PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO GRANDE DO SUL
FACULDADE DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

STOCHASTIC MODELING OF GLOBAL SOFTWARE DEVELOPMENT TEAMS

ALAN RICARDO DOS SANTOS

Dissertação apresentada como requisito à obtenção do grau de Mestre em Ciência da Computação na Pontifícia Universidade Católica do Rio Grande do Sul.

Advisor: Dr. Paulo Henrique Lemelle Fernandes
Co-Advisor: Dr. Afonso Henrique Corrêa de Sales

Porto Alegre
2012

S237s Santos, Alan Ricardo dos
Stochastic modeling of global software development teams / Alan Ricardo dos Santos. – Porto Alegre, 2012.
80 f.

Diss. (Mestrado) – Fac. de Informática, PUCRS.
Orientador: Prof. Dr. Paulo Henrique Lemelle Fernandes.
Co-orientador: Prof. Dr. Afonso Henrique Corrêa de Sales.

1. Informática. 2. Engenharia de Software. 3. Redes de Autômatos Estocásticos. 4. Simulação e Modelagem em Computadores. I. Fernandes, Paulo Henrique Lemelle. II. Sales, Afonso Henrique Corrêa de.
III. Título

CDD 005.1

Ficha Catalográfica elaborada pelo
Setor de Tratamento da Informação da BC-PUCRS

Pontifícia Universidade Católica do Rio Grande do Sul
FACULDADE DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

TERMO DE APRESENTAÇÃO DE DISSERTAÇÃO DE MESTRADO

Dissertação intitulada *"Stochastic Modeling of Global Software Development Teams"*, apresentada por Alan Ricardo dos Santos como parte dos requisitos para obtenção do grau de Mestre em Ciência da Computação, Processamento Paralelo e Distribuído, aprovada em 31/08/2012 pela Comissão Examinadora:

[illegible]

Prof. Dr. Paulo Henrique Lemelle Fernandes – Orientador PPGCC/PUCRS

[illegible]

Prof. Dr. Afonso Henrique Corrêa de Sales – Co-orientador FACIN/PUCRS

[illegible]

Prof. Dr. Rafael Prikladnicki – PPGCC/PUCRS

[illegible]

Prof. Dr. Erran Carmel - *American University, Washington DC, USA*

Homologada em 03/10/2012, conforme Ata No. 21 pela Comissão Coordenadora.

[illegible]

Prof. Dr. Paulo Henrique Lemelle Fernandes
Coordenador.

Para Sérgio e Maria Amália.

AGRADECIMENTOS

Agradeço o orientador Dr. Paulo Fernandes pelo incentivo, apoio e suporte durante o mestrado. Foi um período de aprendizagem constante e gratificante. Sou também grato pela co-orientação dada pelo Dr. Afonso Sales pela dedicação, compreensão e ajuda durante todas as etapas do meu mestrado e também em vários momentos difíceis. Meu sincero obrigado por aceitar o convite para compor a banca de defesa de mestrado aos professores e pesquisadores Dr. Rafael Prikladnicki e Dr. Erwan Carmel. Agradeço o apoio financeiro obtido durante o período, fornecido por Dell Computadores para a bolsa taxas através do programa PDTI. Aos funcionários do PPGCC/PUCRS, particularmente ao Régis Escobal, agradeço o apoio e a paciência durante toda a duração do mestrado. Agradeço também aos amigos que fiz durante o mestrado, principalmente Thais Webber, Ricardo Czekster e Silvio Gomez, pelo conhecimento compartilhado e pela amizade. Agradeço aos meus pais, Sérgio e Maria Amália, obrigado do fundo do coração por sempre acreditar e confiar no meu potencial, eu dedico este trabalho a vocês, pelo empenho constante e por ter me ensinado os valores que tenho hoje. Agradeço também a minha irmã Anne pelo apoio durante o curso de mestrado. Agradeço especialmente a minha esposa Cláudia pela paciência e por tudo que passamos juntos neste período de crescimento. Sem a sua ajuda e apoio, eu não teria conseguido.

MODELAGEM ESTOCÁSTICA DE TIMES DE DESENVOLVIMENTO DE SOFTWARE GLOBAL

RESUMO

Avaliação de desempenho de projetos é um aspecto importante em desenvolvimento de software Distribuído. Empresas e instituições podem obter benefícios através da utilização de análise de performance em times trabalhando em diferentes locais. Este trabalho tem como objetivo apresentar uma definição de modelagem estocástica para projetos Follow-The-Sun (FTS) em diferentes aspectos como tempo, qualidade e custo. Exemplos de uso do modelo são apresentados em conjunto com os resultados de avaliação dos mesmos.

Palavras-chave: Desenvolvimento de Software Distribuído; Simulação de Projetos; Avaliação de Performance; Follow-the-Sun; Modelagem Estocástica.

STOCHASTIC MODELING OF GLOBAL SOFTWARE DEVELOPMENT TEAMS

ABSTRACT

Projects performance evaluation is an important aspect of global software development. Companies and institutions can obtain benefits by the use of performance evaluation of teams working in different sites. The objective of this work is to discuss a stochastic model definition to performance evaluation of Follow-The-Sun (FTS) projects aspects such as time, quality and cost. Example issues that can be addressed using this FTS model are provided with performance evaluation results.

Keywords: Global Software Development; Projects Simulation; Performance Evaluation; Follow-the-Sun; Stochastic Modeling.

LIST OF FIGURES

2.1 Project View 32
2.2 Site View 33
3.1 Research Design 35
3.2 The SAN model that represents Project *ALPHA* 37
3.3 Follow-The-Sun 44
3.4 SAN model 45
4.1 Clock Memory Automaton 47
4.2 Team Site *i* Task Automata 49
4.3 Team Site *i* 51
4.4 Project Memory Automaton 52
4.5 Full Model View 52
5.1 Sample One 58
5.2 Sample Two 62
5.3 Sample Three 66

LIST OF TABLES

3.1 Research Strings 36
3.2 Estimated event rates for an eight-hour workday 38
3.3 Project Estimated Effort $\times$ Actual Hours 39
3.4 Project's impediments 39
3.5 Steady-State probabilities of the proposed model 40
3.6 Project working hours obtained from the proposed model 40
3.7 Project impediment hours obtained from the proposed model 41
4.1 Clock Memory Automaton States 48
4.2 Description of *Clock Memory* events 48
4.3 Site *i* States 49
4.4 Description of *Site i* events 50
4.5 Team Site States 51
4.6 Description of *Teams Site i* events 51
4.7 Description of *Project Memory* events 53
5.1 Sample One Project Estimates 58
5.2 Sample One Parameters 59
5.3 Sample One States Probabilities 61
5.4 Sample Two Project Estimates 62
5.5 Sample Two Parameters 63
5.6 Sample Two States Probabilities 65
5.7 Sample Three Project Estimates 66
5.8 Sample Three Parameters 67
5.9 Sample Three States Probabilities 69

LIST OF ABBREVIATIONS

GSD	*Global Software Development*
FTS	*Follow-The-Sun*
SAN	*Stochastic Automata Network*

SUMMARY

1. Introduction 23
   - 1.1 Research Problem 24
     - 1.1.1 Research Question 24
   - 1.2 Research Goal 24
   - 1.3 Research Justification 25

2. Related Work 27
   - 2.1 *Follow-The-Sun* (FTS) 27
   - 2.2 Stochastic Modeling 28
   - 2.3 Stochastic Automata Networks 28
   - 2.4 Simulation 29
   - 2.5 FTS Performance Evaluation 31
     - 2.5.1 Project View 31
     - 2.5.2 Site View 32

3. Methodology 35
   - 3.1 Research Strategy 35
   - 3.2 Research Design 35
   - 3.3 Literature Review 35
   - 3.4 Practical Case 36
   - 3.5 Academic experiment 41
   - 3.6 Modeling 42
   - 3.7 Final Model 45

4. SAN Model 47
   - 4.1 Automata Definitions 47
   - 4.2 Model Options and Constraints 54

5. Performance Evaluation 57
   - 5.1 Running The Model 57
     - 5.1.1 Samples Settings 57
     - 5.1.2 Sample One 57
     - 5.1.3 Sample Two 61
     - 5.1.4 Sample Three 65

6. Conclusions 71
6.1 Contributions . . . 71
6.2 Research Limitations . . . 72
6.3 Future Work . . . 73

Bibliography 75

1. Introduction

Nowadays companies have been working to re-structure their IT departments extending their operations using software development centers geographically distributed in different time zones. In this context, software engineering has an area named as *GSD - Global Software Development*. For Sarker and Sahay [57]: *"GSD refers to software development geographically, remotely or globally distributed, with the goal of rationalize the process of product development"*. There are challenges on this research field such as different time zones, different cultures, different levels of experience and different technical background [9, 23, 63].

GSD has recently become an active research area [49]. Studies on GSD field started by 1990 when global software development started to become a competitive strategy [50].

There are some domains such as health care and air traffic control working 24h a day [63]. In this sense, on software development area, the benefits of working from the begin to the end of a day can be obtained through work handoff between time zones, in order to eliminate the necessity of work outside regular hours [63].

Follow-The-Sun (FTS) is a type of global software development that aims the use of a 24h workday and focus on projects that search for velocity by the reduction of project life cycle (*time-to-market*) [10, 11] and share many challenges and issues of global software development such as coordination, cultural factors and communication issues [9, 11, 22].

Theoretically, with the use of 24-h software development, the software development cycle time can be significantly reduced [16]. This kind of development may become a valid choice for software companies and can be applied in numerous projects [34, 61, 63].

Nguyen *et al.* [41] explored the effect of distance on communication and task completion time and used social network analysis to obtain insights about the collaboration on distributed teams, *"Global software development promises many benefits such as decreased development cost, access to a larger skill pool, proximity to customer, or twenty-four hour development by following the sun"*. FTS challenges and issues can be mapped through the use of analytical models for analysis of projects behaviors in order to facilitate companies and institutions decision-making process, providing a better understanding of possible issues that can occur in geographically distributed projects.

Related works for stochastic modeling and simulations are developed through the dynamic specification of software projects [13, 44], as the usage analytical modeling for variability performance analysis of software development teams [2, 15]. *SAN - Stochastic Automata Networks* [7, 47] is a formalism based on Markov Chains [62] that provides an abstraction of the model description.

New methodologies are needed to evaluate and simulate projects performance scenarios, considering cost, time and quality. This research aims to apply SAN on FTS software projects. The utilization of Follow-The-Sun strategy for the entire project life cycle can be complex and maybe not feasible [11], however there is research room to simulate a project life cycle in FTS. In this work, the proposed SAN model focus on the project life cycle performance evaluation for different sites.

Under Software Engineering area there are several researches related to GSD [9, 11, 13, 22, 23, 49, 63], and there is space to research the application of stochastic modeling in Software Engineering. The usage of stochastic modeling for software engineering has been applied in few studies, such as [15] which started a modeling effort for GSD teams and continuing this research effort, it was performed a new study to compare performance evaluation of model results against a real project [26]. In this sense, Czekster *et al.* executed an initial essay to apply stochastic modeling to measure *Follow-The-Sun* teams performance [16].

This research focus on FTS which is one type of Global Software Development identifying project model requirements and FTS factors available in the literature. It review simulation, quantitative studies and models characteristics that can be used to understand FTS scenarios. Finally, it proposes a performance evaluation model to explore Follow-The-Sun aspects. Conclusions point out numerical results as well as future works thoroughly for new models.

1.1 Research Problem

Management of Follow-The-Sun projects involves a high degree of uncertainty, and this uncertainty generate risks in terms of cost, quality and project time. To help projects risk mitigation it is possible to use simulation techniques, modeling values distribution within FTS development cycle. There are different methods to evaluate adherence of a given probability model. However to do this it is necessary to exists series of similar operations. When there is no history, how to discover model probabilities ?

1.1.1 Research Question

The issue of this research is to understand how to qualify the decision-making process on *follow-the-sun* projects, evaluating factors such as cost, time and quality, because FTS is a research area with aspects to be explored.

- **Question 1**: How Stochastic Automata Networks can help FTS projects decision-making ?

This question is related to how the use of SAN performance evaluation can help companies and institutions on Follow-The-Sun projects decision-making. This study has no hypothesis defined due its exploratory approach.

1.2 Research Goal

To propose a stochastic model and performance evaluation for Follow-The-Sun projects. In this sense, were defined the following specific goals:

- To perform a study of Analytical Modeling and Performance Evaluation for a better understanding of the functioning and application of those techniques on software development projects;

- To identify Follow-The-Sun aspects that can be used on the analytical model simulation;
- To create a stochastic model to verify through performance evaluation factors that can influence FTS teams performance results;

1.3 Research Justification

There are several practical implications on the coordination of FTS projects, specially on planning and execution project phases. This work explore FTS and stochastic modeling performance evaluation in order to enhance the decision making process before projects start.

According to Ramasubbu *et al.* [53] temporal dispersion reduces the possibilities of communications attributes for design activities *"Temporal dispersion refers to the time zone differences among project members"*. However temporal dispersion may allow distributed groups to accelerate the completion of development tasks using approaches such as *"Follow-The-Sun"*.

The use of Follow-The-Sun as a strategy for software development is not a common practice. It is difficult to find industry cases of software development teams using FTS approach. However, according to Nguyen *et al.* [41] *"in contrast to work in industry commercial environments, open source projects do not seem to have problems with distributed communication, collaboration and development following the sun"*.

Although there are FTS related researches [10,18,28,29,64] to cite a few, there is an opportunity to research and to simulate Follow-The-Sun software development projects. Simulation and numerical analysis can help to identify project gaps and project opportunities before they are started. This research aims to explore stochastic modeling and performance evaluation of teams geographically distributed, focusing on Follow-The-Sun.

2. Related Work

This section provides a literature review identifying existing and related researches methods and approaches.

2.1 *Follow-The-Sun* (FTS)

Innovation has been moving companies and institutions to create new strategies and techniques for software development. Time zone differences, geographical diversity and others aspects actually can help companies to improve their business. Those techniques and strategies can move companies to use an entire day of work using a Follow-The-Sun approach.

There are different definitions about *Follow-The-Sun* available in literature [10, 11, 18, 28, 29, 64]. Follow-The-Sun is a type of complex software development approach. In fact, FTS can be defined as software development activities being handoff from one site to the next on a daily basis [28, 64], where it is possible to work on software development 24 hours a day considering time zone differences between teams. Each team work on a time zone and in the end of each day tasks are handoff to the next team that will continue working on tasks started by previous team.

Treinen and Miller-Frost [64] presented time zone difference as an advantage for teams distribution in order to create a 24h development environment.

Gupta *et al.* [29] introduced the concept of continuous development, considering the aspect of knowledge transfer, *e.g.*, between development and testing teams. In this sense, tasks are transitioned between distributed teams in the end of each work day. However Carmel *et al.* [10] proposed FTS development as a type of global knowledge *workflow* that can reduce projects duration, depending on factors such as handoff efficiency, effective communication and coordination.

A FTS experience was published on 1997 [9] where IBM decided to develop a project applying FTS, to achieve this goal they used 5 different teams in 5 different software development centers. During this project they had several coordination issues on daily *handoffs* [10, 16].

Espinosa *et al.* [20] conducted a FTS study with a research question that guides whether there are gradual differences across time zones that impact team performance. In this study they conducted a laboratory experiment with 42 dyadic teams. The teams were randomly assigned into four time zone overlap conditions: full overlap, 2/3 overlap, 1/3 overlap and no overlap. Using a fictional map task, they found that a small time separation has no effect on accuracy, but that more time separation has a significant effect on accuracy. Another point from that study was the fact that a small amount of time separation had a significant effect on production speed.

The possibility of FTS projects configurations simulations can be an alternative for feasibility studies of FTS practices. One aspect proposed by Carmel *et al.* [10] is the usage of mathematical models for FTS performance evaluation. An approach available to achieve it is the use of stochastic modeling.

2.2 Stochastic Modeling

Structured formalisms have been applied and used along the years increasing the abstraction level and offering another modeling alternative instead of traditional Markov Chains formalism [14].

There are different analytical methods, the most used are based on Markov Chains [62]. Markov Chains describes how a system works based on a group of possible states as well as the transitions between those states, following rates defined by exponential rules [62].

Analytical modeling formalisms are regularly applied to describe different realities. Formalisms based on Markov Chains are applied to several areas such as economy, physics and engineering [62].

In 80s, *Stochastic Automata Networks (SAN)* was introduced as a high level formalism that provides synchronism and parallelism description of systems, and it also provides efficient numerical solutions [25, 48].

This formalism is used for performance evaluation of systems and it is based on the development of a model to represent a real system, where the model has a higher level of abstraction. In this sense, the model is purely mathematical and it works as a real system reduced to mathematical relations [14]. Those mathematical relations describe the system as a group of possible states and transitions between those states [6, 25].

Analytical modeling can be easily applied in different scenarios. One advantage of this technique is that there is no need to be concerned about specific sample groups of the system working in order to obtain performance indices [14].

Analytical models can be *deterministic* or *stochastic*. For a deterministic model, system parameters are previously determined, but for a stochastic model, the system behavior is probabilistic evaluated, in other words, in this case the system parameters are described by random variables with convenient probability distributions [14].

2.3 Stochastic Automata Networks

Stochastic Automata Networks (SAN) are based on Markov chains theory. Through the use of SAN it is possible to describe a global model of a system composed by subsystems, defined as stochastic automata, which have three primitives: *states*, *transitions* and *events* [47]. The definition of stochastic is related to the fact that time is considered as a random variable with exponential distribution for a continuous-time model as geometric distribution for a discrete-time model [6].

A stochastic automaton is defined through a mathematical model of a system with discrete inputs and outputs. The system can be at any state among a finite number of states or internal configurations. The internal state of a system can summarize information about previous inputs and indicate what is necessary to determine the system behavior for next inputs [32].

The local system state is an individual state of each automaton, and the global model state is defined by the combination of local states from the SAN model. The change of a global state is given by changing a local state of any automaton.

In a SAN model, an event is responsible for the occurrence of transitions between states, changing the global model state. Moreover one or more events can be associated to a single transition.

SAN has two different event types, *local* events and *synchronization* events. Local events are responsible for changing a local state of a specific automaton, in order to demonstrate the individual behavior of each automaton. Synchronization events change the state of two or more automata in the model at the same time, describing the interaction between those automata. Events can be associated to functional rates which can be applied to local transitions and synchronization transitions. In addition those rates can be defined as part of functions to reflect the assessment of other states available within a SAN model.

The solution of SAN models, *e.g.*, the numerical results extraction, is usually done by specific algorithms [14] created to deal with a state space derived from the combination of each modeled entity [26].

There are recent studies to apply Stochastic Automata Networks in software engineering area to evaluate software development projects [15, 16, 26], and according to the literature review there are several challenges to simulate projects with teams geographically distributed, such as time, cost and teams composition.

2.4 Simulation

The simulation approach can be described as building a model to simulate a system reality to be evaluated. This model must describe functional system characteristics within an appropriate time scale [46].

During the modeling phase, certain abstraction level must be considered, because the model does not contains all system characteristics, it only contains relevant information to be modeled [6, 25, 52]. In this context, characteristics to be modeled should be careful selected in order to provide simulation results that represent a execution of the system which has been simulated. According to Raffo and Setamanit [51]: *"Software Process Simulation Models can be used as a platform to combine and synthesize previously developed theories and models, and incorporate a wide range of relevant factors"*.

Dafoulas *et al.* [17] presented a research with an approach for setting up pilot studies simulating key features that make global software development teams particularly manageable. Contribution to the FTS model: the importance of the spatial-temporal distance among team members and the configuration of members across different sites. This knowledge provides a foundation of features that make global software development teams particularly attractive to exploit and challenging to manage.

According to Patil *et al.* [45]: *"The nature and the scope of collaboration in knowledge work have changed substantially during the past decade"*. A portion of software development projects is currently being distributed from North America and Europe to countries such as Brazil and India. Contribution to the FTS model: provided an illustrative case of a typical geographically distributed corporate software project. This knowledge described the methodology of a field study of a globally distributed software development project in a multinational corporation.

Raffo and Setamanit's work [51] developed a GSD model incorporating diverse research and theories for GSD projects, including a set of relevant empirical factors, development methods, process, product parameters and project environment factors. Contribution to the FTS model: evaluation of different types of simulation paradigms, identifying important GSD factors as well as the review of quantitative studies and models that could potentially be considered in the FTS model. This knowledge provided a high-level description of a GSD prototype model.

O'Leary and Cummings [43] proposed a list of spatial-temporal indices that account for factors such as teams size, distribution, use of technology and organizational structures. Contribution to the FTS model: They developed a robust view of geographic dispersion in teams focusing on the spatial-temporal distances among team members and the configuration of team members across site. This work described the growing necessity of tools and terms to characterize teams geographic dispersion.

Sooraj and Mohapatra [61] presented a model of 24 hours software development process to help software project managers assess the profitability of a 24-h development configuration and to select the optimal partnering sites. Contribution to the FTS model: the multiplicative model for estimating the length of interaction time. This knowledge provides a foundation in modeling the time and cost of development.

Treinen and Miller-Frost [64] presented time zone difference as an advantage on teams distribution generating a 24h development environment. Contribution to the FTS model: examined whether it is possible to create a development environment in which tasks can Follow-The-Sun. This knowledge provides a foundation of factors that can influence the success or failure of FTS projects.

Setamanit *et al.* [59] described a computer simulation model of software development process which was designed to study alternative ways to configure global software development projects. One of the finds from that work is that although under ideal assumptions follow-the-sun condition was able to produce impressive reductions in time-to-market, under more realistic assumptions the reverse was true. Contribution to the FTS model: the simulation model of the software development process created to study alternative ways to configure global software development projects, including follow-the-sun allocation strategies. This knowledge provided a view that when GSD factors are taken into account, the follow-the-sun strategy is no longer the shortest.

In the context of software engineering simulation and analytical modeling, performance evaluation in software engineering has been successfully used to provide quantitative performance measures [30, 31]. Stochastic models and simulation schemes have been developed towards to the evaluation of dynamics of software projects [40], and also different performance predictions about geographically dispersed teams [9, 27, 60, 70].

Even with the available simulation related work, their focus is not performance evaluation of FTS projects using stochastic models. In this sense, there is an opportunity to research performance evaluation of Follow-The-Sun projects using SAN.

2.5 FTS Performance Evaluation

Czekster *et al.* [15] presented in recent research stochastic models for software development projects, in order to evaluate communication and availability of global software development teams. Following this context, it was created a basic stochastic model for FTS projects evaluation [16]. In this context, it still exists opportunity to capture other important aspects of global software development projects using FTS configuration, such as handoff efficiency, projects phase control, teams capacity and others. This work identifies and catalog performance evaluation aspects that can influence the decision-making process on Follow-The-Sun projects.

A project is a temporary endeavor to generate a product, a service or a result. Projects are done around the world by companies and institutions to achieve several goals. In the software development industry, project management is a common practice and has been widely used for software development projects. Software development is defined as a knowledge intensive and complex activity [1], and software development projects can be done distributed on different sites. According to Prikladnicki *et al.* [50]: *"As with many other industries today, software development must increasingly adapt to teams whose members work together but are geographically distributed"*.

The number of uncertain variables and scenarios to manage global software development projects is bigger than managing local software development projects. There are several studies in literature related to the complexity of global software development projects [1, 10, 28, 29, 50, 51, 61, 64] to cite a few.

This work modeling effort is centered on some characteristics of projects such as: time, cost and quality.

2.5.1 Project View

The performance evaluation of projects under a portfolio can contribute to predict portfolio results helping company and institutions to better drive their investments.

As mentioned on previous sections, FTS projects have several characteristics and challenges. As part of performance evaluation process, projects scenarios must be created. Figure 2.1 presents a project view. In this conceptual mapping we focus our attention to the following factors:

- **Cost**: Blended dollar rate *per* hour, to be assigned to each development site;
- **Time**: Planned project time in number of months;
- **Quality**: Will be measured as the amount of work spent on development *versus* the amount of time on rework (bug fix) activities.

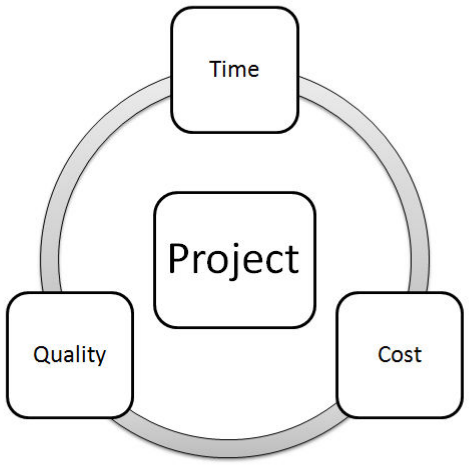

Figure 2.1: Project View

In this context, Follow-The-Sun projects are executed in different geographical locations usually named as sites.

2.5.2 Site View

Site is the location where one or more participants of a distributed project will be performing projects activities. Each site can have a team of *N* people executing a variety of projects tasks and each team member can have a different allocation. As part of a FTS modeling effort the site is an important entity.

According to Jalote and Jain [34]: *"With a better understanding of the benefits and constraints, adequate communication and coordination environments can be developed to support the tight coordination that is necessary to reap the benefits of the 24-hour model"*. In this context a more realistic model also consider time slots that are overlapping [34], however this work model will not consider overlapping of sites as part of this work scope.

Figure 2.2 represents the site view. The performance evaluation model will consider these characteristics as part of scenarios coverage:

- **Country** (Country / team location);
- **Shift size** (number of work hours *per* day);
- **Team size** (number of people engaged on the project);
- **Average cost** (Average dollar rate *per* hour).
- **Number of sites** that will be part of the scenario;
- **Handoff frequency** time spent on handoff between sites;
- **Time zone differences** between sites;

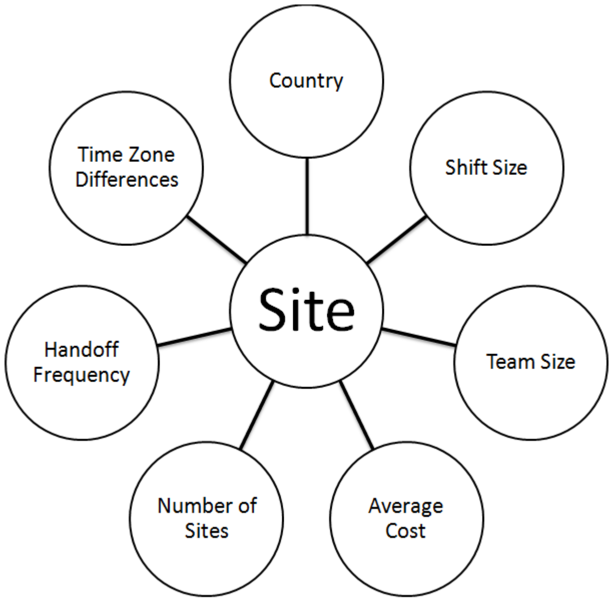

Figure 2.2: Site View

After the description of aspects to be considered at scenarios level, a SAN model should be created. Traditionally, scientific works on software engineering area present different applications for mathematical models, *e.g.*, automated software testing processes [4, 24], and quantitative evaluation of software development teams also evaluating project risks [2, 8].

This work demonstrates the usefulness of analytical modeling applied to Follow-The-Sun projects, presenting the solution of a FTS software development life cycle. It focus on the impact of sites interactions for time, cost and quality. This work does not have a goal to analyze the impact of coordination or cultural diversity, even though these aspects are often relevant. This modeling effort only aims on the impact of different FTS teams configurations in software development projects.

This section presented project aspects that will be used as part of performance evaluation scenarios. Global software development stochastic modeling and FTS stochastic modeling are challenges already discussed by Czekster *et al.* [15, 16] and Urdangarin *et al.* [65].

3. Methodology

This section presents the methodology and procedures used on this research.

3.1 Research Strategy

This research started by a literature review and a practical case study through performance evaluation, manipulating different variables for results analysis. This research is exploratory, because it examines an issue or problem apparently not largely studied or that has not been previously approached in the same way in literature. Exploratory research is challenging because it does not use formal logic methods neither exhaustive experiments, however it requires examples in order to find indicatives results [68]. This type of research aims to explore a research problem in order to explicitly render it. It also involves literature review and the use of examples to demonstrate indicatives to enhance the research problem understanding.

3.2 Research Design

This section describes the process applied to answer the research question. The first stage is composed of a literature review and a practical case, the second stage is composed of an academic experiment and the first modeling exercise and the final stage is composed of the final model. Figure 3.1 graphically shows this research design.

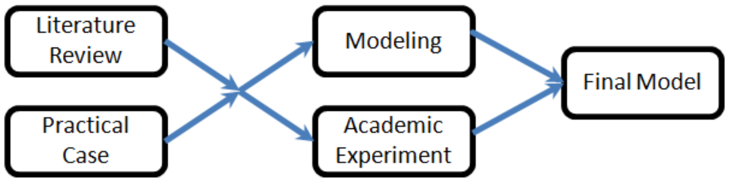

Figure 3.1: Research Design

3.3 Literature Review

This literature review aims to find existing studies related to this research goal. This research began with a literature review with focus on: Analytical Modeling, Stochastic Automata Networks, Global Software Development and Follow-The-Sun.

The literature review used two digital libraries, IEEExplore e Elsevier ScienceDirect. The search was executed between the second semester of 2010 and the first semester of 2011. Using different research strings were listed 248 studies and 21 of them were selected and reviewed.

Table 3.1 presents the research strings used:

Table 3.1: Research Strings

String	IEEExplore	Elsevier	Reviewed
"follow-the-sun"	13	38	[9, 10, 12, 19, 42, 64, 66, 67, 69]
"follow-the-sun" AND *"SAN"*	0	9	
"follow-the-sun" AND *"stochastic automata networks"*	0	0	
"follow-the-sun" AND *"simulation"*	1	0	[59]
"follow-the-sun" AND *"stochastic modeling"*	0	0	
"follow-the-sun" AND *"stochastic simulation"*	0	0	
"global software development" AND *"SAN"*	111	35	[58, 63]
"global software development" AND *"stochastic automata networks"*	0	0	
"global software development" AND *"stochastic modeling"*	0	1	[33]
"global software development" AND *"stochastic simulation"*	0	1	[33]
"global software development" AND *"simulation"*	6	33	[17, 33, 38, 39, 45, 59, 61]

Reviewing the initial 19 selected studies and based on author names and their references other studies were found, creating a list of available references available at the bibliography session. Once the literature review was completed, it was performed an analysis to identify practices that could be used as part of this study. Based on this information, it was created a SAN model for FTS projects which will be presented in details at chapter *SAN Model*.

3.4 Practical Case

In this paper [26], it was presented SAN analytical modeling for a practical case study from an Information Technology company that has multiple sites and different participants' roles and expertise. It has performed the matching of model predictions against actual project observations. Also it had focus on the central entity varying its availability and the level of provided support in order to observe the impact on participants' performance. This work was summarized with further discussions of numerical results and possible model extensions.

It reported findings about the use of SAN for analytical modeling of software development teams in order to predict their performance in different scenarios. Results were based on a case instance of a multi-site project analyzing the effect of availability and levels of support provided by a centralized management entity. In order to verify the prediction accuracy, the numerical results obtained from the proposed model were compared with the actual hours spent in a real project's phases. It had done analysis of possible scenario variations in the project considering different behaviors and participant's skills. Specifically, the availability and quality of the central entity support is analyzed in different scenarios, and the impact on the whole team productivity.

In order to show that analytical modeling is useful to complex state-based performance analysis of software development teams in multi-site context it presented a model of a practical case study named as Project *ALPHA* due to confidentiality.

In that model, the project and delivery managers were abstracted as a central team entity. Therefore, activities of these managers are encapsulated in automaton *Activities*, as well as their availability to interact with other participants are encapsulated in automaton *Availability*.

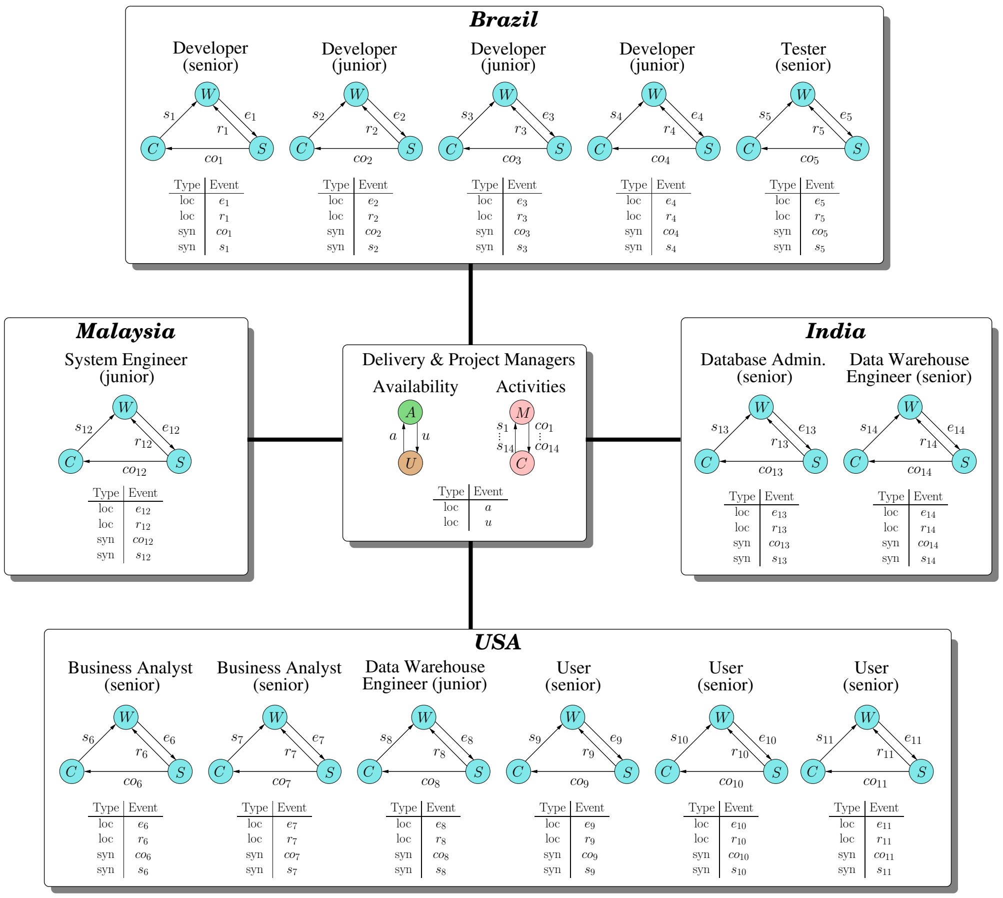

Figure 3.2: The SAN model that represents Project *ALPHA*

Each participant was modeled as an automaton of three states representing its possible activities in a workday: $(W)orking$, $(S)eekingSolution$ and $(C)ollaboration$. The software development team was composed of $N = 14$ participants, where five participants were from Brazil, six from USA, one from Malaysia and two from India. Participants (developers, testers, business analysts, data warehouse engineers, users, system engineers and database administrators) were located in different sites and had different expertises, need to report and collaborate with the project and delivery managers, as well as they collaborate with other participants.

The numerical solution is primarily expressed by the steady-state probability of the SAN model. Based on these probabilities and participants individual hours allocated to Project *ALPHA*, it can be determined the average working hours *per* eight-hour workday. Readers interested in more information about the software tool for numerical solution of SAN models refer to PEPS (*Performance Evaluation of Parallel Systems*) [5] or SAN lite Solver [54].

The whole model with states, transitions, events and their associated rates can be numerically solved in order to obtain the steady-state probabilities of the model.

Table 3.2: Estimated event rates for an eight-hour workday

Type	Event	Description	Rate
loc	a	Central team is available to collaborate with participants on average 2 hours *per* workday, *i.e.*, central team collaborates on a rate of 4 times *per* workday.	$8/2$
loc	u	Central team is unavailable to cooperate with participants on average 6 hours *per* workday.	$8/6$
loc	e	Junior participants work on average 1 hour *per* workday without any kind of central team support.	$8/1$
		Due to their expertise, senior participants work on average 7 hours *per* workday without any kind of central team support.	$8/7$
loc	r	Junior participants spend on average 7 hours *per* workday seeking solutions.	$8/7$
		Senior participants spend on average only 1 hour *per* workday seeking solutions because of their expertise.	$8/1$
syn	co	Once a participant needs to collaborate with the central team, the collaboration occurs immediately if the central team is available.	$disp$
syn	s	Due to the central team support quality (from medium to low), the collaboration takes on average 2 hours *per* workday.	$8/2$

Based on these probabilities, it is possible to calculate the team's performance for different scenarios varying, for example, the central team's availability and the level of expertise of participants.

The project quantitative data was collected focusing mainly on the execution phase working hours and on the impediments occurrence during the project. Impediments were described by participants and central team managers as effective interactions among them.

Table 3.3 shows the estimated effort and actual hours related for each phase of Project *ALPHA*. It is important to remark that approximately 3,317.22 hours were actually spent by participants completing their tasks in the execution phase.

Project *ALPHA* faced issues resulting in a significant amount of impediments in the project execution phase. Table 3.4 summarizes these impediments grouped in eight categories: (i) *Resource* indicates problems related to project resource constraints, allocation, changes and replacements; (ii) *Technology* indicates issues concerning tools used within the project; (iii) *Process* indicates problems related to the development process exceptions during project execution; (iv) *Requirements*

Table 3.3: Project Estimated Effort × Actual Hours

Project phase	Estimated	Actual
Initiating	611.70 h	771.65 h
Planning	1,529.25 h	895.60 h
Execution	3,364.35 h	3,317.22 h
Monitoring and Controlling	611.70 h	438.80 h

indicates the requirement information gathering issues; (v) *Schedule* indicates delays in project deliverable dates; (vi) *Deliverable* indicates code or project artifacts, *e.g.*, documents, issues; (vii) *Scope* indicates issues to project scope changes; and (viii) *Infrastructure* indicates problems with application infrastructure.

In Table 3.4, we present the *quantity*, *category*, *average*, and *total duration* of project's impediments. We can notice in this table that Project *ALPHA* has spent about 332 hours dealing with impediments.

Table 3.4: Project's impediments

Quantity	Category	Average	Total duration
9	Resource	8 h	72 h
3	Technology	4 h	12 h
2	Process	8 h	16 h
11	Requirements	4 h	44 h
4	Schedule	8 h	32 h
4	Deliverable	24 h	96 h
1	Scope	30 h	30 h
6	Infrastructure	5 h	30 h
40	Total	8.3 h	332 h

Table 3.5 shows the results of the main entities of the proposed model (*Central Team*, *Senior*, and *Junior* participants) and their correspondent steady-state probabilities.

Observing the results presented in Table 3.5, it is possible to notice that senior participants have, as expected, high autonomy to deal with their tasks. However, the probability of working state (W) for senior participants is slightly smaller than the 87.5% corresponding to the seven hours of working *per* eight-hour workday. Observing the working percentage of junior participants, we find a quite low value of 14.7% which corresponds to a little more than one hour of work *per* eight-hour workday, while most of their time (more than six hours per workday) is spent in *state* S (seeking solution). Junior participants do not collaborate very often with the central team (half an hour *per* workday), which is probably a side effect of the low availability and low quality provided support.

Table 3.6 shows the participants individual working hours taking into account the average percentage of working hours each participant was allocated to the project, project *ALPHA* spent

Table 3.5: Steady-State probabilities of the proposed model

Entity	State	Probability
Central Team	A	25.00%
	U	75.00%
	M	56.27%
	C	43.73%
Seniors	W	87.09%
	S	11.96%
	C	0.95%
Juniors	W	14.70%
	S	78.26%
	C	7.04%

3,317.22 hours in the execution phase, instead of the initially estimated 3,364.35 hours. This difference corresponds to less than 1.5%, which already is a small difference.

Moreover, Table 3.6 shows that the analytical model instantiated to Project *ALPHA* parameters pointed out an average of 13.69 working hours, considering all participants individual working hours. Consequently, using as basis 242 days (11 months with 22 workdays) of project execution, the total number of working hours calculated from the model probabilities is 3,312.98 hours. Note that the model provided an approximated value for the total working hours, which is even closer to the actual number of hours spent in the execution phase. In fact, the predictions obtained from the SAN model deliver estimations with a relative error of less than 0.2%.

Table 3.6: Project working hours obtained from the proposed model

Quantity	Expertise	Allocation (%)	*State W* (%)	Working hours *per* day
1	Senior	100	87.09	6.97
3	Junior	75	14.70	2.65
1	Senior	20	87.09	1.39
2	Senior	10	87.09	1.39
1	Junior	3	14.70	0.04
3	Senior	3	87.09	0.63
1	Junior	5	14.70	0.06
1	Senior	5	87.09	0.35
1	Senior	3	87.09	0.21
			Total	13.69

Analogously to the prediction of working hours, Table 3.7 presents the results from the analytical model regarding project impediments, using *state* C probabilities as statistical information about needed interactions to solve issues. The cooperation hours are summed in this table to indicate the time spent in solving project issues cooperating with the central team, considering an eight-hour workday. *State* C abstraction indicates that participants have found issues difficult to overcome by themselves and need effectively to cooperate with the central team managers to solve them.

Table 3.7: Project impediment hours obtained from the proposed model

Quantity	Expertise	Allocation (%)	State C (%)	Cooperating hours *per* day
1	Senior	100	0.95	0.076
3	Junior	75	7.04	1.267
1	Senior	20	0.95	0.015
2	Senior	10	0.95	0.015
1	Junior	3	7.04	0.017
3	Senior	3	0.95	0.007
1	Junior	5	7.04	0.028
1	Senior	5	0.95	0.004
1	Senior	3	0.95	0.002
			Total	1.43

The analytical model predictions for the collaboration hours to solve project impediments is on average 1.43 hours *per* day. Considering 242 workdays as the project duration, the model results indicate a total of 346.06 hours of impediments. Comparing this prediction with the 332 impediments hours observed in Project *ALPHA*, a relative error of 4% was found taking into account the amount of information that was abstracted while defining the model states and event rates.

The first contribution of this practical case [26] was to put into a real project scenario a theoretical modeling effort to describe a complex environment of software development. Despite the numerous abstractions made in the modeling stage, the obtained numerical results demonstrate an accuracy when compared to actual project outcome. Such fact by itself justifies the initial assumption that analytical modeling, in the particular case modeled using SAN formalism, may be a worthy option to build teams in software development projects. Another contribution of this work was to numerically demonstrate how much a project success remains on the number of experienced (senior) participants. Not even with a near ideal situation of central team availability and quality, the problems brought by a large number of junior participants can be overcome.

3.5 Academic experiment

This paper [37] reports an academic experiment which has as a goal to verify if an adaptive methodology had more benefits than a prescriptive methodology for a FTS strategy. Prescriptive software development uses the traditional cycle through the phases of Initiation, Analysis, Design, Construction and Testing. However, Adaptive software development replaces the traditional waterfall cycle with a repeating series of planning, development, testing and learning cycles. This study presented the design of a controlled experiment to investigate the differences of project quality and project speed when using and adaptive versus a prescriptive approach for FTS. It used fictitious maps with teams distributed in two sites in order to exercise different teams working on design and development activities. Each site had two teams representing software designers and developers and results demonstrated that the use of the adaptive approach increases the speed, but not the accuracy and the quality of the work.

This study aims to investigate both the adaptive and prescriptive approaches on using a FTS strategy, and compares the results. To reach this goal it performed a controlled experiment with students from a postgraduate Computer Science program at the Pontifical University Catholic of Rio Grande do Sul (PUCRS) in Brazil.

The experiment was planned to have five shifts of 15 minutes each (representing one workday). For each approach, the participants were organized in two sites where each site had two teams. It was used the FTS with no overlap condition between teams. The number of maps that each team delivered divided by 13 (total number of possible maps) results on the working speed. On five shifts, adaptive teams had produced six out of thirteen maps and prescriptive teams had produced four out of thirteen maps and prescriptive teams produced 4 out of thirteen maps. Teams, which had used adaptive approach, obtained 16% more speed than teams that have used a prescriptive approach.

To check the accuracy obtained, were considered derivable tasks. Each map was composed by one background image + two icons + five arrows. In addition, it was also considered the right position of the elements on map (arrows and icons). The total points of correct elements for each map are 12.

Using adaptive approach, teams produced more maps than teams using prescriptive approach. However, when we observe total points of correct elements for each map, prescriptive teams had more accuracy. It also verified the quality criteria between teams: Adaptive: $4.17/6 = 70\%$ Prescriptive: $3.38/4 = 85\%$ Findings presented that teams on adaptive approach had 15% less quality, in contrast, these teams had 16% more speed.

This experiment [37] promoted some insights on how FTS could be used in the software industry. It also contributed to the SAN modeling events and transitions design, specially to set the *start handoff (SH) and finish handoff (FH)* event rates probabilities.

3.6 Modeling

In this paper [16], it was presented a study about Follow-The-Sun (FTS). This work goal was to discuss a formal mapping of FTS characteristics to a stochastic model in order to predict performance indices of teams such as availability and risk assessment. The modeling effort was used to improve understanding and feasibility evaluation for FTS projects [16].

Moreover, it demonstrated benefits of using the Stochastic Automata Networks (SAN) formalism for the modeling and evaluation of distributed teams, since SAN provides a modular description with functional primitives. The modeling effort presented aims to enhance understanding and feasibility evaluation for FTS projects calculating probabilities for availability and project risk factor, such as handoff efficiency, considering the conceptual framework proposed by Carmel *et al.* [10]. Conclusions pointed out future works directed to model extensions to capture advanced characteristics.

This modeling effort was centered on the handoff efficiency concept [11] to evaluate the probabilities of each site being reworking activities or waiting for clarifications in a FTS project. Usually FTS projects have strict rules imposed on their interactions using handoffs between teams at fixed intervals of time.

For the composition of model's states and events, there were assumptions related to FTS characteristics and behavior instantiating a stochastic model having a set of s multiple sites (*e.g.*, each site or team encapsulating n members) working in different time-zones. It considered multi-variate conditions, where sites can shift their work in an efficient (or inefficient) manner, depending on the project configuration.

The computed performance indices are actually a set of quantitative measures from the model, *i.e.*, transient or steady-state probabilities that allow us to estimate the impact of the handoff efficiency on the overall project duration. The model could be used to inspire future instantiations of different models. In this sense, the approach adopted here allows a minimal FTS setting where modelers can extend it to convey other behaviors or to test novel configurations (perhaps optimal ones or settings where teams begin to somehow to be underperformed).

Figure 3.3 shows an example of FTS composed of three sites across the world. In the asynchronous handoff point of view, the sites are unable to interact with the site that shifted the work due to different time zones.

The stochastic automaton representing each site is a five-state model composed of states that can be assumed by the site at each time in a workday:

- ***Off-line*** state, the site is off-line due to time zone, so it is unreachable for collaboration, communication or interaction;
- ***Opening*** handoff state, beginning of the workday;
- ***Working*** state, actual work is being performed in the site (active workday);
- ***Reworking*** state, misunderstandings occurred mainly in the handoff process lead to rework [21];
- ***Closing*** handoff state, end of the workday.

Figure 3.4 present the model where event wk has an associated function that allows (or not) the occurrence of this event. Since we are modeling an asynchronous handoff, a site can open its handoff

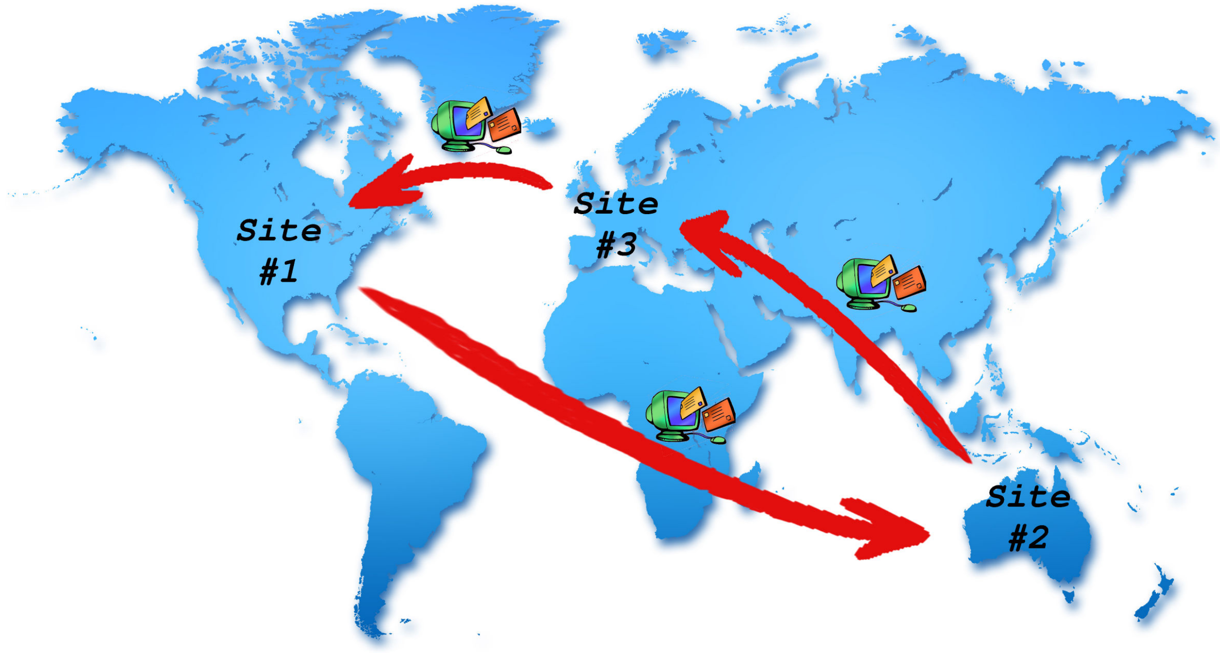

Figure 3.3: Follow-The-Sun

process without changing the state of earlier or next sites. However, each site has a dependency of the state from the earlier site to start to work or rework a task. In this asynchronous model, a function is associated to event wk, which verifies if the earlier site's handoff has been completed. For instance, event wk_1 can occur if *Site* $\#3$ is in *Off-line* state, *i.e.*, if *Site* $\#3$ has closed its handoff. Otherwise, event wk_1 cannot occur and need to wait the closure of handoff from *Site* $\#3$.

This model is composed of three sites, but it can be instantiated for 2, 3, or 4 sites (*e.g.*, considering 24-hour software development with 6-hour shift *per* site) for a model with four sites. The individual behavior of a given *Site* $\#i$, where i is the index of the site, is represented by the following events: *open*$_i$, *wk*$_i$, *rw*$_i$, *nt*$_i$, *cl_wk*$_i$, *cl_rw*$_i$, and *off*$_i$.

With a better understanding of the benefits and constraints, adequate communication and coordination environments can be developed to support the tight coordination that is necessary to reap the benefits of the 24-hour model [34].

This work [16] presented the mapping of the interaction pattern of development sites under FTS methodology for a SAN model. The main entities (*Site* $\#1$, *Site* $\#2$ and *Site* $\#3$) were represented by automata. The abstraction, for instance, represented the handoff within a project: some of the time is spent on *Opening handoff*, *Closing handoff* and the rest in *Working* or *Rework* states. It presented an abstraction of how to model development teams in Follow-The-Sun (FTS) context and it has proposed a scenario that enables the investigation of overall FTS project performance. It was an initial approach that showed how the combination of stochastic modeling and FTS could be mixed to produce important considerations to be used by decision makers, *i.e.*, solving the stochastic models it is possible to obtain a better view of projects bottlenecks before the projects execution.

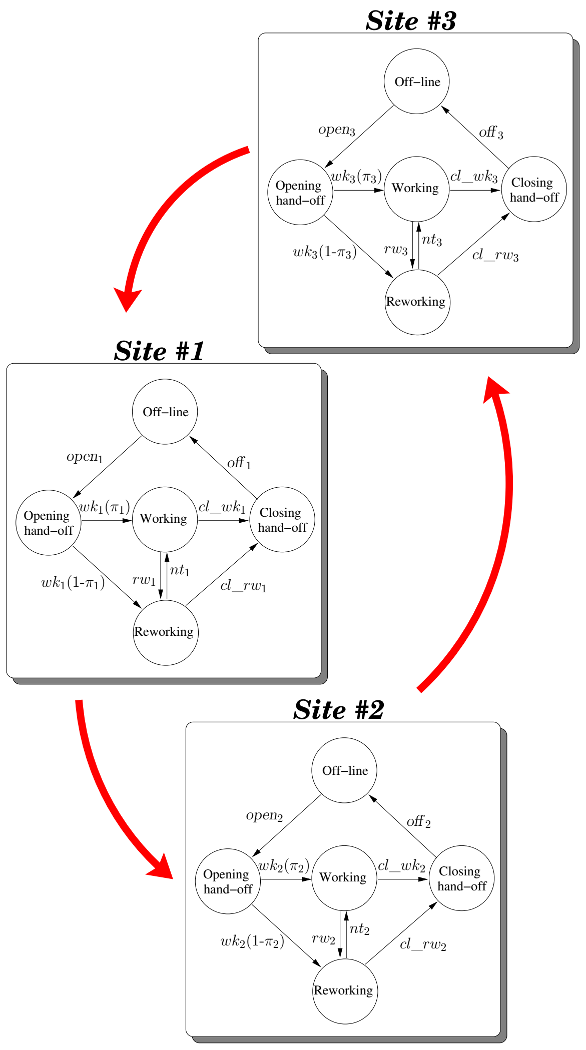

Figure 3.4: SAN model

3.7 Final Model

The previous proposed model focused on the handoff efficiency of follow-the-sun projects and discussed how to parameterize asynchronous models with different measures or observations for different aspects. In this sense, the final model was created based on the studies presented on this methodology and the full description of the proposed final model is presented in Chapter 4, Section 4.1. An abstract about this final model is available on Santos *et al.* [56].

4. SAN Model

According to Czekster *et al.* [16]: *"stochastic modeling is a feasible and flexible alternative to abstract GSD teams configuration in environments such as Follow-The-Sun to measure teams' performance. Moreover, the stochastic modeling can be used during project planning phases in order to find relevant indicative in terms of performance analysis that could help the planning improvement"*.

This section presents a follow-the-sun analytical model using stochastic automata networks formalism. The goal of this model is to instantiate multiple sites (composed by N team members). The main entities represented in this model are described as automata.

4.1 Automata Definitions

Automaton is an entity of a SAN model. Figure 4.1 graphically shows three dispersed software development sites interactions in FTS. In this model, it assumes that development work cycle is divided between different sites. Thus, it considers sites self-contained, *i.e.*, sites only work on tasks under their shift assignments. Handoff interactions occur between *Site 1* and *Site 2*, *Site 2* and *Site 3*, *Site 3* and *Site 1*. Team members do not interact with another site that is different from their previous handoff or their next handoff. There is a handoff to transition activities between shifts from previous site to the next one. This work focus on three sites, however it can be modified and extended to use a different number of sites.[1]

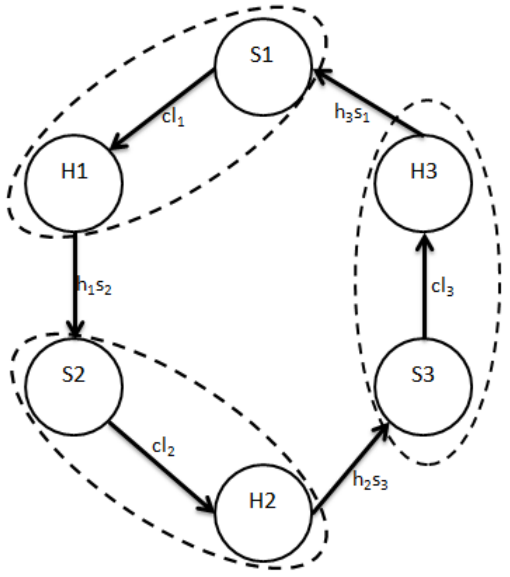

Figure 4.1: Clock Memory Automaton

The clock memory automaton states description is available in Table 4.1.

[1]The reason of been using three sites is described on the Section 4.2.

Table 4.1: Clock Memory Automaton States

State	Description
S1	Site One Working
H1	First Handoff
S2	Site Two Working
H2	Second Handoff
S3	Site Three Working
H3	Third Handoff

Beyond the description of SAN model entities and its relationships it is necessary to assign durations probabilities that every entity rests in a given state named as model parameters and this is paramount within analytical modeling because of its expressiveness [15]. For each transition between states there is an event associated and each event has a probability of occurrence. Clock memory automaton has *six* events associated to it as described in table 4.2.

Table 4.2: Description of *Clock Memory* events

Event	Description
cl_1	This is a synchronizing event and execute the transition from close handoff to offline on *site one*.
$h1_{s2}$	This is a synchronizing event and execute the transition from site one offline to start handoff on *site two*.
cl_2	This is a synchronizing event and execute the transition from close handoff to offline on *site two*.
$h2_{s3}$	This is a synchronizing event and execute the transition from site two offline to start handoff on *site three*.
cl_3	This is a synchronizing event and execute the transition from close handoff to offline on *site three*.
$h3_{s1}$	This is a synchronizing event and execute the transition from site three offline to start handoff on *site one*.

The *Site i* automaton abstracts the activities that will be performed by the sites instantiated in the model. Each site can perform the following activities:

- **Handoff**: Transition of activities between shifts of different sites.
- **Requirements Gathering**: Requirements management activities
- **Development**: Development and unit test activities
- **Test**: All test related activities
- **Bug Fix**: Rework related activities

In this context, the mapping of development team interaction of Site *i* (where *i* is the index of the site) to a SAN model is straightforward, as presented in Fig 4.2.

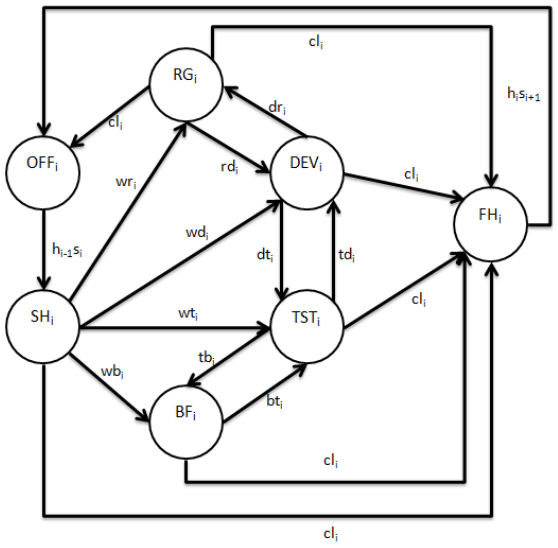

Figure 4.2: Team Site *i* Task Automata

Each site has *N* members that could be used to evaluate different development velocities *per* site, helping the decision making process to be more efficient in terms of resource management. The abstraction considers software development life cycle represented in each site by *seven* states as described in Table 4.3.

Table 4.3: Site *i* States

State	Description
OFFi	Site *i* is off-line and do not perform any project activity.
SHi	Start handoff completing the work transition from previous shift to current shift.
RGi	Requirements Gathering and business system analysis.
DEVi	Software development and unit testing.
TSTi	Quality assurance testing.
BFi	Rework and bug fix.
FHi	Handoff to close activities from current shift to next shift.

There are several other type of activities that could be executed within a site that could be included on the model abstraction, but for the scope of this research it will consider *seven* different activities states per site.

For each state transition there is an event associated, for *Site i* automata all events are type of synchronizing because they are responsible to synchronize information with the automata *Memory Control* which will know what is the activity that the next site or shift should resume working once the current site is set to offline state. Each site automaton has *thirteen* events associated to it as described in table 4.4.

Table 4.4: Description of *Site i* events

Event	Description
$h_i s_{i+1}$	**Start handoff**: this event synchronizes i-*th* site automaton with *Clock Memory* automaton, coordinating start handoff activities between sites.
$h_{i-1} s_i$	**Finish handoff**: this event synchronizes i-*th* site automaton with *Clock Memory* automaton, coordinating finish handoff activities between sites.
cl_i	**Closing activities**: when this event occurs the i-*th* site goes to the state where it needs to finish handoff.
wr_i	**Resume working on requirements gathering**: this event synchronizes i-*th* site automaton with *Project Memory* automaton, enabling the site to work on requirements gathering activities once the start handoff was completed.
wd_i	**Resume working on development**: this event synchronizes i-*th* site automaton with *Project Memory* automaton, enabling the site to work on development activities once the start handoff was completed.
wt_i	**Resume working on testing**: this event synchronizes i-*th* site automaton with *Project Memory* automaton, enabling the site to work on testing activities once the start handoff was completed.
wb_i	**Resume rework**: this event synchronizes i-*th* site automaton with *Project Memory* automaton, enabling the site to work on bug fix activities once the start handoff was completed.
rd_i	**Move to development**: this event synchronizes i-*th* site automaton with *Project Memory* automaton, by the transition of activities from requirements gathering to software development.
dr_i	**Move from development**: this event synchronizes i-*th* site automaton with *Project Memory* automaton, by the transition of activities from software development to requirements gathering.
dt_i	**Move to testing**: this event synchronizes i-*th* site automaton with *Project Memory* automaton, by the transition of activities from software development to testing.
td_i	**Move from testing**: this event synchronizes i-*th* site automaton with *Project Memory* automaton, by the transition of activities from testing to software development.
tb_i	**Move to bug fix**: this event synchronizes i-*th* site automaton with *Project Memory* automaton, by the transition of activities from testing to bug fix.
bt_i	**Move from bug fix**: this event synchronizes i-*th* site automaton with *Project Memory* automaton, by the transition of activities from bug fix to testing.

The use of one automaton to represent each team member within the SAN model increases the time required to solve the SAN model, however it is the way that a model can better represent individuals project allocation. In Figure 4.3 describes the automata that represent the number of members of Site *i*.

Team site automaton abstracts *N* team members working at a given shift and different members allocations can be set on the parameterization phase. Each site can have *N* participants working.

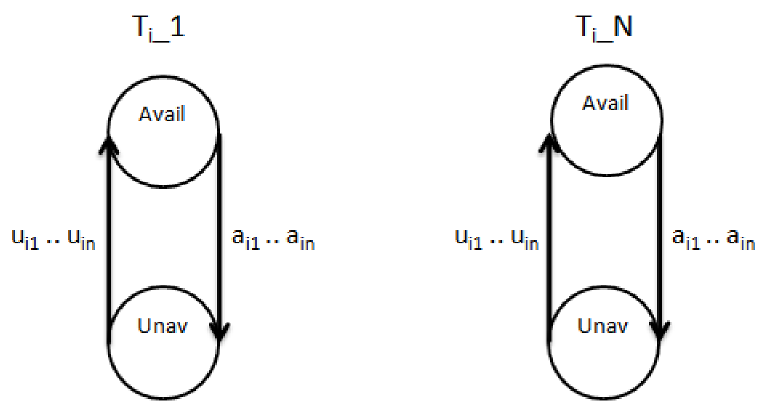

Figure 4.3: Team Site *i*

Table 4.5 presents team site states description.

Table 4.5: Team Site States

State	Description
Avail	Indicate that a participant is available to work
Unav	The participant is unavailable and cannot perform any project task

Team members can have different allocations to different projects, in this sense team members are abastracted in this model as an automata with *two* states representing the availability and unavailability of a given team member. Table 4.6 presents a summary of *team site i* events, previously illustrated.

Table 4.6: Description of *Teams Site i* events

Event	Description
a_i	This is a local event and execute the transition of a team member from available to unavailable.
u_i	This is a local event and execute the transition of a team member from unavailable to available.

Moreover, there is an automaton for project flow in order to control the handoff (activities transitions) between sites. In case a site is working on development, the project flow automaton controls that it is the current task, when the current shift handoff is complete and the next shift start handoff, the next site will follow the activity pointed by project flow automaton.

Figure 4.4 presents the automaton that controls the project memory. Project memory shifts control is used to track what activity has been handled by a given site and when it transition to the next shift, it will resume work from the point where the previous shift stopped to work.

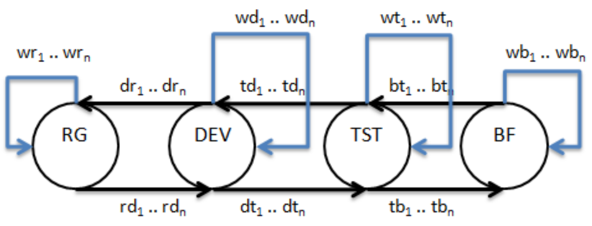

Figure 4.4: Project Memory Automaton

Figure 4.5 presents the full stochastic automata network that represents the model. Project memory shifts control is used to track what activity has been handled by a given site and when it transition to the next shift, it will resume work from the point where the previous shift stopped to work.

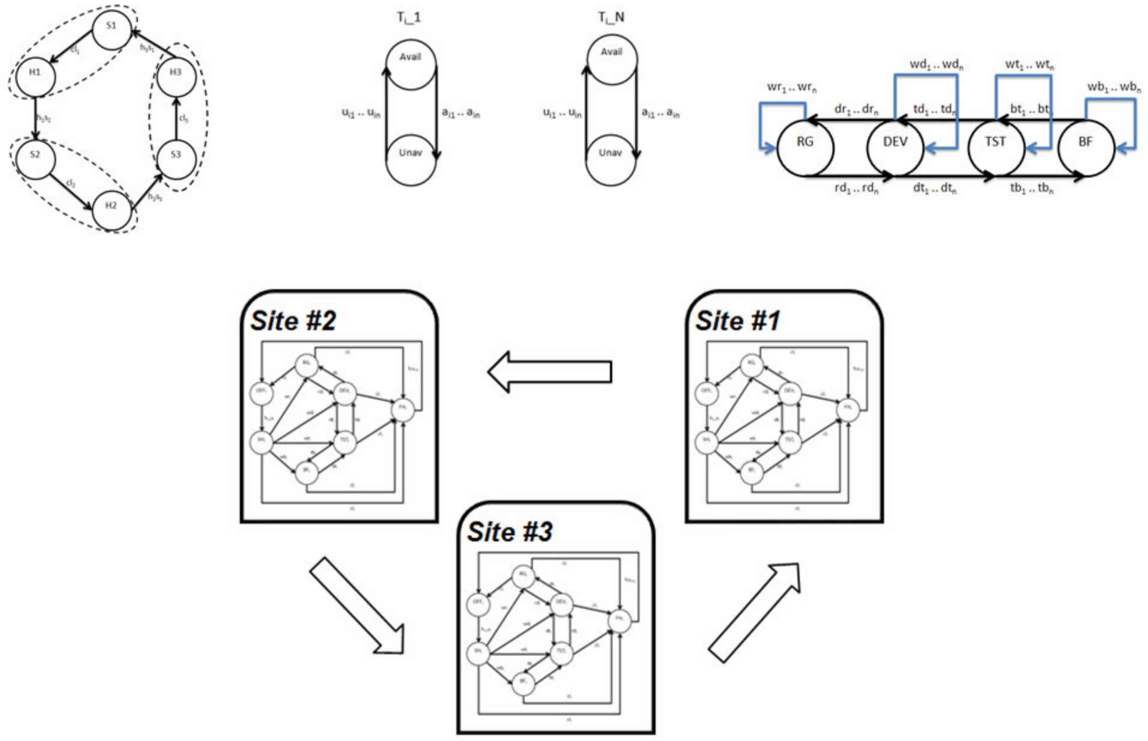

Figure 4.5: Full Model View

The full model view provides an abstraction of a model instance using three sites and N team members, the automata representing team members can have N instances according to the number of people to be considered as part of the performance evaluation analysis.

Also it is important to remark that for each instance of a team member automata it is possible to set different events probabilities enabling the evaluation of different people allocation as well as different development velocities because the according to the number of members instantiated *per* site it also enables different development velocities *per* site.

In the context of project memory states transitions, for each transition between states there is an event associated and each event has a probability of occurrence. Project memory automaton has *ten* events associated to it as described in table 4.7.

Table 4.7: Description of *Project Memory* events

Event	Description
wr_i	**Resume working on requirements gathering**: this event synchronizes i-*th* Project Memory automaton with *Site i* automaton, enabling the site to work on requirements gathering activities once the start handoff was completed.
wd_i	**Resume working on development**: this event synchronizes i-*th* Project Memory automaton with *Site i* automaton, enabling the site to work on development activities once the start handoff was completed.
wt_i	**Resume working on testing**: this event synchronizes i-*th* Project Memory automaton with *Site i* automaton, enabling the site to work on testing activities once the start handoff was completed.
wb_i	**Resume rework**: this event synchronizes i-*th* Project Memory automaton with *Site i* automaton, enabling the site to work on bug fix activities once the start handoff was completed.
rd_i	**Move to development**: this event synchronizes i-*th* Project Memory automaton with *Site i* automaton, by the transition of activities from requirements gathering to software development.
dr_i	**Move from development**: this event synchronizes i-*th* Project Memory automaton with *Site i* automaton, by the transition of activities from software development to requirements gathering.
dt_i	**Move to testing**: this event synchronizes i-*th* Project Memory automaton with *Site i* automaton, by the transition of activities from software development to testing.
td_i	**Move from testing**: this event synchronizes i-*th* Project Memory automaton with *Site i* automaton, by the transition of activities from testing to software development.
tb_i	**Move to bug fix**: this event synchronizes i-*th* Project Memory automaton with *Site i* automaton, by the transition of activities from testing to bug fix.
bt_i	**Move from bug fix**: this event synchronizes i-*th* Project Memory automaton with *Site i* automaton, by the transition of activities from bug fix to testing.

Different number of resources can be allocated on each site in order to enhance the balance on project costs. Once the parameterization is complete, the model can be executed in specialized tools for numerical solution, *e.g.*, PEPS software tool [5] or SAN Lite-Solver [54], in order to extract performance indices and measures [15]. Next section presents model options and constraints.

4.2 Model Options and Constraints

This work has the assumption that workload (daily activities) is equally complex for all team sites and that project resources are assigned from the begin to the end of each project. Also this model abstraction do not consider how factors such as different cultures, tasks complexity and workload impact on the performance evaluation.

Based on Czekster *et al.* [16] research, this work replicates the following assumptions:

- to support team design decisions the model is easily extended to more or less sites, with different working hours, reworking probabilities, average time spent in handoff activities, given a specific project;
- cultural issues, lack of communication or coordination support are not explored in this work.

This work used three sites as default model option because it is the closest way to cover 24 hours of a workday. Van Solingen and Valkema [66] found indications that when the number of sites in a daily cycle increase, on average the overall working speed of sites increased, however they have mentioned that: *"The maximum number of sites in a daily cycle is finite, due to the 24 hours in a day. More sites provide more working capacity, however also require more overhead and increase the likelihood of mistakes"*. Based on that study it was decided to use a number of sites between two and four. Besides, Sangwan *et al.* [55] recommend to starting a global software development project with no more than two or three sites. On this context, it was decided to use three sites on the stochastic modeling.

When instantiating the stochastic model it needs to consider the number of resources working on each site for performance evaluation numerical results. Sangwan *et al.* [55] suggested a rule of thumb where teams should be no larger than 10 people because *"jumping into a large distributed project without having past experience is not recommended"*.

The default rates applied in this model abstraction are based on a workday of 8 hours. The proposed model does not predict shifts overlap. However literature studies [64, 67] point out cases where is recommended a *sticky handoff*, *i.e.*, intense interactions are more favorable than a *clean handoff* (*drop-and-go* approach). The average rates time used on each model state are based on the following literature references Javed *et al.* [35], Basili *et al.* [3], Carmel *et al.* [10].

- Javed *et al.* [35], a work based on four industry projects.
  Average RG time: 23%
  Average DEV time (Design + Coding): 42%
  Average TST time: 24%
  Average time others: 11%

- Basili *et al.* [3], a work based on a lab experience.
  Average RG time: 12%

Average DEV time (Design + Coding): 43%
Average TST time: 30%
Average time others: 26%

- Carmel *et al.* [10], a foundation for understanding FTS.
  Average RG time: 10%
  Average DEV time (Design + Coding): 55%
  Average TST time: 25%
  Average time others: 10%

Regarding quality it will account for each site probabilities of state *Bug Fix (BF)* divided *per* the probability of state *Development (DEV)*. Regarding time it will account for each site the sum of probabilities of all states. It is clear that there are more aspects to be discussed about time, cost and quality, but those are not considered for this scope of work. Regarding cost it will apply an average dollar rate hour value *per* site times the number of project hours.

Based on the sense that this research has some exploratory aspects, examples were created to calibrate and to evaluate the model due the lack of practical FTS cases available in literature, more details are available on research limitations session. In this sense average U$ value *per* hour was *ad hoc* set to exercise the model due the lack of literature with detailed information about real costs, more details available on research limitations session. The cost calculation is given by the sum of the average time spend on each model state in each site, times the average dollar cost per hour for a given site. The average dollar cost per country was ad hoc set due to the lack of available data in the literature.

This research uses heuristics to split different amount of hours on the scenarios creation, those heuristics basically divide a given number of hours for the activities performed by each site such as: requirements gathering, development, testing, bug fix and handoff. The *performance evaluation* chapter will provide the details about the heuristics use and the scenarios creation.

Besides previous mentioned references the model was also calibrated with the following information from literature [9, 16, 17, 34, 43, 45, 51, 61, 64]. It is important to remark that the model abstraction level can be easily changed in order to allow the model to handle four or more sites.Three sites is a suggested parameterization for a FTS performance evaluation covering 24 hours of work. Next chapter demonstrates the model instantiation, performance evaluation and numerical results.

5. Performance Evaluation

There are different practices to measure software development projects and it is difficult to have an agreement about what should be measured and how to analyze measures results. In this sense, the SAN model and performance evaluation can be used as a tool to enhance Follow-The-Sun projects performance control.

Software metrics can help project management planning, in this phase it is possible to identify the amount of required effort and an estimated cost to complete a certain project. In software development projects, there are costs of materials, servers and others, but human resources are usually the main cost factor. Another aspect is how to manage project time, in order to ensure that a project will achieve project goals. Besides quality is a project aspect to be considered. As mentioned in previous sections, this model will focus on scenarios to measure time, cost and quality for Follow-The-Sun software development projects.

5.1 Running The Model

To illustrate and to exercise the functionality and the usefulness of the Follow-The-Sun model, it was created a set of three example projects. Due to the fact of been using stochastic automata networks for performance evaluation, the values generated by the model execution are by nature stochastically precise.

5.1.1 Samples Settings

These examples follow the model defined phases: *Requirements Gathering (RG)*, *Design and Coding (DEV)*, *Testing (TST)* and *Bug fix / Rework (BF)*. Besides this model consider *Start Handoff from previous shift (SH)*, *Finish handoff to the next shift (FH)* and *When the site is off line (OFF)*. Examples are based on *UTC+12-12* and the average time used on each model event rate is based on the following literature references Javed *et al.* [35], Basili *et al.* [3], Carmel *et al.* [10].

5.1.2 Sample One

This project has three development sites: *New Zealand (UTC+12), Russia (UTC+4) and Bolivia (UTC-4)* with the following respective average hour dollar rate 22.5, 30.00 and 20.00. The average dollar rate is ad hoc set due to the lack of cost information per country from the literature. Figure 5.1 present this sample.

It considers a day of 24 hours of work divided by three sites. Assumptions and basis for calculations:

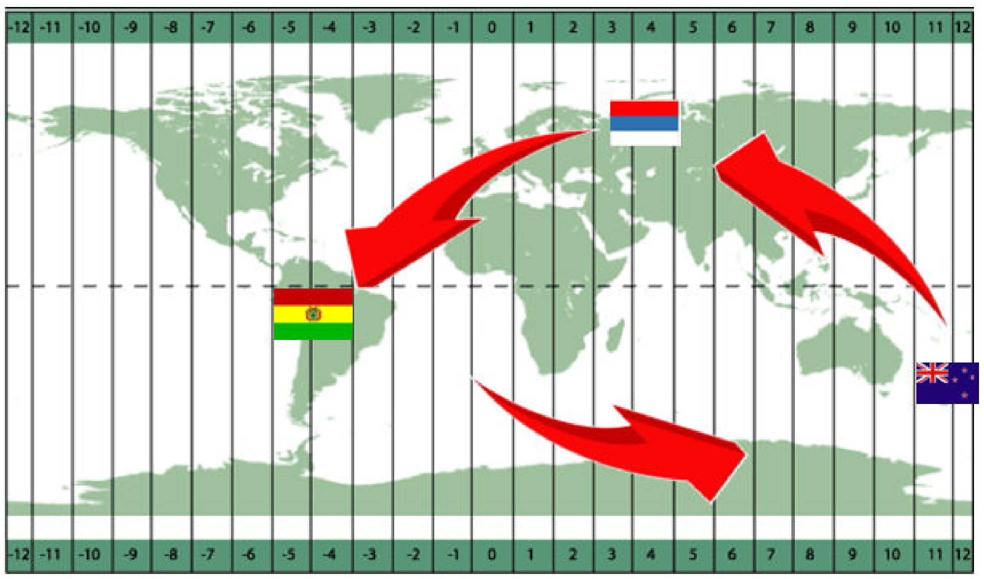

Figure 5.1: Sample One

- Work days *per* week: 5;
- Work hours *per* day: 8;
- Non project work: 8 hours *per* month. These are regular activities, time away, staff meetings;
- Project activities considers all tasks related to software development including coordination time and project meetings;

Each sample scenario was created to ilustraste the use of model instances. This example instance used the estimates described in Table 5.1.

Table 5.1: Sample One Project Estimates

State	Hours	Avg. cost
Requirements Gathering	1,000.00	24,134.80
Development (Design + Coding)	2,500.00	60,791.63
Testing	1,200.00	28,969.40
Bug Fix and Rework	600.00	14,482.24
Operational Handoffs	150.00	3,330.27
Total	5,450 hours	U$ 131,708.33

For each model instance, heuristics were build to create the estimation scenario, in this sense this model instance estimates are based on the following heuristics:

- Requirements gathering = 18.3% of a work day.
- Development = 45.84% of a work day

- Testing = 22.11% of a work day
- Bug fix = 11% of a work day
- Handoff = 2.75% of a work day.

Sample One Performance evaluation estimates

- Expected duration in months: 4
- Average number of hours *per* month: 1,362.50 (5,450 hours ÷ 4 months)
- Average number of hours *per* day: 61.93 (1,362.50 hours ÷ 22 workdays)
- Average number of hours *per* site *per* day: 20.64 (61.93 hours ÷ 3 sites)
- Average number of resources need *per* site: 2.58 (20.64 hours ÷ 8 hours)
- Average monthly cost: 32,927.08 (131,708.33 ÷ 4 months)
- Quality: 23.99% ((6.82 BF hours daily (11.01%) ÷ 28.41 DEV hours daily (45.87%) × 100)

According to the sample project estimates, the main idea is to assign durations to every state in the model, *i.e.*, frequencies at every connection among states.
The parametrization completes the model, which can now be subjected to specialized tools for numerical solution (in this case, SAN lite Solver [54]) in order to extract performance indices and measures.

Table 5.2 demonstrates this sample parameterization (Average time probability in minutes are based on a day of 24 hours (*i.e.*, 1,440 minutes). Considering each site working 8 hours *per* day.).

Table 5.2: Sample One Parameters

State	New Zealand	Russia	Bolivia
OFF	960.00 min	960.00 min	960.00 min
SH	8.00 min	3.82 min	8.00 min
RG	85.00 min	88.00 min	91.22 min
DEV	220.55 min	230.00 min	210.00 min
TST	105.00 min	105.06 min	107.00 min
BF	53.88 min	52.12 min	52.53 min
FH	7.57 min	1.00 min	11.25 min
Total	1,440 min	1,440 min	1,440 min

After the model execution the resultant probabilities rates are multiplied back by the base parameters value for a day of work, in this sense each probability output is multiplied by 1440 minutes representing a day of 24 hours of work. The model was solved using an specialized tool SAN lite Solver [54]. A model solved means that it was found a steady-state solution.

Sample One Performance evaluation results

- Duration in months: 2.70 (5,450 hours ÷ 2,015.99 hours per month)
- Average number of hours *per* month: 2,015.99 (91.64 hours per day × 22 days per month)
- Average number of hours *per* day: 91.64 (30.55 hours per site per day × 3 sites)
- Average number of hours *per* site *per* day: 30.55 (139.64 hours of work per day ÷ 3 sites)
- Average number of resources need *per* site: 3.82 (4 resources working on average 95% of the time on project activities)
- Average monthly cost: 48,719.67 (([New Zealand daily cost 687.27 (30.55 hours × U$ 22.5)] + [Russia daily cost 916.36 (30.55 hours × U$ 30.00)] + [Bolivia daily cost 610.90 (30.55 × U$ 20.00)]) × 22 days per month)
- Quality: 24% ((6.83 BF hours daily (7.45%) ÷ 28.41 DEV hours daily (31.01%)) × 100)

The rate for the OFF model state means that the site will be 960 minutes (16 hours) on off line state. After the model execution the probability of a site stay on this state was 66.67% (16 hours from 24 hours). The probabilities of other states also resulted on some variation based on the average estimated time due to the model events synchronization and due to the model dynamics. However, these are small variations, confirming that the model parameterization is correct. E.g.: The site New Zealand had an estimation of 8 daily minutes to start handoff (SH) and the model execution found 7.87 daily minutes on this state.

Table 5.3 demonstrates this scenario sample average results for a day of work[1].
This first sample performance evaluation generated the following conclusions:

- With four team members this project could be executed using 67% of estimated project time.
- The monthly budget would increase 15,792.59 dollars.
- The estimated quality result and the performance quality result match based on the model parameters set (24% of rework).

Table 5.3: Sample One States Probabilities

Site	State	Time (min)	Probability
New Zealand	OFF	960.00	66.67%
	SH	7.87	0.55%
	RG	85.26	5.92%
	DEV	220.41	15.31%
	TST	105.03	7.29%
	BF	53.86	3.74%
	FH	7.57	0.53%
Russia	OFF	960.00	66.67%
	SH	3.79	0.26%
	RG	87.89	6.10%
	DEV	229.78	15.96%
	TST	105.25	7.31%
	BF	52.29	3.63%
	FH	1.00	0.07%
Bolivia	OFF	960.00	66.67%
	SH	7.87	0.55%
	RG	91.18	6.33%
	DEV	210.60	14.63%
	TST	106.92	7.42%
	BF	52.44	3.64%
	FH	11.00	0.76%

5.1.3 Sample Two

This sample has explored the increase of sites working *per* day. Even with focus on three sites, it was decided to create one sample exploring four sites working six hours each on a total of 24 hours work day. The average dollar rate is ad hoc set due to the lack of cost information per country from the literature. Figure 5.2 present this sample.

It considers a day of 24 hours of work divided by three sites. Assumptions and basis for calculations:

This project has four development sites: *New Zealand (UTC+12), Russia (UTC+6) , UK (UTC 0) and USA (UTC-6)* with the following respective average hour dollar rate 22.5, 30.00, 60.00 and 55.00. We are considering a day of 24 hours of work divided between the four sites. This sample assume that four team members are working on each shift. Assumptions and basis for calculations:

- Work days *per* week: 5;

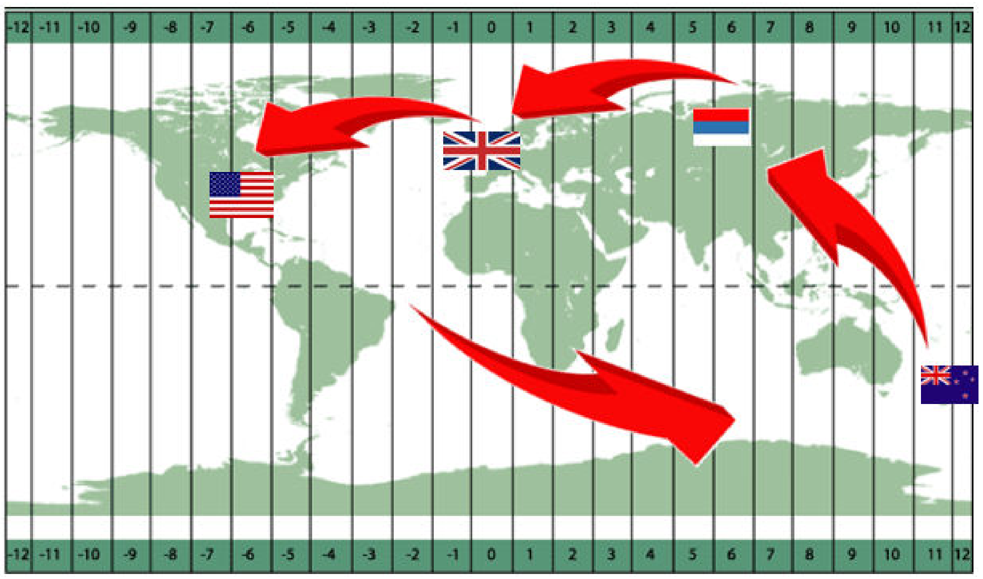

Figure 5.2: Sample Two

- Work hours *per* day: 6;
- Non project work: 6 hours *per* month. These are regular activities, time away, staff meetings;
- Project activities considers all tasks related to software development including coordination time and project meetings;

The following scenario estimates described in Table 5.4 will be applied for this sample.

Table 5.4: Sample Two Project Estimates

State	Hours	Avg. cost
Requirements Gathering	690.00	47,148.15
Development (Design + Coding)	3,430.00	188,592.59
Testing	690.00	47,148.15
Bug Fix and Rework	600.00	38,478.52
Operational Handoffs	670.00	18,859.26
Total	6,080 hours	U$ 340,226.67

For each model instance, heuristics were build to create the estimation scenario, in this sense this model instance estimates are based on the following heuristics:

- Requirements gathering = 11.35% work day
- Development = 56.42% of a work day
- Testing = 11.35% of a work day

- Bug fix = 9.57% of a work day
- Handoff = 11.31% of a work day.

Sample Two Performance evaluation estimates

- Expected duration in months: 4
- Average number of hours *per* month: 1,520 (6,080 hours ÷ 4)
- Average number of hours *per* day: 69.09 (1,520 ÷ 22 work days)
- Average number of hours *per* site *per* day: 17.27 (69.09 ÷ 4 sites)
- Average number of resources need *per* site: 2.87 (17.27 hours ÷ 8 hours)
- Average monthly cost: 85,056.66 (U$ 340,226.67 ÷ 4 months)
- Quality: 20% ((7.68 BF hours daily (9.87%) ÷ 38.38 DEV hours daily (56.41%)) × 100)

According to the sample project estimates, the main idea is to assign durations to every state in the model, *i.e.*, frequencies at every connection among states.
The parametrization completes the model, which can now be subjected to specialized tools for numerical solution (in this case, SAN lite Solver [54]) in order to extract performance indices and measures.

Table 5.5 demonstrates this scenario parameterization(Average time probability in minutes, based on a day of 24 hours (*i.e.*, 1,440 minutes). Considering each site working 8 hours *per* day.).

Table 5.5: Sample Two Parameters

State	New Zealand	Russia	UK	USA
OFF	1,080.00 min	1,080.00 min	1,080.00 min	1,080.00 min
SH	10.00 min	10.00 min	10.00 min	10.00 min
RG	50.00 min	50.00 min	50.00 min	50.00 min
DEV	200.00 min	200.00 min	200.00 min	200.00 min
TST	50.00 min	50.00 min	50.00 min	50.00 min
BF	40.00 min	40.00 min	40.00 min	40.00 min
FH	10.00 min	10.00 min	10.00 min	10.00 min
Total	1,440 min	1,440 min	1,440 min	1,440 min

After the model execution the resultant probabilities rates are multiplied back by the base parameters value for a day of work, in this sense each probability output is multiplied by 1440 minutes representing a day of 24 hours of work. The model was solved using an specialized tool SAN lite

Solver [54]. A model solved means that it was found a steady-state solution. It is important to remark that this second sample is based on four sites working six hours each, covering 24 hours however with a short work day of six hours instead of eight.

Sample Two Performance evaluation results

- Duration in months: 3.01 (6,080 hours $\div$ 2,016.02)
- Average number of hours *per* month: 2,016.02 (91.65 daily hours $\times$ 22 days per month)
- Average number of hours *per* day: 91.65 (22.91 hours per site per day $\times$ 4 sites)
- Average number of hours *per* site *per* day: 22.91 (91.65 hours of work per day $\div$ 4 sites)
- Average number of resources need *per* site: 3.82 (4 resources working on average 95% of the time on project activities)
- Average monthly cost: 84,420.86 (([New Zealand daily cost 515.46 (22.91 hours $\times$ U$ 22.50)] + [Russia daily cost 687.28 (22.91 hours $\times$ U$ 30.00)] + [UK daily cost 1,374.56 (22.91 hours $\times$ U$ 60)] + [USA daily cost 1,260.01 (22.91 hours $\times$ U$ 55.00)]) $\times$ 22 days per month)
- Quality: 20% ((10.17 BF hours daily (11.09%) $\div$ 50.87 DEV hours daily (55.50%)) $\times$ 100)

Table 5.6 demonstrates this scenario execution average results for a day of work[1].

The rate for the OFF model state means that the site will be 1080 minutes (18 hours) on off line state. After the model execution the probability of a site stay on this state was 75.00% (18 hours from 24 hours). The probabilities of other states also resulted on some variation based on the average estimated time due to the model events synchronization and due to the model dynamics. However, these are small variations, confirming that the model parameterization is correct. E.g.: The site UK had an estimation of 40 daily minutes on bug fix (BF) and the model execution found 39.97 daily minutes on this state.

This sample model demonstrated that the (OFF) state was 75% for each site which is tied to six hours of work on a day of 24 hours. Another aspect found is the project execution reduction and the cost reduction, presenting an indicative that more sites can reduce project cycle and project cost. An interesting fact is that using the same average number of resources on a three sites configuration model and on a four sites configuration model, it will result almost in the same average number of working hours *per* day, because even with four resources working in four different sites each site working capacity will be six hours a day instead of eight.

It provided an interesting conclusion that as more sites are working, the more costs compositions are available. It is possible to exercise different costs compositions. Off course there are studies that point a major effort on coordination [10, 66], however it is possible to discuss that the use of more sites and different costs composition possibilities may compensates an increasing cost on project coordination.

Table 5.6: Sample Two States Probabilities

Site	State	Time (min)	Probability
New Zealand	OFF	1,080.00	75.00%
	SH	9.99	0.69%
	RG	49.96	3.47%
	DEV	199.84	13.88%
	TST	49.96	3.47%
	BF	39.97	2.78%
	FH	10.29	0.71%
Russia	OFF	1,080.00	75.00%
	SH	9.99	0.69%
	RG	49.96	3.47%
	DEV	199.84	13.88%
	TST	49.96	3.47%
	BF	39.97	2.78%
	FH	10.29	0.71%
UK	OFF	1,080.00	75.00%
	SH	9.99	0.69%
	RG	49.96	3.47%
	DEV	199.84	13.88%
	TST	49.96	3.47%
	BF	39.97	2.78%
	FH	10.29	0.71%
USA	OFF	1,080.00	75.00%
	SH	9.99	0.69%
	RG	49.96	3.47%
	DEV	199.84	13.88%
	TST	49.96	3.47%
	BF	39.97	2.78%
	FH	10.29	0.71%

5.1.4 Sample Three

This last sample focus more on analysis of handoff and rework using different rates. This project has three development sites: *New Zealand (UTC+12), Russia (UTC+4) and Bolivia (UTC-4)* with the following respective average hour dollar rate 22.5, 30.00 and 20.00. The average dollar rate is ad hoc set due to the lack of cost information per country from the literature. Figure 5.3 present this sample.

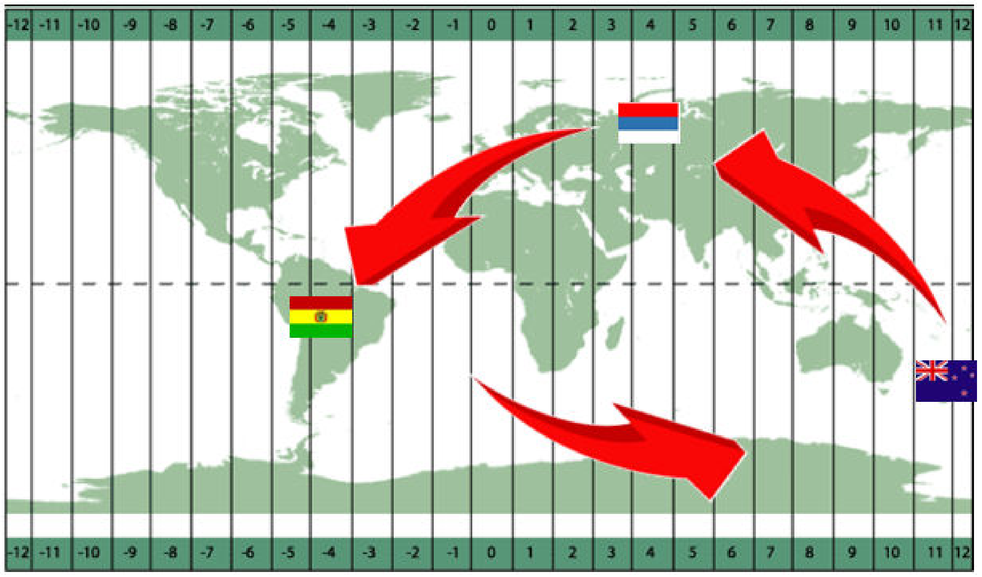

Figure 5.3: Sample Three

This last example considers a day of 24 hours of work divided between the three sites. This sample assumes that two team members will be working on each shift. Assumptions and basis for calculations:

- Work days *per* week: 5;
- Work ours *per* day: 8;
- Non project work: 8 hours *per* month. These are regular activities, time away, staff meetings;
- Project activities considers all tasks related to software development including coordination time and project meetings;

The following scenario estimates are described in Table 5.7 will be applied for this sample.

Table 5.7: Sample Three Project Estimates

State	Hours	Avg. cost
Requirements Gathering	800.00	15,028.82
Development (Design + Coding)	1,800.00	35,089.84
Testing	1,200.00	22,446.40
Bug Fix and Rework	900.00	24,972.16
Operational Handoffs	1,200.00	45,026.12
Total	5,900 hours	U$ 142,583.33

For each model instance, heuristics were build to create the estimation scenario, in this sense this model instance estimates are based on the following heuristics:

- Requirements gathering = 13.56% of a work day.
- Development = 30.51% of a work day
- Testing = 20.34% of a work day
- Bug fix = 15.25% of a work day
- Handoff = 20.34% of a work day.

Sample Three Performance evaluation estimates

- Expected duration in months: 2
- Average number of hours *per* month: 2,950 (5,900 hours /2)
- Average number of hours *per* day: 134.09 (2,950 ÷ 22 workdays)
- Average number of hours *per* site *per* day: 44.70 (134.09 hours ÷ 3 sites)
- Average number of resources need *per* site: 5.59 (44.70 hours ÷ 8 hours)
- Average monthly cost: 71,291.66 (U$ 142,583.33 ÷ 2 months)
- Quality: 71.42% ((23.52 BF hours daily (17.54%) ÷ 32.93 DEV hours daily (24.55%)) × 100)

Table 5.8 demonstrates this scenario parameterization (Average time probability in minutes, based on a day of 24 hours (*i.e.*, 1,440 minutes). Considering each site working 8 hours *per* day.).

Table 5.8: Sample Three Parameters

State	New Zealand	Russia	Bolivia
OFF	960.00 min	960.00 min	960.00 min
SH	97.00 min	55.37 min	75.00 min
RG	50.58 min	51.00 min	50.00 min
DEV	116.68 min	119.90 min	117.10 min
TST	74.37 min	75.00 min	78.00 min
BF	86.00 min	82.73 min	83.90 min
FH	55.37 min	96.00 min	76.00 min
Total	1,440 min	1,440 min	1,440 min

According to the sample project estimates, the main idea is to assign durations to every state in the model, *i.e.*, frequencies at every connection among states.

The parametrization completes the model, which can now be subjected to specialized tools for numerical solution (in this case, SAN lite Solver [54]) in order to extract performance indices and measures.

After the model execution the resultant probabilities rates are multiplied back by the base parameters value for a day of work, in this sense each probability output is multiplied by 1440 minutes representing a day of 24 hours of work. The model was solved using an specialized tool SAN lite Solver [54]. A model solved means that it was found a steady-state solution.

Sample Three performance evaluation results

- Duration in months: 5.85 (5,900 hours ÷ 1,008.00)
- Average number of hours *per* month: 1,008.00 (45.82 daily hours × 22 days per month)
- Average number of hours *per* day: 45.82 (15.27 hours per site per day × 3 sites)
- Average number of hours *per* site *per* day: 15.27 (45.82 hours per day ÷ 3 sites)
- Average number of resources need *per* site: 1.90 (2 resources working on average 95% of the time on project activities)
- Average monthly cost: 24,359.89 (([New Zealand daily cost 343.63 (15.27 hours × U$ 22.50)] + [Russia daily cost 458.18 (15.27 hours × U$ 30.00)] + [Bolivia daily cost 305.45 (15.27 hours × U$ 20.00)]) × 22 days per month)
- Quality: 71.45% ((8.32 BF hours daily (18.15%) ÷ 11.65 DEV hours daily (25.42%)) × 100)

The rate for the OFF model state means that the site will be 960 minutes (16 hours) on off line state. After the model execution the probability of a site stay on this state was 66.67% (16 hours from 24 hours). The probabilities of other states also resulted on some variation based on the average estimated time due to the model events synchronization and due to the model dynamics. However, these are small variations, confirming that the model parameterization is correct. E.g.: The site Russia had an estimation of 75.00 daily minutes on bug fix (TST) and the model execution found 75.52 daily minutes on this state.

Table 5.9 demonstrates this scenario execution average results for a day of work[1].

This sample with three team members this project could be executed using an increasing of 33% of expected project time. Presented samples demonstrated that is possible to exercise different case scenarios in order to create a parameter for project execution on a real worst case. It provides insights for cost, quality and time evaluation.

Based on the literature review and on these three different model instances presented, it is possible to observe that the number of sites makes the difference on project execution time as well

Table 5.9: Sample Three States Probabilities

Site	State	Time (min)	Probability
New Zealand	OFF	960.00	66.67%
	SH	79.66	5.53%
	RG	53.26	3.70%
	DEV	122.96	8.54%
	TST	78.39	5.44%
	BF	90.36	6.27%
	FH	55.37	3.85%
Russia	OFF	960.00	66.67%
	SH	49.07	3.41%
	RG	51.83	3.60%
	DEV	121.88	8.46%
	TST	76.52	5.31%
	BF	84.70	5.88%
	FH	96.00	6.67%
Bolivia	OFF	960.00	66.67%
	SH	63.95	4.44%
	RG	51.82	3.60%
	DEV	121.25	8.42%
	TST	80.46	5.59%
	BF	86.52	6.01%
	FH	76.00	5.28%

as project costs composition. The proposed model allows to exercise different projects compositions depending on the number of team members allocated on each site and on the average resource cost on a given site.

Through provided samples it was observed indicatives that the use of stochastic automata networks can help the decision making process of follow-the-sun projects. It is important to remember that models can be changed to capture other dimensions of FTS projects that were not part of this work scope, these suggestions are available on conclusions and future work sections.

This chapter demonstrated performance evaluation on Follow-The-Sun projects with performance indices from a set of sample scenarios, varying probabilities rates of *RG - Requirements Gathering*, *DEV - Design and Coding*, *TST - Testing* and *BF - Bug fix / Rework* and Handoff (*SH - Start Handoff / FH - Finish Handoff*).

The examples previously provided are hypothetical samples to illustrate how the FTS model can be used to enhance projects planning. They were calibrated with literature information, previously presented as well as with assumptions from model options and constraints section. Since it cannot be created hypotheses based on available data, these results cannot be classified as findings but they are indicatives of how SAN can be used to improve Follow-The-Sun decision making process.

6. Conclusions

Through provided samples as part of numerical results analysis, it was observed indicatives that the use of stochastic automata networks can help the decision making process of follow-the-sun projects. Evidently, the stochastic modeling presented in this work is far from being conclusive, but provides insights and indicatives about how stochastic automata networks can be used for Follow-The-Sun projects.

On the modeling, within the reality abstraction to create a stochastic model, there was no previous data categorization prepared, what created difficulty to define model parameters values, however once the model was instantiated generating the output it was possible to categorize the information to be evaluated.

SAN is a powerful and comprehensive formalism that works for software engineering through the use of the PEPS software tool. PEPS is an efficient tool for resolution of linear equations systems derived from SAN models. For any project decision maker, the most important than the tool itself is the degree of confidence generated by the output results. This work has not focus on measuring the quality level of PEPS tool, instead it focus on the degree of credibility from the results generated by models executions. Besides, PEPS software tool user interface can be improved in order to be used by decision makers, currently this tool is not user friendly for decision makers professionals.

Another difficult found during this research is the lack of a methodology to simulate software engineering projects using SAN models. There is an opportunity as future work to create a methodology for such demand.

Finally future works can create a robust method to apply SAN on software engineering problems, and to enhance PEPS software tool to be more user friendly, specially for modeling activities.

6.1 Contributions

This work outlines how stochastic modeling can contribute to the decision making process on follow-the-sun projects, generating four contributions:

1. A practical case study from an IT company using multiple sites and different participant's expertise reporting finds about the use of SAN for performance evaluation of GSD teams in order to enhance the understanding of their performance on different scenarios, validating a theoretical SAN modeling effort against a real project scenario [26]. The first contribution of this practical case was the comparison of a theoretical modeling effort with a real project scenario to describe a complex environment of software development. Despite the numerous abstractions made in the modeling stage, the obtained numerical results demonstrated an accuracy when compared to actual project outcome. Such fact by itself justifies the initial assumption that analytical modeling, in the particular case modeled using SAN formalism, may be a worthy option to build teams in software development projects. Another contribution

of it was to numerically demonstrate how much a project success remains on the number of experienced (senior) participants. Not even with a near ideal situation of central team availability and quality, the problems brought by a large number of junior participants can be overcome.

2. A controlled experiment verifying if an adaptive methodology had more benefits than a prescriptive methodology for a FTS strategy [37]. Where the speed obtained by teams using an adaptive approach was higher than the speed obtained by teams using a prescriptive approach. The percentage obtained by adaptive teams was 30% higher than the speed obtained by prescriptive teams. However, when comparing the average accuracy of delivered tasks, prescriptive teams were better than adaptive teams. We observed this factor verifying the total accuracy points and the total number of maps delivered by each team. Results obtained showed that adaptive teams had 15% less quality than prescriptive teams. Based on the results found, this experiment suggested that adaptive approaches could perform better than prescriptive approaches in the context of FTS. This is also claimed by Carmel, Espinosa, and Dubinsky [10], and should be deeply investigated in the future, with experiments also executed in real software development settings.

3. A SAN model for FTS that could be extended and reused which was centered on the handoff-efficiency to evaluate the probabilities of each site being working, reworking or waiting for clarifications in a FTS project [16]. It provided insights about how to measure the impact of handoff efficiency based on time spent on work and rework activities by each site or team. It presented an abstraction of how to model development teams in Follow-The-Sun (FTS) context and we proposed a scenario that enables the investigation of overall FTS project performance [56]. This initial approach demonstrated how the combination of stochastic modeling and FTS could be mixed in order to produce important considerations to be used by decision makers, *i.e.*, solving the stochastic models it is possible to obtain evidence of bottlenecks in a given project before the project execution.

4. The future work opportunity to create a methodology for software engineering simulation using SAN, because currently there is no methodology available to simulate software engineering projects using stochastic automata networks. This is a future work methodology to map an end to end process for stochastic modeling and performance evaluation of generic software engineering projects using SAN formalism, to improve coordination and control activities of distributed software development projects.

6.2 Research Limitations

Follow-The-Sun (FTS) is not a common industry practice yet. FTS is this research focus, and this model can be used in any case where time zone difference is considered as an important aspect. This work has a goal of modeling contribution instead of practice industry contribution.

Jorgensen and Shepperd [36] conducted a study that reviewed software cost estimation papers published in journals and tried to support other software cost estimation researchers through a library of estimation classified papers however it has not provided an average cost value *per* hour and *per* country. It has been difficult to find public material about software development cost distributions in different countries. This work has been using ad-hoc cost rates to work on cost simulation and analysis.

Besides, this work has the following limitations:

1. It is difficult to find a set of three countries with eight hours time zone difference between them. In case found, it does not mean that those countries will have off shore culture and qualified staff to execute IT projects geographically distributed. In this context, this work presented three exercise examples.

2. Based on the stochastic model setup and parameters, the cost variation curve is similar to the time variation because it is a function of time in this model.

3. This model does not abstract shifts overlap. The time required to report progress at the end of a day, and to catch up at the beginning of the day (reporting time and catching-up time, respectively) depend on many factors, including for example, the complexity of the task and the opportunity for synchronous communication (*i.e.*, the length of the time overlap). It could have an option for synchronous handoff model, declaring teams overlap at the beginning/end of a shift to allow synchronous coordination.

4. Besides, it does not consider how factors such as different cultures, tasks complexity and workload impact on the performance evaluation. It assumes in the abstraction that the workload (tasks) are equally distributed among the members. Also, it is not considering different levels of complexity for each task, assuming that all the resources needed by a project are assigned at the start and readily available throughout its conclusion.

5. The creation of SAN models is not a trivial activity, it requires knowledge about SAN formalism to create text files to run on PEPS tool (Unix environment), and it is not something simple for a management level user. Also, the PEPS tool has a limitation regarding the number of automata in a SAN model, due the size of states space it can run beyond PEPS tool memory management capacity.

6.3 Future Work

The opportunity to create a methodology for software engineering simulation using SAN is an important research to be done as future work. This methodology can be divided as an end to end process with four stages. The first stage will be the project success key factor identification. The second stage will be the mapping of the project artifacts and documents obtained in the previous stage, converting these artifacts into SAN elements, creating: *automata, states, events,* *and parameters* required for a model. Stages three and four will serve respectively to describe model execution steps for all different performance evaluation scenarios and to the results analysis.

Possible model extensions and future works can be done from the motivation of creating a new modeling exercise for a performance modeling and evaluation toolkit to present a mathematical framework for FTS projects. This conceptual modeling workflow could instantiate different models depending on the configuration for handoff and the desired analysis (or evaluation) to be done, creating three models as follows:

- **Model 1**- Stochastic model with asynchronous handoff: used if the previous site is unable to interact with the next site;
- **Model 2**- Stochastic model with synchronous handoff: a synchronization task during handoff is required between sites, so the next site can resume working after issues clarification;
- **Model 3**- Stochastic model with synchronous handoff extension: an extension from the previous model with synchronization where clarification is required and it could be represented in the model as a new state.

This work focus on one instance of timezone gradation, the FTS one. However, Espinosa *et al.* [20] conducted an study about different timezone gradations. It is possible to create a stochastic model to evaluate different timezone gradations where FTS would be one instance of this model as an extreme timezone difference case. As future work users can extend the existing model to numerically analyze other FTS projects dimensions as well as the impact of time zone gradations on Global Software Development projects.

The motivation of a future stochastic modeling exercise is to explore performance modeling and evaluation process by creating an extensible model and to perform an experiment to compare the current model version against a new model version. The concern for modelers will be directed towards the parameterization of the model with their own measures and their extension of new event compositions.

This work presented how SAN can be used to enhance decision making on FTS projects. This work does not capture all important aspects of Follow-The-Sun projects such as different cultures, different languages, communication barriers and teams coordination challenges that also can be scope of future work.

Finally, it is important to remark that the model herein presented could be extended/modified and applied on analysis of FTS projects using different levels of abstraction.

Bibliography

[1] G. Avram. Knowledge Work Practices in Global Software Development. *Electronic Journal of Knowledge Management*, 5(4):347–356, 2007.

[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, Limerick, Ireland, 2009.

[3] V. Basili, M. Zelkowitz, F. McGarry, J. Page, S. Waligora, and R. Pajerski. Sel's software process improvement program. *Software, IEEE*, 12(6):83 –87, nov 1995.

[4] C. Bertolini, A. G. Farina, P. Fernandes, and F. M. Oliveira. Test Case Generation Using Stochastic Automata Networks: Quantitative Analysis. In *Proceedings of the 2nd International Conference on Software Engineering and Formal Methods (SEFM'04)*, pages 251–260, Beijing, China, 2004.

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

[6] L. Brenner, P. Fernandes, and A. Sales. MQNA - Markovian Queueing Networks Analyser. In *11th IEEE/ACM International Symposium on Modelling, Analysis and Simulation on Computer and Telecommunication Systems (MASCOTS'03)*, pages 194–199, Orlando, FL, USA, October 2003. IEEE Computer Society.

[7] 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.

[8] K. A. Buragga. An analytical model for requirements activities and software product quality. *Journal of Computational Methods in Science and Engineering*, 6:205–215, 2006.

[9] E. Carmel. *Global software teams: collaborating across borders and time zones*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 1999.

[10] E. Carmel, Y. Dubinsky, and J.A. Espinosa. Follow the sun software development: New perspectives, conceptual foundation, and exploratory field study. In *System Sciences, 2009. HICSS '09. 42nd Hawaii International Conference on*, pages 1–9, January 2009.

[11] E. Carmel, Y. Dubinsky, and J.A. Espinosa. Follow The Sun Workflow In Global Software Development. *Journal of Management Information Systems*, 27(1):17–38, 2010.

[12] T. Clear and S. G. MacDonell. Understanding technology use in global virtual teams: Research methodologies and methods. *Information and Software Technology*, 53(9):994 – 1011, 2011.

[13] 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*, 20:420–439, September 2009.

[14] R. M. Czekster. *Solução Numérica de Descritores Markovianos a partir de re-estruturações de termos tensoriais*. PhD thesis, Pontíficia Universidade Católica do Rio Grande do Sul, Porto Alegre, Brazil, 2010.

[15] R. M. Czekster, P. Fernandes, A. Sales, and T. Webber. Analytical Modeling of Software Development Teams in Globally Distributed Projects. In *International Conference on Global Software Engineering (ICGSE'10)*, pages 287–296, Princeton, NJ, USA, 2010. IEEE Computer Society.

[16] R.M. Czekster, P. Fernandes, R. Prikladnicki, A. Sales, A.R. Santos, and T. Webber. Follow-The-Sun Methodology in a Stochastic Modeling Perspective. In *6th IEEE International Conference on Global Software Engineering (ICGSE): Methods and Tools for Project/Architecture/Risk Management in Globally Distributed Software Development Projects (PARIS)*, pages 54–59, Helsinki, Finland, August 2011.

[17] G. A. Dafoulas, K. Swigger, R. Brazile, F. N. Alpaslan, V. L. Cabrera, and F. C. Serce. Global teams: Futuristic models of collaborative work for today's software development industry. *Hawaii International Conference on System Sciences*, 0:1–10, 2009.

[18] N. Denny, I. Crk, and R. Sheshu. Agile Software Processes for the 24-Hour Knowledge Factory Environment. *Journal of Information Technology Research (JITR)*, 1(1):57–71, 2008.

[19] G.R. Djavanshir. Surveying the risks and benefits of it outsourcing. *IT Professional*, 7(6):32 – 37, nov.-dec. 2005.

[20] J. A. Espinosa, N. Nan, and E. Carmel. Do gradations of time zone separation make a difference in performance? a first laboratory study. In *Proceedings of the International Conference on Global Software Engineering*, ICGSE '07, pages 12–22, Washington, DC, USA, 2007. IEEE Computer Society.

[21] J.A. Espinosa and E. Carmel. The impact of time separation on coordination in global software teams: a conceptual foundation. *Software Process: Improvement and Practice*, 8(4):249–266, 2003.

[22] J.A. Espinosa, J.N. Cummings, J.M. Wilson, and B.M. Pearce. Team boundary issues across multiple global firms. *Journal of Management Information Systems*, 19:157–190, April 2003.

[23] S. Faraj and L. Sproull. Coordinating expertise in software development teams. *Management Science*, 46(12):1554–1568, 2000.

[24] 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, Ischia, Italy, 2002.

[25] P. Fernandes, B. Plateau, and W. J. Stewart. Efficient descriptor-vector multiplications in stochastic automata networks. *Journal of the ACM (JACM)*, 45:381–414, May 1998.

[26] P. Fernandes, A. Sales, A.R. Santos, and T. Webber. Performance Evaluation of Software Development Teams: a Practical Case Study. *Electronic Notes in Theoretical Computer Science (ENTCS)*, 275:73–92, 2011.

[27] S. Ferreira, J. Collofello, D. Shunk, and G. Mackulak. Understanding the effects of requirements volatility in software engineering by using analytical modeling and software process simulation. *Journal of Systems and Software*, 82:1568–1577, October 2009.

[28] I. Gorton and S. Motwani. Issues in co-operative software engineering using globally distributed teams. *Information and Software Technology*, 38(10):647–655, 1996.

[29] A. Gupta, E. Mattarelli, S. Seshasai, and J. Broschak. Use of collaborative technologies and knowledge sharing in co-located and distributed teams: Towards the 24-h knowledge factory. *The Journal of Strategic Information Systems*, 18:147–161, September 2009.

[30] J. D. Herbsleb, A. Mockus, T. A. Finholt, and R. E. Grinter. Distance, dependencies, and delay in a global collaboration. In *Proceedings of the 2000 ACM conference on Computer Supported Cooperative Work (CSCW'00)*, pages 319–328, Philadelphia, PA, USA, 2000. ACM.

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

[32] J.E. Hopcroft and J.D. Ullman. *Introduction To Automata Theory, Languages, And Computation*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1st edition, 1990.

[33] D. X. Houston, G. T. Mackulak, and J. S. Collofello. Stochastic simulation of risk factor potential effects for software development risk management. *Journal of Systems and Software*, 59(3):247 – 257, 2001.

[34] P. Jalote and G. Jain. Assigning tasks in a 24-hour software development model. In *Asia-Pacific Software Engineering Conference*, pages 309–315, 2004.

[35] T. Javed, M. Maqsood, and Q.S. Durrani. A study to investigate the impact of requirements instability on software defects. *ACM SIGSOFT Software Engineering Notes*, 29:1–7, May 2004.

[36] M. Jorgensen and M. Shepperd. A systematic review of software development cost estimation studies. *Software Engineering, IEEE Transactions on*, 33(1):33 –53, jan. 2007.

[37] J. Kroll, A. R. Santos, R. Prikladnicki, E.R. Hess, R.A. Glanzner, A. Sales, J. L. N. Audy, and P. Fernandes. Follow-the-Sun Software Development: A Controlled Experiment to Evaluate the Benefits of Adaptive and Prescriptive Approaches. In *Proceedings of the 24th International Conference on Software Engineering and Knowledge Engineering (SEKE 2012)*, pages 551–556, San Francisco Bay, USA, 2012.

[38] P. Laurent, P. Mader, J. Cleland-Huang, and A. Steele. A taxonomy and visual notation for modeling globally distributed requirements engineering projects. In *Proceedings of the 2010 5th IEEE International Conference on Global Software Engineering*, ICGSE '10, pages 35–44, Washington, DC, USA, 2010. IEEE Computer Society.

[39] M.M. Lehman, Mm Lehman, and J.F. Ramil. The impact of feedback in the global software process. *Journal of Systems and Software*, 46:123–134, 1998.

[40] 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.

[41] T. Nguyen, T. Wolf, and D. Damian. Global software development and delay: Does distance still matter? In *Global Software Engineering, 2008. ICGSE 2008. IEEE International Conference on*, pages 45 –54, aug. 2008.

[42] B. Nicholson and S. Sahay. Embedded knowledge and offshore software development. *Information and Organization*, 14(4):329 – 365, 2004.

[43] M. B. O'Leary and J. N. Cummings. The spatial, temporal, and configurational characteristics of geographic dispersion in teams. *MIS Quarterly*, 2002.

[44] F. Padberg. A discrete simulation model for assessing software project scheduling policies. *Journal Software Process: Improvement and Practice (SPIP)*, 7:127–139, 2002.

[45] S. Patil, A. Kobsa, J. Ajita, and D. Seligmann. Methodological reflections on a field study of a globally distributed software project. *Information and Software Technology*, 53:969–980, September 2011.

[46] M. Pidd. *Computer Simulation in Management Science*. John Wiley & Sons, Inc., New York, NY, USA, 3rd edition, 1992.

[47] 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.

[48] B. Plateau and K. Atif. Stochastic automata network of modeling parallel systems. *IEEE Transactions on Software Engineering*, 17(10):1093 –1108, October 1991.

[49] R. Prikladnicki and J.L.N. Audy. Process models in the practice of distributed software development: A systematic review of the literature. *Information and Software Technology*, 52:779–791, August 2010.

[50] R. Prikladnicki, J.L.N. Audy, and F. Shull. Patterns in effective distributed software development. *IEEE Software*, 27(2):12–15, March-April 2010.

[51] D. Raffo and S. Setamanit. A simulation model for global software development project. *Development*, pages 1–7, 2005.

[52] J. Raj. *The Art of Computer Systems Performance Analysis - Techniques for Experimental Design, Measurement, Simulation, and Modeling*. John Wiley & Sons Inc, 1991.

[53] N. Ramasubbu, M. Cataldo, R.K. Balan, and J.D. Herbsleb. Configuring global software teams: a multi-company analysis of project productivity, quality, and profits. In *Software Engineering (ICSE), 2011 33rd International Conference on*, pages 261 –270, may 2011.

[54] A. Sales. San lite-solver: a user-friendly software tool to solve san models. In *Proceedings of the 2012 Symposium on Theory of Modeling and Simulation - DEVS Integrative M&S Symposium*, TMS/DEVS '12, pages 44:1–44:8, San Diego, CA, USA, 2012. Society for Computer Simulation International.

[55] R. Sangwan, M. Bass, N. Mullick, D.J. Paulish, and J. Kazmeier. *Global Software Development Handbook (Auerbach Series on Applied Software Engineering Series)*. Auerbach Publications, Boston, MA, USA, 2006.

[56] A.R. Santos, A. Sales, and P. Fernandes. Setting up a stochastic model for teams working in a follow-the-sun environment. In *Proceedings of the 2012 7th IEEE International Conference on Global Software Engineering*, ICGSE '12, pages 179–179. IEEE Computer Society, 2012.

[57] S. Sarker and S. Sahay. Understanding Virtual Team Development: An Interpretive Study. *Journal of the Association for Information Systems*, 3:247–285, 2002.

[58] F. C. Serçe, K. Swigger, F. N. Alpaslan, R. Brazile, G. Dafoulas, and V. Lopez. Online collaboration: Collaborative behavior patterns and factors affecting globally distributed team performance. *Computers in Human Behavior*, 27(1):490 – 503, 2011.

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

[60] N. S. Shami, N. Bos, Z. Wright, S. Hoch, K. Y. Kuan, J. Olson, and G. Olson. An experimental simulation of multi-site software development. In *Proceedings of the 2004 conference of the Centre for Advanced Studies on Collaborative research*, pages 255–266. IBM Press, 2004.

[61] P. Sooraj and P. K. J. Mohapatra. Modeling the 24-hour software development process. *Strategic Outsourcing: An International Journal*, 1(2):122–141, 2008.

[62] W. J. Stewart. *Introduction to the numerical solution of Markov chains*. Princeton University Press, 1994.

[63] A. Taweel and P. Brereton. Modelling software development across time zones. *Information and Software Technology*, 48(1):1 – 11, 2006.

[64] J. J. Treinen and S. L. Miller-Frost. Following the sun: case studies in global software development. *IBM Systems Journal*, 45:773–783, October 2006.

[65] R. Urdangarin, P. Fernandes, A. Avritzer, and D. Paulish. Experiences with Agile Practices in the Global Studio Project. In *Proceedings of the IEEE International Conference on Global Software Engineering (ICGSE 2008)*, pages 77–86. IEEE Computer Society, August 2008.

[66] R. van Solingen and M. Valkema. The impact of number of sites in a follow the sun setting on the actual and perceived working speed and accuracy: A controlled experiment. In *Global Software Engineering (ICGSE), 2010 5th IEEE International Conference on*, pages 165 –174, aug. 2010.

[67] C. Visser and R. Van Solingen. Selecting Locations for Follow-the-Sun Software Development: Towards a Routing Model. *2009 Fourth IEEE International Conference on Global Software Engineering*, pages 185–194, July 2009.

[68] R.S. Wazlawick. *Metodologia de Pesquisa para Ciência da Computação)*. Elsevier Editora Ltda, São Paulo, SP, BR, 2009.

[69] M. Yap. Follow the Sun: Distributed Extreme Programming Development. In *Proceedings of the Agile Development Conference (ADC '05)*, pages 218–224, Denver, CO, USA, 2005. IEEE Computer Society.

[70] H. Zhang and B. Kitchenham. Semi-quantitative Simulation Modeling of Software Engineering Process. In *Proceedings of the International Software Process Workshop and International Workshop on Software Process Simulation and Modeling*, volume 3966, pages 242–253, Shanghai, China, 2006. Springer-Verlag Berlin Heidelberg.