leader presents low availability due to management duties, *i.e.*, the leader cooperates only *once a day*.
le	**(I)nexperienced:** leader demands cooperation with central team in average *four times a day*.
	(E)xpert: leader requires cooperation with central team in average *once a week*.
es	**(L)ow:** central team requires in average *one day* for responding issues demanded by leaders.
	(H)igh: central team requires in average *30 minutes* for responding issues demanded by leaders.
me	member demands cooperation in average *twice a day*.
r	member requires in average *two hours* to review/correct its tasks.
s	leader requires in average *one hour* for solving issues demanded by members.

Using the event estimated rates for off-site model presented in Table 2, eight possible scenarios were defined [14] as follows: AIL, AIH, AEL, AEH, BIL, BIH, BEL and BEH. These scenarios were parametrized with average values to the team expertise and experience, and level of support provided by the leader, which refers to the events me, r and s, respectively.

The indices obtained from these models bring some questions about the need of a 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 6 (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 he is inexperienced, since the leader provides high external support, compensating the lack of experience. This fact is corroborated for the numeric results for all team sizes. However, the variation of productivity is considerably greater for AIH scenario than BEH scenario, *i.e.*, the decrease of team members productivity is emphasized much more in AIH than BEH scenario.

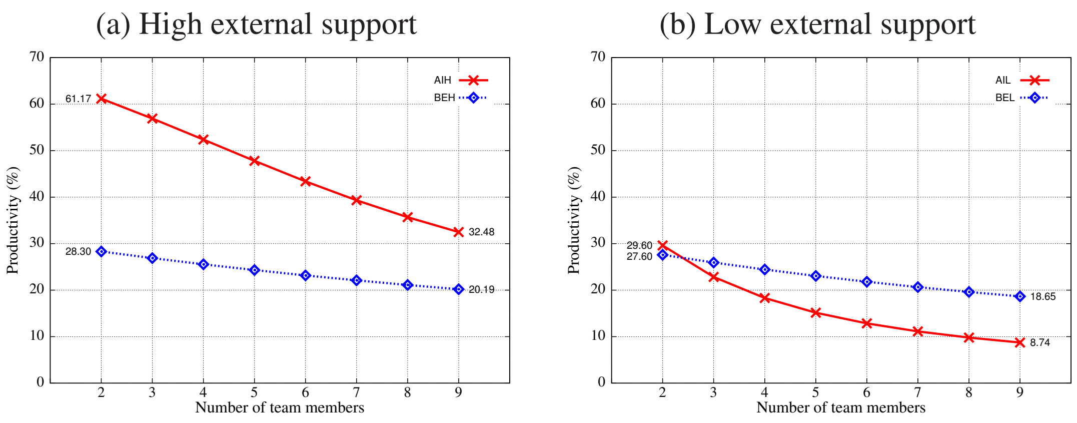


Figure 6: Team members' productivity on a high and low external support scenario

Additionally, these results make possible to answer the question *is it also better to have an available and inexperienced leader than a busy and expert one in low external support scenarios?* Observing the initial productivity values, *i.e.*, the results for very small teams (just two members) the answer is true. On the contrary, as team size increases, the team members productivity is better in overall having an expert leader, despite being busy. In this case, the leader experience improves the members productivity the AIL and BEL curves crossing of Figure 6 (b) indicates.

3 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. Note that the interactions are now dependent of 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.

3.1 Globally Distributed Team Model Automata and Events

Figure 7 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, *e.g.*, time-zones, other meetings, *etc.*). The second automaton (*Activities*) also has two states: M (central team performing *management* activities in general, according to the specific scenario modeled) and C (central team effectively cooperating with a participant).

We model the participants of a software development team with different states as follows: W state means the participant *working*, completing its tasks, or collaborating with other members; S state represents the participant *seeking for a specific solution*, information, documentation, sources of data, or even learning some technical issue by its own; C state means the participant collaborating with the central team due to solve technical questions. Figure 7 presents the stochastic automata network correspondent to 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 by its own, or preferably move to cooperate with the central team, returning to the working state after that. The central team 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.

3.2 Globally Distributed Team Model Event Rates

It is important to notice that events s and co in Figure 7 are fully synchronized, *i.e.*, as soon as the central team is available, a participant requesting a collaboration can start immediately. This abstraction is represented by a functional rate involving the verification of the central team availability each time a participant

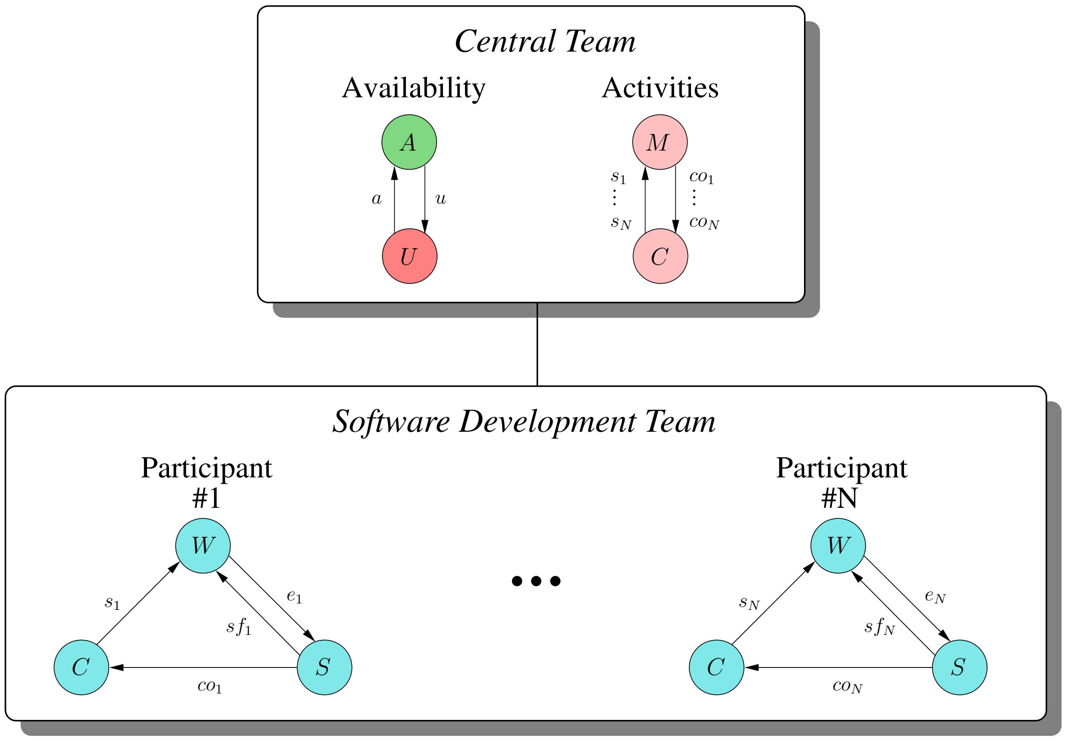


Figure 7: Global software development team model

wants to collaborate while seeking a solution. The collaboration state has an average fixed time in which central team and a participant remain in the state, as well as the given participant returns to the working state when finished the meeting/collaboration. Table 3 presents the events depicted in the model of Figure 7, 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 its associations with some estimated rates (Table 4). 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.

Using the event estimated rates for the model presented in Table 4, we consider eight possible scenarios combining the defined rates as follows: AIL, AIH, AEL, AEH, BIL, BIH, BEL, and BEH. For example, AEL scenario represents a team in which the central team is frequently Available to cooperate with the participants (7 of 8 hours); the participants are Experienced (working for 7 hours before seek-

Table 3: Description of the events in Figure 7

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 i-th participant with the central team.
e_i	**Participant's expertise:** this event represents the i-th participant starting to solve his/her issues without cooperating with the central team.
s_i	**External support:** this event represents the i-th participant resuming work after the end of the cooperation with the central team.
sf_i	**Solution found:** this events represent the i-th participant resuming work after founding a solution by him/herself.

ing for cooperation or own solution), and they receive Low quality support from central team (issues handled in 1 hour).

4 Teams Performance Analysis

In this paper, team 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 promptly manner, *i.e.*, tasks assignments and completion are encapsulated in the working state. The time spent on this state reflects the amount of useful work that has been produced as they influence the other states for each participant including central team states. The following figures and tables demonstrate the main results obtained from the global software development team model, varying the team size from two to ten participants, since specialists of the domain suggest to keep the number of participants within this limit [28].

Table 4: Event estimated rates for the model in Figure 7

Event	*Estimated rates for an eight-hour workday*
$a \mid u$	**(A)vailable:** central team is available to cooperate with participants in average during *420 minutes per* workday, *i.e.*, 7 of the 8 hours of a workday. Note that events a and u present complementary rates for a eight-hour workday, *e.g.*, 420 minutes actually collaborating (available), and 60 minutes unavailable (busy).
	(B)usy: central team presents low availability due to time spent in management duties, or very small time-zone overlap, *i.e.*, central team is available during only *60 minutes per* workday, *e.g.*, software development team in India and the central team in the United States.
e_i	**(I)nexperienced:** participant works in average *one hour* before stopping seeking for cooperation or finding its own solution.
	(E)xperienced: participant works for a long period of approximately *seven hours* without requiring any external support or start looking for an own solution.
s_i	**(L)ow:** central team takes in average *60 minutes* for responding participant issues. A low support is often characterized by communication issues such as language, available channels, time-zones and cultural diversity, but also central team expertise.
	(H)igh: central team requires in average *30 minutes* for responding participant issues.
co_i	since this event occurs immediately when the central team is available, a numerically high functional rate that verifies the automaton *Availability* state is used.
sf_i	this event assumes that the participant finds a solution by him/herself in *one hour*.

4.1 Analysis of Inexperienced Participants Scenarios

As expected the worst case scenario is configured by an inexperienced participants coping with a busy central team providing a low quality support when collaborating, *i.e.*, BIL scenario. Obviously, this scenario presents the worst productivity because they are often seeking for solutions themselves and collaborating with the central team than actually working in their tasks.

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

Another alternative to increase productivity is to increase the quality of the support, *i.e.*, change from the BIL to the BIH scenario. But the productivity increase is negligible (a 3% gain with two participants and 1% gain with ten participants. However, combining the two improvements, *i.e.*, 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). Figure 8 plots the productivity results for this four scenarios with inexperienced participants.

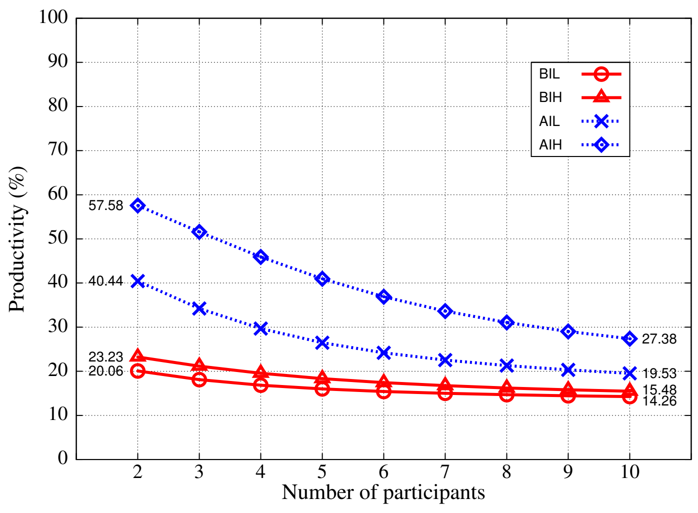


Figure 8: Productivity analysis for teams with inexperienced participants

Figure 9 depicts the productivity flow on the possible scenarios for the inexperienced teams. The availability and support quality parameters are changed to determine some aspects considering inexperienced participants in terms of communication. As seem in the results of Figure 8, the productivity increases in the scenarios with more available external resources (AIL and AIH scenarios) and, of course, the best productivity is achieved in the one with high quality external support to the participants, *i.e.*, AIH scenario.

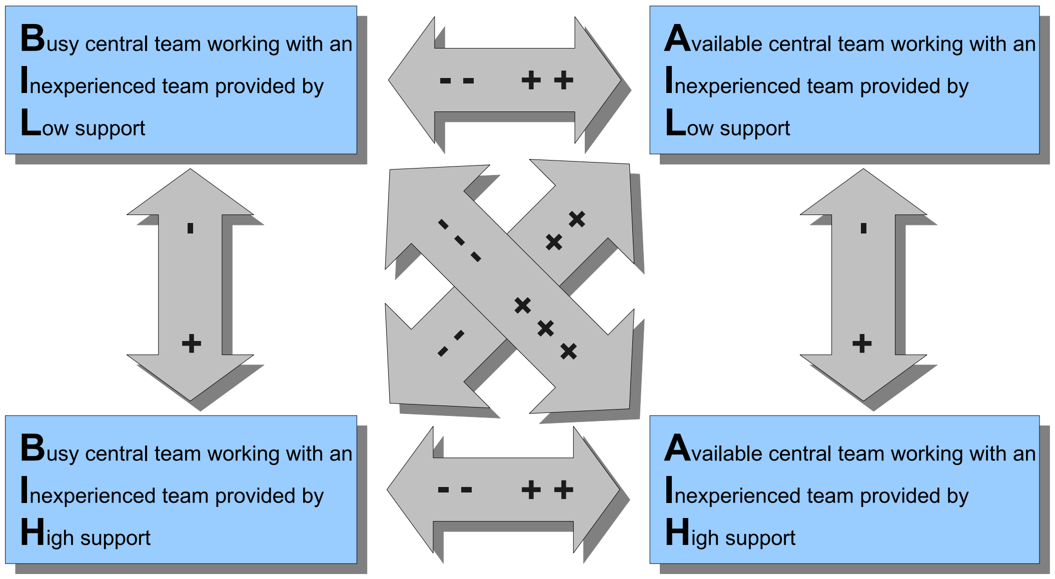

Busy central team working with an Inexperienced team provided by Low support
Available central team working with an Inexperienced team provided by Low support
Busy central team working with an Inexperienced team provided by High support
Available central team working with an Inexperienced team provided by High support
- -
+ +
-
+
- - -
+ +
+ + +
- -
-
+
- -
+ +

Figure 9: Teams with inexperienced participants productivity flow

4.2 Analysis of Experienced Participants Scenarios

Observing now more experienced participants, we analyze the impact of the support quality and availability provided by the central team on productivity. Figure 10 presents the numerical productivity achieved for the scenarios composed of experienced teams, *i.e.*, BEL, BEH, AEL, and AEH. The first clear observation is that the results with experienced participants are far more productive than the results for the inexperienced participants (Figure 8).

Analogously to the results of inexperienced participants scenarios, Figure 11 depicts the productivity flow for the scenarios with experienced participants. Note that for the scenarios where the central team provides a low quality support (BEL and AEL), the productivity does not change, regardless the central team's availability. On the contrary, for the BEH and AEH scenarios, *i.e.*, central team delivering high quality support, the increase in the central team's availability represents a considerable increase in productivity.

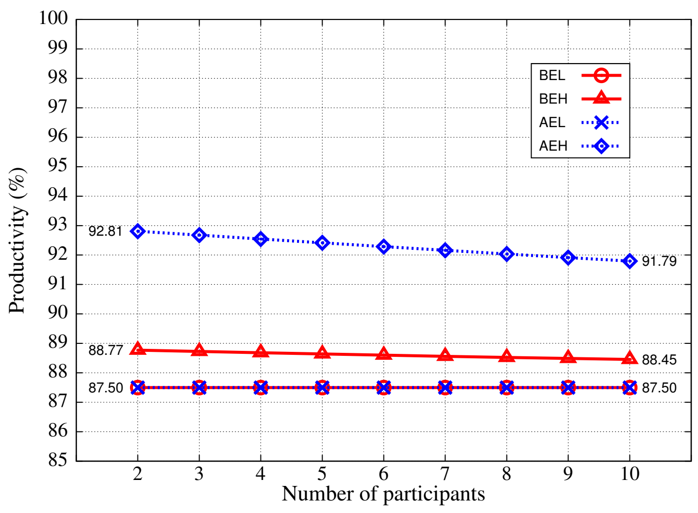
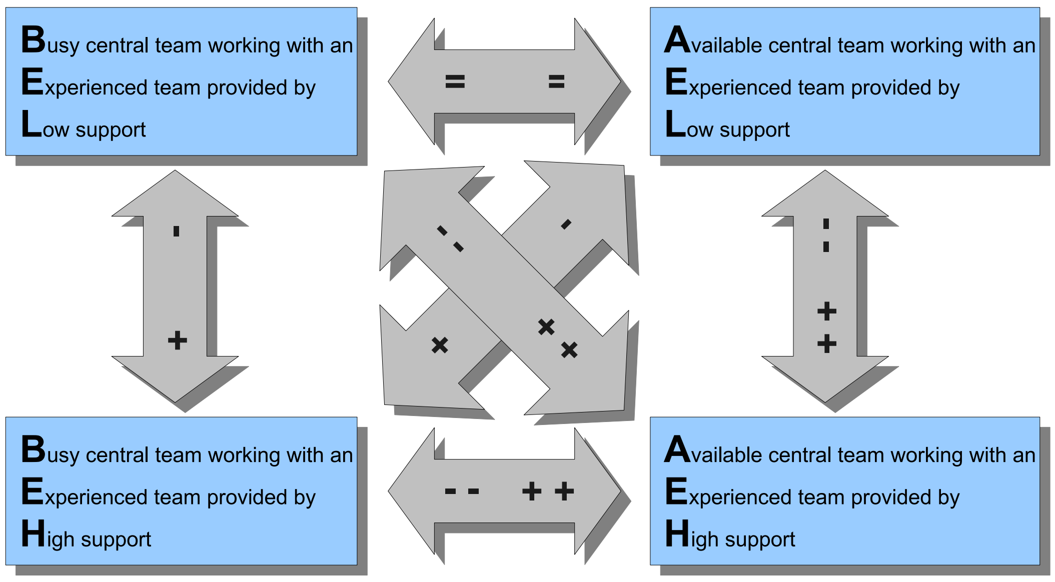


Figure 10: Productivity analysis for teams with experienced participants



Busy central team working with an Experienced team provided by Low support
Available central team working with an Experienced team provided by Low support
Busy central team working with an Experienced team provided by High support
Available central team working with an Experienced team provided by High support
=
=
-
+
- -
+ +
- -
+ +
-
-
+ +
- -
+ +

Figure 11: Teams with experienced participants productivity flow

4.3 Analysis of Very Low Quality Central Team Support

Let us consider a different scenario, where the central team is quite available (7 of 8 hours of a workday), the participants are experienced (7 hours of work before requesting support or seeking own solution), but a very low quality support, *i.e.*, the central team takes 2 hours to answer requests from participants. Such scenario is quite different from the previous ones, since in this configuration the central team does not help the participants, but the central team hinder the productivity of the participants. The numerical results for two and ten participants are, respectively, 79.55% and 83.86%, *i.e.*, increasing the number of participants the productivity increases. In fact, the experienced participants find solutions to their problems faster by themselves, than with collaboration of the central team. Nevertheless, the productivity for such scenario stays always below the 87.50% productivity of the analogous AEL scenario with central team one hour response time.

4.4 Analysis of Participants Experience Variation

The participant expertise is an important parameter to increase productivity. In Figure 12, we present a set of results for scenarios where an available central team provides a low support to the participants, varying the level of participants experience from 60 to 420 minutes *per* workday. In order to highlight the increase $(+)$ or decrease $(-)$ in terms of productivity, we have calculated a percentual 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 productivity of the maximum productivity. For example, considering AIL scenario which is the first x-axis value of in Figure 12, 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 Δ=51.70% = $\frac{40.44\%-19.53\%}{40.44\%}$. Our aim is to analyze the relation between participants experience and central team availability. In other words, Δ index quantifies the visual impression that as the participants experience grows (from 60 to 420 minutes of uninterrupted work) the productivity decrease is less relevant. Numerically, Δ decreases from 51.7% with inexperienced participants (AIL scenario) to 0.25% with experienced participants (AEL scenario).

Analyzing the same situation for scenarios with busy central team and high quality support (Figure 13), we observe similar results with a smaller difference between two and ten participants. It is also noticeable that the Δ index variation here from BIH to BEH scenarios is smaller than the variation in AIL to AEL scenarios (Figure 12).

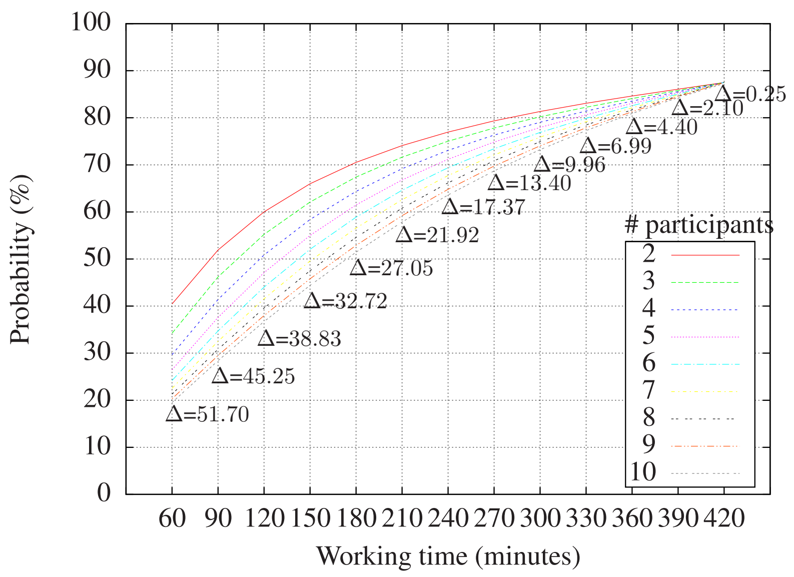


Figure 12: Team productivity analysis for A_L scenarios, varying the participant experience level

4.5 Hypotheses Analysis

Once the previous scenarios analyzed, we turn our attention to verify some hypotheses about the case study considering the numerical results obtained.

H1: *Teams with experienced participants do not need high availability of the central team that provides low quality support to keep a good productivity.*

Observing the results from Figure 10 this hypothesis is **clearly confirmed**, since AEL and BEL scenarios results were practically the same. In fact, experienced teams are sufficiently capable to solve their own issues in a quick manner, without the need of constant interactions with the central team.

H2: *According to productivity, teams with experienced participants scale better than team with inexperienced participants.*

This hypothesis is also **confirmed**, since the productivity for scenarios BEL, BEH, AEL and AEH (Figure 10) remains practically the same as the number of participants increase. However, the teams with experienced participants are not the only cases that scale well. Scenarios BIL and BIH (Figure 8) scale almost as

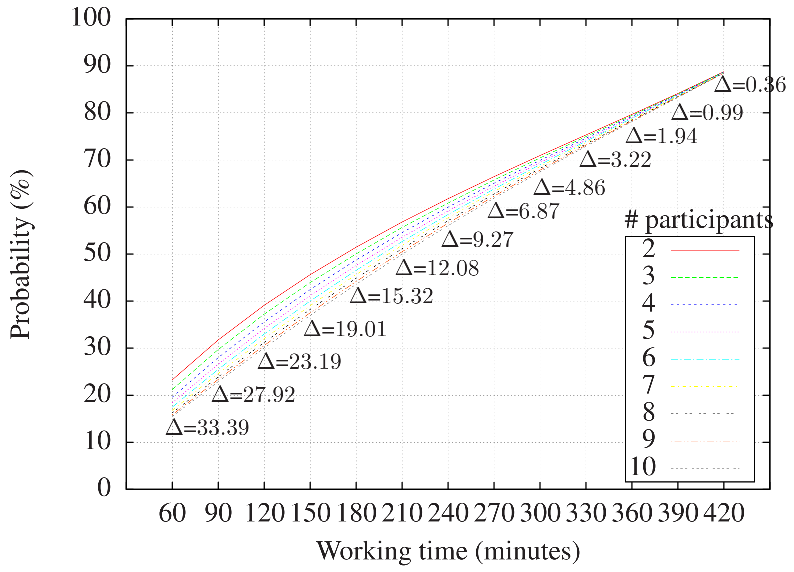


Figure 13: Team's productivity analysis for B_H scenarios, varying the participant experience level

good as the scenarios with experienced participants. Therefore, a better statement will be: "*According to productivity, only teams with available central teams and inexperienced participants do not scale well*".

H3: *Teams with experienced participants with a high quality external support provided by the central team have better productivity if the external staff is often available.*

With a strict analysis, this hypothesis is **confirmed**, since the best productivity results were obtained 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 10. These numbers probably indicate that a highly available central team is not really necessary, specially considering the usual high costs to make high quality central team often available.

H4: *Even teams with experienced participants need central team high quality support and availability to keep a high productivity.*

For teams with inexperienced participants the high quality support and availability is very important, as seem in Figure 8, where the AIH scenario practically double the productivity of BIL scenario. However, this hypothesis is just **barely confirmed** because the gains achieved by AEH scenario compared to BEL scenario represents a 5% increase in a already highly productive situation (Figure 10). Once again, in order to obtain these small gains, it is necessary to spent a considerable amount of effort and cost.

H5: *A central team highly available and providing high quality support can keep the productivity.*

Even though a good and available central team could increase the productivity, as seem in Figure 8, the key factor to productivity seems to be the experience of the participants. Therefore, this hypothesis is **not confirmed**.

5 Conclusion

In this paper we present two variations to model development teams in a global context and we analytically solve a set of scenarios to investigate some performance indices. For our analysis, it is important to remark that basically three parameters play a direct influence in team productivity: central team's availability, quality of external support and experience level of participants. The results pointed out valid insights to the analyzed examples, but we believe that this paper main contribution is the use of a formal stochastic modeling approach to describe and predict the performance of a globally distributed development project.

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 of all, it allows a very thorough reflexion 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 numeric conclusions are based on solid probabilistic analysis over a given set of input parameters.

Unfortunately, there resides also the major disadvantage of a formal approach, which is the high-level abstractions made 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 such estimation may be based only on the project manager skill, and 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, such problem is far from invalidating the applied formal approach because informal approaches also may suffer from this incorrect assumptions. Despite the fact that formal approaches are more prone to this kind of errors, in informal approaches these errors are usually harder to detect.

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

Therefore, in this paper we worked out a simple 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 the applicability to real GSD projects. Such improvements will probably require 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 indices to set up development teams, multi-site projects and resource allocation decisions.

Another possible future work is to enhance the formal model considering more subtile 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 tasks misconceptions or changes in requirements, and even concentrate the analysis on accomplished project tasks. The measure of team productivity can be also influenced by the developers morale, as they must constantly feel that the project is smoothly progressing [28]. 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 work consists in extend 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.

Once again, it is our believe that the main contribution of our paper is to propose an initial, an yet already valid exercise, in order to formally describe development projects and extract meaningful performance indices. Some insights on the analyzed models seems to be at the same time reasonable, since they confirm intuitive expectations, and also demystifying of some quick assumptions. The best result achieved with our stochastic modeling exercise was to express our performance predictions by numeric values in an area that is usually driven by qualitative opinions based only on previous experiences. Certainly, much research on formal modeling of GSD projects remains to be done, and we believe that SAN formalism may play an important role in doing so.

References

[1] M. Ajmone-Marsan, G. Balbo, G. Conte, S. Donatelli, and G. Franceschinis. *Modelling with Generalized Stochastic Petri Nets*. John Wiley & Sons, 1995.

[2] A. Avritzer and A. Lima. An Empirical Approach for the Assessment of Scheduling Risk in A Large Globally Distributed Industrial Software Project. In *Proceedings of the 4th International Conference on Global Software Engineering (ICGSE'09)*, pages 341–346, 2009.

[3] L. Baldo, L. Brenner, L. G. Fernandes, P. Fernandes, and A. Sales. Performance Models for Master/Slave Parallel Programs. *Electronic Notes In Theoretical Computer Science*, 128(4):101–121, April 2005.

[4] M. Bass, J. D. Herbsleb, and C. Lescher. Collaboration in global software projects at siemens: An experience report. In *Second IEEE International Conference on Global Software Engineering (ICGSE'07)*, pages 33–39, Aug. 2007.

[5] A. Benoit, L. Brenner, P. Fernandes, B. Plateau, and W. J. Stewart. The PEPS Software Tool. In *Computer Performance Evaluation (TOOLS 2003)*, volume 2794 of *LNCS*, pages 98–115, Berlin, Germany, September 2003. Springer-Verlag Heidelberg.

[6] M. Bernardo and J. Hillston. Formal Methods for Performance Evaluation, SFM 2007, Bertinoro, Italy, Advanced Lectures. In *SFM*, volume 4486 of *LNCS*. Springer, May/June 2007.

[7] C. Bertolini, A. G. Farina, P. Fernandes, and F. M. Oliveira. Test Case Generation Using Stochastic Automata Networks: Quantitative Analysis. In *Pro-*

ceedings of the 2nd International Conference on Software Engineering and Formal Methods (SEFM'04), pages 251–260, 2004.

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

[9] L. Brenner, P. Fernandes, B. Plateau, and I. Sbeity. PEPS2007 - Stochastic Automata Networks Software Tool. In *Proceedings of the 4th International Conference on Quantitative Evaluation of SysTems (QEST 2007)*, pages 163–164, Edinburgh, UK, September 2007. IEEE Press.

[10] L. Brenner, P. Fernandes, and A. Sales. 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, February 2005.

[11] R. Chanin, M. Corrêa, P. Fernandes, A. Sales, R. Scheer, and A. F. Zorzo. Analytical Modeling for Operating System Schedulers on NUMA Systems. *Electronic Notes in Theoretical Computer Science (ENTCS)*, 151(3):131–149, June 2006.

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

[13] R. M. Czekster, P. Fernandes, A. Sales, D. Taschetto, and T. Webber. Simulation of Markovian models using Bootstrap method. In *Proceedings of the 2010 Summer Simulation Multiconference*, pages 564–569, July 2010.

[14] R. M. Czekster, P. Fernandes, A. Sales, and T. Webber. Analytical Modeling of Software Development Teams in Globally Distributed Projects. In *Proceedings of the International Conference on Global Software Engineering (ICGSE'10): Methods and Tools for Project/Architecture/Risk Management in Globally Distributed Software Development Projects (PARIS'10)*, pages 1–10. IEEE Computer Society, August 2010.

[15] R. M. Czekster, P. Fernandes, A. Sales, and T. Webber. 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)*, pages 1–4, September 2010.

[16] R. M. Czekster, P. Fernandes, and T. Webber. GTA express - A Software Package to Handle Kronecker Descriptors. In *Proceedings of the 6th International Conference on Quantitative Evaluation of SysTems (QEST 2009)*, pages 281–282. IEEE Press, September 2009.

[17] A. G. Farina, P. Fernandes, and F. M. Oliveira. Representing software usage models with Stochastic Automata Networks. In *Proceedings of the 14th International Conference on Software Engineering and Knowledge Engineering (SEKE'02)*, pages 401–407, 2002.

[18] P. Fernandes, J.-M. Vincent, and T. Webber. Perfect Simulation of Stochastic Automata Networks. In K. Al-Begain, A. Heindl, and M. Telek, editors, *Proceedings of 15th International Conference on Analytical and Stochastic Modelling Techniques and Applications (ASMTA '08)*, volume 5055 of *Lecture Notes in Computer Science*, pages 249–263. Springer-Verlag, June 2008.

[19] J. D. Herbsleb and A. Mockus. An empirical study of speed and communication in globally distributed software development. *IEEE Trans. Softw. Eng.*, 29(6):481–494, 2003.

[20] J. D. Herbsleb and D. Moitra. Global software development. *IEEE software*, pages 16–20, 2001.

[21] J. D. Herbsleb, D. J. Paulish, and M. Bass. Global software development at siemens: experience from nine projects. In *Proceedings of the 27th International Conference on Software Engineering (ICSE'05)*, pages 524–533, 2005.

[22] J. Hillston. *A compositional approach to performance modelling*. Cambridge University Press, New York, USA, 1996.

[23] M. I. Kellner, R. J. Madachy, and D. M. Raffo. Software process simulation modeling: why? what? how? *The Journal of Systems & Software*, 46(2-3):91–105, 1999.

[24] A. Mockus. Succession: Measuring transfer of code and developer productivity. In *Proceedings of the 31st International Conference on Software Engineering (ICSE'09)*, pages 67–77, 2009.

[25] A. Mockus and J. D. Herbsleb. Expertise browser: a quantitative approach to identifying expertise. In *Proceedings of the 24th International Conference on Software Engineering*, pages 503–512, 2002.

[26] F. Padberg. A Discrete Simulation Model for Assessing Software Project Scheduling Policies. *Software Process: Improvement and Practice*, 7(3-4):127–139, 2002.

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

[28] R. Sangwan, N. Mullick, M. Bass, D. J. Paulish, and J. Kazmeier. *Global Software Development Handbook*. CRC Press, 2006.

[29] S. Setamanit, W. Wakeland, and D. M. Raffo. 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)*, pages 8–14, 2006.

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

[31] W. J. Stewart. *Probability, Markov Chains, Queues, and Simulation*. Princeton University Press, USA, 2009.

[32] K. Swigger, F. C. Serce, F. N. Alpaslan, R. Brazile, G. Dafoulas, and V. Lopez. 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)*, pages 267–274, 2009.

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

[34] J. A. Whittaker and M. G. Thomason. A markov chain model for statistical software testing. *IEEE Transactions on Software Engineering*, 20(10):812–824, 1994.