FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Lean Forecasting Software Project

Pedro Miranda

PREPARAÇÃO DA DISSERTAÇÃO

Mestrado Integrado em Engenharia Informática e Computação

Supervisor: João Carlos Pascoal Faria

Supervisor: Filipe Figueiredo Correia

February 6, 2020

Lean Forecasting Software Project

Pedro Miranda

Mestrado Integrado em Engenharia Informática e Computação

February 6, 2020

Abstract

When developing a software project, it's recognisable that accurate estimations of development effort play a important part in the successful management of the project. Although this process is so important, developers and experts can't usually estimate accurately the effort, time and cost of a project to be developed. This is inherit to the uncertainty that underlies their activity. After the first estimation of the effort, the project may, with some likelihood, need to adapt to evolving circumstances, which may lead to changes in its scope, and consequently lead to managers putting pressure in the developers to respect delivery dates. In the end, the project's development will, probably, get delayed and this delays not only affect the development team but also other parts of the company, such as staffing or marketing. This could, in some situations, lead to the company losing time and in many times the trust of the stakeholder.

Even if the estimate is accurate enough so that delivery dates are respected, methods that relay on Human estimation are, often, time consuming, what can represent a problem when teams waste precious time in making estimations.

In order to mitigate this problems, we will seek to identify the motivations and forces playing in a accurate estimate and determine which forecast method could provide the better accuracy with some generalization, in order to satisfy the existing variety of software projects. We will focus on forecast methods because of their automatability, that will help reduce the time teams waste on estimations, still delivering accurate results. This method must also be easy to understand, implement and use, so the number of inputs required and the difficulty to collect this inputs should be low. The output of the method should contain a certain level of uncertainty, in order to better represent the problem. In order to validate this method, a tool based on it will be developed, tested in terms of effectiveness and accuracy against other existing methods, and it will be integrated with software development management tools to validate it's ability to be used in real projects during their development phase. Following this lines, the main goal of this dissertation is to help reduce the time wasted in estimations, while maintaining or even increase the accuracy of the prediction made and maintaining the understandability and usability easy for the teams and developers using it.

Keywords: Estimation, Forecasting, Software Development

Resumo

Quando se desenvolve um projeto de software, é reconhecível que estimativas precisas do esforço envolvido no desenvolvimento são uma parte importante na gestão bem-sucedida do projeto. Embora este processo seja tão importante, desenvolvedores e especialistas não conseguem normalmente estimar precisamente o esforço, tempo e custo que o projeto a ser desenvolvido terá. Isto é inerente à incerteza subjacente à sua atividade. Depois da primeira estimativa do esforço ser feita, o projeto pode, com alguma probabilidade, necessitar de se adaptar a circunstâncias em evolução, o que pode levar a mudanças nas características do projeto, e subsequentemente levar a que os gestores ponham mais pressão nos desenvolvedores para que sejam respeitados os prazos de entraga. No fim, o desenvolvimento do projeto irá, provavelmente, atrasar-se e estes atrasos não só afetam a equipa de desenvolvimento, mas também outras partes da empresa, como os departamentos responsaveis pelos funcionários e pelo marketing. Isto pode, em algumas situações, levar a que a empresa perca tempo e muitas vezes a confiança do cliente interessado no projeto.

Mesmo que a estimativa seja precisa o suficiente para que as datas de entrega sejam respeitadas, métodos que dependem das estimativas de humanos consomem, normalmente, muito tempo, o que pode representar um problema quando equipas gastam tempo precioso a fazer estimativas.

De maneira a mitigar estes problemas, iremos procurar identificar as motivações e forças em jogo no processo de fazer estimativas precisas e determinar que métodos de previsão alcançam os resultados mais precisos com alguma generalização, de modo a satisfazer a variedade de projetos de software existente. Vamos nos focar nos métodos de previsão devido à sua automaticidade, que irá ajudar a reduzir o tempo que as equipas gastam em estimações, mantendo a precisão dos resultados. Este método deve, também, ser fácil de perceber, implementar e usar, logo o número de dados que deve receber e a sua dificuldade de obter deve ser reduzida. As previsões do método devem conter um certo nível de ambiguidade, de modo a representar melhor o problema. Para a fase de validação do método, uma ferramenta baseada no método irá ser desenvolvida, testada em termos de eficácia e precisão contra outros métodos existentes, e irá ser integrada com ferramentas de gestão de desenvolvimento de software, de modo a validar a sua usabilidade em projetos reais durante a fase de desenvolvimento destes. Assim, o objetivo principal desta dissertação é o de ajudar a reduzir o tempo perdido em estimações, mantendo ou até melhorando a precisão das previsões feitas e mantendo a facilidade de percepção e de uso para os desenvolvedores e equipas que utilizem este método.

Keywords: Estimar, Prever, Desenvolvimento de Software

Contents

List of Figures

List of Tables

Abbreviations

COCOMO	Constructive Cost Model
SLIM	Software Lifecycle Management
ASD	Agile Software Development
XP	Extreme Programming
SEER-SEM	System Evaluation and Estimation of Resources – Software Estimation Model
ANN	Artificial Neural Networks
SVM	Support Vector Machine
BN	Bayesian Network
WIP	Work in Progress

Chapter 1

Introduction

Software developers can't usually estimate accurately the effort, time and cost of a project to be developed. This is inherit to the uncertainty that underlies their activity, since after the first estimation of the effort, the project may, with some likelihood, need to adapt to evolving circumstances, which may lead to changes in its scope, and consequently in delivery dates. These new circumstances may be impossible to predict in any estimation effort taken when the project started. This delays not only affect the development team but also other parts of the company, such as staffing or marketing. This could, in some situations, lead to the company losing time and in many times the trust of the stakeholder. A snowball effect that had it's begging in that poor estimation.

Methods that relay on Human estimation are, often, time consuming, what can represent a problem when teams waste time in making estimations that will, often, underestimate the effort and duration.

1.1 Motivation

The motivations of this dissertation project are: the fact that effort predictions in the industry, most of the times, are done using expert-opinion methods that could be time consuming and subject to human bias, and when these type of methods are not used, reports are that other types of methods don't perform as well as they are described in the literature and have a certain degree of difficulty to be applied to agile context projects, being the expert-opinion solutions much more apt to this task.

1.2 Objectives

The objectives of this dissertation project are to identify the motivations and forces at play when making forecasts and what are the main reasons that make this forecasts sometimes fail in delivering a good accuracy, determinate what methods and strategies may work better and achieve more close-to-reality forecasts in agile context, put the chosen methods and strategies to test using real-world data from different real projects and prove this way their effectiveness and accuracy. After analysing what method delivers better results, we will develop a tool based on this method, in order to validate the method used and test it against other existing methods.

1.3 Structure

The document has 4 more chapters. In Chapter 2, a background analysis is made of what earlier methods made the foundations of software development estimation and forecasting what important movements and premises this dissertation will take as bases for it's development and what metrics of both measurement, for input and output, and validation will be mentioned and used during the dissertation.

In Chapter 3, a complete analyses of the most relevant methods found is made, structuring a more in depth research about this methods, what benefits and limitation they present and a brief comparison between some of them.

In Chapter 4, a formalization of the problem this dissertation intents to approach is presented, a hypothesis is raised, two possible solutions to be followed and better analysed are proposed and the validation and workplan are defined.

In Chapter 5, some conclusions of this brief part of the dissertation are made and the expected results for the rest of the dissertation project are identified.

Chapter 2

Background

In this chapter we present an overview of concepts that provide the basis for the rest of the document. We will look into important notions for the better understanding of software development effort forecasting/estimation and look into methods that establish themselves as pioneers on this field, but that nowadays aren't really part of the state of the art in this field. We will also look into important metrics both for input/output and validation that will be mentioned during the rest of the dissertation.

2.1 Mythical Man-Month

When talking about software development management and effort forecasting or estimation, we need to mention the Mythical Man-month phenomenon, elicited by Brooks[7]. In his book, F.Brooks mainly explains that adding more manpower during the development of the project will probably end up delaying even more the project. When bad predictions are made and deadlines are built on them and then the project starts to show signs of being delayed, a common response is to add more manpower to fulfill the deadline. This is a bad response, since the time to integrate the new developers into the project and the increased communication overhead will cause even more delays.

2.2 #NoEstimates Movement

We will now look into the #NoEstimates movement, a movement that defends that maybe estimations are not the better way, since often they are not accurate and are wrongly used to put pressure into teams to deliver. S.McConnell recommend some tips for making good predictions [27], one of them being "*Distinguish between estimates, targets, and commitments.*" reveling that these terms are many times compressed in the estimate definition, that in case of a underrated estimation, will lead to teams not being able to correspond to the commitments and targets defined. That's why is important to look at estimates and see them for what they really are, and don't overuse them.

R.Jeffries also mentions that estimates are almost impossible to do in a way that they will be accurate when there's missing data or when requirements are vague, a characteristic they often present [20].

L.Keogh goes a bit further and presents another idea, a separation and definition of two different terms for effort prediction[24]. These definitions are:

- **Estimate**, a single value prediction of what will be the real effort.
- **Forecast**, a probabilistic prediction of a range of values or a exact value with confidence levels associated.

L.Keogh defends that forecasting, due to it's probabilistic characteristics, will often provide more accurate predictions and more reliable ones that estimates.

2.3 Metrics

In software development effort forecasting and estimation, there are several metrics available to classify both inputs and outputs. Some of this metrics are better understandable, represent better the variable associated and are easier to obtained than others.In this section we will look into some of the most used metrics. We will also look into some of the most used validation metrics, metrics useful to test and compare different methods.

2.3.1 Takt Time

Takt is a German word for a rhythm or beat. Takt Time describes the regularity of that beat. In production line manufacturing, Takt Time is the rate at which each manufactured item is finished. Using Takt Time, a manufacturer is able to run the line at a pace that is in correspondence with demand. It also provides an easy way to work out how long it would take to produce large batches of items [11].

2.3.2 Story Points

A story point is a metric used in agile project development management to estimate the difficulty of implementing a user story, which is an abstract measure of effort required to implement it. It's a number that tells the team about the difficulty level of the story. Difficulty could be related to complexities or risks. The story points are used to then calculate the velocity of the team.

2.3.3 Throughput

Throughput is the number of items completed in a given time window or iteration. It is a measure of output - how much is getting done. Measuring daily throughput as well as trends over time provides insight into both the rate and rhythm of delivery[6].

2.3.4 Cycle Time and WIP

Cycle time is time that elapsed from the start of work on a given feature till it was finished. Work in Progress (WIP) is a number of items that the team already started to work on but haven't finished them till the moment[6].

2.3.5 Validation metrics

As validations metrics, the main used metrics are the Magnitude of Relative Error(MRE) with the Equation (2.1), the Mean Magnitude of Relative Error (MMRE) with the Equation (2.2), the Percentage Error (PE) with the Equation (2.3) and the pred(x), percentage of estimates that are within x% of the actual value.

$$MRE = \frac{|\text{Actual value - Estimated value}|}{|\text{Actual value}|} \tag{2.1}$$

$$MMRE = \sum_{i=1}^{n} MRE_i \tag{2.2}$$

$$PE = \frac{\text{Actual value - Estimated value}}{\text{Actual value}} \tag{2.3}$$

This will be used in the validation phase of the method implemented and developed and also to compare the state of the art methods.

2.3.6 Other metrics

Besides the metrics presented earlier, there are other highly used metrics. Lines of Code, number of features, number of interfaces, use cases are some examples of this other metrics used for inputs. For outputs we can have duration metrics like days, weeks, months or effort metrics like man-day or man-month.

2.4 Types of Methods

After this more in depth contextualization, we will start to analyse the exiting methods and models. The methods are divided into 4 main types [35]:

- **Empirical Methods**, methods that gather information by means of direct observation and experience.
- **Theory-Based Methods**, methods based on an explicit theory or logic model.
- **Regression Methods**, statistical methods for calculating the relationships between a dependent variable and one or many independent variables.
- **Machine Learning Methods**, methods that work with machine learning typologies and focus on the learning ability inherent.

2.5 COCOMO

COnstructive COst MOdel (COCOMO), a theory-based model, was created by Barry Boehm in 1981 and is one of the first probabilistic methods.

COCOMO is divided in Basic, Intermediate or Detailed type, depending on the processing power and detail required and this types use one of the following models:

- Organic, a mode used for small projects developed within stable environments with a relaxed requirement specification
- Embedded, a mode used in large projects that are operated under tight constraints.
- Semi-detached, a mode used in intermediate projects that lie between the two previous extreme models.

COCOMO takes as main inputs,all subject to scale factors, the LOC of the project and in the intermediate and Detailed versions, it implements Cost Drivers. Cost Drivers are refinements to the estimate to take account of local project characteristics. Cost drivers are spread across four categories (product, computer, personnel and project) and have ratings (Very Low, Low, Nominal, High, Very High) assigned, depending on the extent to which they affect the project in question.

The 17 available cost drivers are: in Product(Required system reliability; Complexity of system modules; Extent of documentation required; Size of database used; Required percentage of reusable components), in Computer (Execution time constraint; Volatility of development platform; Memory constraints), in Personnel (Capability of project analysts; Personnel continuity; Programmer capability; Programmer experience in project domain; Analyst experience in project domain; Language and tool experience) and in Project(Use of software tools; Development schedule compression; Extent of multisite working and quality of inter-site communications)[28].

The output effort is given in Man-month, with pessimist and optimist deviation forecastes.

2.6 SLIM

The Software LIfecycle Management(SLIM) method, created by L. Putnam, is a theory-based technique to forecast software development effort.

This method is based on Putman's lifecycle analysis of the Rayleigh distribution. The Rayleigh distribution is used to forecast the number of developers that should be allocated to the development and maintenance during the lifecycle of a software project.

The SLIM model uses two main equations: the software productivity level equation and the manpower equation [10]. The software productivity level equation expresses the project size and development time, while the manpower equation expresses the buildup of manpower as a function of time.

This method was already very criticized by many researchers and Putnam himself found from implementing the method in 750 projects that the model worked properly for only 251. This method was also criticized by it's underlying assumptions, including the fact that it disregards the initial stages (exploratory design /specification) of a project.

Chapter 3

State Of The Art

In this chapter, we provide an overview of relevant breakthroughs, relevant existing methods and tools based on this methods and the overall state of the art in software development forecasting. We also take important conclusions for the next steps to take on this dissertation.

3.1 Empirical methods

In this section, we will look into some of the existing empirical methods, methods that gather information by means of direct observation and experience.

3.1.1 Expert-Opinion

Firstly, let's begin with the most used set of methods in the industry for agile context projects, expert-opinion methods, also known as expert-judgement methods.

The main advantage of this method is that the expert can analyse the differences between past project experiences and requirements of the new project to better estimate, also analysing the impact of new technologies, applications and programming languages.

Muhammad Usman et al. showed that this type of techniques are the most commonly used for estimate effort and duration of software projects when agile software development(SAD) methodologies like Scrum or Extreme Programming(XP) are being used [39].

Magne Jørgensen reported that 10 out of 16 studies reviewed showed that using expert-opinion based effort estimation methods led to more accurate effort estimates than using sophisticated formal forecasting models [21]. Main reason appointed for this conclusions are:

- The large number of input variables that many methods require can lead to overfitting and lower accuracy when we're using small data sets to learn from.
- The effort and complexity of building a proper model.
- The difficulty of understanding what input variables are really important and those who are not.

This allied to the fact that expert-opinion methods are more easily explained and understood that other complex methods, shows that expert-opinion based methods are commonly better accepted by companies and their staff.

3.1.1.1 Wideband Delphi

Wideband delphi, developed by Barry Boehm and John Farquhar , is a estimation method by expert-opinion where the project manager selects the estimation team and a moderator [37].

The team should consist of 3 to 7 project team members. Also, the moderator should be familiar with the Delphi process, but should not have a stake in the outcome of the session if possible, to try to avoid human bias. If possible, the project manager should not be the moderator because he should ideally be part of the estimation team. Every member should understand the Delphi process, read the vision and scope document and any other documentation, and be familiar with the project background and needs. The team should agree on a unit of estimation and after some brainstorm, write down the assumptions made.

Individually, each member should right down his estimates and assumptions that lead him to that estimation. The estimations are collected and a plot is made with every estimate and the team resolves any issues or disagreements. However, no individual estimation is discussed and the team tries based on that discussion arrive in a consensus. In the end the project manager and the team review the final results to check if everything is in order.

The main downsides of Wideband delphi are the huge amount of time spent in estimation that could be used in other tasks and the subjectivity to human bias, because even though the method recommends that the moderator doesn't have a stake in the outcome of the meeting, this is not always ensured and other members in the reunion could make estimations to favor their own stakes.

3.1.1.2 Planning Poker

Planning poker, developed by J. Greening [13] and based on the Wideband delphi method, is a estimation method used during the release planning phase, when every story is getting a effort value and when the team makes the first draw of how the sprint will be organized. Basically, when using planning poker, a story is read and a brief explanation of the story is made so that every team member participating understands it. After that, every team member has to write down on a card their effort estimation for that particular story. Instead of writing down on a paper a value for every estimation, there are decks of cards available, like the classical deck in which the cards are numbered with the Fibonacci series in order to reflect the inherent uncertainty in estimating larger items [36]. Then, when every member has already his estimation made, the cards are shown. Based on the estimations of each member of the team, a discussion starts, to analyse the highest and lowest estimations and to try understand their perspective and then try to achieve common ground among the team to proceed. If this common ground is not achieved, the team should just stop and defer the story, split it, or take the lowest estimation. Also if a story is getting consensus from the participants in attributing it a high value, the story should be split, since it probably represents a super story and can, probably, be divided in smaller ones.

In summary, this method aims to involve all the team members in the estimation process and make it more quick. Although being presented as a great technique, it presents some problems and issues that may rise:

- Teams could just ignore the discussion phase, a big feature of the method, and just take the most consensual value.

- Being a manual process and more time consuming than computational methods, what translate in more time spent in estimation.

- Subject to human bias, personal agendas of the participants who may influence the estimation in order to achieve personal goals.

Even though planning poker present some problems, is noted to be one of the most known and used methods in the industry. This could be explain by it's simplicity and satisfactory accuracy. This method is revealed to be easy to understand, not only by members of the team, but also by stakeholders and upper management.

3.1.2 Analogy

Analogy methods are based in the analogical reasoning inherent to the human cognition. Analogical reasoning takes a fundamental part in recognition, classification, learning and in the process of creativity. Whenever we are about to make a decision, we use analogical reasoning by recalling related past experiences.

The use of analogy for software effort estimation is the process of forecast the effort and duration of a project, based on previous relevant analogous projects. It's main advantage are the fact that forecasts were based upon experiences that could be analysed to determine the specific similarities and their possible impacts on the new project.

According to Ali Idri et al. [16], analogy processes are generally composed of three steps:

- **Characterizing:** Feature and project selection, feature weighting, and the selection of other dataset properties, such as dataset size, outliers, feature type, and missing values.

- **Similarity evaluation:** Retrieval of the cases that are the most similar to the project under development using similarity measures, in particular the Euclidean distance.

- **Adaptation:** Prediction of the effort of the target project based on the effort values of its closest analogs. This requires choosing the number of analogs and the adaptation strategy. The number of analogs refers to the number of similar projects to consider for generating the estimates. Based on the closest analogs, the effort of the new project is derived using an adaptation strategy.

The main advantages of Analogy based methods are:

- Can achieve good levels of accuracy.

- Simple if the organization repeats similar work.

- Estimates are immediately available.

Although presenting some advantages, when using this type of methods it is extremely tough to evaluate the dissimilarity among the environments and therefore assess the correctness of the data [36].

3.1.2.1 ANGEL

Chistopher Schofield's ANaloGy Estimation tooL (ANGEL) [33] is one of the most used analogy estimation tools. In this approach, for the characterization phase we must characterise a project with one or more descriptor features. These can be represented as quantifiable (interval, ratio or absolute) or categorical (nominal or ordinal) variables. These must be available at the point when an estimate of the goal feature (effort) is required. It is recommended that features that have a direct impact on effort are used. Collecting features that have no perceived relationship to the project effort is likely to result in the selection of poor analogies, what will lead to wrong forecasts. However, no expert is required to decide what features will be selected, since the ANGEL tool incorporates a mechanism for finding the best combination of features presented to it, based upon the historical data it is presented with.

For the similarity measurement phase, the measures that have found practical use are predominantly the nearest neighbour algorithms. The nearest neighbour algorithm used in forecasting by analogy involves measuring Euclidean Distance (3.1) in n dimensional space.

$$ED(x,y) = \sqrt{\sum_{i=1}^{i=n}(x_i - y_i)^2} \tag{3.1}$$

The Euclidean Distance represents the simplest proximity measure and the most commonly used. To use this measure, ANGEL first needs to standardise the features, dividing them by their range, so that every feature is one value between 0 and 1. Two different feature types can be incorporated into the similarity measure, both quantifiable and categorical features but while the use of quantifiable data is not problematic, categorical features have no notion of interval and therefore can only be described as identical,assuming the value of 1 or different, assuming the value of 0. For measuring the confidence that can be placed in a forecast, each project is successively removed and its effort estimated using the remaining projects as the analogy source. The estimated effort for each of the projects is then compared to their actual effort, which yields an indication of how much reliance can be placed upon new estimates from that Project case base.

For the adaptation phase, ANGEL finds the n closest projects and makes an average of their total effort or alternatively applies a weighting so that the closest projects contribute more to the eventual estimate. The Angel tool has 4 different strategies: choosing 1 analogy(closest project), 2 analogies(either with the average or weighted average of the 2 closest projects) or 3 analogies(average of the 3 closest projects), since more analogies would mean more computational power required. The main limitation of the ANGEL tool and analogy methods in general are:

- Not applicable in agile context since it takes final efforts of analogous projects and not the final efforts for each iteration of those projects.
- Is based on a partially wrong premise:"Similar projects have similar costs".

To Dealing with noisy features, features that don't translate in useful analogies and only add noise, ANGEL performs an exhaustive comparison of every possible combination of features until the subset of features that returns the best confidence is found. This could be problem since this exhaustive comparison can be computationally expensive. In 1998, Christopher Schofield estimated that in a set of 20 projects processed by a Pentium 200, ANGEL would take 5 seconds to process 5 features, 2 minutes and 40 seconds to process 10 and 45 hours to process 20 features [33]. This estimation shows the exponential behavior of this comparison.

The ANGEL tool was used to forecast effort in several databases to check which of the strategies (1, 2, 2(weighted) or 3 analogies) was performing better. The Table 3.1 presents those results.

Data Sets	N° Project Cases	Optimum N° of analogies
Albrecht	24	2 (weighted)
Deshamais	77	3
Finnish	38	2 (weighted)
Hughes	33	1
Kemerer	15	2
Mermaid	28	1
RealTime1	21	2
Mermaid	18	1

Table 3.1: Optimum no. of analogies for each dataset, reported by Schofield [33]

Summarizing the table, 1 and 2 analogies work better than 3 and 2 (weighted) in most of the datasets analysed by Schofield.

3.2 Theory-Based Methods

In this section, we will look into some of the existing Theory-Based methods, methods based on an explicit theory or logic model.

3.2.1 Monte Carlo

Monte Carlo techniques are used to forecast the results of problems that can't be classified by a single exact value, or that are to difficult to solve in order to arrive to a exact value.

Monte Carlo methods uses random sampling techniques, where the inputs are chosen randomly from a subset, and recording the results over many computational cycles. Instead of a single value, a range of values is returned, forecasting the most probable result in that range of values. Random sampling techniques were first applied in the decade of 1930 to estimate the value of π, being later used, in 1946, by Stanislaw Ulam and Enrico Fermi to formalize the Monte Carlo method in the Los Alamos Laboratory, USA [26]. They used Monte Carlo methods to solve problems in the radiation shielding and neutron diffusion. The method only gained the name Monte Carlo after John von Neumann coined it, in memory of his uncle's favorite casino.

Nowadays, Monte Carlo methods are mainly used to manage risk in the financial field, oil and gas exploration and project management.

When applying the Monte Carlo method to software projects effort forecast, the simplest model would include the amount of work in the initial backlog, the columns that work flow through(like Design, Develop, Test) and a cycle-time estimate for each of those columns. This model would allow simple forecasts, but it is not good enough to give a clear idea of the impact of events that occur during development that delay the goal date. Defects and bugs are found and added to the backlog of work, questions arise about the project and features and modifications in the project scope and design are made. Although this examples of variables are not primary to a project, when happening, they cause delays, many times doubling the duration of the project. So when implementing a Monte Carlo method we should pay attention to what variables are more important and which ones will impact the most in the project's duration. This is the reason why a more granular model is so important.

In the end, the method returns a diagram representing the probabilities of ending the project by a given date, for example "*there is 70% chance that we will be done by the end of June, 80% chance it will be half of July*" [6].

We can use Monte Carlo method to forecast the Defects Costs, making two simulations, one taking in account defect defined previously and another one without this definitions. Subtracting the two simulations will give us the forecast for the amount of days that the team will spend fixing bugs and defects.

Another possibility is to forecast what staff skills are in demand, taking advantage of the description of the columns input. Columns are, usually, associated with specific staff skills. For example, the testing column is associated with testing skills, the development column is associated with developing skills,etc. This way we can analyse what staff skills will impact the most the project duration and tackle this problem, arranging the team to achieve better forecasts.

Another advantage is that Monte Carlo methods can be integrated with Agile methods, working with iterations and, this way, improving the final forecast every sprint that has passed, since the model will before each sprint make readjustments in the final forecast with the information available from past sprint of the same project. Although it seems really wonderful, as stated by D.North in his blog [29], this can also lead to wrongest forecast, and so to make more accurate forecasts, some cautions must be taken:

- "*The past work should be representative of the future work*". If the past work was about adding new features and the next work will be about integrating the system with other systems, the past work is unlikely to be representative, since the team will have different paces in the task ahead.
- "*The future delivery context will not change significantly compared to the recent context*". If the team is changing or a reorganization as happened, this is likely to affect the delivery diagram and also change the parameters from that point on.
- "*The Monte Carlo function should be a reasonable indicator of the past work*". Identifying the parameters that genuinely capture the behaviour, and using them to define a function that reasonably represents the historical and future data, is hard.

In terms of practical experimentation of the method, J. Zang shows that when implementing a Monte Carlo model in a real project, the acceptance of business users, stakeholders and upper management was great [40]. Since the method returns a highly graphic diagram, it was easy for people with no knowledge in software development to better understand the premises of the forecast achieved. This shows another feature of Monte Carlo models. Since this models are easy to understand, in part due to it's easy implementation, people outside of the area can deeply recognise the main factors playing during a software project development and more intrinsically understand the final forecast of the model.

There is a online tool implementing the Monte Carlo Model, developed by Sevawise[1]. This tools takes as inputs the historical data type(Velocity or Story Count), the number of Sprints per Program increment(sum of the time windows of every sprints to analyse), historical data input (previous sprints in excel) and Number of Sprints of historical data(previous sprints). As a output it returns a data grid which displays the level of confidence that a specific Velocity or Story Count will be achieved for the PI timebox selected. A great feature implemented by Sevawise was the export to excel feature. This is highly advantageous because allows for better share of information between departments and better storage of past forecasts.

[1] Sevawise's Monte-Carlo

The Table 3.2 shows a example of a output grid from this model showing the percent chance some PI velocity and Sprint Velocity will be achieved. Also classifies presents forecasts for each bet taking in count: Very conservative bet, Conservative bet, Realistic bet, Optimistic bet or Very Optimistic bet.

Percent Chance	Bet	PI Velocity	Sprint Velocity
99%	Very Conservative	93	31
80%	Conservative	124	41
50%	Realistic	142	47
20%	Optimistic	158	53
1%	Very Optimistic	181	60

Table 3.2: Example of a Output from Sevawise model, observed by Betts [4]

3.3 Regression Methods

In the present section, we will look deeper into a type of regression methods, methods that based themselves in statistical models for calculating the relationships between a dependent variable and one or many independent variables.

3.3.1 Parametric Models

Parametric models are useful for subjecting uncertain situations to the rigors of a mathematical model. They usefully embody a great deal of prior experience and are less biased than human thought processes alone, what extract the uncertainties of personal agendas and motivations of the forecast process. Parametric models provide forecast techniques based on facts, including mathematical equations as well as interpretation of historical data. Projects are break down into sub-components, like stages for deliveries and match each stage with appropriate equation to obtain the forecasts. Although a main advantage of parametric models is the simplicity of implementation, unfortunately, this doesn't always is translated into simplification in terms of understanding of the methods by the team members or even the stakeholders and upper management.

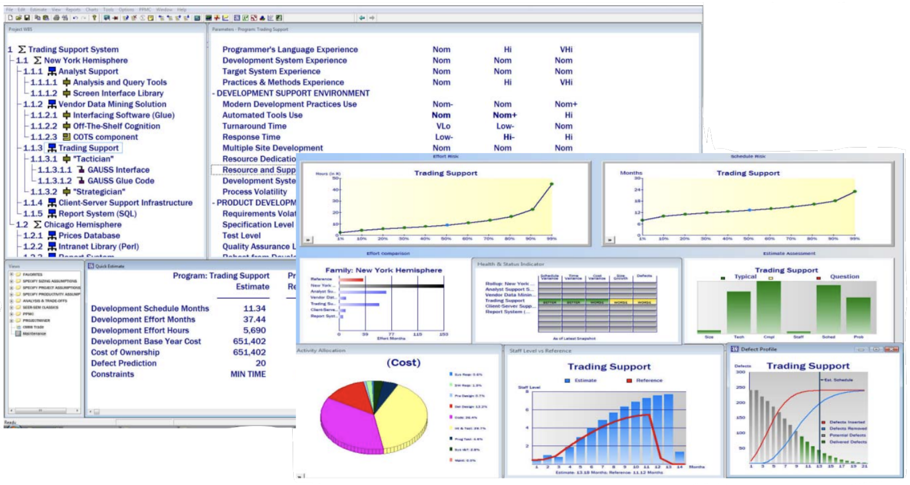
3.3.1.1 SEER-SEM

The System Evaluation and Estimation of Resources-Software Estimation Model(SEER-SEM) is a parametric regression model developed by Galorath Inc., based on the Jensen model developed during the decade of 1970 by the Hughes Aircraft Company's Space and Communications Division.

SEER-SEM forecast based on knowledge bases, parametric algorithms and historical precedents. It makes forecast of effort, duration, staffing, and defects.

SEER-SEM implements over 50 inputs that will influence directly the output of the method. Among them, 34 parameters are used by the SEER-SEM effort estimation model [19]. In SEER-SEM effort estimation, each input has sensitivity levels, with the ratings ranging from Very Low(negative) (VLo-) to Extra High(positive) (EHi+). Each main rating level is divided into three sub-ratings, such as VLo-, VLo, VLo+. These ratings are then translated to the corresponding quantitative value used to calculate the effort.

SEER-SEM takes as input the project scope, mainly size measurements (Lines of Code, Function points, Use cases), Historic Data, staffing levels and capabilities, requirements specification and volatility, development and target environment and economic factors. As outputs the SEER-SEM model has a variety of grids, diagrams and reports, as Figure 3.1 shows.



1 Σ Trading Support System
1.1 Σ New York Hemisphere
1.1.1 Analyst Support
1.1.1.1 Analysis and Query Tools
1.1.1.2 Screen Interface Library
1.1.2 Vendor Data Mining Solution
1.1.2.1 Interfacing Software (Glue)
1.1.2.2 Off-The-Shelf Cognition
1.1.2.3 COTS component
1.1.3 Trading Support
1.1.3.1 "Tactician"
1.1.3.1.1 GAUSS Interface
1.1.3.1.2 GAUSS Glue Code
1.1.3.2 "Strategician"
1.1.4 Client-Server Support Infrastructure
1.1.5 Report System (SQL)
1.2 Σ Chicago Hemisphere
1.2.1 Prices Database
1.2.2 Intranet Library (Perl)

Programmer's Language Experience	Nom	Hi	VHi
Development System Experience	Nom	Nom	Nom
Target System Experience	Nom	Nom	Nom
Practices & Methods Experience	Nom	Hi	VHi
- DEVELOPMENT SUPPORT ENVIRONMENT			
Modern Development Practices Use	Nom-	Nom	Nom+
Automated Tools Use	Nom	Nom+	Hi
Turnaround Time	VLo	Low-	Nom
Response Time	Low-	Hi-	Hi
Multiple Site Development	Nom	Nom	Nom
Resource Dedicatio			
Resource and Supp			
Development Syste			
Process Volatility			
- PRODUCT DEVELOPM			
Requirements Volat			
Specification Level			
Test Level			
Quality Assurance L			

Quick Estimate

	Program: Trading Support Estimate
Development Schedule Months	11.34
Development Effort Months	37.44
Development Effort Hours	5,690
Development Base Year Cost	651,402
Cost of Ownership	651,402
Defect Prediction	20
Constraints	MIN TIME

Figure 3.1: Example of the SEER-SEM output reports as presented in SEER-SEM product brief [19]

Initially, SEER-SEM generates an global effort and schedule forecast based on project size, complexity and productivity factors. Initial forecasts are generated in minutes. The problem with this simple forecasts is that they are not very accurate because they are based in little information about the project. To achieve better results the forecast must be refined increasing the number of inputs and the forecast complexity. Figure 3.2 shows the functioning of the SEER-SEM method through a software project effort forecast.

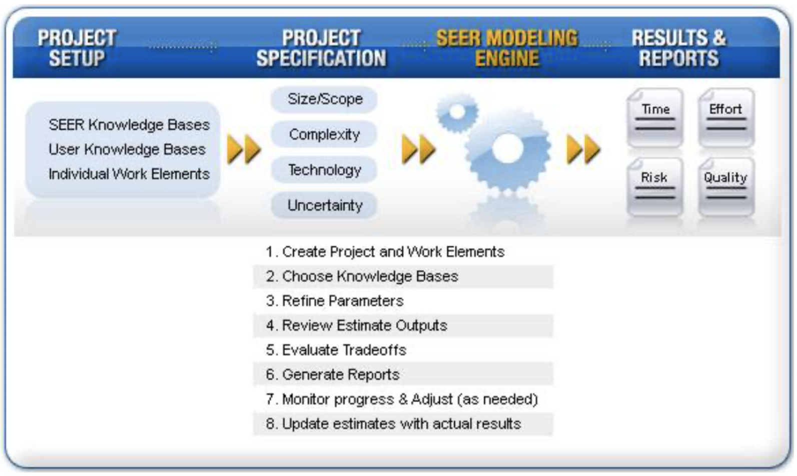


Figure 3.2: SEER-SEM functioning Diagram, reported in SEER-SEM product brief [19]

The main advantages of SEER-SEM is that produces a effort forecast with a good level of accuracy, the multi-platform integration ability, being able to be integrated with Microsoft Project, Microsoft Office, IBM Rational, and other 3rd-party applications[19] and it's ability to be implement in software projects using agile methodologies, what helps in maintaining the model actual, since Agile methodologies are the most used methods for software development management at the moment. Despite this two advantages, SEER-SEM present two main limitations that affect greatly it's usage in the industry [36]:

- Over 50 input parameters related to the various factors of a project, which increases it's complexity.
- The specific details of SEER-SEM make much harder to understand the relationship between the parameter inputs and the corresponding outputs.

3.4 Machine Learning Methods

In this section, we will look into some of the existing machine learning methods, methods that work with learning typologies and focus on this machine learning ability.

3.4.1 Neural Network Model

Neural Networks or Artificial Neural Networks(ANN) is Regression based method that are able to, in a more efficient and accurate way, establish complex relationships between the dependent (effort) and independent variables(inputs). It also has the ability to generalize from the training set of data. Most methods based on ANN use either feed-forward networks where no loops in the network path occur or feedback networks that have recursive loops. Limitations for ANN are:

- The difficulty of understanding why an ANN makes a particular decision [5], reason why this methods are many times called "*Black boxes*", since it's really difficult to understand what is happening inside the model.
- Two forecast from the same ANN will hardly be equal, due to the arbitrariness of the random weights.
- There are no guidelines for the construction of the neural networks (number of layers, number of units per layer, initial weights) [17]
- The amount of data required to usefully train a network since it's difficult to collect a considerable amount of data to train and test the model so it becomes viable.

Even with this limitations, there is a lot of interest in this models because of their ability of forecasting with good accuracy levels. S. Hydarali has conducted a survey where he asked some questions about ANN to 33 developers [15]. Among other questions, when asked which method had better performance in software development effort forecasting, 60% of the developers replayed ANN, what translates this interest in ANN.

In the Figure 3.3 is a example of a simple neural network architecture with one input layer (having 3 neurons for each input), one hidden layer (with 3 neurons) and an one output layer.

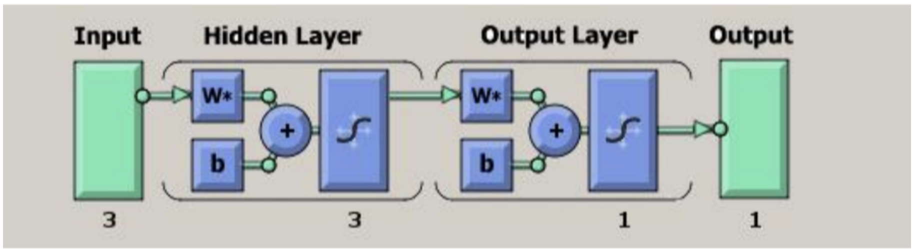


Figure 3.3: ANN Architecture example, explained by Bhatnagar [5]

A neuron computes a weighted sum of received inputs and generates an output if the sum exceeds a certain threshold. This output then acts as an excitatory or inhibitory input to other neurons in the network and the process continues until an output is generated. Artificial neurons are intended to be similar to biological ones, for example, the weighted inputs represent biological synapses, the interconnections between neurons represent the dendrites and axons, while the threshold function represents the activity in the biological neurocyton.

3.4.1.1 Support Vector Machine

Support Vector Machine (SVM) is a neural network method used in various areas both for forecasting and classification. In a linearly separable case, the SVM classifier separates the instances from two different classes by using a hyper-plane that tries to maximize the margin, increasing the capacity of generalization of the classifier. The instances closer to the borders(or on the borders) are the support vectors. The number of support vectors represents the complexity of a model. In the Figure 3.4 is a representation of the SVM algorithm.

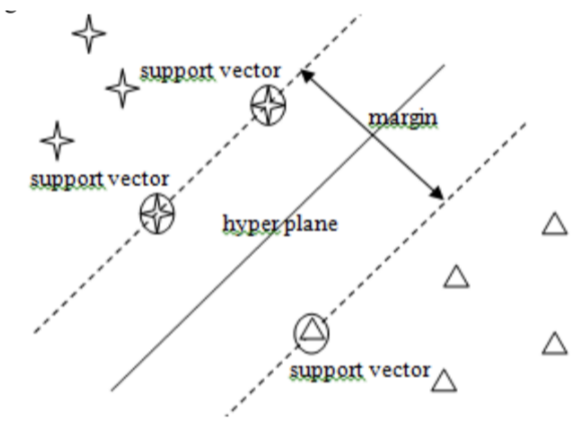


Figure 3.4: SVM Algorithm Representation, reported by Hidmi [14]

First, we must proceed to the calculation of the total number of story points and project velocity, that will be our input. This inputs are scaled using a nonlinear function, to be within the range [0,1].

S.Satapathy et al. suggest the Function 3.2 [32], with X representing the values of the dataset, x the values of a project of the dataset and x' the scaled value.

$$x' = \frac{x - min(X)}{max(X) - min(X)} \tag{3.2}$$

After achieving the scaled inputs, the projects are separated in the hyper-plane in order of those scaled inputs. If the dataset is not linearly separable we can either use soft margin, where the separation is made as in linear mode but there's a tolerance for one or few misclassified dots, or use a kernel trick, that applies some transformations to the existing features and creates new features. These new features are the key for SVM to find the nonlinear decision boundary. Kernel trick uses one of the following non-linear kernel methods:

- **SVM Polynomial Kernel**, a classifier that generates new values by applying the polynomial combination of all the existing ones.

- **SVM RBF Kernel**, a classifier that generates new values by measuring the distance between all other dots to a specific dot.

S.Satapathy et al. compared the different kernel methods accuracy when applying them to real project datasets [32]. The Table 3.3 illustrates the results obtained.

Actual Effort	SVM Linear Effort	SVM Polynomial Effort	SVM RBF Effort
0.4615	0.3436	0.2332	0.4194
0.7692	0.7698	1.1027	0.7865
0.2637	0.2295	0.2059	0.2325
1.0000	0.7030	1.9282	0.8908
0.1538	0.1635	0.1691	0.1489

Table 3.3: Comparison between different kernel methods, reported by Satapathy et al. [32]

This results show that the RBF kernel method achieves a better accuracy than the rest of the methods.

E. Kocaguneli et al. implemented a SVM model and used it to forecast the effort of various datasets of software projects available in the database PROMISE [25]. They concluded that the SVM model was the one that achieved better results among the models they tested.

Finally, this method can also be applied in forecasts of software projects effort when using agile methodologies and since has a increased capability of generalization, it's not confined to a specific type of project, what is very helpful for companies that work with different types of projects, like consulting companies.

3.4.2 Bayesian Network Model

Bayesian networks (BN) are graphical models that describe probabilistic relationships between related variables. BNs have been used for decision-making in problems that involve uncertainty and probabilistic reasoning. The first applications of BNs focused on classical diagnostic problems in medicine(during the 80s and 90s) [1]. The medical and biological decision-support domain was and continues to be the area in which more BN applications are published. However, companies like Microsoft have also used BNs for fault diagnosis and software companies have started to look into this methods benefits.

A BN is represented by a pair BN = (G,P), where G is a directed acyclic graph, and P is a set of local probability distributions for all the variables in the network. The directed acyclic graph ($G = [V(G), E(G)]$) is composed of a finite, non empty set of tuples $V(G) = \{(s_n, V_n), (s_{n+1}, V_{n+1}), \dots\}$ and a finite set of directed edges $E(G) \subseteq V(G) \times V(G)$.

Each node V takes on a certain set of variables s. Each node also contains the associated probability table, called the Node Probability Table, with the probability distribution of the variables of s.

The edges $E(G) = \{i, j\}$ represent dependencies among variables. A directed edge (i, j) from V_i to V_j shows that $V_i(parentnode)$ is a direct cause of $V_j(childnode)$.

If V has no parents, its probability distribution is unconditional. The probability distribution of variables must satisfy the Markov condition, which states that each variable of a node V is independent of non-descendants Nodes [8].

The 4 main advantages of Bayesian Networks are mentioned by Roger [30]:

- capability of handling missing data.
- capability of learning causal relationships.
- capability of combining prior knowledge and data.
- capability of avoiding overfitting the data.

The method takes as inputs mainly the Project Scope (User Stories, Features or Lines of Code), but can also implement other metrics in order to improve it's accuracy. This other metrics could be Staff factors(developer skills, experience and motivation) or Process characteristics(organization and maturity of the development process).

The output of this method is a single point forecast with probability bounds, providing some essential uncertainty to the forecast value.

M. Perkusich et al. made a survey and an analysis of the applicability of Bayesian Networks in Agile context software projects, more specifically, projects using scrum as it's software development management methodology [31]. With the results of the survey and the opinion of some experts, they build the model, which a partial of the directed acyclic graph is represented by Figure 3.5.

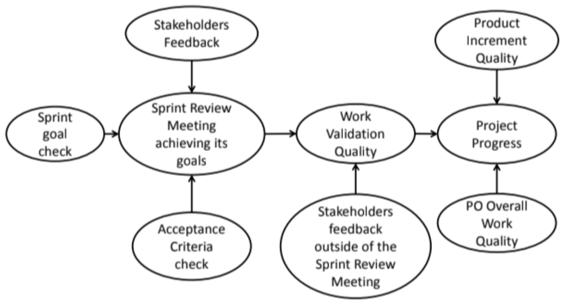

Stakeholders Feedback
Product Increment Quality
Sprint goal check
Sprint Review Meeting achieving its goals
Work Validation Quality
Project Progress
Acceptance Criteria check
Stakeholders feedback outside of the Sprint Review Meeting
PO Overall Work Quality

Figure 3.5: Partial directed acyclic graph for the BN, defined by M. Perkusich et al. [31]

3.4.3 Fuzzy Logic

Fuzzy logic was founded and first applied by Lotfi Zadeh in 1965. Fuzzy logic is a mathematical tool for dealing with uncertainty and also it provides a technique to deal with imprecision and information granularity. Fuzzy Reasoning is divided in 3 main components [3]:

- Fuzzification process, where the objective term is transformed into a fuzzy concept. Membership functions are applied to the actual values of variables to determine the confidence factor. This allows for inputs and outputs to be expressed in linguistic terms.
- Inference, a defuzzification of the conditions of the rules and propagation of the confidence factors of the conditions to the conclusion of the rules. A number of rules will be fired and the inference engine assigned the particular outcome with the maximum membership value from all the fired rules.
- Defuzzification process, where the translation of fuzzy output into objectives is made.

A Membership function, used in the Fuzzification process, is a curve that defines how each point in the input space(also referred to as the universe of discourse) is mapped to a degree of membership between 0 and 1 [2]. The main types of Membership Functions used are the Triangular Function (TMF), Trapezoidal Function and the Gaussian Function (GMF).

D. Ahlawat and R. Chawla tested these 3 meambership functions to observe which one presented better results [2]. Table 3.4 shows the results obtained.

Actual Effort	E.E with Trap	E.E with TMF	E.E with GMF
2040	2089	2075	1945
321	201.6	200.4	187.8
79	102	101.15	94.35
6600	7373	7329.2	6854.7
724	690.9	687.14	643.9
83	95.71	95.14	89.2
156	159.12	158.04	148.32
15	16.3	16.22	15.17

Table 3.4: Comparison between different Membership Functions, reported by D. Ahlawat and R. Chawla [2]

As we can conclude, the Gaussian membership function is the one that achieves better accuracy, according to this study.

During the inference process, Fuzzy sets and fuzzy operators are used. Fuzzy Sets are the subjects while fuzzy operators are the verbs of fuzzy logic. Fuzzy logic is comprised by conditional statements formulated as the rule statements "*if-then*. A single fuzzy if-then rule takes the form *if x is A then y is B* where A and B are linguistic values defined by fuzzy sets on the ranges of X and Y. The if-part of the rule "*x is A*" is called the premise or antecedent, while the then-part "*y is B*" is called the conclusion or consequent [2].

One major problem of software development effort forecasting is that data available in the initial stages of a project is often unclear, inconsistent, incomplete and uncertain. A fuzzy model is more appropriate when the systems are not suitable for analysis by conventional approach or when the available data is uncertain, inaccurate or vague. Some of the difficulties of other methods, as described by Gray and MacDonell [12], are :

- Providing exact values for inputs, with methods often using values that they anticipate will occur, since for many metrics the actual value is never known with certainty until the project is completed.
- Over commitment, with most of methods committing to a value and not even provide confidence intervals.
- The size of data sets, since most of the times the datasets used to forecast the effort of the project are too small, not contributing with significant values to generate a good forecast.
- Interpretability of models, with some models being too hard to understand or giving outputs that are too hard to interpret.

This are problems that fuzzy models focus on solving, with easy interpretation of the model and output, handle of missing data, handle of non exact linguistic values and reducing the commitment through fuzzy outputs.

Based on this benefits of Fuzzy Logic, implementations of Fuzzy Systems combined with other existing methods have also appear.

V.Sharma and H.Verma proposed a Fuzzy system implemented on the Cocomo model, using the Gaussian Membership Function, since on their study they also reported the best accuracy of this membership function [34]. After establishing the fuzzy rules for the nominal effort components of Cocomo, they also proceeded to the fuzzification of the cost drivers available in Cocomo and subsequent establishment of fuzzy rules for this components.

W.Du et al. developed a fuzzy system implemented with the SEER-SEM model [9]. With this combination of processes, they helped to mitigate some of the problems and limitations of SEER-SEM model. After the development of the model, they putted it to test against the SEER-SEM model alone and were able to achieve good conclusions. In summary, the combination of methods proved in every experiment to be more accurate that SEER-SEM alone, also mitigating some of it's limitations, already described.

A.Idri et al. proposed a fuzzy analogy system, a combination of the fuzzy logic method and analogy method [18]. Like a normal analogy method, the identification of cases, retrieval of similar cases and cases adaptation steps are still present in this model, but here these are fuzzification of its equivalent in the classical analogy procedure. With the implementation of fuzzy logic, they were able to introduce some machine learning abilities like the three criteria part of human nature that machine learning systems implement: tolerance of imprecision, learning and uncertainty. This method also help updating the false premise of analogy methods "*similar projects have similar costs.*" to a premise with some uncertanty added "*'similar projects have possibly similar costs'.*"

3.5 Final Discussion

After a more in depth analysis of each method, this section will summarize the methods seen and compare their benefits, limitations and accuracy.

COCOMO, in synthesis, takes too much inputs when using the intermediate or Detailed types, because when using the Basic, the accuracy is lower. Most of this inputs are expert-based, meaning that a expert must estimate some inputs in order to make the forecast. This could lead to subjectivity to human bias. Some benefits are the fact that simple to implement and is also relatively simple to understand. A possible implementation that mitigates some of the problems of COCOMO and increases it's accuracy is combining it with fuzzy logic. The isn't easily applied in agile context, since it doesn't analyse past iterations of a project.

SLIM's major limitation is the fact that the method in a experience of 750 projects only worked properly in 251, what represent a high number of projects where the model didn't make any sence. Also it is a target of criticism because of it's assumptions, like the disregard of initial stages of a project. Compare to the other methods presented, SLIM doesn't add anything that could be considered as a benefit in relation to the others. The method is difficult to apply in agile context since it doesn't analyse past sprints of a given project. Although, it provides a satisfactory accuracy and is relatively easy to understand.

Both Wideband Delphi and Planning Poker have proven to achieve a satisfactory accuracy and because of their simplicity, don't increase the difficulty of understanding the results. Their main liabilities are the fact that more time is necessary to make the reunion for making the estimation and is also subject to human bias. A great benefit both Planning Poker and Wideband delphi have, in relation to the others, is the fact that development teams are more comfortable using this expert-opinion methods because of the discussion between different perspectives and the great communication it provides. Among the expert-opinion methods, planning poker is simpler and with a more focus discussion achieves better results than wideband delphi. Both methods can be used in agile context, since new meetings can be done for estimate at the end of every iteration.

Angel, for being a analogy method, proves itself to be easy to understand and to interpret. The main limitations are the inability to handle missing data and non exact values and the fact that the premise of this type of methods is partly wrong. When combining methods like Angel with Fuzzy logic, we saw that this limitations are mitigated and better accuracy is achieved. The method is not easily implemented in agile context since it doesn't analyse past iterations of the actual project.

Monte-Carlo proves itself to be a method that presents good accuracy and besides that, is easy to implement and easy to understand, with some literature even giving simple examples of the process of this method. Another great benefit is the Granularity inherent, with a easy interpretation of the level each input affects the final forecast. The method is easily implemented in agile context.

SEER-SEM, although showing satisfactory accuracy, presents itself as a hard method to understand and interpret, partially because of it's numerous inputs (could go over the 50 inputs) and because of it's specific details that make really hard the interpretation of the relation between inputs and outputs. When used in combination with Fuzzy logic, this difficult understanding is mitigated and the accuracy even improves, in relation to SEER-SEM alone. The method can be adopt when using agile methodologies.

Both SVM and BN are presented as tough to understand, with some studies even calling this methods, black boxes. However, these methods show great accuracy, the ability to handle missing data, the fact of avoiding overfitting to data and the capability of generalization, not being confined to one type of projects. Both methods can work in agile contexts, since both can analyse past iterations of a project.

Fuzzy Logic also handles missing data, avoids overfitting and has satisfactory accuracy but, unlike SVM and BN, since it works with linguistic values, is easier to understand and interpret what presents a great benefit. This method can also be applied in agile context projects since can be implemented to analyse past iterations of the project.

Table 3.5 synthesises the methods limitations, benefits and their applicability in Agile context projects.

Method	Understandability	Accuracy	Overfits to Data	Agile
Cocomo	Easy	Not Good	Yes	Yes
SLIM	Easy	Satisfactory	Yes	No
Wideband Delphi	Easy	Satisfactory	No	Yes
Planning Poker	Easy	Satisfactory	No	Yes
Angel	Easy	Not Good	Yes	No
Monte-Carlo	Easy	Good	Yes	Yes
SEER-SEM	Hard	Satisfactory	Yes	Yes
SVM	Hard	Good	No	Yes
BN	Hard	Good	No	Yes
Fuzzy Logic	Easy	Satisfactory	No	Yes

Table 3.5: Synthesis of methods

D. Toka and O.Turetken compared the SEER-SEM method, the SLIM method and the COCOMO method when applied to a project effort forecast [38]. Table 3.6 shows the results obtained.

Method	MMRE	PE
COCOMO	74%	-54%
SEER-SEM	36%	-10%
SLIM	41%	-7%

Table 3.6: Comparison of different methods, reported by Toka and Turetken [38]

This results show that COCOMO provides the least accurate results.

B. Boehm defends [22] that the reason why expert-opinion methods are being more used in the industry and the reason why they show better results in the industry and then don't correspond in the literature, when compared to other types of methods is because this other types are many times wrongly used in companies. Companies often don't make good implementations of the methods to their specificity cases. This will lead to methods not corresponding to what researchers claim in the literature.

M. Jørgensen defends [23] that most times methods that work with ranges instead of exact values, have the ranges too narrow. Both the minimum and maximum values of the rage are too close what very often doesn't translate the uncertainty inherent to the project, making the forecast more probable to be inaccurate.

Chapter 4

Problem and Proposed Solution

This chapter contains the details and formalization of the problem we are trying to solve according to the notions already presented above and the hypothesis raised by this problem formalization. The solutions that will be followed and better analysed are discussed and we define how the chosen method will be validated. A Workplan for the next phase of the dissertation is presented.

4.1 Problem Formalization

As stated before, software developers can't usually estimate accurately the effort, time and cost of a project to be developed, a problem inherent to the uncertainty that underlies their activity. This allied to the fact that methods that rely on human estimations, usually, cost a lot of precious time and, often, provide underestimates of effort and duration, rises the problem of having a method that focus on forecasting, and this way not as time consuming as estimation methods, easy to use and understand, making it more explainable and user-friendly, and that provide us results with good accuracy. This method should also work with any type of software development management methodologies, specially with agile methodologies since, nowadays, these are often used to manage software projects.

4.2 Hypothesis

This dissertation is based on a hypothesis that serves as a basis for its development and implementation. This hypothesis is:

"*To what extent can data-driven methods for software project forecasting be applied in agile projects bringing benefits in terms of reduced forecasting errors and user's effort?*"

This hypothesis is supported by the following research questions:

- What method will be used?
- Is the chosen method more efficient than the others?
- Which type of data will this method work with as input?
- What type of output will be delivered by this method?

4.3 Proposed Solutions

To solve the stated problems, we chose the Monte-Carlo method and the Bayesian Networks. Since the literature didn't provided enough information and data to compare both this methods, and since both present different benefits, we decided to choose both this methods and with some more practical research, to be done in the future, select the one that presents better results, proceeding then to the development phase. Monte-Carlo main benefits are: the applicability in agile context, it's output, being easy to understand and implement, it's granularity and the good accuracy it provides. Bayesian Networks main benefits are: the applicability in agile context, it's output, the fact that avoids the overfit to data, the capacity of generalization and also the good accuracy that achieves.

4.4 Validation

After having selected a method, we will validate it by building a tool based on this method and putting this tool to test, against other tools and methods, revised in the previous chapters, using some of the most used validation metrics, already presented above. This metrics are Magnitude of Relative Error (MRE), the Mean Magnitude of Relative Error (MMRE) , the Percentage Error (PE) and the pred(x), percentage of estimates that are within x% of the actual value.

For validating the tool, we will use real projects data from the PROMISE[1] database that contains several datasets of projects, such as the CocomoNasa dataset with 60 projects, the Desharnais dataset with 81 projects or the Cocomo81 dataset with 63 projects.

[1]PROMISE Database

This tool will also be integrated with software projects development management tools, like Jira[2] or Gitlab[3]. This will allow our tool to be tested by users, helping in the validation of the method.

4.5 Workplan

The workplan proposed is represented in Figure 4.1.

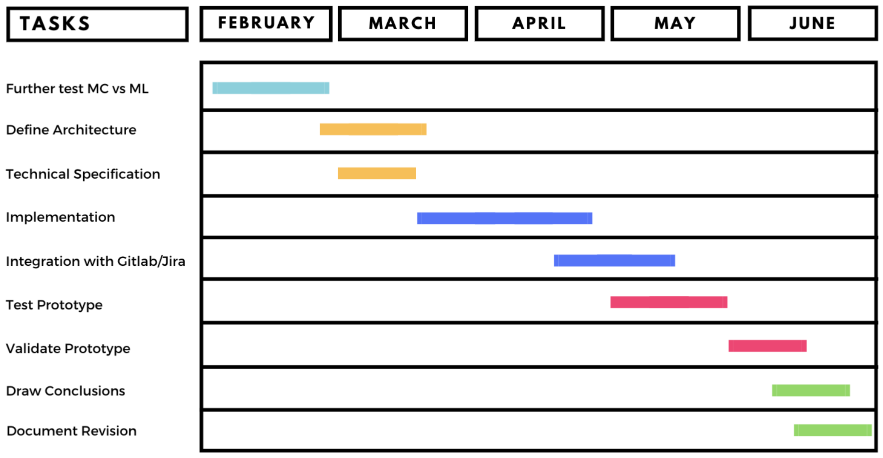


Figure 4.1: Proposed Workplan

In the "*Further testMC vs ML*" phase, we will do the practical analyses of both methods and decide which one is the most promising to follow, something we weren't able to do with the revision of the literature.

On "*Define Architecture*" and "*Technical Specification*", we will specify the architecture of the solution to implement and it's define it's technical specifications.

In the "*Implementation*" and "*Integration with Gitlab/Jira*" phase, we will develop the solution and integrate it with software development management tools, like Gitlab or Jira.

On "*Test Prototype*" and "*Validate Prototype*", we will test the solution against other tools and methods and analyse it's benefits and limitations. We will also validate the results provided by the solution.

Finally, in "*Draw Conclusions*" and "*Document revision*", we will draw conclusion about the solution and further work and prepare the final document for presentation.

This workplan is subject to alteration due to complications that could arise during the development phase.

[2]Jira
[3]Gitlab

Chapter 5

Conclusion

In conclusion, the document presented, analyses and reports important notions that set the tone for the rest of the dissertation, presents a detailed report of some of the most used metrics,both for measurement of input/output and for validation. It analyses some of the most used methods and reports their major benefits, limitations and make a comparison between them, highlighting the most promising methods. It was also concluded that the literature didn't presented enough information to compare some of the methods, and because of this, in the future, more practical comparisons of this methods will be made, in order to select one of them and proceed to validate it.

By the end of the dissertation, it's expected to have a tool based on machine learning or on a Monte Carlo model, that can be integrated with software development management tools and have a complete analysis and validation of the benefits of this method in relation to others.

Since the beginning of the dissertation project, a complete state of the art analysis, comparing different methods and its benefits in different situations, was already achieved and presented.

References

[1] Mohamed Abouelela and Luigi Benedicenti. Bayesian network based xp process modelling. *International Journal of Software Engineering & Applications*, July 2010.

[2] Deepak Ahlawat and Rshma Chawla. Software development effort estimation using fuzzy logic framework - an implementation. *IRD India*, June 2015.

[3] Iman Attarzadeh and Siew Hock Ow. Software development effort estimation based on a new fuzzy logic model. *International Journal of Computer Theory and Engineering*, October 2009.

[4] Thomas Betts. Web-based monte carlo simulation for agile estimation. Available at https://mobilemonitoringsolutions.com/web-based-monte-carlo-simulation-for-agile-estimation/, 2019.

[5] Roheet Bhatnagar, Vandana Bhattacharjee, and Mrinal Kanti Ghose. Software development effort estimation – neural network vs. regression modeling approach. *International Journal of Engineering Science and Technology*, 2010.

[6] Pawel Brodzinski. Estimation and forecasting. Available at https://www.agilealliance.org/estimation-and-forecasting//, 2015.

[7] Frederick P. Brooks. *The Mythical Man-Month: Essays on Software Engineering*. Addison-Wesley, 1995.

[8] Stipe Celar, Srdjana Dragicevic, and Mili Turicc. Bayesian network model for task effort estimation in agile software development. *The Journal of Systems and Software*, January 2017.

[9] Wei Lin Du, Danny Ho, and Luiz Fernando Capretz. A neuro-fuzzy model with seer-sem for software effort estimation. In *25th International Forum on COCOMO and Systems/Software Cost Modeling*, 2010.

[10] Richard E. Fairley. Recent advances in software estimation techniques. In ACM, editor, *ICSE '92: Proceedings of the 14th international conference on Software engineering*, page 382–391. ACM, 1992.

[11] Adrian Fittolani. Agile project forecasting – the monte carlo method. Scrumage. Available at http://scrumage.com/blog/2015/09/agile-project-forecasting-the-monte-carlo-method/, 2015.

[12] Andrew Gray and Stephen G. MacDonell. Applications of fuzzy logic to software metric models for development effort estimation. In Computer Society Press, editor, *Proceedings of the Annual Meeting of the North American Fuzzy Information Processing Society*, pages 394–399. IEEE, 1997.

[13] James Grenning. Planning poker. April 2002.

[14] Omar Hidmi and Betul Erdogdu Sakar. Software development effort estimation using ensemble machine learning. *Journal of Computing, Communications & Instrumentation Engineering*, March 2017.

[15] Sultan Mohiuddin Hydarali. *Comparison of Artificial Neural Network and regression models for estimating Software development effort*. PhD thesis, University of Limerick, 2019.

[16] Ali Idri, Fatima azzahra Amazal, and Alain Abran. Analogy-based software development effort estimation: A systematic mapping and review. *Elsevier*, July 2014.

[17] Ali Idri, Taghi M. Khoshgoftaar, and Alain Abran. Can neural networks be easily interpreted in software cost estimation? In IEEE, editor, *2002 IEEE World Congress on Computational Intelligence*. IEEE, 2002.

[18] Ali Idri, Taghi M. Khoshgoftaar, and Alain Abran. Estimating software project effort by analogy based on linguistic values. In IEEE, editor, *Proceedings Eighth IEEE Symposium on Software Metrics*. IEEE, 2002.

[19] Galorath Incorporated. *SEER-SEM Product Brief*. Galorath Incorporated, 2011.

[20] Ron Jeffries. The noestimates movement. Available at https://ronjeffries.com/xprog/articles/the-noestimates-movement/, 2013.

[21] Magne Jørgensen. Forecasting of software development work effort: Evidence on expert judgement and formal models. *International Journal of Forecasting 23*, 2007.

[22] Magne Jørgensen, Barry Boehm, and Stan Rifken. Software development effort estimation: Formal models or expert judgment? *IEEE Software*, May 2009.

[23] Magne Jørgensen, Tore Dybå, and Helen Sharp. What we do and don't know about software development effort estimation. *Voice of Evidence*, April 2014.

[24] Liz Keogh. The estimates in #noestimates. Available at https://lizkeogh.com/2015/05/01/the-estimates-in-noestimates/, 2015.

[25] Ekrem Kocaguneli, Ayse Tosun, and Ayse Bener. Ai-based models for software effort estimation. In IEEE, editor, *36th EUROMICRO Conference on Software Engineering and Advanced Applications*. IEEE, 2010.

[26] Troy Magennis. Managing software development risk using modeling and monte carlo simulation. In *Lean Software and Systems Conference 2012*, 2011.

[27] Steve McConnell. *Software Estimation: Demystifying the Black Art*. Microsoft Press, 2006.

[28] Darko Milicic. *Applying COCOMO II - A case study*. PhD thesis, Blekinge Institute of Technology, 2004.

[29] Dan North. Monte python simulation: Misunderstanding monte carlo. Available at https://dannorth.net/2018/09/04/monte-python-simulation/, 2018.

[30] James Rodger Parag Pendharkar and Girish Subramanian. A probabilistic model for predicting software development effort. *IEEE Transactions on Software Engineering 31*, July 2005.

[31] Mirko Perkusich, Hyggo Oliveira De Almeida, and Angelo Perkusich. A model to detect problems on scrum-based software development projects. In ACM, editor, *SAC '13: Proceedings of the 28th Annual ACM Symposium on Applied Computing*, page 1037–1042. ACM, 2013.

[32] Shashank Mouli Satapathy, Aditi Panda, and Santanu Kumar Rath. Story point approach based agile software effort estimation using various svr kernel methods. In SEKE, editor, *The 26th International Conference on Software Engineering and Knowledge Engineering*. SEKE, 2014.

[33] Christopher Schofield. *An Empirical Investigation into Software Effort Estimation by Analogy*. PhD thesis, Bournemouth University, 1998.

[34] Vishal Sharma and Harsh Kumar Verma. Optimized fuzzy logic based framework for effort estimation in software development. *International Journal of Computer Science Issues*, March 2010.

[35] Alberto Sillitti, Giancarlo Succi, Witold Pedrycz, Raimund Moser, and Pekka Abrahamsson. Effort prediction in iterative software development processes – incremental versus global prediction models. In IEEE, editor, *First International Symposium on Empirical Software Engineering and Measurement*. IEEE, 2007.

[36] Amit Sinhal and Bhupendra Verma. Software development effort estimation: A review. *International Journal of Advanced Research in Computer Science and Software Engineering*, June 2013.

[37] Andrew Stellman and Jennifer Greene. Applied software project management: Estimation. Available at https://www.stellman-greene.com/LectureNotes/03%20estimation.pdf.

[38] Derya Toka and Oktay Turetken. Accuracy of contemporary parametric software estimation models: A comparative analysis. In IEEE, editor, *39th Euromicro Conference Series on Software Engineering and Advanced Applications*. IEEE, 2013.

[39] Muhammad Usman, Emilia Mendes, Francila Weidt, and Ricardo Britto. Effort estimation in agile software development: A systematic literature review. In ACM, editor, *PROMISE '14: Proceedings of the 10th International Conference on Predictive Models in Software Engineering*, page 82–91. ACM, 2014.

[40] Juanjuan Zang. Agile estimation with monte carlo simulation. In Springer, editor, *Agile Processes in Software Engineering and Extreme Programming*, pages 172–179. Springer, 2008.