A structured stochastic model for software project estimation in Waterfall models

Ildo Massitela *, Joaquim Assunção †, Alan R. Santos *, Paulo Fernandes *

* PUCRS, School of Technology, Porto Alegre, Brazil
† UFSM - Department of Applied Computing - Santa Maria, Brazil
* {ildo.massitela, alan.ricardo}@acad.pucrs.br, † joaquim@inf.ufsm.br, *paulo.fernandes@pucrs.br

Abstract

Evaluate team's performance on long-duration projects can be a challenge. It relies on human expertise to perform estimations, which can generate uncertainty and dependency of experienced specialists. Statistical techniques, such as modeling and simulations, are suitable options as tools to support projects estimations. Our goal in this paper is a formal mapping of the main Waterfall model characteristics to stochastic model to predict performance indices of teams such as the real working time.

1 Introduction

Simulations of teams' performance can help in solving a series of issues concerning software development, yet estimations and analysis for non-short projects can be a challenge [8]. Considering all the project phases, from extracting requirements to deliver and maintain a product, involves careful planning and resources management. Thus, estimations for a project, play an important role in any medium or big project [7, 10].

Traditional software development methodologies, also known as prescriptive approaches (such as Waterfall), claim their support to comprehensive planning, detailed documentation, and expansive design. Agile approaches have gained significant attention from the software engineering community in the last few years. Unlike traditional methods, Agile methodologies employ short iterative cycles and rely on tacit knowledge within a team as opposed to documentation [1]. In this context, traditional software development approaches will still have their need in large, long-lived projects that have particular safety, reliability or security requirements [1]. Also, Waterfall projects tend to have much longer releases than Agile methods, which makes the estimation of resources even more important.

Performance evaluation for software development projects has become a challenge for both industry and academia. Theoretical models can be adopted as a tool to analyze and to understand different dynamics on software development process to help project leaders to perform a better assessment on issues related to the development context [6, 11, 7].

Characteristics, such as mixed teams, different levels of expertise, different knowledge about the project *etc.*, make it difficult for accurate predictions [13, 5, 3, 12]. In such cases, we may rely on stochastic techniques to represent such apparently random situation. More specifically, a model that uses known data as parameters and known behavior as its structure. Such models should be able to simulate a team's activity, laying on probabilities to estimate its performance.

Although our model does not intend to be a complete solution, we can well cover some of these key points by achieving probabilities through a structured stochastic model. Our model relies on stochastic methods based on a structured formalism, which allows us to perform simulations efficiently, adapt its parameters and scales the model itself depending on the number of team members. This compact and adaptable model can be used both to estimate and evaluate teams' performance. Moreover, we demonstrate the benefits of using the Stochastic Automata Networks (SAN) formalism for the modeling and evaluation of Waterfall development teams.

2 Estimation through stochastic models

A stochastic model is a structure which represents a dynamic, and non-deterministic, system or phenomenon through the use of statistical techniques. In our case, a structured modular model, which allows us to manipulate and update the model for different scenarios easily. Such models have a low cost to gain insights compared to the cost, risk or logistics of manipulating a real system.

Our model relies on the stochastic description of the events. With a structure representing a given team, we use parameters caring information concerning the time effectively spent by Junior, Standard, and Senior members. SAN is a structured formalism which describes a complete system as a collection of subsystems that interact with each other [9]. Its behavior is similar to a Markov formalism, except that it is internally represented by a Kronecker descriptor and it allows modular representations, which can be especially useful to create compact representations. SAN models can be solved using specialized tools such as PEPS [2].

*DOI reference number: 10.18293/SEKE2018-033

3 The Model

Our model parameters data was collected through interviews with a project manager at an IT company, a Waterfall project, here in this work, named as project X. We used the data collected to feed our model parameters. We use the expression team members to describe the roles of developers and testers.

The model is composed of n automata representing n team members; also, we use an abstraction of five different states to describe the team, Working (W), Seeking solution (S), Collaborating (C), Helping others (H) and Reworking (R). Table 1 describes each state.

Table 1: Model states

State	Description
(W)orking	Execution of a given task by each member.
(S)eeking solution	A team member trying to find a solution.
(C)ollaborating	A synchronous contribution between 2 team members, a member receive help or a mutual collaboration is established.
(H)elping	A synchronous contribution with benefit only for another member, j.
(R)eworking	A given team member reworking to solve a problem in a given task.

We assume that a team member can be working (W) and then stop due to the need of information or some necessary condition for a task, we call this state *Seeking solution* (S). Yet, when this team member is not working, sometimes he/she is available to help others, be helped or exchange knowledge about the problem; thus, the states *Collaborating* and *Helping* are used to represent synchronous task between two team members.

Helping (H) is a state that is achieved only when a member i is helping another member without any benefit for its own problem. Collaborating (C) is a state that represents a member i being helped by another team member. When an H activity ends, a member returns to seek a solution for its problem. A C activity ends into two different possibilities, which are: 1, a team member i can resume working on a new task (going to W). 2, a team member i must return to fix an already started task (R), *i.e.*, to rework on a known problem.

For each automaton, five states are linked by a transition associated with events that fire according to specific rates. Those rates are assigned according to flexible parameters based on the average working hours achieved, described by the members of project X. For instance, as the event a is related to an average rate indicating a member that stops working to seek a solution. A frequency of once per day is represented as $1/9$, six times per day as $6/9$, and so on (Table 2). The event q is the opposite and has different taxes due to the fact that a member can stop to seek a solution for its problems or to help others.

Table 2: Collected average working hours per team member

Event	Expertise	Rate
a_i	Senior	$\tau_{a_i} = 1/9$
	Plain	$\tau_{a_i} = 3/9$
	Junior	$\tau_{a_i} = 6/9$
q_i	Senior	$\tau_{q_i} = 6/9$
	Plain	$\tau_{q_i} = 4/9$
	Junior	$\tau_{q_i} = 2/9$
r_i	Senior	$\tau_{r_i} = 1/9$
	Plain	$\tau_{r_i} = 1/9$
	Junior	$\tau_{r_i} = 1/9$

We evaluated the model using the project X data to set the parameters, however the model parameters can be adjusted and assigned according to different scenarios. For instance, a Junior developer seeks a solution or support more often than a Senior developer, and the combination of professionals with different levels of expertise can be taken into account to accurately describe each specific project behavior. A team, formed by n members, is formed by a joint network of n automata, each as illustrated on Figure 1. The events that trigger transitions among the states are presented in Table 3.

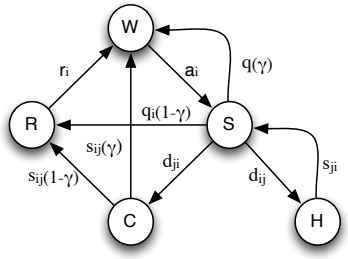

Figure 1: Automaton $A^{(i)}$ describing possible activities of the i^{th} member in a team of n members, $0 > i, j \leq n \mid i \neq j$.

Each automaton is described by a set of states (W, S, C, H, R) and a set of events associated with its transitions. For the analysis task, we focus on the state W to get probabilities concerning the estimated useful working hours. More clearly, we consider the probability of a state W as the productivity of each team member.

In this sense, to determine the rate of the synchronizing events, we started with the following functions, which is given by the expertise of a member, in conjunction with the level of expertise of the other member seeking collaboration (Table 5). So, the better a member is, the fewer this member will be available at state S. However, once in S, the more likely this member will help the others (Event d_{ij}). Also, these members tend to solve problems faster, *i.e.*, they leave the collaboration state faster than those with lower expertise

Table 3: Model events

Event	Description	Rate
a_i	Team member i is stop working to seek for the solution for a problem.	τ_{a_i}
q_i	Team member i has found a solution by itself and is either resume working (with probability γ), since its problem was not an issue, or he/she is going to re-work (with probability $1-\gamma$) to fix his/her problem.	τ_{q_i}
d_{ij}	Team member i is either going to help team member j (d_{ij}), or he/she is going to be helped by team member j (d_{ji}).	$\tau_{d_{ij}}$
s_{ij}	A collaboration between team members i and j is ended and, since i is being helped by j, team member i is going to work or re-work in his/her problem (s_{ij}), while team member j is going back to seek his/her own solution. Probability $1-\gamma$ defines if member i will need to re-work.	$\tau_{s_{ij}}$
r_i	A team member i going to work after re-work on a task.	τ_{r_i}

(Event s_{ij}). We first use temporary variables (β) to find the rate set for these events:

$$\beta(s_{ij}) = \tau_{q_i} + (1-\tau_{q_j}) \;\; ... \;\; \beta(s_{ji}) = \tau_{q_j} + (1-\tau_{q_i})$$
$$\beta(d_{ij}) = \tau_{q_i} + (1-\tau_{q_j}) \;\; ... \;\; \beta(d_{ji}) = \tau_{q_j} + (1-\tau_{q_i})$$

These temporary variables are needed due to the expertise level's dependency on each team member pair of team member concerned by each synchronous event. Imagine two members, one that has half an hour per day to cooperate and the other has three hours. Between these two members, the cooperation time will be just half hour, which is the available time concerning the busier member. Nevertheless, considering another member of the team, the odds of productive collaboration among members increases as the number of possible member pairs to collaborate increases. Therefore, we use the cumulative sum of all β rates for starting these events. Now, concerning a team with n members, and a member represented by an automaton $A^{(i)}$ with $0 > i, j < n \mid i \neq j$, the final values for the rates of synchronizing events d and s are given by a cumulative sum of the β values:

$$\tau_{d_{ij}} = \sum_{\forall j \mid j \neq i} \beta(d_{ij}) \;\; ... \;\; \tau_{d_{ji}} = \sum_{\forall j \mid j \neq i} \beta(d_{ji})$$
$$\tau_{s_{ij}} = \sum_{\forall j \mid j \neq i} \beta(s_{ij}) \;\; ... \;\; \tau_{s_{ji}} = \sum_{\forall j \mid j \neq i} \beta(s_{ji})$$

Thus, the cumulative rate for all output events from S towards C or H is given by: $\sum_{\forall j \mid j \neq i} \tau_{d_{ji}}$. The same applies for the synchronizing events departing from C. The cumulative rate for events from C towards R and W is given by: $\sum_{\forall j \mid j \neq i} \tau_{s_{ij}}$. Note that from H there is only one possible transition that leads back to S. Once we have a team with n members, we can define a model composed by n automata. In such model, it is clear that each event with the member indication j must represent all team members, but itself (i). As for the probability to return to work (γ) or to re-work ($1-\gamma$), the more senior a member is the higher will be γ. For project X a Senior member has $\gamma = 0.67$, a Plain member has $\gamma = 0.44$ and a Junior member has $\gamma = 0.22$.

4 Results

Our model target is the state W since it represents when a team member is performing useful work. Thus, a total working time, Ω, is multiplied by the probability of the stationary probabilities of W. We previously have the real project time and the estimated time, which were given by the project members. Example, for the *Requirements* phase, the estimated total time was 179.7 hours and the real, observed, time was 192.8 hours. Our goal is to achieve a precision as good as the original estimations made by the senior members of the project. Thus, we calculate, for each phase and for each member, the stationary probabilities; which should retrieve a probability of a team member be effectively working. Each of these probabilities is then used to get the calculated working time of phase k (CWT_k). The current parameter makes a simple equation. Given:

- Ω, team members daily absolute working time;
- $P(W^{(i)})$, the calculated probability of the i^{th} team member be actually working;
- D, the total days in the project;
- n, the total number of team members.

$$CWT_k = \sum_{i=1}^{n} \Omega * P(W^{(i)}) * D$$

Our numerical analysis is obtained by the calculated working time previously described. This is performed for each project phase and for each team member. The model results retrieves the calculated probability for each member. The project phases are: *Requirements*, *Prototype*, *Specifications*, *Development* and *Deployment*.

Table 4: Probabilities achieved to the *Requirements* phase.

Analyst	State	Probability	Analyst	State	Probability
Senior	W	66.4089	Junior	W	20.9505
	S	10.6419		S	56.1932
	C	2.6546		C	17.9955
	H	17.9955		H	2.6546
	R	2.2988		R	2.2060

Considering Table 4, the total time spent to this phase is given by: $\Omega * P(W^{(i)}) * D$. Being Ω is a constant for the project, 9 hours; and D is a constant for each phase. In the

first phase, 21 days were spent. Thus, one Senior and one Junior Analyst is respectively:

$$\Omega * P(W^{(1)}) * D = 9 * 0.66 * 21 = 124.7$$
$$\Omega * P(W^{(2)}) * D = 9 * 0.2 * 21 = 37.8$$

The total time is the sum of all team members in the phase, which makes 162.5 hours. Table 5 shows our calculated estimations in hours compared with the actual time and the project manager estimation.

Table 5: Project X, estimated *versus* calculated differences.

Project Phases	Estimated	Actual	Calculated
Requirements	189	192.8	162.5
Prototype	63	61.5	61.1
Prog. specifications	144	138	168.4
Development	912	1069	1285.2
Documentation	171	213	213.7
Deployment	63	51	81.9

Despite a relatively poor performance for the deployment phase, we achieved a precision quite similar to the actual time, and sometimes better than the estimated values by the project manager. These results indicate that this type of approach can be used to support project leads and project managers to evaluate waterfall projects before their execution.

5 Final Remarks

Given specific project data, this model can be extended to more sites with different working hours, reworking probabilities and average time spent in activities. The modeler needs to understand more deeply the project and the participants' profile to collect significant data from interviews, surveys or historical data in companies databases. Indeed, we do not explore characteristics such as; the team expertise for a specific task, the historical performance for each phase, the complexity of the tasks and requirements, etc. Yet, these characteristics can be added to adjust the model according to different situations, therefore leading to constant quality and, possibly, improving the accuracy. Nonetheless, the main contribution of this work is a conceptual model for the analysis of effectively working time in Waterfall software development context.

Our model main parameter can be set according to the scenario and previous experiences concerning the targeted team. Despite limitations, our model is structured using Stochastic Automata Network (SAN), which allows new system compositions and behaviors by appending new states and relationships among previously defined entities. As a future work, we will identify characteristics in companies with different sizes and different project types for tests with our model and further research to improve it, which can lead to a more detailed model for various projects.

References

[1] M. AWAD, *A comparison between agile and traditional software development methodologies*, University of Western Australia, (2005).

[2] L. BRENNER, P. FERNANDES, B. PLATEAU, AND I. SBEITY, *PEPS 2007 - Stochastic Automata Networks Software Tool*, in Proceedings of the Fourth International Conference on Quantitative Evaluation of Systems (QEST '07), Washington, DC, USA, September 2007, IEEE Computer Society, pp. 163–164.

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

[4] R. M. CZEKSTER, P. FERNANDES, AND T. WEBBER, *GTA express - A Software Package to Handle Kronecker Descriptors*, in Proceedings of the 6th International Conference on Quantitative Evaluation of SysTems (QEST 2009), Budapest, Hungary, September 2009, IEEE Computer Society, pp. 281–282.

[5] S. FARAJ AND L. SPROULL, *Coordinating expertise in software development teams*, Management Science, 46 (2000), pp. 1554–1568.

[6] 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 (2009), pp. 1568 – 1577. SI: YAU.

[7] M. I. KELLNER, R. J. MADACHY, AND D. M. RAFFO, *Software process simulation modeling: why? what? how?*, The Journal of Systems & Software, 46 (1999), pp. 91–105.

[8] A. MAYRHAUSER, *Experimental Software Engineering Issues: Critical Assessment and Future Directions: International Workshop Dagstuhl Castle, Germany, September 14–18, 1992 Proceedings*, Springer Berlin Heidelberg, Berlin, Heidelberg, 1993, ch. The role of simulation in software engineering experimentation, pp. 177–179.

[9] B. PLATEAU, *On the stochastic structure of parallelism and synchronization models for distributed algorithms*, SIGMETRICS Perform. Eval. Rev., 13 (1985), pp. 147–154.

[10] 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.

[11] S. SETAMANIT, W. WAKELAND, AND D. M. RAFFO, *Using simulation to evaluate global software development task allocation strategies: Research Sections*, Software Process: Improvement and Practice, 12 (2007), pp. 491–503.

[12] C. U. SMITH AND L. G. WILLIAMS, *Software performance engineering: a case study including performance comparison with design alternatives*, Software Engineering, IEEE Transactions on, 19 (1993), pp. 720–741.

[13] A. TAWEEL AND P. BRERETON, *Modelling software development across time zones*, Information and Software Technology, 48 (2006), pp. 1–11.