POLITECNICO DI TORINO

Department of Management and Production Engineering

Master of Science Thesis

Improving SCRUM Project Duration Forecasting through Learning Curve Theory

Supervisors:

Prof. Alberto De Marco

Dott. Filippo Maria Ottaviani

Candidate:

Elisa Alfieri

Academic year 2021-2022

Abstract

My thesis focuses on the Project Management applied in the Software industry. I will go throw the history of Project Management from the traditional methodologies to the more recent ones applied in software development, focusing my attention on the SCRUM framework. More attention will be paid to the main different techniques to manage and predict the costs and the duration of a software. In particular: the Earned Value Model, the Burndown charter and the Putnam model. In this thesis, I am reporting the direct application of the Scrum methodology during my curricular internship in an IT company in Sophia Antipolis, France. The internship research focuses on data collected in 6 months of work, in which I was part of an Agile team having the purpose of developing an internal project for the company. The product delivered is a web application used by Business Managers and Human Resources of the company to manage consultants. Our team was composed by only inters and we were simulating a real Scrum team with the occasion to turn all the roles of this methodologies. Although I worked for the whole duration also as developer, I will not detail the development of the project, but I will focus my attention on the data I have collected in these months and the application of the Scrum theory in a real working environment. Thanks to these data it was possible to study the projects performance in an Agile setting. I apply the different techniques listed before to monitor and forecast duration and cost on the project I was working on. In particular, I will monitor the project throw the burndown charter and the Earned value model and I will obtain a graphical and mathematical forecast of performance. The data I have collected from the internship will be useful also to criticize the Putnam resource allocation model and propose a Revised theory. The Putnam model is seen as pioneering work in the field of software process modelling. Nevertheless, a hypothesis made by Putnam has been investigated and in this thesis, I am proposing a variation of the model. The main difference focuses on the concept of the Productivity. While Putnam considers it as constant in his equation, I will suppose that it is increasing during time following the learning curve. This variation implies a different progress of the project in terms of time and effort. In the Conclusion, the forecast obtained with EMV, Burndown charter, Putnam and the Revised model has been compared with the result data of the project at the end of the internship in order to investigate which model better estimate the duration in a SCRUM environment.

Index

1. Introduction: Project Management approaches

1.1 Introduction to Project Management

A Project is undertaken to achieve planned objectives, which could be defined in terms of outputs or benefits. It has a defined beginning and end in time, a defined scope and resources with specific set of operations designed to accomplish a singular goal. The project manager is the person in charge of achieving the project objectives. Project management is the application of knowledge, skills, tools, and techniques to project activities to meet the project requirements. It is often closely associated with engineering projects, which typically have a complex set of components that have to be completed and assembled in order to create a functioning product.

The key to successful project management is to focus on the 4p:

1) **People**

Identifying the roles people play in almost any given project is the first step to a successful project. People are the primary resource on every project, and a well-managed team can greatly increase the chances of success. Some of the different roles people play in project management includes project manager, project team members, sponsors, stakeholders, business analysts and information technology developers.

2) **Product**

As the name implies, this is the deliverable of the project. The project manager should define the product scope to ensure a successful outcome, control “scope creep”; as well as technical hurdles that he or she may encounter.

With that said, the product does not necessarily need to be restricted to software; project management can be applied to all industries with software development being one of the key elements. The product of a project can also be something that is intangible; such as moving a company to a new headquarters or setting up a new company as a registered legal entity to commence trading activities on day one.

3) **Process**

The third P of project management is Process. Project managers and team members should have a methodology and plan that outlines their approach. Without a clearly

defined process, team members will not know what to do and when to carry out project activities. However, this problem can be avoided through comprehensive early stage process planning. Using the right process will increase the project execution success rate that meets its original goals and objectives.

4) **Project**

The fourth and final P of project management is Project. This is where the project manager's roles and responsibilities come into play. He or she must guide team members to achieve the project's goals and objectives. The project manager must delegate tasks, help team members when needed, and ultimately strive to accomplish all requirements set forth in the project scope.

1.1.1 Project definition

Project in general refers to a new endeavor with specific objective and varies so widely that it is very difficult to precisely define it. Some of the commonly quoted definitions are as follows. Project is a temporary endeavor undertaken to create a unique product or service or result.

(AMERICAN National Standard ANSI/PMI99-001-2004)

Project is a unique process, consist of a set of coordinated and controlled activities with start and finish dates, undertaken to achieve an objective confirming to specific requirements, including the constraints of time cost and resource.

(ISO10006)

It is important to note that, regardless of the organization and sector of reference, a project is characterized by some distinctive elements:

- Unique in nature
- Have definite objectives (goals) to achieve.
- Requires set of resources.
- Have a specific time frame for completion with a definite start and finish.
- Involves risk and uncertainty.
- Requires cross-functional teams and interdisciplinary approach.

The specificity of the objective determines the exceptional nature of the project compared to ordinary activities and therefore the absence of previous experience. The non-recurring nature of the project implies the definition of a start date, the end of the project is established by the achievement of the objective, by the exhaustion of resources, by the dissolution of the project group or by the closing of the "window of opportunity", in fact the time limitation also manifests itself in the need to carry out the activities within precise time windows not dictated by the availability of resources, but linked to the object of the project. A project is usually deemed to be a success if it achieves the objectives according to their acceptance criteria, within an agreed timescale and budget.

1.1.2 Project Life cycle

The project manager and the team have one common goal: to carry out the work of the project for the purpose of meeting the project's objectives. Every project has beginnings, a middle period during which activities move the project toward completion, and an ending (either successful or unsuccessful). A standard project typically has the following four major phases: initiation, planning, implementation, and closure. Taken together, these phases represent the path a project takes from the beginning to its end and are generally referred to as the project life cycle.

- Initiation phase

During the first of these phases, the initiation phase, the project objective or need is identified; this can be a business problem or opportunity. An appropriate response to the need is documented in a business case with recommended solution options. A feasibility study is conducted to investigate whether each option addresses the project objectives, and a final recommended solution is determined. Issues of feasibility ("can we do the project?") and justification ("should we do the project?") are addressed. Once the recommended solution is approved, a project is initiated to deliver the approved solution and a project manager is appointed. The major deliverables and the participating work groups are identified and the project team begins to take shape. Approval is then sought by the project manager to move on the detailed planning phase.

- Planning phase

The next phase, the planning phase, is where the project solution is further developed in as much detail as possible and you plan the steps necessary to meet the project's objective. In this step, the team identifies all of the work to be done. The project's tasks and resource requirements are identified, along with the strategy for producing them. This is also referred to as scope management. A project plan is created outlining the activities, tasks, dependencies and timeframes. The project manager coordinates the preparation of a project budget; by providing cost estimates for the labor, equipment and materials costs. The budget is used to monitor and control cost expenditures during project implementation. Once the project team has identified the work, prepared the schedule and estimated the costs, the three fundamental components of the planning process are complete. This is an excellent time to identify and try to deal with anything that might pose a threat to the successful completion of the project. This is called risk management. In risk management, "high-threat" potential problems are identified along with the action that is to be taken on each high threat potential problem, either to reduce the probability that the problem will occur or to reduce the impact on the project if it does occur. This is also a good time to identify all project stakeholders, and to establish a communication plan describing the information needed and the delivery method to be used to keep the stakeholders informed. Finally, you will want to document a quality plan; providing quality targets, assurance, and control measures along with an acceptance plan; listing the criteria to be met to gain customer acceptance. At this point, the project would have been planned in detail and is ready to be executed.

- Implementation phase

During the third phase, the implementation phase, the project plan is put into motion and performs the work of the project. It is important to maintain control and communicate as needed during implementation. Progress is continuously monitored and appropriate adjustments are made and recorded as variances from the original plan. In any project a project manager will spend most of their time in this step. During project implementation, people are carrying out the tasks and progress information is being reported through regular team meetings. The project manager

uses this information to maintain control over the direction of the project by measuring the performance of the project activities comparing the results with the project plan and takes corrective action as needed. The first course of action should always be to bring the project back on course, i.e., to return it to the original plan. If that cannot happen, the team should record variations from the original plan and record and publish modifications to the plan. Throughout this step, project sponsors and other key stakeholders should be kept informed of project status according to the agreed upon frequency and format. The plan should be updated and published on a regular basis. Status reports should always emphasize the anticipated end point in terms of cost, schedule and quality of deliverables. Each project deliverable produced should be reviewed for quality and measured against the acceptance criteria. Once all of the deliverables have been produced and the customer has accepted the final solution, the project is ready for closure.

- Closing phase

During the final closure, or completion phase, the emphasis is on releasing the final deliverables to the customer, handing over project documentation to the business, terminating supplier contracts, releasing project resources and communicating the closure of the project to all stakeholders. The last remaining step is to conduct lessons learned studies; to examine what went well and what didn't. Through this type of analysis, the wisdom of experience is transferred back to the project organization, which will help future project teams.

1.1.2.1 Characteristics of Life Cycles

There are some common characteristics of project life cycles that can influence how an organization approaches complex or high-risk projects.

- Cost and staffing curve.

In general, costs and staffing for a predictive project life cycle are low during the earlier Process Groups—Initiating and the first pass at Planning—because few team members are on board and the project is using fewer resources. Costs increase quickly during the later Process Groups of Executing and Monitoring and Controlling,

when most of the project work is being done. Costs begin to decrease at the end of Executing and Monitoring and Controlling as deliverables are completed and resources are released. Costs decrease more quickly during the final Process Group, Closing.

The life cycle of a project from start to completion follows either a “S” shaped path or a “J “ shaped path (*Figure 1* and *Figure 2*). In “S” shape path the progress is slow at the starting and terminal phase and is fast in the implementation phase. At the beginning detailed sectoral planning and coordination among various implementing agencies etc. makes progress slow and similarly towards termination, creating institutional arrangement for transfer and maintenance of assets to the stakeholders progresses slowly.

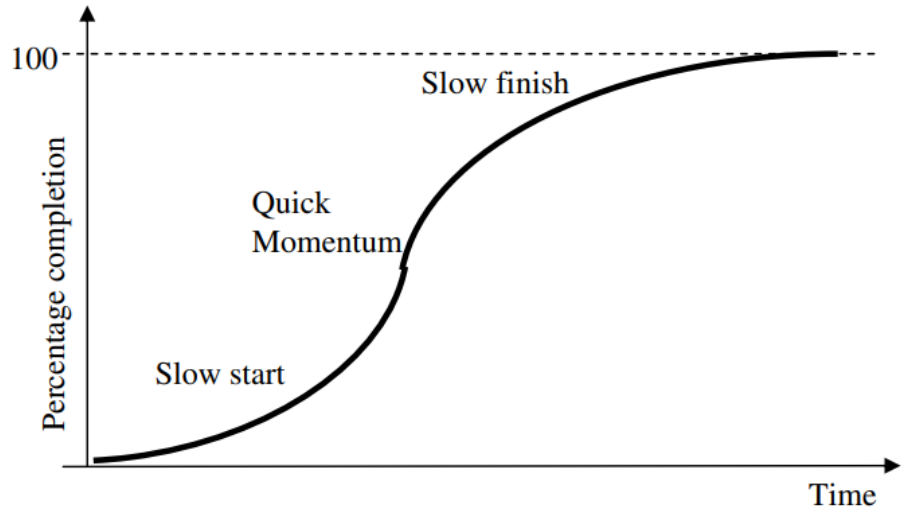

Figure 1. Project life path – “S” shape

In “J” type cycle path the progress in beginning is slow and as the time moves on the progress of the project improves at fast rate. Example, in a developing an energy plantation. In this the land preparation progresses slowly and as soon as the land and seedling are transplantation is undertaken. This is shown in *Figure 2*.

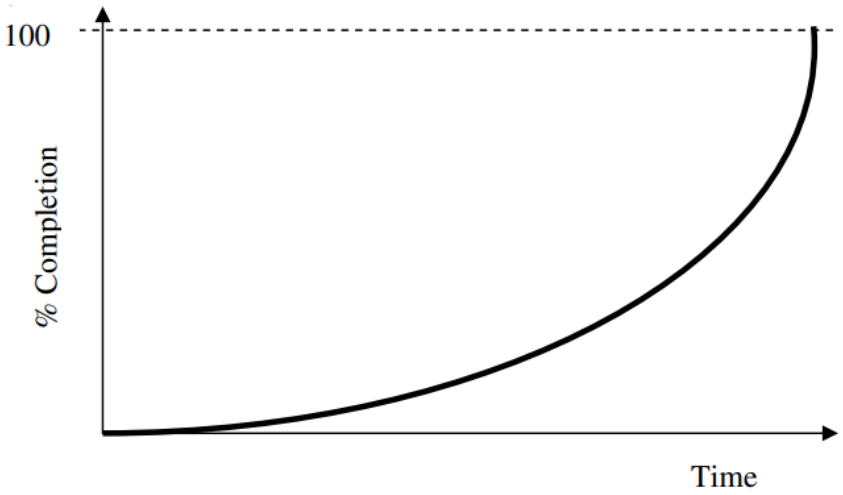

Figure 2. Project life cycle path - "J" Shape

- Risk/uncertainty curve.

In general, uncertainty about a project's ability to meet its objectives decreases during the project life cycle. As the project proceeds, decisions are made, knowledge and experience increase, and steps can be taken to manage risk more effectively. For example, a predictive project aiming at producing a new type of manufacturing equipment has more risk during Initiating and Planning, because the sponsor and the project team don't know how or if the objectives can actually be achieved in proposed designs and if the designs can be translated into production within scope, budget, and time frames. Once a project is accepted by the customer, no further risk exists. On an agile/hybrid project, as phases are completed and their deliverables are accepted, the level of risk and uncertainty gradually is reduced.

- Cost of changes curve.

For a predictive project, the least expensive time to change scope or product characteristics is early on, before work has been done and resources spent. As the project work progresses, the ability to make changes without significantly affecting cost and schedule (and possibly other measures such as quality and project team motivation and engagement) decreases. Changing course may mean that investments made as a result of earlier decisions have been wasted. Work based on a mistake or incorporating a flawed element has to be redone. Changes may have a "ripple effect" on work that has already been completed. A team working on new

manufacturing equipment could handle this issue by proving their design concepts through a prototype test before proceeding to building the equipment. (Prototyping is an example of an agile mindset.) This would decrease the risk of later changes to project baselines. Agile, by design, works to keep the cost of changes lower throughout a project so that changes can be accommodated even late in the project. The relationship between the anticipated curves for risk and the cost of changes is illustrated *Figure 3*.

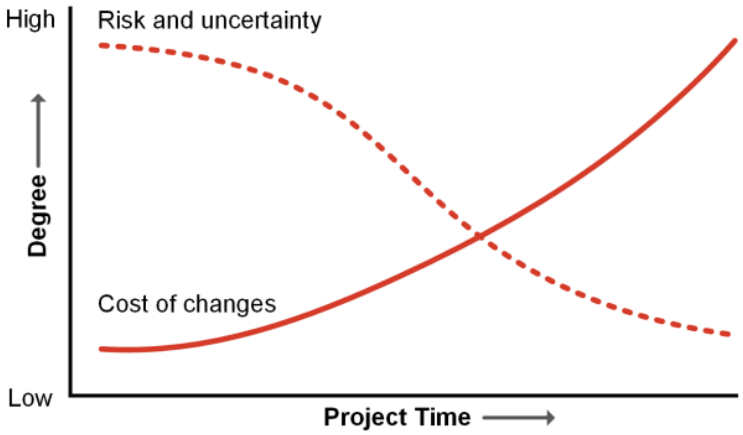

Figure 3. Influence of risks and cost of changes over time

1.1.3 Project Performance dimensions

In the mid 1980s Dr. Martin Barnes created the Triangle of objectives shown in *Figure 1*. Called also the "Scope Triangle" or the "Iron Triangle" this shows the trade-offs inherent in any project. The triangle demonstrates that scope cost and time are interrelated. The constant effort to balance these three factors impacts on the quality of the project. High-quality projects deliver the requested product, service or result within the established scope, within the set time and within the limits of the defined budget. Time is the available time to deliver the project, cost represents the amount of money or resources available and scope represents the fit-to-purpose that the project must achieve to be a success.

The variation of even one of the three factors of the triple constraint implies that at least one other factor is affected. For example, if the scope is enlarged, project would require more time for completion and the costs would also go up. If time is reduced

the scope and cost would also reduce. Similarly, any change in cost would be reflected in scope and time. Successful completion of the project would require accomplishment of specified goals within scheduled time and budget. In recent years a forth dimension, stakeholder satisfaction, is added to the project. However, the other school of management argues that this dimension is an inherent part of the scope of the project that defines the specifications to which the project is required to be implemented. Therefore, the performance of a project is measured by the degree to which these three parameters (scope, time and cost) are achieved.

Mathematically: Performance = f (Scope, Cost, Time)

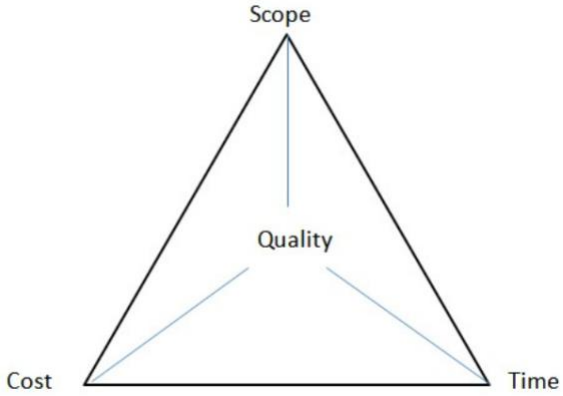

Figure 4. Representation of the Triangle of objectives

Interest increases when two of the points are fixed. Normally this occurs when costs are fixed and there is a definite deadline for delivery, in this case you can just cut functionality, as long as the core requirements remain. Additional functionality can always go into "the next release", but the core functionality has to be always high prioritized. Project managers are also responsible for managing projects also taking into account project risks, i.e. uncertain events or conditions which, if they occur, have a positive or negative effect on at least one of the project objectives.

A phenomenon known as "scope creep" can be linked to the triangle too. Scope creep is the almost unstoppable tendency a project has to accumulate new functionality. Some scope creep is inevitable since, early on, your project will be poorly defined and will need to evolve. A large amount of scope creep however can be disastrous. This is represented by the quality of the project.

More requirements fulfilled means a better-quality product and in this situation the Project Manager has three options:

1. Add time - delay the project to give you more time to add the functionality
2. Add cost - recruit, hire or acquire more people to do the extra work
3. Cut quality - trade off some non-essential requirements for the new requirements

1.2 Traditional project management

Project management is applied in today's business world to a variety of different projects. Principles established in the 1950's have prescribed that the methods and procedures should be applied to every project in a uniform way. Such uniform implementation should ensure robustness and applicability to a wide range of projects, from the simple and small projects to most complex and large ones. The basic idea behind that traditional, rational and normative approach is that projects are relatively simple, predictable and linear with clearly defined boundaries which all makes it easy to plan in detail and follow that plan without many changes. The ultimate goal of the traditional project management approach is optimization and efficiency in following initial detailed project plan, or, having said in usual way, to finalize project within planned time, budget, and scope.

Traditional project management involves very disciplined and deliberate planning and control methods. With this approach, distinct project life cycle phases are easily recognizable. Tasks are completed one after another in an orderly sequence, requiring a significant part of the project to be planned up front. For example, in a construction project, the team needs to determine requirements, design and plan for the entire building, and not just incremental components, in order to understand the full scope of the effort. Traditional project management assumes that events affecting the project are predictable and that tools and activities are well understood. In addition, with traditional project management, once a phase is complete, it is assumed that it will not be revisited. The strengths of this approach are that it lays out the steps for development and stresses the importance of requirements. The

limitations are that projects rarely follow the sequential flow, and clients usually find it difficult to completely state all requirements early in the project.

1.2.1 Critical path method

Critical Path Methods (CPM), which is a subcategory in the traditional project management is based on the idea that in order to achieve a task, the previous task should have been done already. The past is the critical one if any postponement in a given task in the that sequence of activities will results in a delay of the project. It was created in the late 50's, El DuPont de Nemours Company, an American chemical company, was seriously falling behind its schedule, and they needed something that would get them back on track. They came up with a solution to divide their project into thousands of tasks, measure the time each task will take, and how assess critical they are to the entire process. They called this technique Critical Path Method (CPM). CPM was first tested in 1958 in a project to construct a new chemical plant and has ever been one of the most frequently used techniques of project management. Any project can have one or more critical paths and, knowing the critical paths, allow managers to set priorities and allocate resources more efficiently. This process helps determine the most important tasks to perform and prioritize them accordingly. Such methodology supports rescheduling, in favor of optimizing performance of the team, with the objective of getting the jobs done on time.

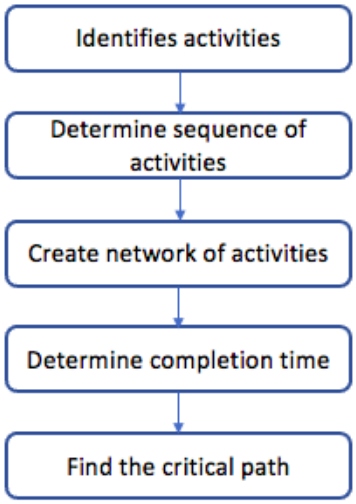

Figure 5. CPM process

A critical path method includes the following steps:

1. Identifying activities

By using the project scope, you can break the work structure into a list of activities and identify them by name and coding; all activities must have duration and target date.

2. Determining sequence of activities

This is the most important step as it gives a clear view of the connection between the activities and helps you establish dependencies as some activities will depend on the completion of others.

3. Creating a network of activities

Once you determined how activities depend on each other you can create the network diagram, or critical path analysis chart; it allows you to use arrows to connect the activities based on their dependence.

4. Determining completion time for each activity

By estimating how much time each activity will take will help the Project Manager to determine the time needed to complete the entire project.

5. Finding the critical path

A network of activities will help you create the longest sequence of activities on the path or the critical path using these parameters:

- Early Start ES; earliest time to start a certain activity providing that the preceding one is completed.
- Early Finish EF; earliest time necessary to finish activity
- Late Finish LF; latest time necessary to finish the project without delays
- Late Start LS; latest start date when the project can start without project delays

If there is a delay in any task on the critical path, the whole project will have to be delayed. The critical path is the path where there can be no delays. Naturally, not all the project activities are equally important, in fact, while some have a huge impact on the critical path and are therefore critical, others don't make much difference to the project if they are delayed.
The critical path method helps us determine which activities are "critical and which have "total float". However, if any of the floating activities get seriously delayed, they can become critical and delay the entire project.

1.2.2 Critical chain project management (CCPM)

Critical chain project management is another subcategory in the traditional, sequential methodologies. Dr. Eliyahu M. Goldratt developed the concept of CCPM in 1997. CCPM relates very closely to one of Dr. Goldratt's other theories—the theory of constraints. The theory of constraints helps you identify key bottlenecks or limiting factors standing in the way of your project's completion. The idea of this latter is that every project has one main constraint and this constraint has the ability to disrupt the entire project by breaking the weakest chain. In the Critical Chain Project Management (CCPM) the key project constraints are the finite resources such as people, equipment, and physical space. The goal of CCM is to eliminate project schedule delays due to uncertainties, overestimation of task duration, and wasted internal buffers. It is very similar to the critical path method, both focus on schedule development and estimation and furthermore, like CPM, you must finish a task before starting the other. No multitasking is accepted. Its objective is to prioritize organization and efficiency of the project by having enough resources, that why every task should properly be finished before entering a next phase.
There are also few differences between the two methods, the main points of inequality are:

- The critical path (CPM) focuses on managing tasks, whereas the critical chain concentrates on managing the resources and buffer.
- The critical path method (CPM) is more of an estimation than the critical chain, as it assumes that every resource will be available when needed. On the

contrary, the critical chain takes that limited resource and uses the readily resources to plan out a realistic schedule.

- In critical chain method (CPM), the buffer (added extra time) is for the whole project, whereas in critical path is for the individual activities.
- Therefore, there are delays of schedule in critical chain on non-critical activities whereas in critical path the project members can immediately start on non-critical activities.
- In the critical path, work increases if an activity has extra time, which isn't the case in a critical chain with the real-time length and buffer for every activity
- In critical path, the float or slack (added extra time) to each activity of the project were misused and lead to delays in projects.

A critical chain project network strives to keep resources levelled and requires that they be flexible in start times. This method accepts the inherent uncertainty of estimates, but instead of letting everyone give themselves their own contingency to safeguard their individual estimates, the method rids the individual estimates of their local contingencies.

CCPM applies three types of buffers:

- Project buffer is placed in between the last task and the project's final date, particularly after the last task and before the end date. It is basically used as the plan B for the critical chain. That's where the delays can be found.
- Feeding buffers prevent non-critical chain tasks from delaying the start of critical chain tasks.
- Resource buffers are more specific feeding buffers and are placed to guarantee resource availability for each task on the critical chain.

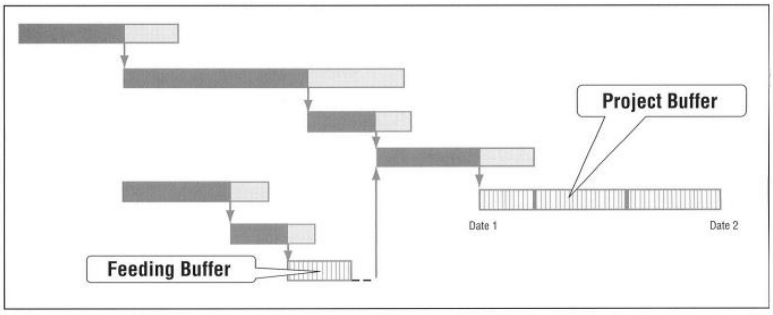

Figure 2. Project Network With Feeding Buffer

Figure 6. Types of buffers

1. Identify the critical chain

First of all, there is the need to identify the most important tasks, as well as the tasks that will take the longest to complete. These tasks will become your critical path. To help you identify your critical path, we recommend creating a work breakdown structure. This structure breaks down large projects into smaller, more manageable pieces. In CCPM, a work breakdown structure can help you determine where you will need the most resources. Additionally, it will show which tasks are going to take the most time. For a successful CCPM process, begin your project with the most important tasks first and then work in descending order.

2. Determine Resource Constraints to Create the Critical Chain

Critical Chain Project Management focuses on resources — so consider any constraints you might experience as you assign employees, workstations, materials, etc. to tasks. To create your Critical Chain, you will need to determine the set of activities that, if delayed, will extend the end date of the project by looking at resource availability. This path, along with the resource constraints it contains, is the Critical Chain. All tasks not in the critical chain are part of a feeding chain.

3. Eliminate Multitasking

When employees switch between different tasks, productivity drops, and task durations increase. And, ultimately, team morale decreases as the team members try to keep the project moving forward. The Critical Chain Project Management process keeps employees focused on fewer items at a time, which allows teams to execute projects faster. Moreover, In critical chain methodology, it’s important to keep your team focused on individual tasks as the practice will lead to more productive, harmonious, collaborative, and innovative behaviors. All these factors contribute to timely task completion and efficiency. For successful CCPM, you want to ensure your team has enough on their plate to stay focused, but not enough that they will have to multitask in order to get things done.

4. Create 50/50-time estimates

Successful critical chain processes cut the estimated time needed for projects in half. The idea isn't to stress out the team members; it's to avoid wasted time and push them toward a more efficient timeline. Sometimes employees procrastinate, waiting until the last possible moment to start or even stretching out a task to fill time. By cutting the time needed for a task in half, you create a sense of urgency in your team. This practice will push them to stay focused and finish their tasks on time.

5. Insert Buffers

After you cut the estimated time needed by 50%, that 50% is then used as a buffer. It acts as a shock absorber for the project should a task take longer than anticipated to complete. One study showed that implementing project buffers with the CCPM process leads to employees finishing projects 25% faster. To use buffers the PM needs to decide where buffers should be inserted along the Critical Chain and which type of buffer to use. There is the need also to determine the appropriate size of buffer. A general rule of thumb is that the bigger the risk or uncertainty an activity entails, the bigger the buffer should be.

6. Create a Detailed Project Model

The use of the Critical Chain methodology likely implies to take on a large and complicate project. To ensure timely project completion, create a detailed project model that your entire team can use. The model allows your team to see how well the project is progressing. The project model should include time estimates, task descriptions, assigned resources, time buffers, and finish dates.

7. Oversee the Project

Continue to monitor the project, check for completed milestones to ensure the project remains on track. Manage buffers and gain the information necessary for controlling the plan and take recovery actions if needed.

1.2.3 PERT

PERT Method (Program Evaluation and Review Technique) was first developed by the US Navy SPO (Special Projects Office) in 1967 during the Polaris missile development program then it was applied to the other industries. PERT stands for program evaluation and review technique. It is a statistical technique used for getting highly accurate time estimates for complex projects. It focuses on events and milestones, and the PERT chart emphasizes this by making those the nodes in the wireframe. Unlike other methods, it is an event-oriented technique that uses three-point estimation approach for each task. Any task filled with uncertainties can have a wide range of estimates in which the task actually will get completed. Uncertainties include both favorable conditions (opportunities) as well as unfavorable conditions (threats). The 3 points of the estimate are: Optimistic, Most Likely, Pessimistic. The optimistic time estimate is the best-case scenario, the case in which every person in the project finished his part perfectly and there is no need for extra parts and no time hang up anywhere in the process. In the most likely scenario is the average completion time for a project. Not everything goes according to plan on the project, but it is not the worst-case scenario either. For the most part, everyone completes her part of the project on time, even if there was the occasional slow down. The worst case scenario for project completion is when a project was derailed due to factors outside of your control.

PERT calculates a weighted average as the PERT estimate by using the formula:

$$PERT = \frac{O + 4M + P}{6}$$

That means that we are weighting the most likely estimate by a factor of four (4) and then determining the average of the weighted most likely time, the best-case scenario and the worst-case scenario. We use a weighted average calculation because statistically we find that the largest portion of a randomly occurring population of data will be found close to the mean as can be seen from the following bell curve, also called a normal distribution curve. This bell curve in *Figure 7*, shows us that if we divide the data into 6 equal portions, then by far the majority of the instances will be found in the 2 portions next to and on both sides of the middle line (mean) while the probability of the occurrence falling in either the optimistic or

pessimistic block is much lower. We divide the graph into 6 equal blocks by calculating the standard deviation also called the sigma (identified by the sigma symbol “σ”). By using the simple formula Sigma = (Pessimistic – Optimistic) / 6 we divide the graph in 6 equal blocks. From this we can see that the PERT estimating formula using (Optimistic + (4 X Most Likely) + Pessimistic)/6 is based on this natural distribution. This also gives us a tool with which we can quantify the probability of how the data will be distributed. Statistically we know that:

- 68% of the data will be found in one standard deviation from the mean in other words between the mean minus one standard deviation and the mean plus one standard deviation. We can also say that there is a 68% probability that the activity will be completed within one standard deviation from the mean.
- We also see that 95% of the data will be found within two standard deviations or, put another way, that we can say with 95% certainty that the activity will be completed withing the two sigma range.
- At three sigma there is a 99,7% probability.

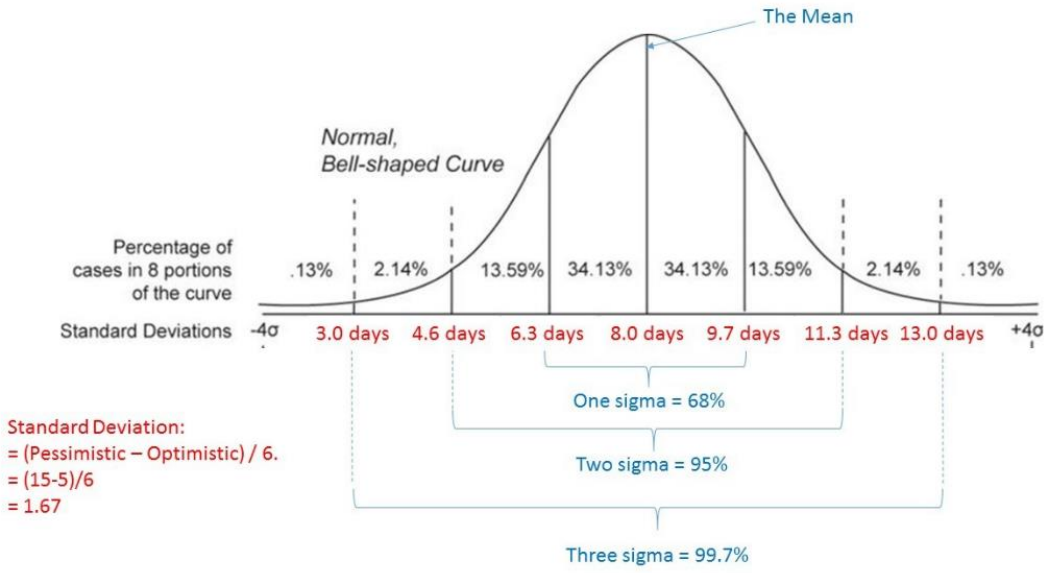

Figure 7. Normal distribution

Both CPM and PERT are complementary tools, and they are developed at roughly the same time. The two scheduling methods use a common approach for designing the network and for ascertaining its critical path. They are used in the successful completion of a project and hence used in conjunction with each other. Nevertheless, the truth is that CPM is different from PERT in a way that the latter concentrates on time while the former stresses on the time-cost trade-off. In the same manner, there are many differences between PERT and CPM, the main are reported:

- PERT is a probability model, it is that technique of project management which is used to manage uncertain (i.e., time is not known) activities of any project. On the other hand, CPM is a deterministic model, and it is that technique of project management which is used to manage only certain (i.e., time is known) activities of any project.
- PERT is an event-oriented technique which means that network is constructed on the basis of event. CPM is an activity-oriented technique which means that network is constructed on the basis of activities.
- In PERT there is no chance of crashing as there is no certainty of time, while in CPM there may be crashing because of certain time bound.
- PERT is suitable for high precision time estimation of projects that are unpredictable and whose activities are non-repetitive, and CPM is more suitable for projects that are predictable and whose activities are repetitive. Time estimation for such projects is made under reasonable forms.

1.2.4 Waterfall method

The Waterfall model was first presented in 1970 by American computer scientist Winston W. Royce—though he didn't actually use that term to describe it—in his article titled, "Managing the development of large software systems." The first mention of "Waterfall" is often attributed to a paper written by T.E. Bell and T.A. Thayer in 1976. Since then, the Waterfall approach has made an impact on many projects and project managers. It's still widely used across industries and has even inspired formalized education around project management. The Waterfall method is a traditional project management methodology that takes a well-defined project idea to completion through a sequential series of linear steps, tasks, and hand-offs. This

straightforward and somewhat rigid method uses early planning and estimation to define and document project requirements prior to executing on the work. The Waterfall methodology centers around a visual timeline—or Gantt chart—of your project. This makes it easy to see how long every task should take, who should be working on it, and what order work should be done in. Waterfall project management follows a linear process designed to deliver project quality and cost-efficiency. Each phase of the Waterfall process happens in sequential order, meaning one step must finish before the next one begins. You start at point A, finish that step, move on to step B, and continue that way until your project's complete. The Waterfall lifecycle doesn't allow for a ton of iteration unless it's planned. So, if you're working with a client, be very clear about how much time is scoped for feedback and iteration on your deliverables. Those steps will be built directly into your project plan. If a client wants to change the direction of your Waterfall project midstream, you'll face challenges with your project scope, budget, and deadline. That's because the Waterfall method is grouped by phases and tasks that depend wholly on previous tasks and decisions. The minute you go off track with the plan, things start to fall apart.

The Waterfall development process can be broken down into 6 key phases that are shown in *Figure 8*.

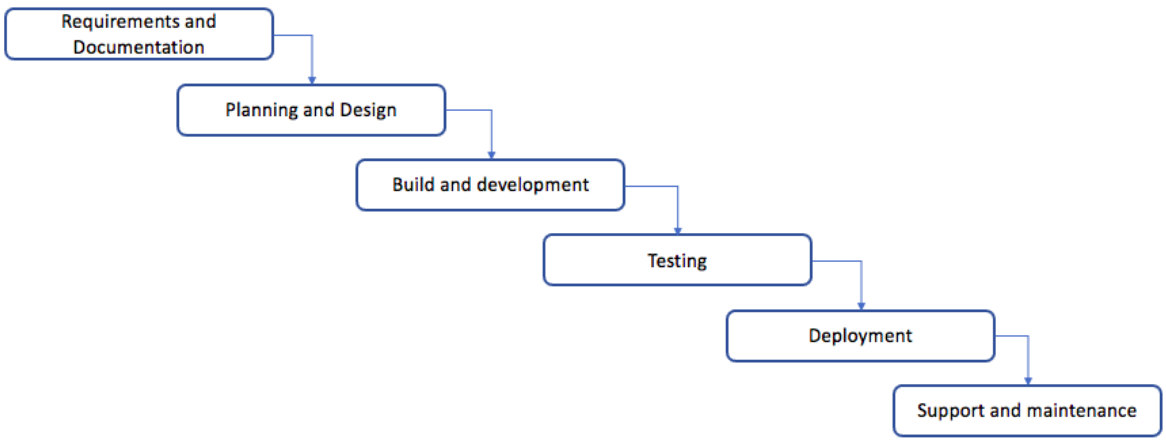

Figure 8. Waterfall Process

- Phase 1: Requirements gathering and documentation

The first step of any Waterfall project is to question and analyze business needs and understand project goals with a focus on documenting project requirements. This phase is critical to project success because it fully explains what's needed—in detail—to complete the project both at a high level and as it relates to each requirement, which will be tracked throughout the project.

- Phase 2: Planning and design

The second phase of the Waterfall lifecycle builds on the first step by creating an overall plan for what's being built. After all, you've got to know what you're designing before setting out to design it. In this Waterfall phase, the goal is to come away with a foundational design document everyone agrees on that act as a true north for your project. Once that plan's complete, you can hand it to a designer who will bring the plan to life. That said, it's important to remember to keep an eye on your project requirements and documentation so the design work can be handed over in the next phase to begin implementation.

- Phase 3: Build and development

This is where the real work begins. The project team starts building the actual product/ service. This is where the documentation you've created in the previous 2 steps proves critical, as it will guide your team to implement the design work. The project team may decide to conduct preliminary testing at this stage to ease the pain of rework or fixes in the testing phase. This is also the most time-consuming part of the process. Depending on the nature of the project, the implementation can be anything from writing source code to pouring the foundation. At this point, the team has an excellent understanding of project requirements.

- Phase 4: Testing

Once the deliverables are ready, the project team can start looking for bugs and other issues that may plague end users. This is usually the responsibility of the testing team. The verification phase ensures that the product/service works

according to initial requirements. The testing step carries the most risk in a Waterfall project because you just don't know what issues or defects will pop up and how they'll impact the timeline. Due diligence during the Requirements and Design helps the project team save time they'd have to spend fixing things after deployment. The better the documentation and project requirements, the shorter the testing phase gets. Testing usually concludes with a test report. Adding a buffer into your testing schedule can help ensure your team has adequate time to make fixes. The project is close to completion, so the team need to do everything to perfect it in the testing phase before it launches. Here are just a few activities testing might include:

- Review and check of the project requirements and goals
- Design review to ensure the integrity of the look and feel
- Review of usability

- Phase 5: Deployment

At this point, requirements have been met, the product is fully tested and approved, and everyone is confident your product is 100% ready to release. Depending on the type of product you're launching, you'll have a plan to ensure your deployment is smooth and drama-free. Be sure to discuss what the deployment or launch will look like far in advance of actually doing it. Working out the details early will enable you to approach your release day with a checklist and some confidence. It may feel like time to celebrate, but you're not done yet.

- Phase 6: Support and maintenance

In the final phase, the product/service goes live. Once the product is operational and in the hands of users, the project enters a maintenance phase. The team actively works on any outstanding issues missed during testing and pushes updates according to client feedback. The maintenance stage can last until the product or service is retired. The team may focus on corrective (issues and bugs), adaptive (new features), perfective (user requests), and preventive maintenance. The maintenance phase makes up a substantial part of the project lifecycle.

Waterfall project management has its roots in non-software industries like manufacturing and construction, where the system arose out of necessity. In these fields, project phases must happen sequentially. You can't put up drywall if you haven't framed a house. Likewise, it's impossible to revisit a phase. There's no good way to un-pour a concrete foundation. A project's requirements must be clear upfront, and everyone involved in a project must be well aware of those requirements. Each team member should also understand what their role will be in the project and what that role entails. All of this information must be thoroughly documented and then distributed to everyone on the project. It is useful to outline this information as a flowchart so the team can quickly understand and reference requirements as needed. Team members will refer to the documentation you provide throughout the process. When followed properly, this document makes clear precisely what is expected, thus guiding the creation of the product. It will also provide project milestones that will make it simple to determine progress. Consequently, thorough documentation is a priority in the waterfall project management methodology. Documentation should take place throughout every phase of the process, ensuring that everyone involved is on the same page despite the sequential progression of the project.

Subject matter expert Patrick Rockwell advises on the types of situations in which using the waterfall method can be beneficial.

"Though less common these days, when your end product's requirements are fixed yet time and money are variable, choose the waterfall method. I like to imagine a scientist doing research for a big company—through trial and error, he'll likely restart his whole process many times and at different stages to get the coveted final result. Through waterfall project management this behavior is anticipated and even preferred! This enables members to adjust and re-think their approach time and time again."

As Patrick mentions, waterfall project management can be problematic if the project requirements are not perfectly clear, which happens when a user has a general idea of what they want but can't nail down specifics. The waterfall system's linear nature is not suited to discovery, and the project will likely suffer without more specific

requirements. Late-stage testing makes any revision a serious undertaking. In fact, strict adherents to the waterfall system would argue that a need for revision means the product requirements were not clear, and therefore the project must return to stage one. This can be a serious problem in many industries, such as the ever-changing world of software. Because of its inability to adapt to change, the waterfall methodology is best suited to short projects that are well-defined from the beginning. If you are certain that the project requirements are static, then waterfall project management provides a straightforward way to push a project through a clearly defined process. It's simple to manage and easy to track.

1.3 The shift towards new management approaches

Even though traditional approach to project management emphasizes robustness as one of its advantages, prescribing that the same methods and techniques could be applied to all projects uniformly, it is increasingly mentioned as one of the crucial disadvantages of such approach. Today, increasing number of authors stress the fact that "one size does not fit all". Projects, same as business environments in general, become progressively complex, with higher number of tasks and complex interrelations, while traditional project management approach is based on mostly hierarchical and linear task relations and can not properly reflect all complexity and dynamics of today's projects. Furthermore, assumption that project is isolated from its environment causes the second major disadvantage of the traditional. Change in any form is the reality of today's business environments and the projects within those environments. Changes in the initial plan are inevitable due to adjustments to unpredictable and dynamic changes in the project environment or within the project. Also, it is sometimes very hard to create complete project plan at the outset of the project due to inability to clearly define project goals. Williams in 2005 summarizes those main reasons of inappropriateness of the traditional approach to majority of today's projects are structural complexity, uncertainty in goal definition and project time constraints.

1.3.1 Software project management

In this thesis, the field of project management on which I'm focusing and which I was able to observe during the internship experience, is the Software project management.

A software is an intangible product consisting of strings of codes that represent useful information for the management of a process. It can be defined as the set of computer codes

and procedures, to which annexed documentation, which allow an electronic or computer system to process data. Before software was born, project management was fully done through papers. This eventually produced a lot of paper documents and searching through them for information which was not a pleasant experience. Once software came available for an affordable cost for the business organizations, software development companies started developing project management software. This became quite popular among all the industries and these software were quickly adopted by the project management community.

1.3.3.2 Life cycle of a software

To talk about the software life cycle we must start by identifying the peculiar characteristics of this "product". The software, in fact, cannot be compared to the generic industrial product, as it develops and is not built, and is engineered but not manufactured.

The life cycle of a described IT project turns out to be different from that relating to the Project Management of standard projects (i.e., definition, planning, execution, monitoring and closure). However, they are connected and parallel and it is therefore necessary to use both for the management of an IT project: this is because the second includes the phases necessary for the organization of the project and the guarantee that it proceeds efficiently from the beginning. at the end (Hadaya et al., 2012).

The life cycle of an IT project outlines the technical work and defines the deliverables that are required to complete the IT project. Based on the definition provided by the Project Management Institute (2008), it includes the following steps:

1. Requirements and analysis
2. Architecture

3. Design
4. Construction
5. Integration and test
6. Delivery

- Phase 1: Requirements and analysis

The collection of requirements and their analysis make up the first phase required for the execution of an IT project. The requirements represent the problem that needs to be solved and the objectives of the project. The analysis is conducted in order to identify the relationships between the components (i.e., functionality) of the product or system for which the requirements were collected.
During this phase, the Project Manager works closely with the client and with the technical experts of the subject matter in the scope of the project, in order to define the requirements coming from the business and determine how they can be met. It is also the task of the Project Manager to identify the products or systems that will be delivered to the customer, the resources needed to carry out the project and the skills required for its completion.

- Phase 2: Architecture

The architectural definition phase is required for the identification of the elements that will be included in the product or system that will be delivered to the customer. The architecture expert reviews the requirements coming from the client and identifies possible solutions from a technical point of view. The project stakeholders analyze the proposed options and choose the one they consider the most efficient and effective solution.

- Phase 3: Design

The design phase of the solution to be provided to the customer consists of two distinct parts: high-level design and detailed design.
The high-level design is necessary to define how the components of the product or system that will be supplied will function from a technical point of view, what their

interactions will be with the other components of the system and how they will interact with hardware and software.
The detailed design, on the other hand, is descriptive and provides precise details for each component of the product or system. It identifies and defines each module that belongs to each component of the product or system, describing its functionality, how it will work in detail, and what are the detailed relationships that exist between different modules.

- Phase 4: Construction

The construction phase of the product or system defined in the design phase is the actual development activity and coincides with the writing of the code and the construction of the components by the team of developers.
In this phase, the Project Manager is responsible for the control process of the project being developed: it coincides with the monitoring and control phase characteristic of the Project Management life cycle applied to standard projects.
Project control includes a set of activities and processes aimed at identifying and monitoring the progress and performance of the project and identifying the areas that require any modification. It can be defined as a continuous process that requires the Project Manager to observe, gather information and understand the need for change in case of need.
As defined in the detailed chapter, the control process aims to compare the current project performances with those planned, in order to understand if the project is under control. This phase is also necessary for the identification and tracking of any new project risks and problems.
Another method used to determine if the project is under control is that of project status reports. The Project Manager schedules regular meetings with the Project Team in order to gather information on the status of developments from each separate working group that is involved in the implementation of the project. In this way, the Project Manager is able to identify any problems that may arise that may require corrective actions.

- Phase 5: Integration and Test

The integration phase consists in the combination of all the modules and components made in a single product or system functioning in such a way that it can be tested. The testing phase is crucial for the development of IT projects and is aimed at determining the quality of the product or system supplied.
Five distinct types of tests can be identified:
1. Unit and function test
2. System test
3. Integration test
4. No regression test (NRT)
5. User Acceptance test (UAT)

- Phase 6: Delivery

The Delivery phase of the developed product or system consists of multiple key tasks and activities, necessary for the delivery of the application to the end customer to take place efficiently and successfully.
It is possible to distinguish between two different phases of the same, characterized by activities carried out by the Project Manager that are distinct from each other: those necessary before the Go-Live of the application and those required in the Go-Live phase.

The first are the following:

- Creation of a detailed Deployment Plan, necessary for identifying and remembering all the critical steps in the delivery of the functionality.
- Review of the Deployment Plan with the members of the project team, necessary for the identification of any tasks that have been underestimated or even forgotten, thus avoiding costly mistakes.
- Communication of the key milestones of the plan to the project stakeholders, which allows the customer to prepare their teams adequately and in time for the new functionality.

- Training of the Support Team, who must be informed about the new functionality and adequately prepared to provide the necessary support in case of doubts, questions or problems raised by end users following the go-live.

The second are the following:

- Go-Live Monitoring, during which the Project Manager has the task of checking that the application release process takes place easily and smoothly. Typical activities of this phase are the restarting of the servers, the upload of the code and the deployment of the mobile applications and are carried out by the team of developers
- Follow-up with the Support Team, necessary to be made aware of any bugs and problems arising after the release whose resolution must be prioritized.

1.3.2 Waterfall in software project

Software projects date back as far as the late 1960s. The software industry grew fast and computer companies saw the potential in software production, which had a low cost compared to hardware production and circuitry which were more common. Software companies adopted the already well known waterfall model for its software projects. It turned out that this linear approach for developing software was less than optimal (Mens, 2008). The inflexible separation of phases and the fact that requirements are not always clear at the start of a project, were two major limitations of this model. The main causes for software project failures were; unrealistic project goals, poor estimates, badly defined requirements, poor status reporting, unmanaged risk, poor communication, use of immature technology, high project complexity, poor development practices, poor management, stakeholder politics and commercial pressures (Charette, 2005).

1.3.3.2 The evolution of the traditional waterfall approach over the past years

In 2004, the third edition of A Guide to the Project Management Body of Knowledge (PMBOK Guide) described the project life cycle of a typical project to have phases that 'are generally sequential and are usually defined by some form of technical information transfer or technical component handoff' and software development projects were specifically included in this description of a traditional waterfall life cycle. The pressure exerted by the then relatively recent publication of the agile manifesto on this view is only visible in the acknowledgement that phases can and often do overlap as an example of 'fast tracking'.

The rapid adoption of an agile approach to project management in especially the technology sectors leading up to 2008 saw the PMBOK Guide acknowledge this requirement of projects in 'largely undefined, uncertain, or rapidly changing environments' to allow for an iterative phase-to-phase relationship in which phases are planned one at a time, and the next phase is only planned for while work is progressing on the current phase. One could argue that the PMBOK Guide at this stage still supported a view that an agile approach could somehow be considered a variant of the traditional waterfall life cycle.

In 2013, the PMBOK Guide confirmed that the traditional waterfall project life cycle, referred to as 'predictive life cycles' or 'fully plan-driven', is relevant and applicable to all projects where the project scope and the time and cost associated in delivering that scope are determined and relatively stable at the earliest possible opportunity. Predictive life cycles (traditional waterfall) are well suited where both the goal of the project and the solution to the problem addressed by the project are known and clear, as is the case with most infrastructure and construction projects, for example.

The above notion is further supported by the fact that the 2013 edition of the PMBOK Guide clearly distinguishes between predictive, iterative and incremental, and adaptive life cycles. This is an indication that the traditional waterfall approach is now considered to be a specific type of project life cycle as opposed to being a 'catch all' that could not possibly meet the requirements of the entire modern project environment.

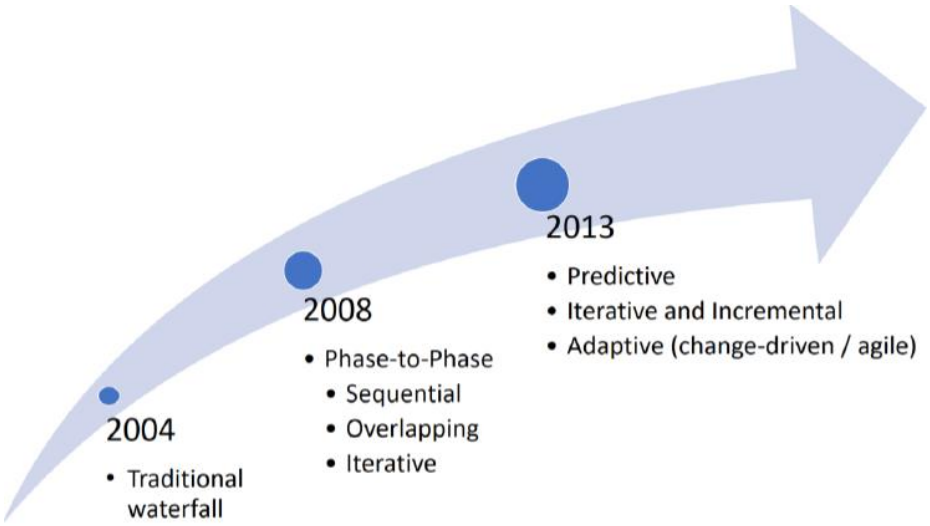

Figure 9. The changing view of the project life cycle in different editions of the PMBOK Guide.

1.3.3 Agile methodology

The term Agile is used to refer to a mode of software development that proposes a less structured approach than the traditional methods implemented up to that point, iterative and focused on customer satisfaction, delivering working and quality outputs quickly and frequently. The methodology has been so successful that it has been adapted to other contexts as well, thus being used for generic project management, projects that are becoming increasingly "extreme." The current context of high competition, which manifests itself through a series of actions carried out by various companies such as price battles, communication campaigns, promotional actions, new product launches, pre- and post-sales services, has given birth to the Agile methodology, an approach that aims to become a lever of competitive advantage for the company, supporting the conquest of new market positions or defending those acquired over time. The principles of the Agile method, born in software engineering, are inspired by the Toyota Production System, a method of organizing production that originated in Japan in the 1950s, a post-war period of severe crisis for the country. The Toyota Production System has as its pivotal goal the provision of products and services at a "world class" level of quality, a goal that can be pursued through the empowerment of employees, whose work is to be differentiated from the repetitive and alienating work of machines, cost reduction through the elimination of waste for profit maximization, and flexible production (Just in time) according to

market demand. The basic idea of the Toyota Production System is that a product cannot be better than the process that generated it, therefore, to have satisfied customers one needs processes of excellence. Quality is a result of the efficiency and effectiveness of processes. A company that is excellent in processes is one that produces zero inventory, that is, one that produces only what is required, one that produces with zero defects, so that there is no waste, and one that produces with zero downtime, so that productivity is increased and the entire working time is devoted to the realization of added value for the customer. The driving force behind the Toyota method is Lean Thinking, an operational strategy that pursues continuous improvement (Kaizen) of the enterprise; it starts by defining what is of value to the customer, proceeds by aligning value-creating activities in the right sequence, and then moves them forward without interruption according to a pull logic, i.e., only the activities required downstream are carried out, eliminating over- or under-production. Also emphasized is the idea that production is more efficient when carried out in small batches.

Based on these concepts, which originated in the automotive industry, a pool of software development experts in 2001 initiates the famous Agile Manifesto in order to outline and define the core values and principles of thinking. And so, paralleling the Toyota philosophy, from the idea of working in small batches, a methodology apt to provide for small frequent versions of a product is developed. The focus is on teamwork and staff involvement, and we are inspired by the concept of Kaizen: the project is to be constantly reviewed aiming at continuous improvement, changing factors where necessary based on new information or feedback returned by the system. The final result is the value for the customer; with this methodology, one is able to constantly readjust to customer needs by means of an iterative process and reduce the risk of failure.

1.3.3.2 Principles behind the Agile Manifesto

The Agile Manifesto is a document that identifies four key values and 12 principles that its authors believe software developers should use to guide their work. Formally called the Manifesto for Agile Software Development, it was produced by 17 developers during an outing on 2001, at The Lodge at Snowbird ski resort in Utah.

The developers called themselves the Agile Alliance. They were seeking an alternative to the existing software development processes that they saw as complicated, unresponsive and too focused on documentation requirements. The aim of the four values outlined in the Agile Manifesto is to promote a software development process that focuses on quality by creating products that meet consumers' needs and expectations. The 12 principles are intended to create and support a work environment that is focused on the customer, that aligns to business objectives and that can respond and pivot quickly as user needs and market forces change.

The four core values of Agile software development as stated by the Agile Manifesto are:

1. *Individuals and interactions over processes and tools*

In the past, a lot of software teams would concentrate on having the best possible tools or processes with which to build their software. The Agile Manifesto suggests that while those things are important, the people behind the processes are even more. Having the right group of individuals on a software team is vital to success, the best possible tools in the wrong hands are worthless. Perhaps even more important is how these individuals communicate with each other. The interactions between team members are what helps them to collaborate and solve any problems that arise.

2. *Working software over comprehensive documentation*

Previously, software developers would spend ages creating detailed documentation even before to start writing one line of code. Even if documentation is not a negative thing, it comes a point when the team should focus on providing the customers with working software. The Agile Manifesto places shipping software to your customers as one of the highest priorities, then the team can gather feedback to help improving future releases.

3. Customer collaboration over contract negotiation

Contract negotiation refers to any agreements including deadlines, budget agreements, and scope agreements with internal stakeholders or customers. You would draw up contracts with your customers who would then detail the finished product. As a result, there was often a contrast between what the contract said, what the product did, and what the customer actually required. According to the Agile Manifesto, the focus should be on continuous development. There is the need to build a feedback loop with your customers so that the team can constantly ensure that the product works for them.

4. Responding to change over following a plan

Following a plan means to deal with a static roadmap but the needs and requirements are always shifting, and priorities are always changing. That static roadmap will soon grow outdated. That's why the Agile Manifesto suggests that a software team should have the ability to pivot and change direction whenever they need to, with a flexible roadmap that reflects that. A dynamic roadmap can change from quarter to quarter, sometimes even month to month, and agile teams are able to keep up with those changes.
The 12 principles articulated in the Agile Manifesto are:

Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.

Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.

Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.

Business people and developers must work
together daily throughout the project.

Build projects around motivated individuals.
Give them the environment and support they need,
and trust them to get the job done.

The most efficient and effective method of
conveying information to and within a development
team is face-to-face conversation.

Working software is the primary measure of progress.

Agile processes promote sustainable development.
The sponsors, developers, and users should be able
to maintain a constant pace indefinitely.

Continuous attention to technical excellence
and good design enhances agility.

Simplicity--the art of maximizing the amount
of work not done--is essential.

The best architectures, requirements, and designs
emerge from self-organizing teams.

At regular intervals, the team reflects on how
to become more effective, then tunes and adjusts
its behavior accordingly.

While the Agile Manifesto doesn't rank the principles in any particular order, there are several key patterns that emerge. The need for continuous feedback is one. The role of effective communication in product development, another. Finally, the ability to be flexible and change directions quickly and smoothly.

Perhaps the most important lesson to take from the 12 agile principles is that you need to remember what you're there for. As a software development team, your

primary goal is to develop and release software. Agile principle 7 says that working software should be the primary measure of success. Having that as your north star will ensure all the other agile values and principles will fall into place.

1.3.3.2 Methodologies in the Agile industry

Agile approaches and agile methods are umbrella terms that cover a variety of frameworks and methods. *Figure 10* places agile in context and visualizes it as a blanket term, referring to any kind of approach, technique, framework, method, or practice that fulfills the values and principles of the Agile Manifesto. It also shows agile and the Kanban Method as subsets of lean. This is because they are named instances of lean thinking that share lean concepts such as: “focus on value,” “small batch sizes,” and “elimination of waste.”

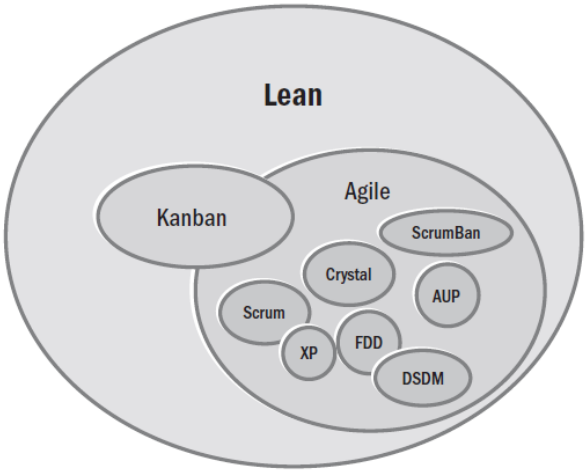

Figure 10. Relation between Lean, Agile and Kanban

One way to think about the relationship between lean, agile, and the Kanban Method is to consider agile and the Kanban Method as descendants of lean thinking. In other words, lean thinking is a superset, sharing attributes with agile and Kanban. This shared heritage is very similar and focuses on delivering value, respect for people, minimizing waste, being transparent, adapting to change, and continuously improving. The Kanban Method is inspired by the original lean-manufacturing system and it was adapted to the software development. Kanban is a popular framework used to implement agile and DevOps software development. Work items are represented visually on a Kanban board, allowing team members to see the state of

every piece of work at any time. As we can see in *Figure 10*, there are several methodologies through which we can implement Agile Projects. I will not go in details about all the techniques because our focus will be just in the Scrum method. Here I have discussed just three of them which are most widely used in the Industry. The agile methods are focused on different aspects of the software development life cycle. Some focus on the practices (extreme programming, pair programming), while others focus on managing the software projects (the scrum approach).

- Extreme Programming (XP)

XP is the most successful method of developing agile software because of its focus on customer satisfaction. XP requires maximum customer interaction to develop the software. It divides the entire software development life cycle into several number of short development cycles. It welcomes and incorporates changes or requirements from the customers at any phase of the development life cycle.

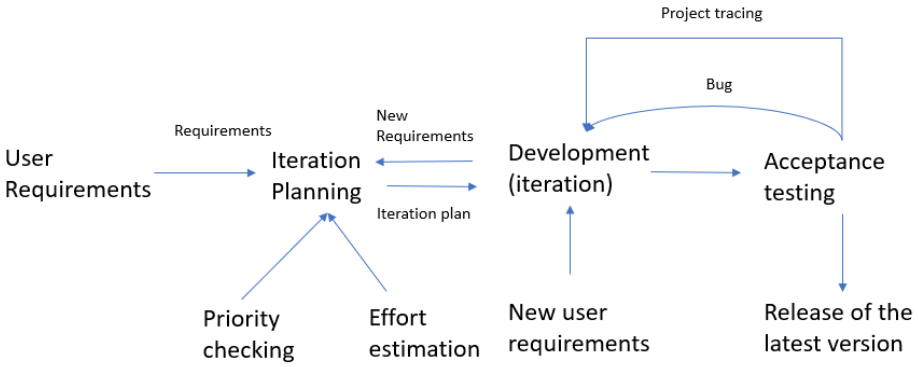

Figure 11. Method of Developing Agile Processes using Extreme Programming

The above diagram shows the complete method of developing agile process using XP method. Extreme programming starts with collecting user requirements. Depending upon these requirements the whole development process is divided into several small number of cycles. The next phase is iteration planning i.e. deciding the number of cycles, prioritizing the requirements and estimating the amount of effort required to implement each cycle. Now each iteration is developed using pair programming. During the development phase new user requirements may come and the iteration plan should be adjusted according to that. Next step is to test the latest

developed version for bugs, if detected; the bugs will be removed in the next iteration. After every acceptance testing project tracing should be done in which feedback is taken from the project that how much job has already been done.

- Scrum

Scrum is another popular method of agile development through which productivity becomes very high. It is basically based on incremental software development process. In scrum method the entire development cycle is divided into a series of iteration where each iteration is called as a sprint.

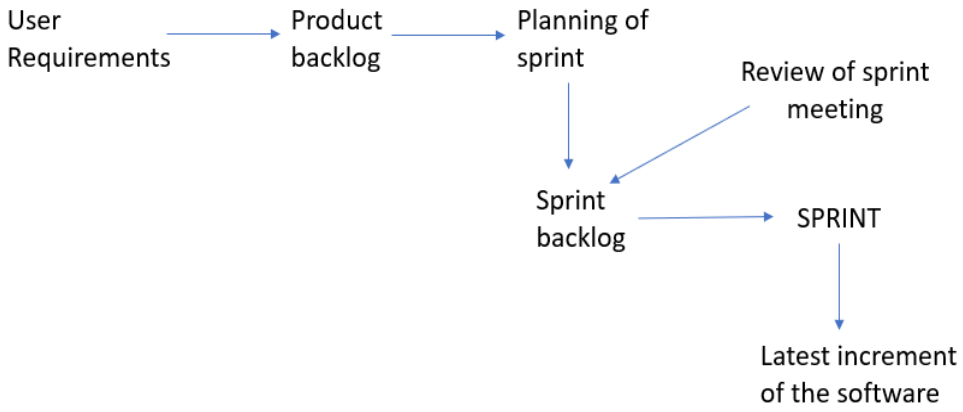

Figure 12. Method of Developing Agile Processes using Scrum

The method starts with collecting user requirements but it is not expected that all the requirements should come out from the user at the beginning. User can change their mind at any time during development; they can add new features, remove or update some existing features. Next phase is to prioritize the requirements and the list is known as product backlog. A proper planning for sprint should be done i.e. how many sprints are needed to develop the software, duration of the sprint, and what are the requirements from the product backlog should be implemented in each sprint. This particular list is known as sprint backlog. During each sprint one sprint meeting is held every day to take the feedback of how much work has been done. After each sprint review is taken to determine whether all the requirements for that particular sprint have already been implemented or not and to decide the requirements should

be implemented at the next sprint. After each sprint we get a working increment of the software.

- Feature Driven Development (FDD)

FDD is one of the agile development methods. The key advantage of this method is to design the domain of the software to be produced before development.

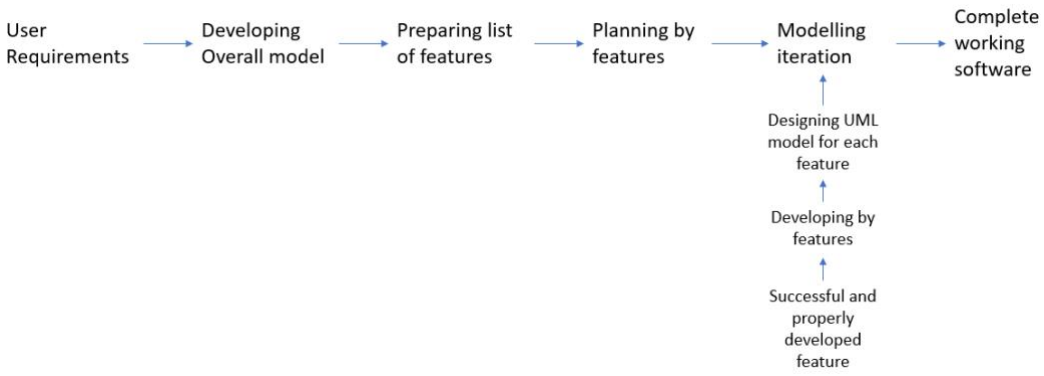

Figure 13. Method of Developing Agile Processes using FDD Graph

The method starts with collecting the requirements from the users and building up the overall model of the project. The model gives the clear idea about the scope of the software. Next step is to make a list of features which are the client-valued functions. For example, 'authenticate the password', 'calculate the salary for each employee', 'calculate the income tax for each employee'. Now several groups of features are made based on their domains i.e. related features are combined into a single group. Next step is to make a plan for developing the features. Each group of features is assigned to a development team which is headed by one chief programmer. Last step is modelling iteration in which first UML modelling is done for each feature and then developing that particular feature. This step continues unless all the features get implemented successfully.

1.4 SCRUM

In 1995, Ken Schwaber started formalizing the rules of Scrum and compiled his findings into a book, "Agile Software Development with Scrum" (Sutherland & Schwaber, 2007). As aforementioned, Scrum is counted as a member of the Agile model which was modified to be more flexible and adaptable method that can be

applied to different business situations (Bomber, 2007). Scrum is a simple process in software development that focuses on the quality of the team, designed techniques, and methods of maintenance (Weerasak, 2007). In addition, Scrum focuses on people rather than on the development process. This method emphasizes on communication, collaboration, and rapid exchange of information between team members. Due to its ability to increase the rate of success in software development, Scrum is one of the most widely used processes in Agile software development (Nuevo, 2011). Scrum is one of the most use Agile frameworks followed to complete challenging projects wherein there are dynamic changes in the requirements by using one or more cross-functional, that are self-organizing teams of about 7 +/- 2 people on each team. It has risen from being a method used by a number of enthusiasts at the Easel Corporation in 1993, to one of the world's most popular and well-known frameworks for development of software. The continued expansion of the global rollout of Scrum is testimony to the fact that Scrum delivers on its promise. Its inspect and adapt approach to continuous quality improvement can do serious damage to outmoded business practices. By focusing on building communities of stakeholders, encouraging a better life for developers, and delivering extreme business value to customers Scrum can release creativity and team spirit in practitioners and make the world a better place to live and work. Scrum has emerged from a rough structure for iterative, incremental development to a refined, well-structured, straightforward framework for complex product development.

1.4.1 SCRUM values

In July 2016, the Scrum Values were added to The Scrum Guide. Successful use of Scrum depends on people becoming more proficient in living five values:

1) Commitment; People personally commit to achieving the goals of the Scrum Team
2) Focus; Everyone focuses on the work of the Sprint and the goals of the Scrum Team
3) Openness; The Scrum Team and its stakeholders agree to be open about all the work and the challenges with performing the work
4) Respect; Scrum Team members respect each other to be capable, independent people

5) Courage; Scrum Team members have courage to do the right thing and work on tough problems

The Scrum Team commits to achieve its goals and to support each other. Their primary focus is on the work of the Sprint to make the best possible progress toward these goals. The Scrum Team and its stakeholders are open about the work and the challenges. Scrum Team members respect each other to be capable, independent people, and are respected as such by the people with whom they work. The Scrum Team members have the courage to do the right thing, to work on tough problems. These values give direction to the Scrum Team with regard to their work, actions, and behavior. The decisions that are made, the steps taken, and the way Scrum is used should reinforce these values, not diminish or undermine them. The Scrum Team members learn and explore the values as they work with the Scrum events and artifacts. When these values are embodied by the Scrum Team and the people they work with, the empirical Scrum pillars of transparency, inspection, and adaptation come to life building trust.

1.4.2 SCRUM Roles

The fundamental unit of Scrum is a small team of people, a Scrum Team. The Scrum Team consists of one Scrum Master, one Product Owner, and Developers. Within a Scrum Team, there are no sub-teams or hierarchies. It is a cohesive unit of professionals focused on one objective at a time, the Product Goal. Scrum Teams are cross-functional, meaning the members have all the skills necessary to create value each Sprint. They are also self-managing, meaning they internally decide who does what, when, and how. The Scrum Team is small enough to remain nimble and large enough to complete significant work within a Sprint, typically 10 or fewer people. If Scrum Teams become too large, they should consider reorganizing into multiple cohesive Scrum Teams, each focused on the same product. Therefore, they should share the same Product Goal, Product Backlog, and Product Owner. The Scrum Team is responsible for all product-related activities from stakeholder collaboration, verification, maintenance, operation, experimentation, research and development, and anything else that might be required. They are structured and empowered by the organization to manage their own work. Working in Sprints at a

sustainable pace improves the Scrum Team's focus and consistency. The entire Scrum Team is accountable for creating a valuable, useful Increment every Sprint.

1.4.2.1 Scrum Master

The Scrum Master is the team role responsible for ensuring the team lives agile values and principles and follows the processes and practices that the team agreed they would use.
The responsibilities of this role include:

- Clearing obstacles
- Establishing an environment where the team can be effective
- Addressing team dynamics
- Ensuring a good relationship between the team and product owner as well as others outside the team
- Protecting the team from outside interruptions and distractions.

The scrum master role was created as part of the Scrum framework. The name was initially intended to indicate someone who is an expert at Scrum and can therefore coach others. The general benefit expected from having a scrum master on a team is providing a self-organizing team with ongoing access to someone who has used agile, and Scrum in particular, in another setting and can help the team figure out the best way to apply it in their situation. Another expected benefit is that the scrum master is someone that can address distractions, disruptions, and obstacles so that the remainder of the team is free to focus on the work of producing output that will generate the desired outcome.

1.4.2.2 Product Owner

The product owner is a role on a product development team responsible for managing the product backlog in order to achieve the desired outcome that a product development team seeks to accomplish.
Key activities to accomplish this include:

- Developing and explicitly communicating the Product Goal
- Creating and clearly communicating Product Backlog items

- Ordering Product Backlog items
- Ensuring that the Product Backlog is transparent, visible and understood

The product owner role was created as part of the Scrum framework in order to address challenges that product development teams had with multiple, conflicting direction, or no direction at all with respect to what to build. Many infer that a product owner is someone who can spend a considerable amount of time with the product development team providing clarification on product backlog items, and making decisions about which product backlog items to do and regarding the specifics of those particular product backlog items.

The Scrum framework was originally created to address issues that product development teams faced. The product owner role was established in order to provide that single source of information for a product development team about the product they are trying to build.

This single point of information keeps the team focused and reduces churn resulting from waiting for answers, or conflicting priorities.

Product owners also represent a single point of responsibility, leading to the epitaph “single, wringable neck” which is a benefit for the team and those outside the team when looking to easily identify the responsible party, but may not be as beneficial for the product owners specifically.

1.4.2.3 Developers

The developers are the people in the Scrum Team that are committed to creating any aspect of a usable Increment each Sprint. The specific skills needed by the Developers are often broad and will vary with the domain of work. However, the Developers are always accountable for:

- Creating a plan for the Sprint, the Sprint Backlog
- Instilling quality by adhering to a Definition of Done
- Adapting their plan each day toward the Sprint Goal
- Holding each other accountable as professionals

1.4.3 Scrum artifacts

Scrum provide three Artifacts show in *Figure 14* that are meant to be used along the development process. They represent work or value to provide transparency and opportunities for inspection and adaptation. They are specifically designed to maximize transparency of key information so that everybody has the same understanding of the artifact.

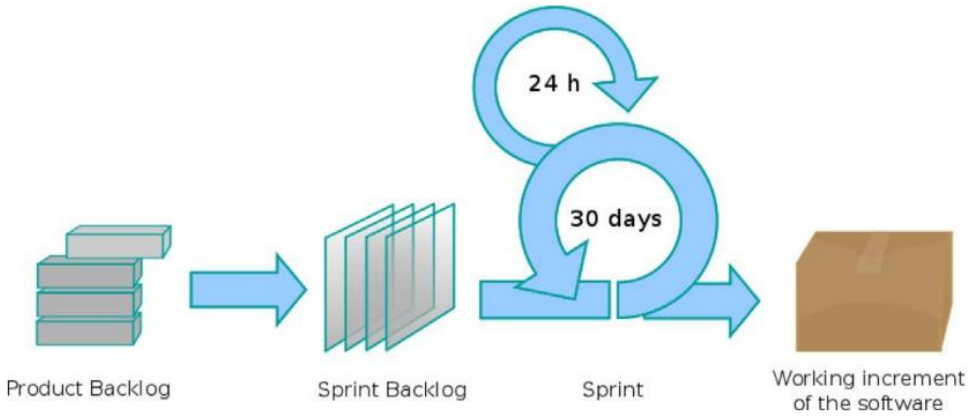

Figure 14. Scrum Artifacts

1.4.3.1 Product Backlog

A product backlog lists and prioritizes the task-level details required to execute the strategic plan set forth in the roadmap. The backlog should communicate what's next on the development team's to-do list as they execute on the roadmap's big-picture vision. It is the single source of requirements for any changes to be made to the product. The Product Owner is responsible for the Product Backlog, including its content, availability, and ordering. The product backlog is the only authoritative source for the things a team works on. This means that nothing is done that is not in the product backlog. Conversely, having a product backlog item on a product backlog does not guarantee that it will be delivered. It represents an option the team has to deliver a specific result rather than a commitment. Items in the product backlog take a variety of formats, with user stories being the most common. The team using the product backlog determines which format they have chosen to use and considers the backlog items as a reminder of what aspects of a solution they could work on. Items in the product backlog vary in size and extent of detail based

largely on how quickly a team will work on them. The ones a team will work on soon should be small in size and contain enough detail for the team to start working on. Items in the product backlog that are not intended for the job can be quite large and have few details. The sequence of product backlog items on a product backlog changes as a team gains a better understanding of the outcome and identified solution. This reordering of existing product backlog items, the continual addition and removal of product backlog items, and continual refinement of product backlog items give a product backlog its dynamic characteristic. A product backlog can be an effective way for a team to communicate what they are working on and what they intend to work on next. Story maps and information radiators can provide a clear picture of your backlog for the team and stakeholders.

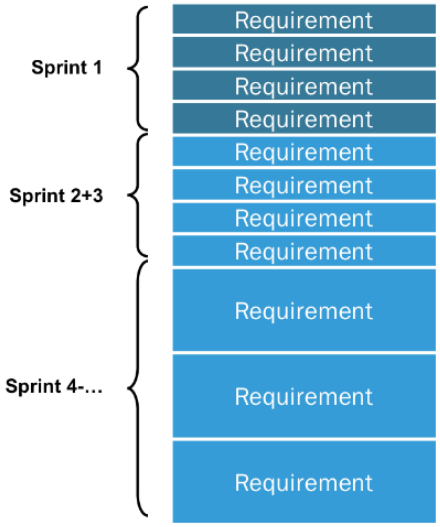

Figure 15. Sprint Backlog

1.4.3.2 Sprint Backlog

The sprint backlog is a list of activities identified by the Scrum team to complete during the current sprint. During the sprint planning meeting, the team selects a number of items from the product backlog, usually in the form of a user story, and identifies the activities needed to complete each user story. The Sprint Backlog clarifies the work required by the development team to achieve the Sprint goals. While the Product Owner owns the Product Backlog, the development team is consulted when deciding 1) the goal of the sprint and 2) the specific Product Backlog items that will help achieve that goal. Sprint Backlog is a sufficiently specific plan to

make progress changes understandable in daily meetings. The development team will modify the Sprint Backlog throughout the Sprint, and the Sprint Backlog will gradually emerge in the Sprint process, such as the development team working according to plan and knowing more about the work needed to accomplish the Sprint goals. When a new job comes up, the development team needs to add it to the Sprint to-do list. As tasks proceed or are completed, the estimated remaining workload for each task needs to be updated. If a part of the plan loses the meaning of development, it can be removed. Within Sprint, only the development team can modify the Sprint Backlog. The Sprint Backlog is highly visible, a real-time reflection of the team's plans to complete work within the current Sprint, and it belongs only to the development team.

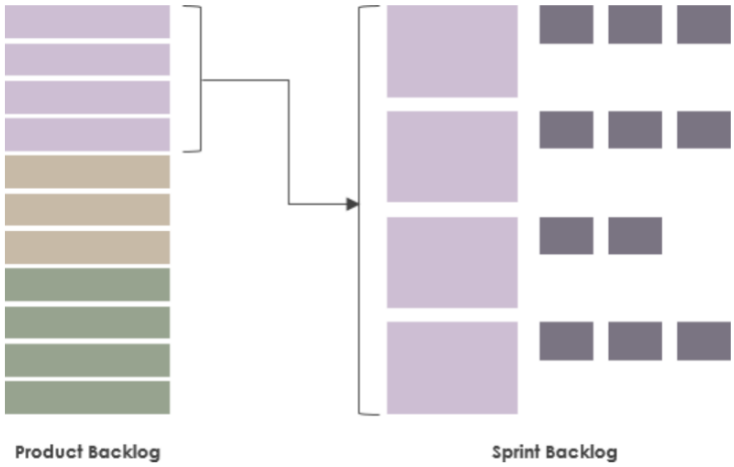

Figure 16. Sprint Backlog

1.4.3.3 Potentially Releasable Product Increment

An Increment is a concrete stepping stone toward the Product Goal. Each Increment is additive to all prior Increments and thoroughly verified, ensuring that all Increments work together. In order to provide value, the Increment must be usable. At the end of Sprint, the new increment must be "done", which means it must be available and meet the standard of the scrum team's definition of done. The Agile definition of done is a collection of criteria that must be completed for a project to be considered "done." It is essentially a checklist used by Scrum teams to create a shared understanding of what is required to make a product releasable. The Definition of Done promotes

transparency by providing everyone a unified understanding of the work completed during the Increment. Increments are viewable and completed product components that support empiricism at the end of Sprint. Whether or not the Product Owner decides to release it, the product increment must be available and able to meet the business requirements to accomplish the delivery plan. Product increments are additive, which means that they build on previous iterations of the product. By the end of several sprints, the Scrum team will have delivered multiple product increments, each improving on the next in the endless quest to meet user needs.

1.4.4 Scrum Events

Scrum defines several events (also called ceremonies) serving a specific purpose that occur inside each **sprint**. These events are designed to reduce the need for other events and they are all timeboxed, which means that they have a maximum length. The 5 ceremonies are summarized in the *Figure 17*.

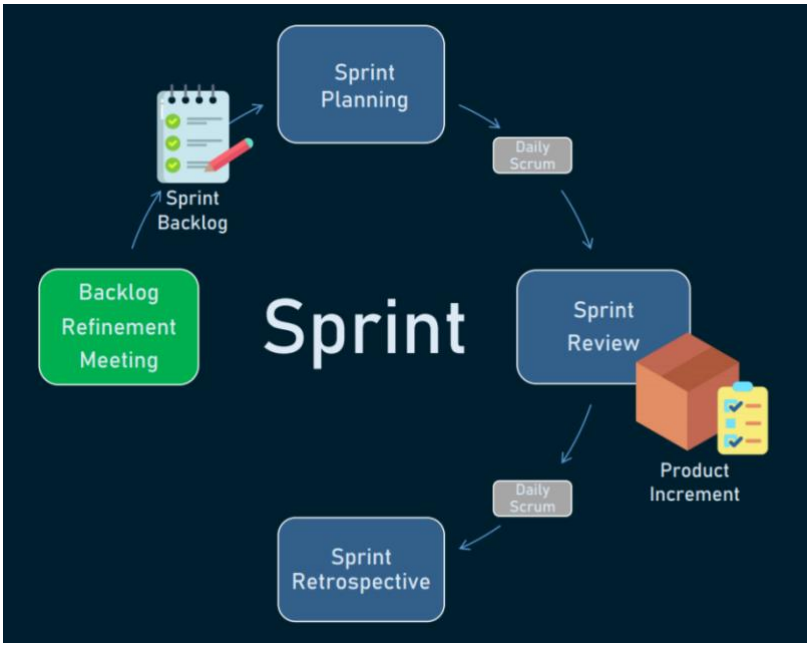

Figure 17. Scrum events: Backlog Refinement, Sprint Backlog, Sprint Planning, Sprint Review, Sprint Retrospective

1.4.4.1 Backlog Refinement meeting

The purpose of the backlog refinement meeting is to decompose the highest priority items in the product backlog into user stories which are suitable for inclusion in the

next sprint. The backlog refinement meeting (or backlog management meeting, or backlog grooming session) usually takes place at the beginning of the current sprint, before the sprint planning. The benefits of having a separate backlog management meeting are threefold: it reduces the need for a long sprint planning meeting and which can tax even the longest of attention spans and it also gives the team a chance to reflect on the backlog items before committing to the sprint backlog and sprint goal. The team is given a chance to ask the questions that would normally arise during sprint planning. These questions do not need to be fully resolved in a backlog refinement meeting. Rather, the product owner needs only to address them just enough so that the team feels confident that the story can be adequately discussed during the coming planning meeting.

1.4.4.2 Sprint planning

In this event the Product Owner, Scrum Master and Developers are present. Their goal is to decide which Product Backlog items to develop within the time limit. It also provides how to organize to achieve this. In practice, the Product Owner presents to the rest of the team what will be the goal of the Sprint that is starting and the members are invited to clarify all elements of the Sprint backlog together. Through discussion with the Product Owner, the Developers select items from the Product Backlog to include in the current Sprint. They break down the Items into technical and operational tasks and they foresee the duration of the tasks taking into account in the estimation the phases of testing and documentation. The Product Owner proposes how the product could increase its value and utility in the current Sprint. The whole Scrum Team then collaborates to define a Sprint Goal that communicates why the Sprint is valuable to stakeholders. The Sprint Goal must be finalized prior to the end of Sprint Planning. Indeed, each Sprint can be seen as a project in itself, in which at the end, the team must be able to present a potentially deliverable product. The duration of Sprint Planning cannot exceed 8 hours for a Sprint of 4 weeks or a calendar month. For shorter Sprints, the meeting time can be scaled proportionally. It is possible to summarize this meeting in 3 questions:

- What is the specific objective of the Sprint that is starting?
- Which priority items of the Product Backlog can be converted into a potentially shippable increment before the end of the event?

- How should the team organize itself to convert the selected elements into a potentially shippable increment?

1.4.4.3 Sprint review

A sprint review is an informal meeting held at the end of a sprint, during which the team shows what was accomplished, while the stakeholders provide feedback. Based on this information, attendees collaborate on what to do next and the Product Backlog may also be adjusted to meet new opportunities. The attendees of a sprint review usually include the product owner and product manager, the Scrum Master, the development team, the management, various business stakeholders, and anyone else involved in the product management process. The main purpose of a sprint review is to inspect the outcome of the sprint, collect feedback from all the stakeholders, and adapt the backlog going forward. In a sprint review is always shown the work done, usually with a demo session and it help to create transparency, foster collaboration, and generate valuable insights. A sprint review may last up to 4 hours in 4-week-sprints. The general rule is that the sprint review should take no more than one hour per week of sprint duration.

1.4.4.4 Daily scrum

A daily scrum meeting, daily stand-up meeting or "DSM" for short, is one of the "official" rituals organized on a recurring basis as part of an agile project management process. It is called ''stand-up' because it should take more than 15 minutes. It is then up to the Scrum Master to enforce this deadline. The objective of this daily meeting is to make it possible to know the progress of the team in order to achieve the achievement of one or more objectives determined during the sprint planning. It is essentially a ritual dedicated to development teams, accompanied by the Scrum Master. Stakeholders can come but only as observers. The objective is not so much to establish a rigid organization as to allow maximum velocity of the team on each sprint in order to achieve the achievement of the sprint objectives. The presence of the PO is optional. Each member of the team specks answering 3 basic questions: What did I do yesterday? What am I going to do today? What are the existing blocking points that prevent me from moving forward?

The first question may be useful to Testers and the Product Owner because it will allow them to know the progress of one or more user stories and to determine which ones are testable. The second one gives to the stakeholders the real time progression of the development and for the team is useful to know on what the other members are working in to be more aligned. The third question is fundamental because it allows the whole team to have a transverse vision on all the contentious points adjoining a specific user story or the overall project, and this in complete transparency.

1.4.4.5 Sprint retrospective

This meeting takes place after the review with the clients, once the sprint is concluded. During each Sprint Retrospective, the Scrum Team plans ways to increase product quality by improving work processes or adapting the definition of "Done" if appropriate and not in conflict with product or organizational standards. To do this, the Scrum Team discusses what went well during the Sprint, what problems it encountered, and what could be improved. In doing this, they examine how the last Sprint went with regard to individuals, interactions, processes, tools and their standard. The main purpose of the Sprint Retrospective is to increase quality and effectiveness, for this purpose the Scrum Team identifies the most useful changes to be made. The most impactful improvements are addressed as soon as possible and can also be added to the Sprint Backlog for the next Sprint. It is set for a maximum of three hours for a one-month Sprint. For shorter Sprints, the event is generally shorter. By the end of the Sprint Retrospective, the Scrum Team should have identified the improvements it will implement in the next Sprint. The implementation of these improvements in the next Sprint is the adaptation to the inspection of the Scrum Team itself.

2. Methodology: Techniques for the analysis

In order to analyze the project performance and foresee the duration and the costs I have used several methods that are explained in this chapter. First of all, I unfold the Earned Value, a traditional waterfall-related method that is also adopted over the past years in the software product development. The method, relying largely on baselined task-driven plans with fixed and well-defined scope, assume linear progression on task execution and completion that pave the way to the devising of schedule and cost performance indices that can be used for forecasting and making decisions on project controls. Studying Scrum projects, the efficiency of this method was doubted because it contrasts with the Agile theory. Another model is presented to be more suitable, the Burndown charter. The Burndown charter is born in the Agile environment to gather information about both the work the team have completed on a project and the work that is yet to be done within a given time period. This method doesn't provide mathematical calculation for the forecast of time and cost as the EMV but from the visualization of the chart we can have an idea about the speed of the team and this helps to foresee the future progress and the optimal tasks the team will be able to do in the next iteration. Moreover, this kind of tool is more adaptable in an Agile environment where the scope often changes. The last method that is taking into consideration is the Putnam model. Putnam studied software project by collecting project data and fitting a curve to the data. He hypothesized an equation able to predict the effort and time required to finish a software project of a specified size. Plotting effort as a function of time yields the Time-Effort Curve. It is one of the earliest of these types of models developed and is among the most widely used. This seems to be very efficient to foresee project duration and cost but in the next chapter I will go throw a limitation of the model and I will propose a theory that underline the importance to take into consideration this limited factor.

2.1 Earned value

The Earned Value method has been developed as a tool facilitating project progress control. It is used for determining a project's status and the scale of current variances from the plan. To implement earned value management, the PM must first make a detailed plan of both time and budget, then make a corresponding detailed valuation of the work on the project before it starts. As the project progresses, the PM must assess at predetermined reporting periods how much value should have been achieved according to the plan, how much value has been produced and how much money has been spent. These three assessments form the basis for all earned value analysis techniques. The method has been recognized as a useful tool by many practitioners and government agencies and has become a standard in project management. It proved to be versatile enough to be applied to any type of a project, ranging from defense schemes worth millions and extending on many years to minor IT projects. The analysis can be conducted on any level of work breakdown structure and used by both clients and contractors. The purpose is to detect any deviation as soon as possible, so that there is enough time to assess if the deviation is dangerous for the project and, if necessary, to take corrective actions.

To analyze the functioning and limits of this methodology, however, it is necessary to deepen the concepts and definitions that constitute its foundations.
The variables on which the EVM analysis is based are:

- BC, budget cost

- WS, Work Scheduled

- AC, Actual cost

- WP, Work Performed

Starting from these variables and parameters, the following indicators are calculated:

- Planned Value (PV, BCWS) – The authorized budget assigned to the scheduled work to be accomplished for a schedule activity or work breakdown structure component.
  It is calculated as the product of the amount of work scheduled for the period and the cost of the resources involved.

$$BCWS = Budget\ cost * Work\ scheduled$$

The total planned cost of the whole project (BAC) equals BCWS at the planned finish.

- Actual Cost (AC, ACWP) – Total costs actually incurred and recorded in accomplishing work performed for a schedule activity or work breakdown structure component.
  This is given by the product of current cost and work actually performed.

$$ACWP = Actual\ cost * Work\ performed$$

- Earned Value (EV, BCWP) – The value of work performed expressed in terms of the budget assigned to that work for a schedule activity or work breakdown structure component.
  It is calculated as the product of the budget initially planned and the work carried out up to the moment in question.

$$BCWP = Budget\ cost * Work\ perfomed$$

From the analysis of the variables just described, it is possible to calculate the following performance indicators:

- Schedule Variance (SV) – A measure of schedule performance on a project. It is the algebraic difference between the earned value (EV) and the planned value (PV).

$$SV = BCWP - BCWS$$

SV tells the PM if the actual progress achieved is ahead of or behind the baseline schedule. A negative value is a symptom of delay with respect to the schedule. Instead SV greater than 0 indicates that it is ahead of the initial plan.

- Cost Variance (CV) – A measure of cost performance on a project. It is the algebraic difference between earned value (EV) and actual costs (AC).

$$CV = EV - AC$$

CV indicates whether actual project expenditures are exceeding the planned amount for the corresponding value achieved. In other words, if it has been spent more or less than planned to achieve a defined level of progress. A positive value indicates a favorable condition and a negative value indicates an unfavorable condition.

- Cost Performance Index (CPI) – A measure of cost efficiency on a project. It is the ratio of earned value (EV) to actual costs (AC).

$$CPI = \frac{EV}{AC}$$

CV tells the PM if costs are being expended efficiently. In other words, if they are getting more, less or the planned value for actual amounts spent. A value equal to or greater than one indicates a favorable condition and a value less than one indicates an unfavorable condition.

- Schedule Performance Index (SPI) – A measure of schedule efficiency on a project. It is the ratio of earned value (EV) to planned value (PV).

$$SPI = \frac{EV}{PV}$$

SV tells the PM if effort is being expended efficiently. In other words, if they are we getting more, less or the planned value for effort (time) expended. An SPI equal to or greater than one indicates we are ahead the time schedule and a value of less than one indicates a delay.

The trend of earned value, planned value and actual cost are often reported cumulatively as a function of time periods. In this way you can see the progress trends immediately. Physiologically, a project whose life cycle tends to be waterfall

presents this S-shaped trend, therefore increasing in a very accentuated way in the intermediate phases of the project, while less marked at the start and towards the end. Planned (Scheduled), Actual and Earned Value S-curves can have six possible arrangements, as in the chart presented in *Figure 19*.

An example of this trend is shown below, in *Figure 18*.

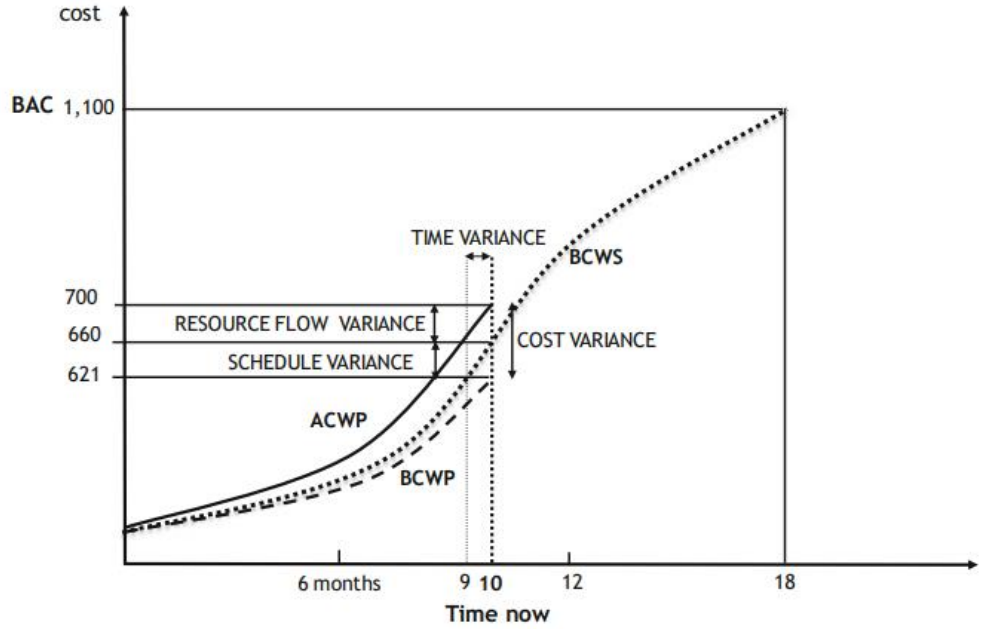

Figure 18. Project progress and S curves

One must look at the position of the EV S-curve as the reference curve line: whether the AV or the PV above the EV curve indicates the project is over budget or is behind schedule. The best case-scenario is the one where both the AV and the BV curves are below the EV: in this case both the cost and the duration are better than scheduled. The more the distance of the AV and BV curve lines from the PV one, the larger the loss or gain of value and schedule. Close reasoning can be conducted for Cost and Schedule performance indexes: aggregate indexes can be obtained for the overall project by weighted sum of activity performance indexes. The best case-scenario is when both the aggregate CPI and aggregate SPI for a project are more than 1.

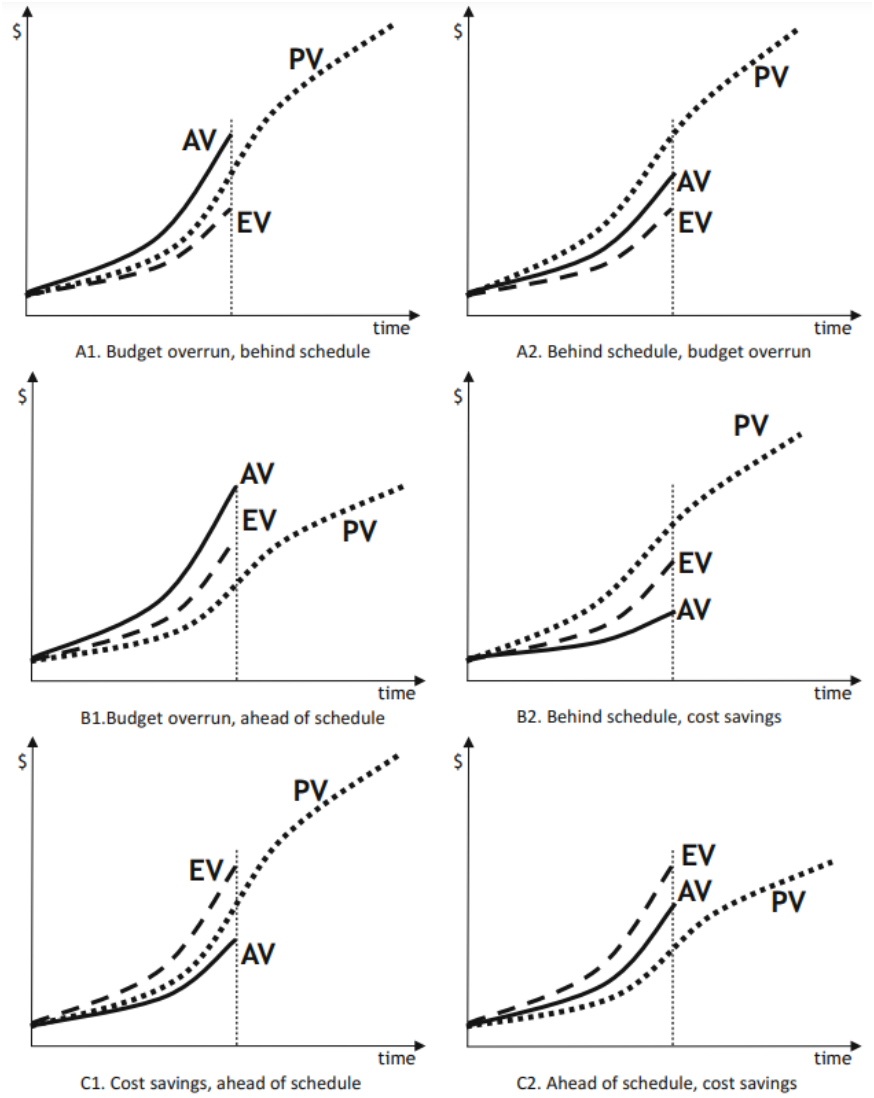

Figure 19. Possible arrangements of S-curves indicating planned value (PV), actual cost (AV) and earned value (EV). For example, A1 indicates both cost overruns and schedule delay, with more serious problems on cost than on schedule. A2 is a similar situation, where schedule delay is more significant than extra cost

Starting from the indices listed above and considering the project parameters, it is possible to calculate the project time and cost estimates, ie the time at completion (TEAC) and the cost at completion (CEAC).

2.1.1 Earned Value “forecasting” parameters

2.1.1.1 Cost estimates at completion

Cost estimate at completion (CEAC or EAC) defines the final estimate of the total costs of a project. The Project management discipline recognizes several approaches to the calculation of this forecast:

- Traditional estimation approach:

The first original approach states that future remaining cost will be in line with the budget (i.e. the total Budget at Completion minus the budgeted cost of work performed). This means:

$$CEAC = ACWP + (BAC - BCWP) = BAC – (BCWP - ACWP) = BAC – CV$$

This approach is rather optimistic, assuming that cost overruns are old problems and will not incur in the future. A better way for calculating EAC is a revised estimate approach.

- Revise estimate approach:

$$\mathrm{CEAC} = \mathrm{ACWP} + \frac{(\mathrm{BAC} - \mathrm{BCWP})}{CPI} = \frac{BAC}{CPI}$$

This principle assumes that the project future will, at least, reflect the past performance, if no corrective actions are undertaken. Therefore, starting from the total cost allocated to the budget, the BAC, a Cost Index lower than one, ie if you are incurring costs higher than those estimated, will lead to a CEAC higher than the BAC. Conversely, a CI greater than one will cause the cost on completion to be lower than the budgeted one.

- Pessimistic approach

$$CEAC = \frac{BAC}{SPI * CPI} = \frac{BAC}{CR}$$

This last approach considers, in addition to the influence of a possible delay or advance on the final cost of the project, also the effect of a possible loss or gain from the point of view of costs (therefore the possible variation of the actual costs with respect to those budgeted). This approach is defined as pessimistic because when the two SPI and CPI indices show similar trends (at the same time they are both greater or less than 1), the effect on the CEAC is amplified. Conversely, the effect of these levers tends to counterbalance when they present opposite trends. In practice, when the SPI is less than 1, this estimate is precautionary; conversely, when the SPI is greater than 1, its contribution tends to be reasonably neglected.

2.1.1.2 Time estimates at completion

Similarly to the deductible assumptions on the cost just presented (CEAC), it is possible to make similar forecasts on the completion times of the project. It is defined Time estimate at completion (TEAC or AC)

The two approaches to calculating this estimate are as follows:

- Traditional estimation approach:

$$TEAC = AT + (BAC - BCWP) * \frac{PD - T}{BAC - BCWS}$$

Where PD (Planned duration) is the estimated completion time in the preliminary phase of the project. The parameter T, on the other hand, indicates the moment in which the analysis is carried out. As with the calculation of the CEAC, the traditional approach is limiting in fact, it assumes that the work rate, from moment T until the end of the project, is constant and equal to that estimated in the budget a priori.

Note that: if BCWP = BCWS, then TEAC will be exactly equal to PD.

- Revised estimation approach

$$TEAC = AT + (BAC - BCWP) * \frac{PD - T}{BAC - BCWS} * \frac{1}{SPI}$$

This approach inserts the SI (or SPI) index into the calculation of the completion time. If the schedule index is less than 1, therefore the project is progressing in a situation of inefficiency, the expected time for completion will lengthen.

Note that: if BCWP = BCWS, this is always true at time 0, then the TEAC will be equal to BAC.

2.2 Alternative to the Earned value method: Earned schedule

The Earned Schedule methodology, whose authorship is attributable to Walter Lipke, was born recently, in 2003, and represents an extension of the theory and practice of Earned Value Management. Integrated since 2005 in the standards dictated by the PMI, it allows to elaborate more meaningful considerations on completion times. The notion of Earned Schedule (ES) overcomes some limitations inherent to the Earned Value method: it measures the schedule progress in time units instead of dollar amounts and eliminates the defect of the SPI to tend to unity as the project closes to completion, regardless of any early or late progress. The value of the ES is obtained by projecting to actual time AT the EV curve onto the PV curve line assuming that the current EV should actually have been earned at that projected point in time. A graphical representation of the Earned schedule is in Figure 16.

Therefore, the ES is defined as:

$$ES = C + \frac{EV - PV(\mathrm{c})}{PV(\mathrm{c}) + 1 - PV(\mathrm{c})}$$

Where C and the associated subscript c denote the number of time units for which the EV exceeds the PV. As a consequence, time-based SV(t) and SPI(t) can be defined as:

$$SV\ (t) = ES - AT$$

Where, positive SV (t) indicates an advance with respect to the initially scheduled plan, vice versa SV (t) less than 0 indicates a delay. This index, calculated in this

perspective in which it is a function of time, results in returning more confident considerations than the schedule variance calculated with the traditional method of EV, in which, precisely, it is calculated with the unit of measurement of the value (SV ($)).

$$SPI\ (t) = \frac{ES}{AT}$$

TEAC is therefore calculated as:

$$TEAC = AT + \frac{(PD - ES)}{SPI\ (t)}$$

$$TEAC = T + \frac{PD - C + \dfrac{EV - PV(c)}{PV\ (c+1) - PV(c)}}{\dfrac{C + \dfrac{EV - PV(C)}{PV\ (c+1) - PV(c)}}{AT}}$$

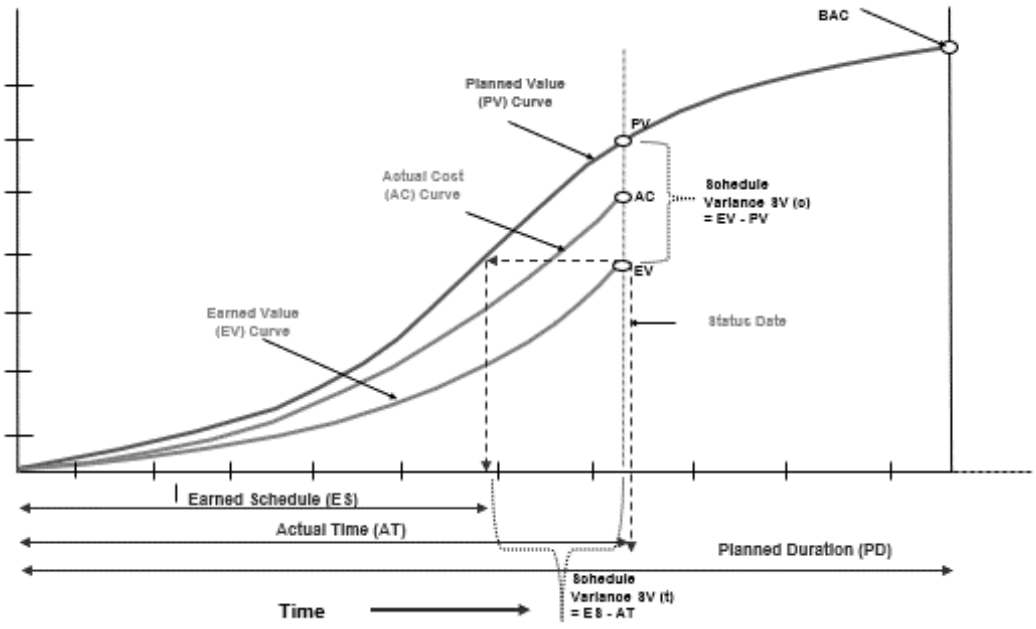

Figure 20. Graphical representation of the Earned Schedule

2.3 Burn Down charter

For agile project progress visualization, the burn down chart was introduced, which can be a powerful tool for any project. At a Sprint-level, the burndown presents the easiest way to track and report status, i.e., whether your Sprint is on or off-track, and what are the chances of meeting the Sprint goals. The burndown chart can provide near real time updates on Sprint progress. At the beginning of a Sprint, the Scrum team perform Sprint Planning and agree to take on development work worth a certain number of Story points. This forms the basis for the Sprint Burndown chart. The total story points agreed at the beginning of the sprint make up the y-axis, and the individual dates in the Sprint make up x-axis. There are two lines represented in the Burndown charter: one represents the ideal progress rate, the other the team progress rate. Of course, not all sprints are made equal. So actual Sprint Burndown may not look as the ideal progress. For instance, Scrum teams are prone to overestimate their ability to deliver during their first development Sprint on a new project. Or if they are a newly formed team. Or if they are learning to work Scrum. In such cases, it's quite possible that the team fall behind schedule and the burndown chart helps bring issues to the surface. The ideal progress bar serves as a guide for the Scrum team by showing them where they need to be. The team progress rate expresses the number of story-points (or tasks) completed per iteration. Only tasks completed at the end of the iteration are counted.

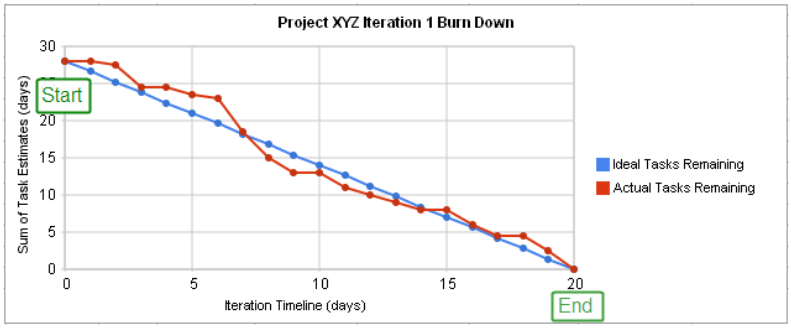

Figure 21. Example of Burndown charter in a Scrum project

After a few sprints, the speed of a Scrum team will most likely be predictable and allow for a fairly accurate estimate of how many points it produces during a sprint.

The sprint velocity is a metric that helps the team to achieve the ideal Sprint burndown. It represents the average number of story points a team can take on for a Sprint. This number is based on observing how many story points were delivered during the previous Sprints, and simply calculating the average story points delivered per sprint. When you know your team's velocity, it is then going to be easy to manage how much work they can commit to at the beginning of a Sprint.

2.3.1 Burnup charter

Essentially, a Burn-up chart seems to be an inverted Burn-down chart. However, a Burn-up chart offers a piece of very crucial information that a Burn-down chart doesn't offer, which we will shortly get to that in a moment. Both the axes in a Burn-up & Burn-down chart are the same. The only difference is that - the Burn-up chart showcases the amount of work that is completed along the way, instead of showing the remaining amount of work. The main focus of a Burn-up chart is to show how much work the team has completed throughout the product life cycle. The Burn-up chart also shows the ideal Burn-up rate, similar to the Burn-down chart. But, there's one vital piece of data that the Burn-down chart doesn't show and the Burn-up chart does show, i.e. Burn-up charts show a scope line that simply tracks the time at which tasks are removed or added. If a certain product tends to have a fixed scope, then it's always a better choice to use Burn-down charts, to keep things less complicated for the viewer. However, if the scope of a product is ever-changing and there's a need to showcase progress at every step to the stakeholders, then it's better to go ahead with a Burn-up chart.

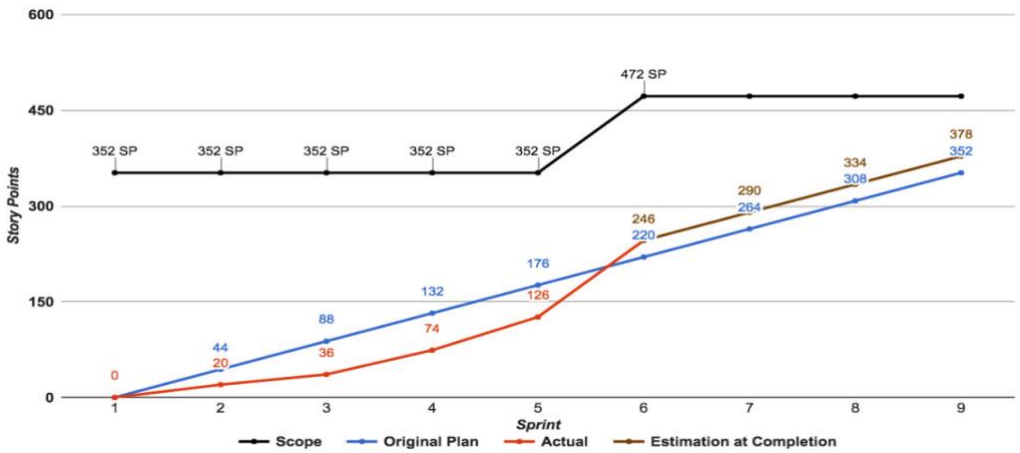

Figure 22. Example of Burn Up charter in a Scrum project

2.4 Putnam model

The Putnam model can be used to predict and manage software development projects. It is a management tool that takes, as input, easily obtained manpower data and produces cost and schedule estimates. The original paper by Lawrence H. Putnam published in 1978 is seen as pioneering work in the field of software process modelling. Created by Lawrence Putnam, Sr. the Putnam model describes the time and effort required to finish a software project of specified size. SLIM (Software Life cycle Management) is the name given by Putnam to the proprietary suite of tools his company QSM, Inc. has developed based on his model. It is one of the earliest of these types of models developed, and is among the most widely used.

Traditional methods develop to work in an Agile environment are essentially static. On contrary, the Putnam model reflects a dynamic software Life-Cycle Model Approach. It will be instructive to examine the data that led to the formulation of the life-cycle model. The data are aggregate manpower as a function of time devoted to the development and maintenance of large-scale business type applications.

Putnam noticed the software staffing profiles followed the well-known Norden-Rayleigh distribution. Norden took the Rayleigh distributions and observed that it provides a good approximation of the manpower curve for various hardware development processes, so he plotted the Norden's curve in *Figure 23* that represents the manpower as a function of time.

Norden's proposed an equation to represent the Rayleigh curve:

$$Effort = \frac{A}{t} * t_d * e^{-\frac{t^2}{2t_d^2}}$$

Where:

- E is the effort required at time t
- A is the area under the curve
- Td is the time at which the curves attain maximum value; td is the time the curve reaches a maximum. Empirically this is very close to the time a system becomes operational and it will be assumed hereafter than t-Pmax = td = development time for a system.

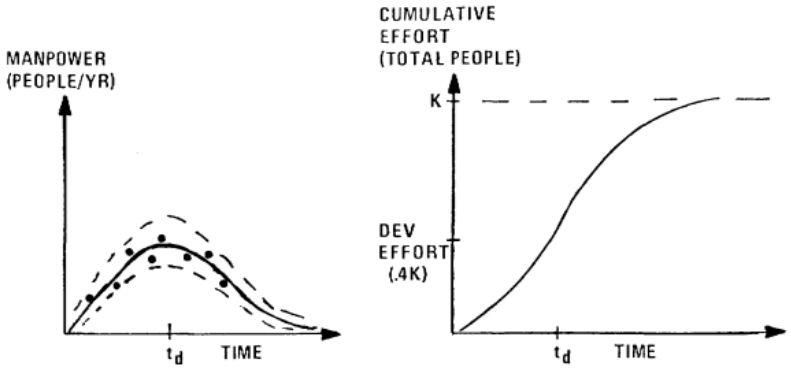

Figure 23. Expected manpower behavior of a software system as a function of time

The manpower included in the graph is for the entire project from requirements specification through planning, design, modelling, release, and product support. The 40:60 rule says the effort requires to develop the product (design and code) is equal to 40% of the total budget while the effort requires to maintain the product (modification and upgrades) is equal 60%. Knowing that the peak occurs after about 40 percent of the schedule gives the first good prediction of the project completion date from the left graph in Figure 18. It is more difficult to project a completion date from the cumulative data. The area under the curve represents the total number of man years expended. When multiplied by the average cost per man year, the area under the curve represents the cost of the job. By manipulating the time of the peak and the size of the peak, different costs are obtained.

The first assumption of the Putnam model is that all software projects follow the Rayleigh-Norden curve. Only a small number of engineers are required at the beginning of a plan to carry out planning and specification tasks and as the project progresses and more detailed work are necessary the number of engineers reaches a peak. After implementation and unit testing, the number of project staff falls.

Putnam agreed with the base observations of Norden and revised it. He further proposed another simpler model.

The software equation:

$$\frac{B^{\frac{4}{3}} * Size}{Productivity} = Effort^{\frac{1}{3}} * Time^{\frac{4}{3}}$$

- Size is the product size (whatever size estimate is used by your organization is appropriate). Putnam uses ESLOC (Effective Source Lines of Code) throughout his books.
- B is a scaling factor and is a function of the project size.
- Productivity is the Process Productivity, the ability of a particular software organization to produce software of a given size at a particular defect rate.
- Effort is the total effort applied to the project in person-years.
- Time is the total schedule of the project in years.

The software equation in terms of Effort becomes:

$$Effort = \left[\frac{Size}{Productivity * Time^{4/3}}\right]^3 * B$$

From this equation we can easily see that the effort will not be linear during time, it depends on the life cycle of the project. A constant level of manpower throughout the project duration would lead to:

- Wastage of effort
- Increase the time and effort to develop the project
- Some phase would be overstaffed and understaffed
- Increase the cost of development

The Putnam-Norden-Rayleigh (PNR) Curve, *Figure 24*, provides an indication of the relationship between effort applied and delivery time for a software project. The base concept is that putting more people on a project does not decrease time linearly, the curve obtain is the one in Figure. The implication is that delaying project delivery can reduce costs significantly. The curve indicates a minimum value t_d that indicates the least cost for delivery (i.e., the delivery time that will result in the least effort consumed). As we move left of t_d, so as we accelerate delivery, the curve rises

nonlinearity. A level of effort E_d will be required to achieve a nominal delivery time t_d, that is optimal in terms of schedule and available resources. Although it is possible to accelerate delivery, curve rises very sharply to the left of t_d. If we attempt further compression of time the project moves into the 'impossible region' and the risk of failure becomes very high. In this region the project delivery time cannot be compressed more beyond 0.75 t_d. The PNR curve also indicates a lowest cost delivery option, t_o. A small compression in delivery schedule can result in substantial penalty on human effort and hence project development cost.

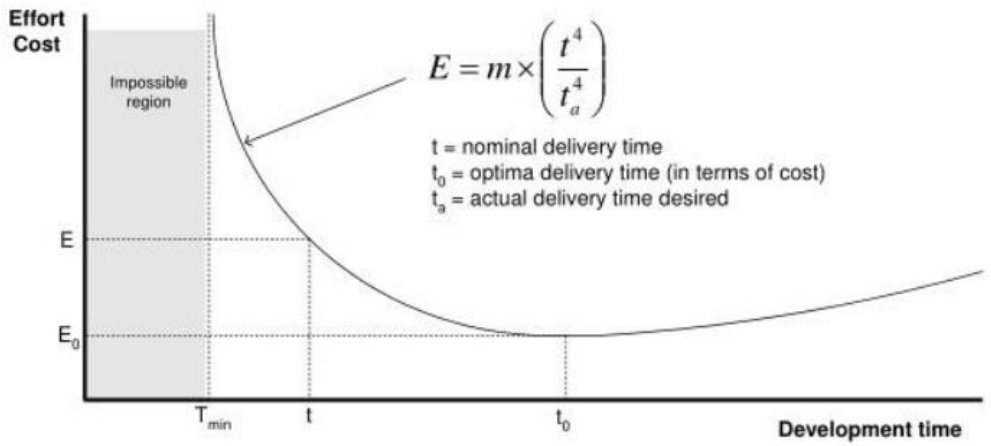

Figure 24. Putnam-Norden-Rayleigh (PMR) Curve

There is a common management myth that if the project is late from its delivery dates then by increasing more people in the development team can complete the project on time. It does not work in the software development environment because increasing more people in the final phases will delay the project even further. The main reason are as follows:

- new people need time to synchronize with the existing team
- new people need training work on the project
- there will be no or little work on the project because trainers are the team members working on the current project.

In the SW development environment projects schedule are flexible i.e., can be reduced or extended. It is possible to compress a desired project completion date by adding resources to some extent and it is possible to extend a competition date by

reducing the number of resources. Most software organizations, regardless of maturity level can easily collect size, effort and duration (time) for past projects. Process Productivity, being exponential in nature is typically converted to a linear productivity index an organization can use to track their own changes in productivity and apply in future effort estimates.

Relevant for the Revise project that I will propose in the internship project application is the Putnam theory for the Productivity variable. In his model the productivity is the average productivity that remains constant for the whole duration of the project. Putnam takes in consideration another variable that is the Difficulty of the project. This latter is in relation with the average productivity following this equation:

$$PR = C_n * D^{-\frac{2}{3}}$$

So, the productivity for different projects will be the same only if the difficulty is the same. This does not seem reasonable to expect very frequently since the difficulty is a measure of the software work to be done, i.e., $K/t_d = D$ which is a function of the number of files, the number of reports, and the number of programs the system has. Thus, planning a new project based on using the same productivity a previous project had, is fallacious unless the difficulty is the same in the Putnam theory. This important limitation of the Putnam model implies that the team works with a constant learning curve. The revise model will highlight the importance to take into consideration this variable. In the learning theory there is a proportion between the performance on a task and the number of attempts or time required to complete it. This can be represented on the continuous line in the graph below in *Figure 25*, while the dotted line shows a linear increase in the learning process.

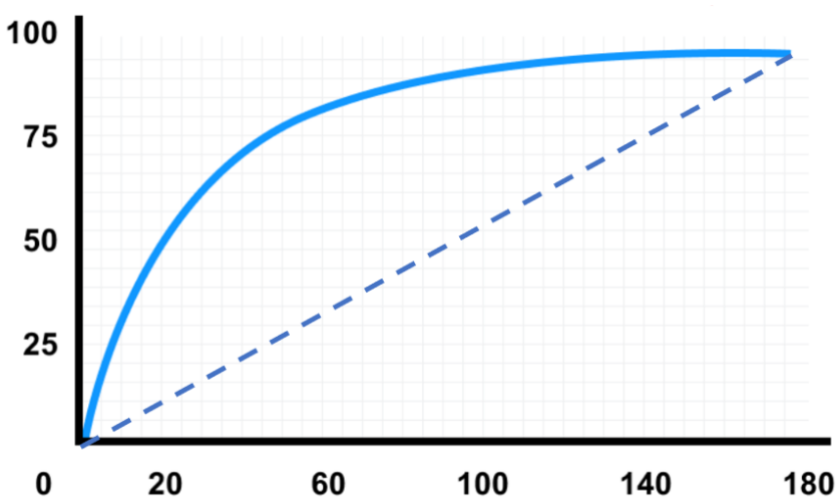

Figure 25. Learning curve

In case the team works following a learning curve the productivity increases until a max of 100%. As a consequence, in the software equation the denominator will increase and the effort will decrease during the project.

3. Analysis and results of a Scrum Project

I attended six months internship in an IT company in Sophia Antipolis, Cote D'Azur, France. In this experience the company made us simulate a Scrum team working for an internal project. Although in this internship I worked for the whole duration also as Developer I will not detail the development of the application, but I will just briefly introduce it. I will focus on how the agile framework is implemented in a IT company and I will analyze the data collected during my experience studying the project performance. In particular, I will monitor my project throw the burndown charter and the Earned value model. These two methods suggest unlike decisions to take in the project due to a different forecasting of performance. The data I have collected from the internship will be useful also to criticize the Putnam resource allocation model and propose a Revised theory. The Putnam model, as I have already explained in the chapter before, is very used in IT companies and it bases his focus on the software estimation of time and effort. Nevertheless, a Putnam hypothesis was investigated and here I'm proposing a variation of this model. The main difference from the two models is in one variable that for the old model is constant while in the Revise one is increasing during time following the learning curve. The importance of take in consideration the learning curve is pointed up also speaking about the Burn Down charter, in fact I will show that a Project Manager can make errors in prevision if he/she considers it constant, and this can comport delay in the schedule, bad partition of the effort and wrong allocation of resources. The forecast obtained with EMV, Burndown charter, Putnam and the Revised model has been compared with the result data of the project at the end of the internship in order to investigate which model better estimate performance.

3.1 Overview of the project

The project consists in the development of a web-application mainly used for Business Manager and HR. Its main purpose is to ease the collaboration between the two in order to have a flawless and interactive way to handle the scouting, hiring and career follow-up of consultants. This project is called MiamiV2. The application already exists but it wasn't very appreciated by the clients because hard to use and with a lot of features missing. The client asked also to implement in the old version

the “Alten Grains”, a way to remunerate consultants for their activities. The product is a completely new web application, creates from scratch starting from the analysis of the needs of the clients. For this reason, they didn’t allow us to see the old version of the project. The product has been realized using exclusively native Cloud Technologies (Microsoft Azure, AKS, Redis Stream, ServiceBus, FaaS, …). During the development phase we have tested real world project management methodologies and see which ones are best suited for the development team. The main real goal for the company, apart to deliver a product, was to train us, as interns, to work in a real Scrum environment. In particular, at the beginning, we spent a lot of time analyzing all the roles, the ceremonies and the philosophy of the Agile framework. To succeed to this goal, we were personally involved in all the roles that take part in the development of a modern application. For the whole duration of the Project, usually every two iterations, we have turned the Scrum roles. I was the one with a more functional path of studies, for this reason I kept the role of the Product Owner for the majority of time. In the meanwhile, I had always worked as frontend developer.

To be ready to start the Project we read the paper of specifications to identify the customer needs, which are the different roles in the company and what they can see on it. The main initial requirement from the costumer (ALTEN) was a user-friendly website that employees can easily use. One important feature is the multi-tenancy, Miami V2 must be developed as a single cloud-native SaaS solution. Multiple business unit should be able to use Miami V2 without interferences between them. Initially we were involved in several meetings and workshop with roles with different responsibilities in the company with the aim to answer all our questions and so to better understand the functionalities that the website will have. Something that has taken up a lot of our time at the beginning were the lessons done in order to explain:

- The Agile manifesto and the Scrum framework
- The role of the Quality Assurance and how to test the software
- Architectures
- Cloud infrastructure creation
- Story mapping
- backlog and sizing (in Jira)
- How to organize and incur a workshop

The two screenshots below are the layout of two pages of the developed project ‘Miami V2’. I entered in the web page as ‘Business Manager’, so these are filtered by his visibility. In the *Figure 26* is displayed the dashboard with reactive and moveable widgets. Every person has a dashboard as entering page but the BM differently from others has widgets related to the statistics of Grains and details about consultants. In the Consultant page in the menu is possible to see all their information, the project in which they are working on and the history of their performance. In the *Figure 27* we can see the ‘Alten Grains’ page. All the requests made by consultant to the BM are listed. The BM can accept, deny or not reply at them. In this last case they are displayed in the ‘pending’ requests.

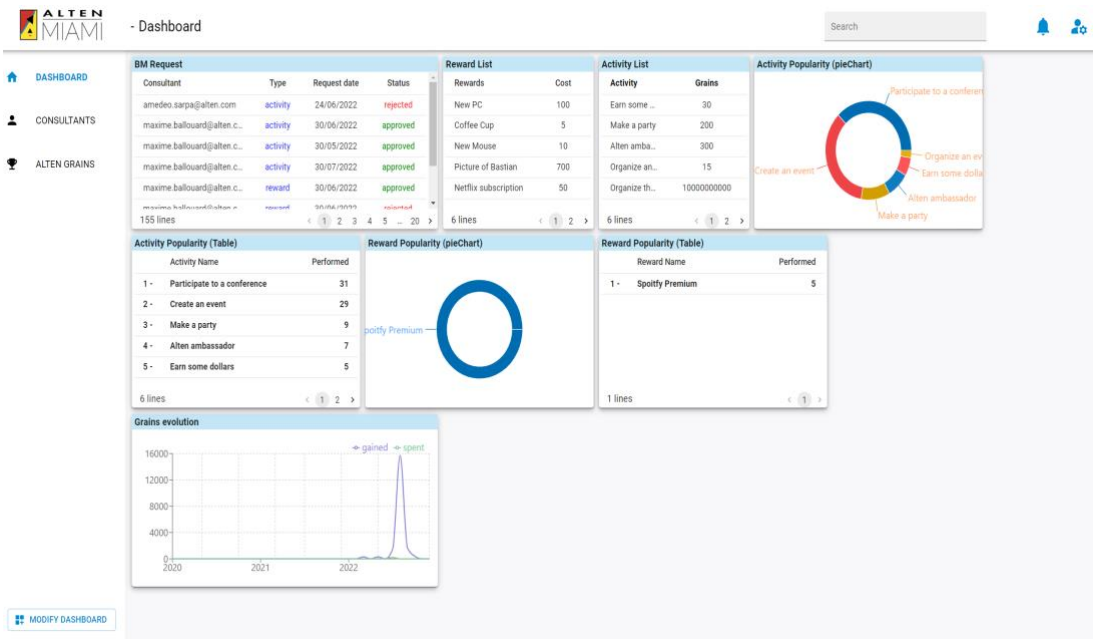

Figure 26. Screenshot of the Dashboard of Miami V2

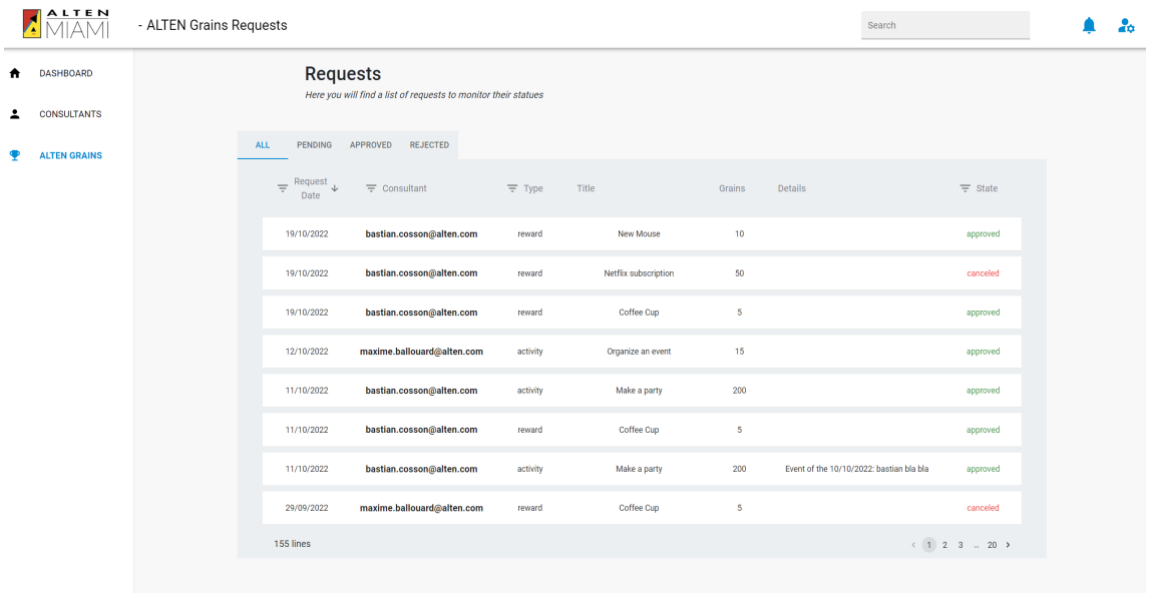

Figure 27. Screenshot of a page of Miami V2

3.2 Initialization and implementation

After having read all the specifications of the client we had an initial 'Sprint 0' in order to transform these requirements in functional and technical diagrams.
We created the C4 model, the vision board, the functionalities, the costumer journey and the story mapping. The C4 model is a simple way to communicate software architecture at different levels of abstraction, the four "C" stay for: Context, Container, Components, Code. This type of models is useful for the technical staff of the team, the developers. As I previously said, I will focus more on the functional diagrams respect to the technical one.

3.2.1 Vision board

Regarding the functional side we fill the vision board to have a clear image of the product. A vision statement is a written declaration clarifying your business's meaning and purpose for stakeholders, especially employees. It describes the desired long-term results of your company's efforts. The target group is the final users that will use our application and the needs are the reason why they will use it. In the product we wrote the main features the product is going to fulfil. The value is the benefit added in the development of the project. In our case, an old application

with the same goal already exists but it is not used much by employee because it's not "user friendly". Therefore, as we can see from the *Figure 28* one aim is to make it easier to use. Another added value from the application, as I have already said, is the Alten Grains feature.

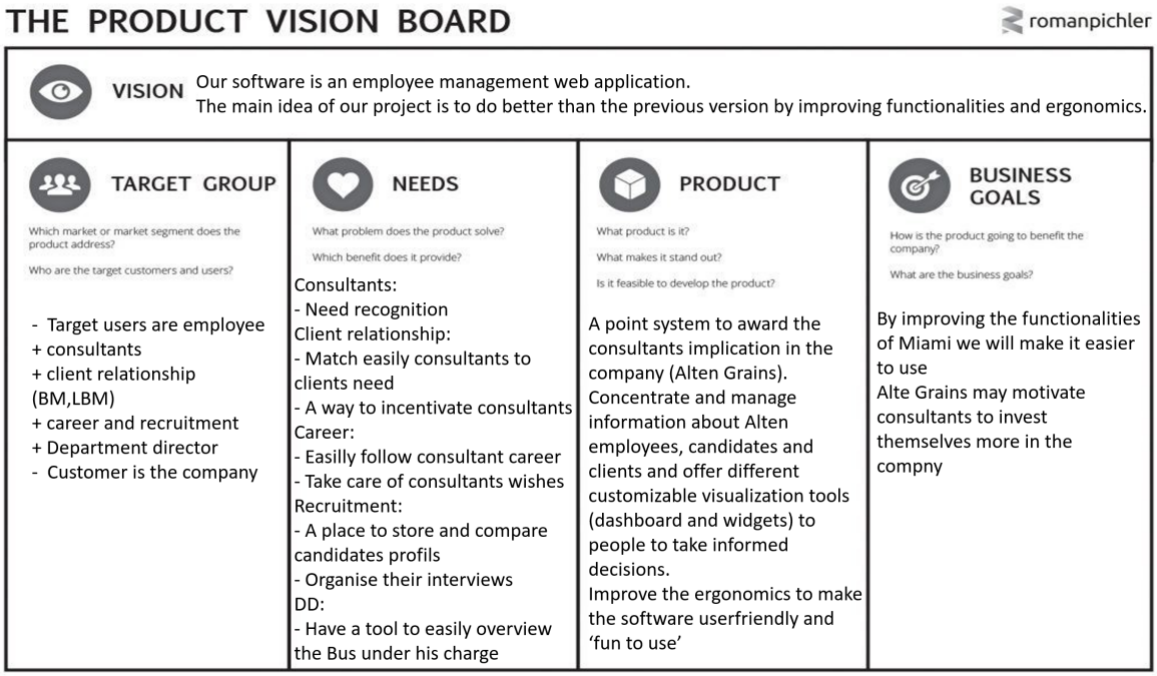

Figure 28. Vision Board

3.2.1.1 Functionalities

Thanks to the 'functionalities' diagram we have a general overview of what a user can do as soon as he logs in. To have a better vision of our application we analyze each Persona indicating each functionality he is able to do. The main function, so the main reason why the user wants to use our platform, is positioned in the center and circled in this graph.

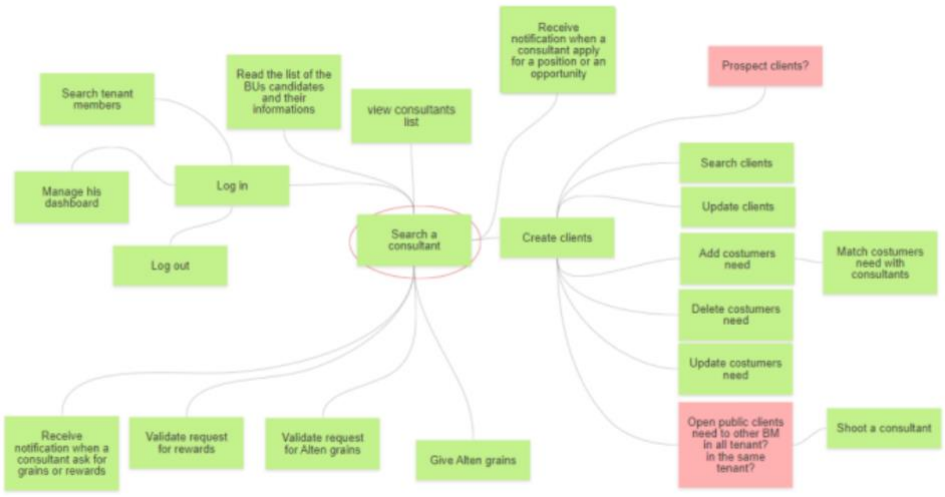

Figure 29. Functional diagram of MiamiV2

3.2.1.2 Costumer Journey

The costumer journey is a graph that allows to represent the succession of actions of each user. This leads the team to propose some idea of how they want to design the application. Starting from the login each user will have a similar dashboard but the pages he is allow to see and the functionalities he will be able to use are different due to the different goal they are using the app. For space problem I do not report the graph here. For the big dimensions a image of the Costumer journey is not shown.

3.2.1.3 Story mapping

For both the goal of the application so, the Alten Grains (shown in the *Figure 5*) and MiamiV2 we develop a user mapping. In the green post-it are shown the people that are interested in the pink post-it, that specific functionality. With “Everybody” we mean that each user is going to use that function. In the yellow post-it we wrote Who, How and Why they should use that function. In the story mapping the standard sentence "AS …I CAN …. SO THAT …” is used to write the user requirements. The last part regarding the why (so that) sometimes is not important if the aim is easy to predict, and for this reason sometime is left out. The user stories may be organized

within hierarchies, we use a prioritization from P1 to P3 to understand from where we should start.

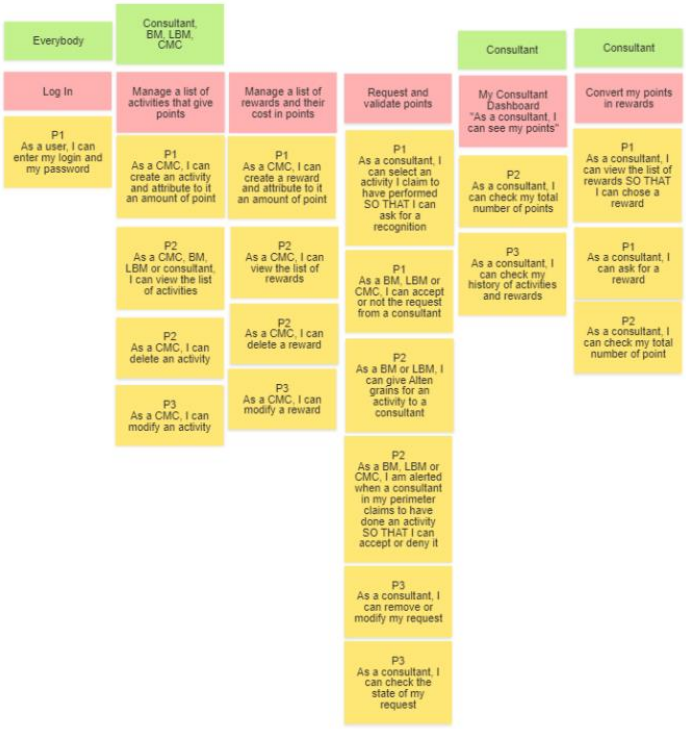

Figure 30. Story Mapping of 'Alten Grains'

3.2.2 Implementation of the Scrum methodology in an IT company

The main objective of this project is to understand how the agile methodology works. For this reason, each member of the team has worked autonomously in a transparent environment, respecting all the ceremonies of this framework.

Our Sprint 1 started the March 28 and every sprint lasts two weeks, for a total of ten days of work on each. The resources allocated on the project were not constant for the whole time. In particular, from the iteration 1 to the iteration 4, nine people worked for this project. Starting from the sprint 5 four resources were assigned in a different project, for the back office. In the sprint 6 a new inter enjoyed the team and the project has continued with six resources until the end of it.

Each morning the team meets for the stand-up meeting. This event has a duration of 15 minutes regardless of the people who were to speak. Often the time is not

adhered to precisely as additional issues are often discussed even though it is not the right meeting to do so. It is very important, therefore, to have defined timelines.

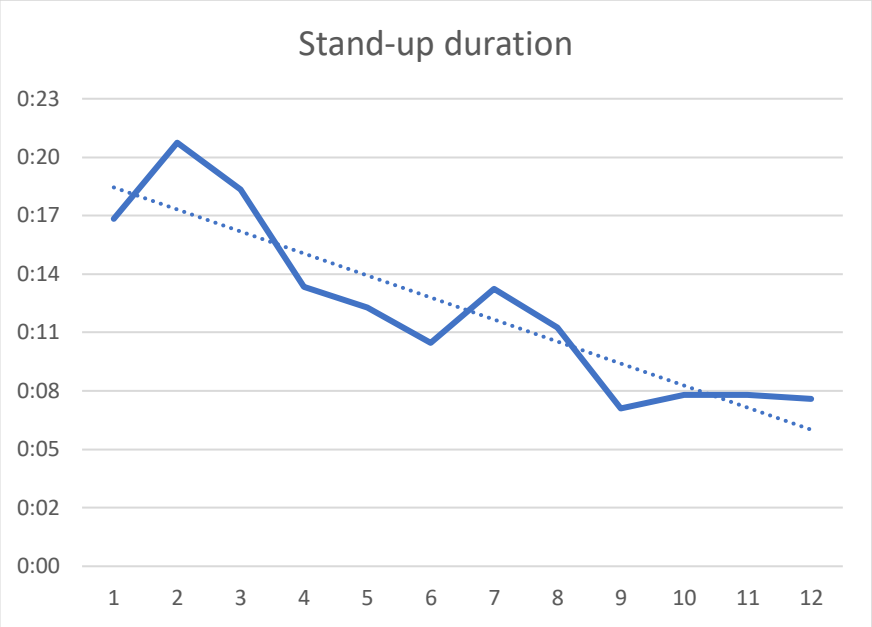

Figure 31. Trend of the Stand-up meeting

As we can see from the graph in *Figure 31*, the team does not keep the duration of the stand-up meeting constant and this can be attributed from not having a true scrum master, as he is who has to make sure that the duration of this meeting is met. Moreover, we can see that the trend is decreasing, at first the team did not know how to deal with this ceremony and took more time than necessary to explain what they had done the previous day and what they should do on the current day. Over time we realized that the importance of this event is to focus on the main events without going into the details of the tasks addressed. Reducing a meeting that is addressed every day implies a not insignificant increase in time spent on the project. Not relevant was the decrease in the number of people on the team because although there were fewer members the responsibilities and tasks performed by each one increased and consequently increased the amount of time each person spent explaining what he or she did the previous day and what he or she will do on the current day.

With the presence of an experienced Scrum Master we probably would have had a constant meeting duration of 15 minutes, as we can see though, very often this time

was excessive. During the retrospective meeting, in the company I'm working now, we discuss often about how to make the stand-up meeting more efficient. In fact this is meeting is usually criticized and is very common to find companies working in full Agile but not conducting this event every day, but on alternate days. Also, since the agile philosophy is very focused in transparency usually the teams work in an open space environment, which facilitates communication, and a very complete view of the product backlog indicating what tasks everyone is doing is also had through a common dashboard. To track and manage our project we use the tool Jira.

The morning of the last day of the iteration was dedicated to the Sprint Review with stakeholders. This meeting is an hour and a half long and was divided into three parts. Initially, the scrum master discusses the tasks that were completed in the current sprint and what could not be completed, explaining the reasons why. Next, what was done is shown by the product owner through a demo. At this time, it is very important to gather the customer's opinions and show the various implementation options. In conclusion the goals that the team wants to achieve in the next sprint are shown.

After the review the team meet for the Retrospective. In this meeting the team exposes general feedbacks about how the Sprint has gone. Each person had to reflect on the problems and the positive things of the sprint just finished both on a personal and group level. No technical comments are addressed.

In the afternoon we carry out the Sprint Planning. After receiving customer feedbacks the team with the help of the Product Owner decides which tasks to carry out in the next sprint. The Product owner first of this event is in charge to prioritize the tasks and writing the user stories in a way that are easily understood by everyone. In order to decide what it will be developed in the next sprint the team has to re-size each task considering the difficulty on implementing it. To size the tasks we use the 'Poker planning'. Each estimator is holding a deck of Planning Poker cards with values that follow the Fibonacci sequence from 1 to 13 as maximums. The values represent the number of story points, so how much time the team thinks they will spend on it. In this way the members can compare if the estimate values are very different and as a result, better understand the task. The final size is the average of the estimate values.

The stand-up meeting, the review, the sprint planning and the retrospective were regularly done in each sprint while the refinement was usually planned when the team needed it.

3.3 Monitoring

During sprint 6, almost in the half of the internship, I monitored the project in order to understand its progress, predict its end and suggest any decisions in case of delay. In this part I'm monitoring throw two techniques already explained: The Burndown chart and the Earned Value. The Burndown charter graphically shows where we are compared to the schedule and thanks to the velocity of the team, directly proportional to the learning curve, we are able to predict the future progress of the project and its deadline. I applied the mathematical formulation of the Earned Value method to monitor and to predict the project time and duration. We will see through this chapter that these two methods will lead to different scenarios.

3.3.1 Burndown charter

Agile teams use this simple, visual tool to determine the work they have completed on a project and the work that is yet to be done within a given time period, or as Scrum teams call it, a Sprint.

In order to study the progress of the project from its initial, March 28, up to the monitoring date, June 15, I took into consideration the "theoretical points" and the "actual points" in 6 iterations. The "theoretical points" are the points that the team planned to make in each sprint while the "actual points" are the points that have actually been completed, which is equivalent to the tasks in 'Done' in the Sprint Backlog plus the percentage completed tasks in 'In progress'.

In these six sprints the team was scheduled to perform a total of 185 points, which are equivalent to the "theoretical points", instead only 125 "actual points" have been completed, which show a delay in the scheduling.

The speed was obtained by dividing the points done by the actual duration of each sprint, which is equivalent to 10 days. In order to complete all the tasks planned, the team should had worked with an optimal speed of 3.08 (AVERAGE

(3.5,2.3,3.3,2.9,2.7,3.8))in these 6 sprints, so 60 days in total, but it managed to sustain an average speed of 2.2 (AVERAGE(0.9,1.1,2.3,2.5,2.5,3.2)).

	Iteration 1	Iteration 2	Iteration 3	Iteration 4	Iteration 5	Iteration 6
Theoretical points	35	23	33	29	27	38
Actual points	9	11	23	25	25	32
Theoretical velocity	3.5	2.3	3.3	2.9	2.7	3.8
Actual velocity	0.9	1.1	2.3	2.5	2.5	3.2

Table 1. Data monitored in the six Sprints

Through this information I was able to design the Burndown charter. The "Optimal" line was found subtracting at each iteration the points that had to be carried out with an optimal speed, while the "Effective" curve was found by subtracting the points that were actually completed. From the graph it can be easily seen that the actual curve shows a delay in the project compared to the theoretical one. The reason of the delay is not understandable just from the graph but throw the data of the advancement of tasks and the velocity progress we can have an idea.

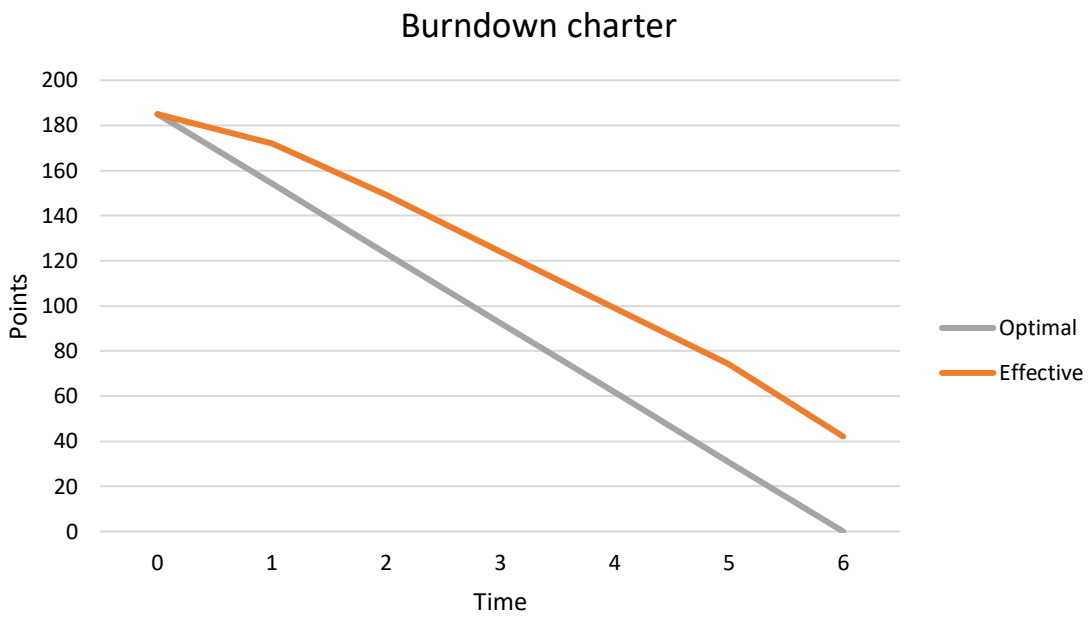

Figure 32. Burndown charter until Sprint 6

For the comparison with the other models and for the forecasting I will use the Burndown charter. Here below in *Figure 33* the Burnup charter is reported in order to point up that with the data collected is possible to plot both the graphs.

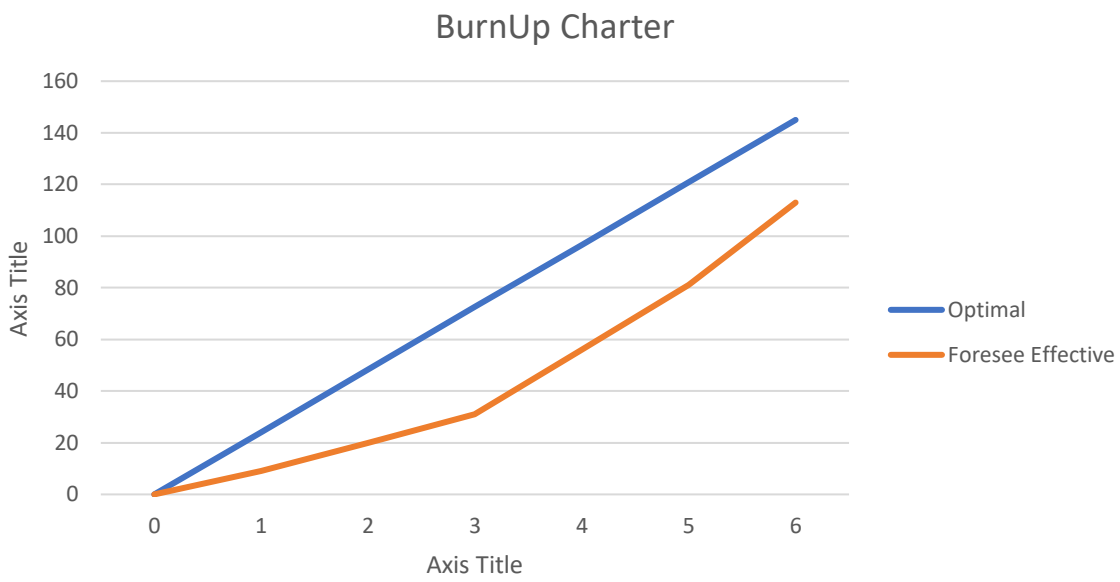

Figure 33. Burnup charter until Sprint 6

We can identify 3 major reasons that cause this delay:

- High optimism in the first sprint due to inexperience in sizing and lack of knowledge of an average team speed. This led the team to accumulate unfinished tasks.
- In the assignment of sprint tasks, especially in the first period, the team's learning curve was not considered. The team is initially composed of 9 interns, with no experience behind them, different courses of study and different knowledge of the technologies used. For this reason, it is very important to consider in the sizing that the team is not working at 100% of its knowledge and that therefore part of its time will be dedicated to the "Strike". Different initial knowledge leads the resources to position in different learning curves shown in the *Figure 34*.

- The curve 1 in the case the resource does not have high IT knowledge.
- The curve 2 in the case the resource has a path of IT studies but without knowing the technologies used
- The curve 3 in the case the resource has an IT path and a discrete knowledge of technologies.

The curve in blue is the average learning curve of team and it depends on the number of resources. In this case a new resource arrived in sprint 4 while in sprint 5 a part of the team was assigned to another project for this reason the learning curve decreases.

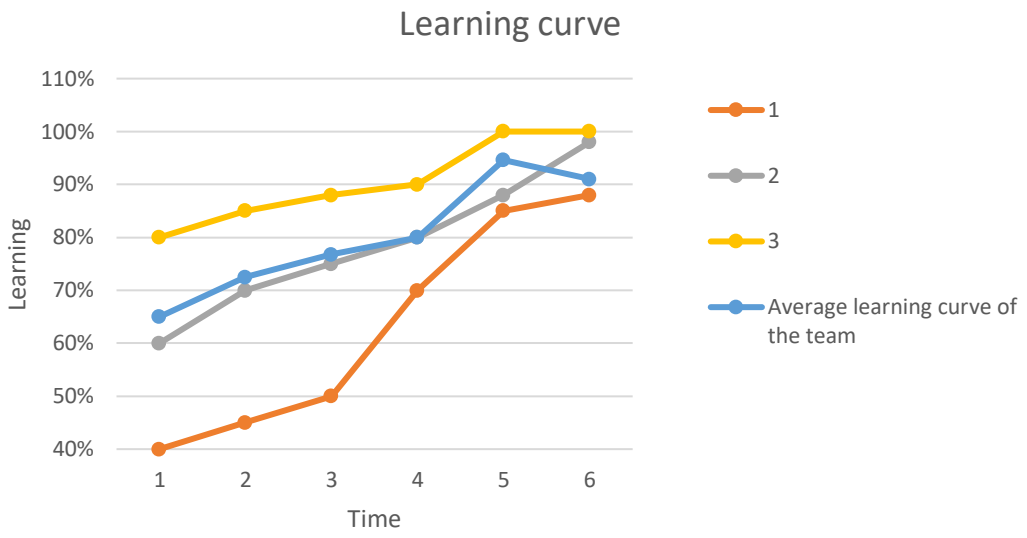

Figure 34. Learning curve of each group and average learning curve of the team

- The third factor that had not been taken into account in the sizing is the actual time of work, ie the time that really adds value to the project. Especially in the initial phase, the team was engaged in various lessons on technologies (which positively have influenced the learning curve), on the Agile methodology, and on the company's philosophies.

As a demonstration of what has been said, I have repeated the calculations considering the team's learning curve and the time that actually gave an added value to the project, that is the theoretical time minus the time due to the various

ceremonies and the strike time. With these considerations we can see from the *Table 2* that the points actually made and those the team has planned in the sprit planning are much more aligned. It is, therefore, very important for future planning to take into account the team's learning curve and actual working time.

t	1	2	3	4	5	6
Average learning curve of the team	65.0%	72.5%	76.8%	80.0%	94.6%	91.3%
Points able to do	13.9	13.0	25.3	23.2	25.5	34.7
Points really done	9.0	11.0	23.0	25.0	25.0	32.0

Table 2. Monitoring considering the real working time and the Learning curve

3.3.1.1 Burndown charter forecast

The Burndown charter gives us an idea of how the project is performing in the moment of monitoring. To get more information on which could be the progress of the project in the next sprints we need to make predictions. The end date of our project is 26/08, so the project must be completed in exactly 6 months of internship since sprint 1 started on 28/03. Taking into consideration the data I have found in the monitoring, I foresee the behavior of the project drawing a forecast curve in the Burndown charter, from Sprint 7 until its end.

In order to plot the curve, some data have been hypothesized thanks to historical data. The future speed of the team has been considered as the average of the speed in the last three iterations. Moreover, the speed was multiplied by the delta of learning curve gained in each sprint because it is directly correlated to the knowledge of the team. To take in consideration only the time the team is working to add value to the project one day was removed considering all the meetings of the Scrum and in alternating iteration another day was considered off for possible future lessons and events. From these data was possible to obtain the "Points able to do", multiplying the foresee velocity by the time that is actually expected to work. The learning curve of the team will reach the 100% just from the Sprint twelve and after this moment the speed will be constant for future Sprints.

t	7	8	9	10	11	12
Average learning curve of the team	93.7%	95.8%	96.7%	98.0%	99.7%	100.0%
Average speed	2.98	3.05	3.07	3.12	3.17	3.18
Point able to do	26.86	24.40	27.67	24.93	28.51	25.43

Table 3. Forecast of the next Sprints

Thanks to these forecasts, it was possible to design the Burndown charter of the entire project in order to see if the team will be able to catch up the delay. According to these estimations, the project will be concluded at the beginning in the Sprint 12, instead of the end of Sprint 11, the deadline is therefore set for 9 September 2022. Thanks to the increase in speed in carrying out the tasks, the previously accumulated delay is largely recovered despite this does not imply to completely finish the project on the deadline, but an iteration later. In this case the team in the twelfth Sprint should develop just seven points, so the delay can be completely recovered just increasing a bit the speed in the last iterations. Considering that the values have been estimated and that the delay is not alarming, the risk has been considered 'Accept' and no actions have been taken.

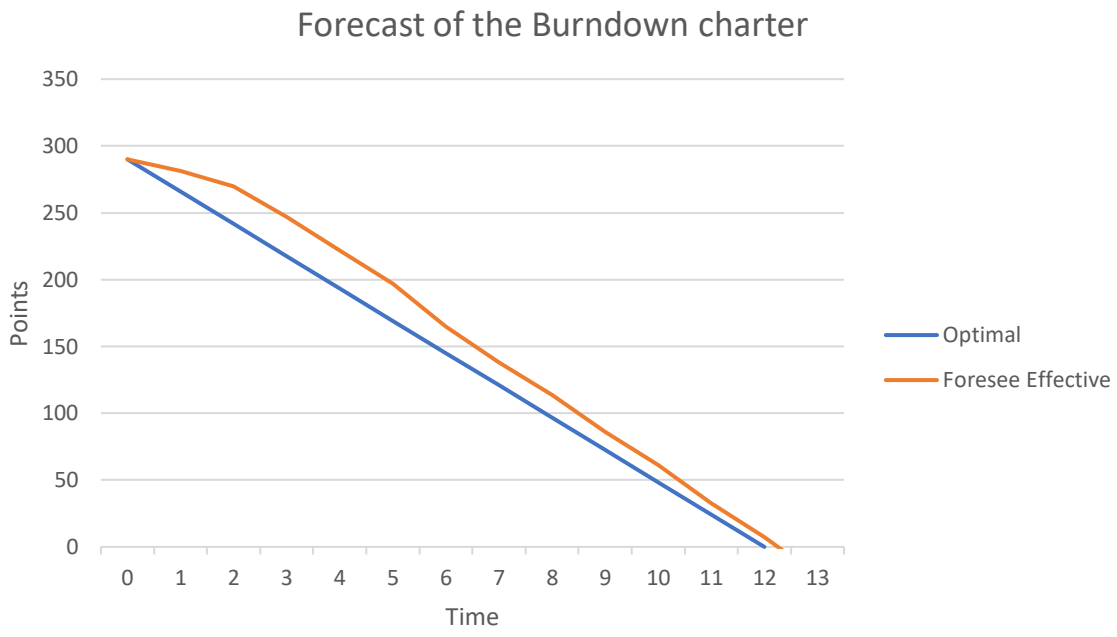

Figure 35. Forecast of the Burndown charter for the next Sprints

3.3.2 Earned value

The Earned Value is one of the most diffuse method to measure and forecast the project cost and time performance, comparing if the costs and work done are in line with what was planned. Compared with the burndown charter here we can have a detailer view of the delay.
The model is based on some input variables, which are explained below and which are represented in the *Table 4*:

- Budget cost; it is the cost that was planned at the beginning of the project which is equivalent to the cost of the interns plus the cost of the project managers in charge of following our internship. In this case it was assumed to be constant, not having additional information on planned costs. The sum of the total planned cost is called BAC, budget at completion.

- Work Schedule: it is the work that was planned to be done at the beginning of the project. Since the project consists of 290 tasks, it was considered that in order to complete all of them all it was necessary to finish at least 9% of the total for each sprint.

- Actual cost; it is the cost that in reality has been spent up to the monitoring date. Initially it is in line with the planned cost but from the fourth sprint it changes as there is a decrease in the resources allocated to the project compared to what was planned.

- Work performed; this is the work that has actually been done up to the monitoring date. In particular they are the tasks that the team has completed in each sprint with respect to the total.

Sprint	N. of resources	Budget cost	Actual cost	Work schedule	Work performed
1	9	5409.1	5409.1	9%	5%
2	9	5409.1	5409.1	9%	5%
3	9	5409.1	5409.1	9%	9%
4	8.75	5409.1	5284.1	9%	10%
5	5	5409.1	3409.1	9%	10%
6	6	5409.1	3909.1	9%	12%
7	6	5409.1		9%	
8	6	5409.1		9%	
9	6	5409.1		9%	
10	6	5409.1		9%	
11	6	5409.1		9%	

Table 4. Input data of the EV

Thanks to these variables and parameters is possible to calculate the indicators of project cost and time performance that are explained and reported in the *Table 5*:

- BCWS, also called PV (Present value), it was calculated as the product between the amount of work planned in each sprint (Work schedule) and the planned cost of the resources involved (budget cost).

- ACWP, also called AC (Actual cost), it was calculated as the product between the work actually performed (Work performed) and the cost actually incurred (Actual cost).

- BCWP, also called EV (Earned value), it was calculated as the product between the work actually performed (Work performed) and the planned cost of the resources involved (Budget cost)

Sprint	BCWS	ACWP	BCWP
0	0	0	0
1	491.7	167.9	167.9
2	491.7	205.2	205.2
3	491.7	429.0	429.0
4	491.7	455.5	466.3
5	491.7	293.9	466.3
6	491.7	431.3	596.9

Table 5. EVM variables

From these indicators is possible to obtain the cost variance and the schedule variance. We can see from the *Figure 36* that the cost variance is positive, the project is running under budget. The budget cost of work performed is higher than the actual cost of work performed so the difference between them is higher than zero, we are saving money. The schedule variance instead is negative, the project is behind schedule. The difference between the budget cost of work performed and the budget cost of work scheduled is lower than zero so, we are late. The information to be late in the project was already pointed up by the Burndown charter so, we need now to calculate other indicators to investigate more on the causes.

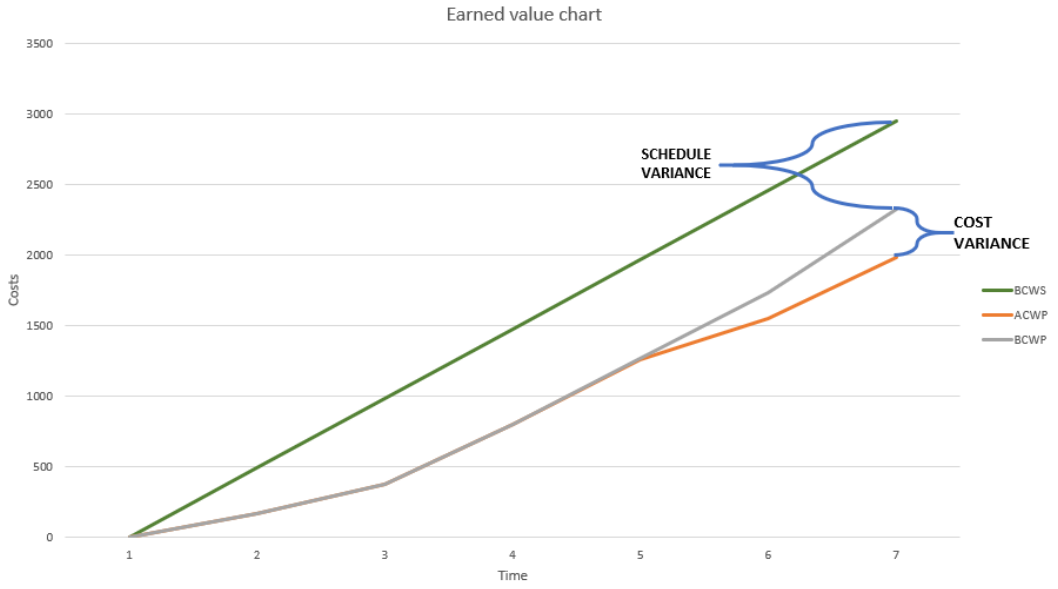

Figure 36. Graphical representation of the EV

Despite we are late, the curve of the BCWP is increasing and could reach the BCWS. To have a clearer value of the delay I calculated the SI, schedule index, as BCWP divided by BCWS in each sprint. If this value is higher than one we are beyond the schedule, instead if it is lower than one we are behind the schedule. In the *Table 6* we can see that the velocity of the team is quickly increasing so, the team is recovering the delay.

t	1	2	3	4	5	6
SI (€)	0.34	0.42	0.87	0.95	0.95	1.21

Table 6. Schedule index in monetary terms

In any case the total SI (€) for the whole project since the beginning until now is obtained as:

$$\text{SI (€)} = \frac{\text{BCWP}}{\text{BCWS}} = 0.79$$

So, the project is behind schedule of 1-0.79 = 0.21 €.
To have a clearer overview of the project performance I calculate the Cost index.

$$\text{CI (€)} = \frac{\text{BCWP}}{\text{ACWP}} = 1.18$$

This value is higher than one so we are saving money. The initial planned budget is more than what we are needing for the completion of the project and exactly is 18% more.

3.3.2.1 Earned Value Method forecast: TEAC and CEAC

The TEAC is calculated as:

$$\text{TEAC} = \frac{Duration}{SPI} = \frac{110}{0{,}79} = 139.2.$$

So, the team will spend 139.2 days to finish the project, instead of 110 days. I use the Revise formula to take in consideration that the team will have consequences of the delay accumulated.

These 40 days of difference imply a delay that is not indifferent. In fact, in this case the project will be concluded in the Sprint 14. The Earned Value is not as intuitive as the Burndown charter to give an idea about the progress and velocity of the team in the next sprint. To design the forecast trend, I foresee the work that will be performed in the future Sprints with the Moving Average method from Sprint 0 until Sprint 6. I used this method because take into consideration the cumulated delay until the beginning as the revised model to calculate the TEAC. The 100% of the work schedule will be completed in the Sprint 14 as we can see in the *Figure 37.*

Figure 37. Forecasting of the future Sprints progress throw EVM

The CEAC is found as:

$$\text{CEAC} = \frac{BAC}{CPI} = \frac{59.500\,€}{1.18} = 50.601\,€$$

Also in this case it is calculated with the revised method. The foresee final budget will be lower compared to the initial BAC of 59.500 €, we are saving money. The graph of the forecasting for the future costs is not very relevant because the Burndown charter doesn't say nothing about cost prevision and our goal is to compare this latter with the EVM and understand which one is more accurate.

We can integrate the cost and the schedule index calculating the CR, the Control Ratio. If this value is drastically lower than 1, we should take decisions to rake off the project. In our case is equal to 0.9. In this situation we could use the over budget to hire a new resource and so to crash the project.
Despite this any decision to crash or revise the project was done, for the reasons listed below:

- The Burndown charter shows that the team can catch up
- A new inter enjoyed the team in the last sprint
- The CPI is positive and the SPI is quickly increasing

3.3.2.2 Earned schedule

The schedule index SI has a monetary value even if it is a value of schedule performance and this can be confusing. For this reason, I calculated the ES, earned schedule. In fact, in contrast to the cost-based indicators from EVM, the ES schedule performance indicators are time-based, making the index easier to comprehend.
I calculated the ES (t) when the project is six months completed throw this formula:

$$ES\,(t) = c + \frac{BCWP(t) - BCWS\,(t-1)}{BCWS\,(t) - BCWS\,(t-1)} = 5 + \frac{2331.5 - 2458.7}{2950.4 - 2458.7} = 4.74$$

The constant c is the number of completed time increments for which the EV is higher than the PV, and in this case is equal to 5. From this result we can see that the team

for the tasks complete should be in the Sprint 5.74 compared to the Sprint 6, where it was planned to be.
I calculated the SV, schedule variance as:

$$SV = ES\ (t) - \text{time planned} = 4.7 - 6 = -1.3$$

This mean that the project is lagging behind schedule, with -0.3 delay.
Calculating again the SI, but in temporary terms, I found that it is equal to:

$$SI\ (t) = \frac{ES\ (t)}{\text{Schedule time}} = 0.8$$

Was possible to observe that the ES was around 4.75-4.85 also in the *Figure 36* above. Nevertheless, the ES gives as a clearer idea of the progress we can see that some limitations are still present. For this reason, I will apply the Putnam model, a model that was created to monitor software project.

3.3.3 The Putnam model and a revised model

The Putnam model is an empirical software effort estimation model. It describes the time and effort required to finish a software project of specified size. Thanks to this model is possible to predict project completion times and manpower requirements as the project evolves. There are numerous factors that affect efforts and time to develop a software. These factors include complexity of software and experience of the development team. In this part of the analyses, I propose the use of a model that take in consideration this latter and I will compare it with the Putnam model and afterwards with the actual results.
The dependent variable is usually software effort, where the independent variables include software size and some non-functional requirements.
The formula that allows to calculate the effort is here reported:

$$E = \left[\frac{Size}{Productivity * Time^{\frac{4}{3}}}\right]^3 * B$$

- Size; The model is estimated at the planning time and for this reason, consistent with what was said earlier, it was considered that in order to complete the 290 task points the team can work with a linear trend. The team then, in each sprint must finish 9% of the tasks to finish on time.
- Time, is the time in which the project should be finished. I consider the 11 sprint as the final deadline with milestones every ten days of work, corresponding to the sprint.
- B is a scaling factor and is a function of the project size.
- Productivity; in the Putnam model the dependence of the productivity on the time is essentially absent. The productivity is considered a constant which is somehow a measure of the state of technology being applied to the project and it can be increased by applying better technology. The constant seems to relate to the overall information throughput capacity of the system and it seems to be more heavily dependent on machine throughput than other factors. Typically, this constant is: Ck = 2 for poor development environment, Ck= 8 for good software development environment and Ck = 11 for an excellent environment. To make the comparison with the next theoretical model I consider it as a percentual and in particular, equal to 100% in each sprint. With this consideration the productivity is independent to the time, this mean that the team is not following a learning curve.

Plotting effort as a function of time yields the Time-Effort Curve in *Figure 38*. The points along the curve represent the estimated total effort to complete the project at some time. The curve as the shape of a bell curve moved more on the right. This type of curve is called the Rayleigh distribution. Cumulating the Effort value we can see that the effort follows a S- curve. So, the effort is low in the first phase, while in the middle phases of the life cycle of the project the curve becomes drastically steeper, reaching a pick. As time passes and the project is in its last phases the effort decreases.

One of the distinguishing features of the Putnam model is that total effort decreases as the time to complete the project is extended. Putnam says that only a small number of engineers are required at the beginning of a plan to carry out planning

and specification tasks. As the project progresses and more detailed work is necessary, the number of engineers reaches a max, that in our case is in the Sprint 4. After implementation and unit testing, the number of project staff falls.

As the theory says, the peak occurs around 40 percent of the project and from it the delivery date can be found quite accurately. Coherently, we have the peak in the Sprint 4 that corresponds to the 36.36% of the duration of the project.

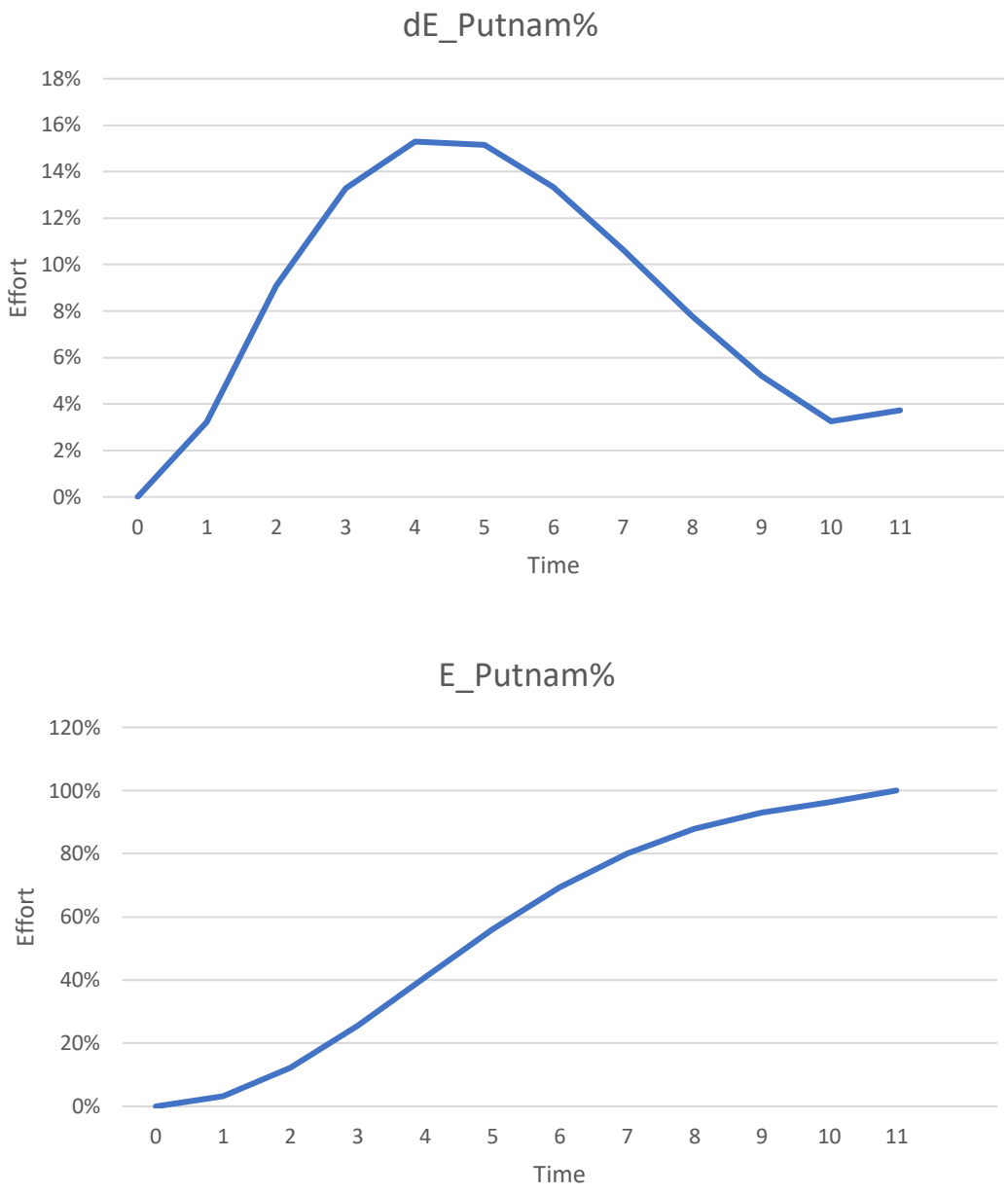

Figure 38. Cumulative and derivative curve of the Effort for the Putnam model

Starting from the Putnam model I tried to hypothesize a theory that follows the base idea of Putnam. In both theory the effort in time is function of the work load but in the Putnam model the Productivity is considered constant and, as a consequence, independent from the learning curve. As I previously said, in the planning phase is very important to take in consideration that the team is starting with different experience and learning path. For this reason I suppose the Productivity not as a constant parameter as in the Putnam model but rather, as a variable increasing with time. In particular, it is the average learning curve of the team. The other parameters as the size and the time are kept as in the Putnam model.

The model obtained as we can see in the *Figure 39* is not overlapping with the Putnam curve. The main difference is where the curve becomes steeper and where the pick is reached. The revise model reflects the hypothesis that I did regarding the learning curve. In particular, when the learning curve is lower the team needs a major effort to perform the project. The effort grown function in the revise case is always a S-curve, but a bit more moved in the left. This because it starts to exponentially steep in the first phase. Putnam in his model suppose that the effort depends on the project phases and in specifically the pick is reached in the development phase of the project itself. The revised model, exhibits a different behavior. The effort will be higher in the first phases because the team doesn't know the requirements, doesn't know well the teammates and the project can require new knowledge. For this reason, the delta of productivity it is not constant but it will decrease with time, while the skills improve.

This theory is relevant because we can allocate resources ex ante knowing the effort required from them. In the revise model several consequences can be deduced:

- It is better that the people with more experience start to work from the beginning, because it is the moment that requires more effort.
- In the first phases the resources cannot work in parallel with others. They can work just in one project per time because they will be full absorbed form it.

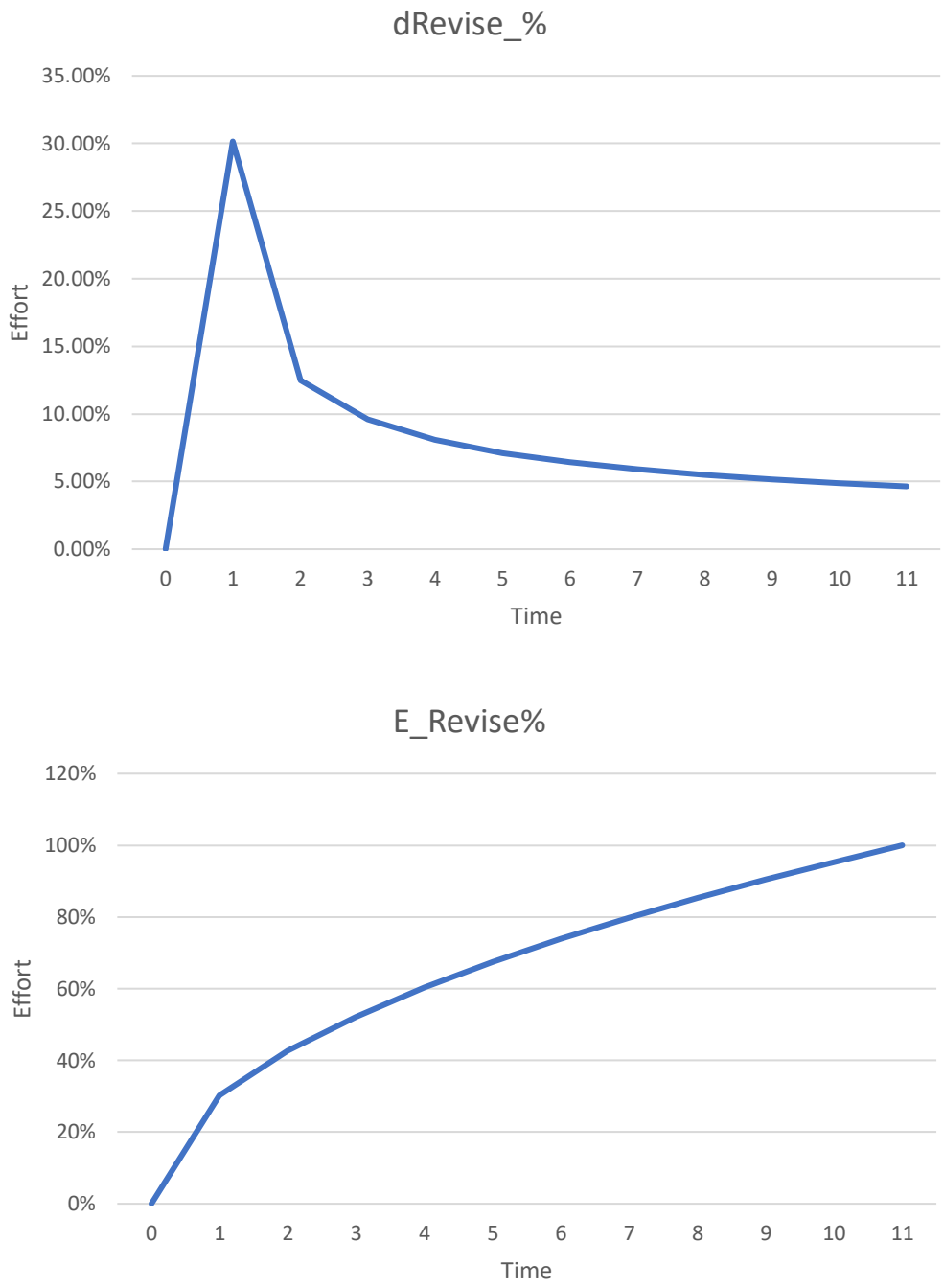

Figure 39. Cumulative and derivative curve of the Effort for the Revise model

4. Conclusion

The Earned Value predicted that it would take the team 140 days to complete all tasks. On the other hand, the Burndown charter had a more positive outlook on the progress of the project and foresaw that at the time of the deadline the team was only missing 7 points to complete and that therefore, by increasing the speed a little, it would catch up with no problems. At the end of the monitoring phase, it was decided to give greater confidence to the Burndown charter and not to crash or take other measures to the project.

The *figure 40* below shows the two forecast curves, the one obtained with the Burndown charter and the one obtained with the EVM and the current curve. Although the EM gives very detailed information regarding a delay/advance in scheduling and an over/under budget, it is not very intuitive through the data found by the model to imagine and design a trend of the project's tasks.

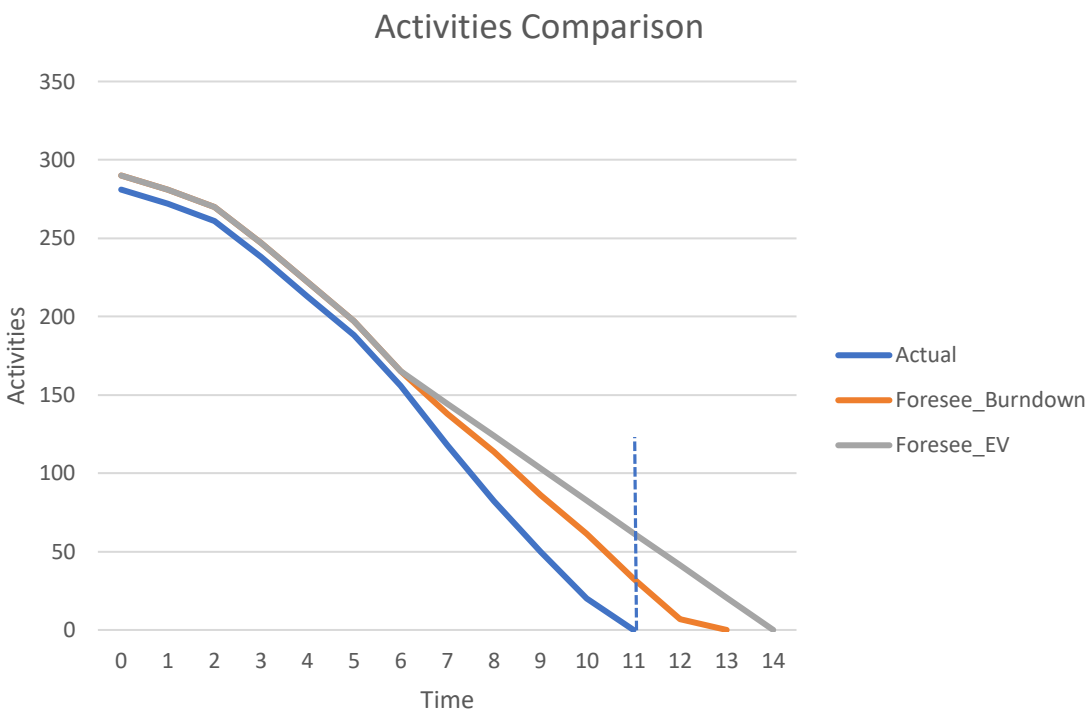

Figure 40. Comparison between the Burndown chart and the EVM to monitor Agile projects

We can see from the graph that the Burndown charter is more accurate than the original model, to not crash the project was a good idea. Although widely used and very effective, EVM seems to be less suitable in the Agile methodology.

One of the biggest problems with EVM is that it is all based on having detailed specifications upfront and not having too much change which doesn't fit with agile initiatives. In fact, the problem here is that EVM is based on a detailed and accurate "plan" at the start of the initiative while agile avoids it because it usually represents waste and throwaway work. The team complete 281 tasks compared to the 290 planned and this is not due to a delay but to the continuous change that occurs in the Agile methodology. In fact, through reviews with the client and discussions with the team, some tasks have been removed and some have been added. It is very difficult in this methodology to respect exactly what was planned.

From one side the Burndown do not provide at-a-glace project cost information and from the other the Earned Value do not provide an idea of the progress of points done. In fact, the EVM and burndown chart have a different focus. EVM is more focused on cost and schedule of the project. It can tell us the cost and budget overruns which burndown chart cannot. Burndown chart can on the other hand tell us about the remaining effort and can help in adaptation to meet the schedule.

Burndown charts are part of the agile management philosophy which promotes frequent inspection and adaptation, teamwork and collaboration.

Quentin Fleming, Joel M. Koppelman in "Earned Value Project Management, Second Edition" list 3 critical success factors for Earned Value:

- Quality of the project's baseline plan. Earned Value is compared against the baseline plan, whether the plan is accurate or not. Therefore, cost 'over runs' will occur if the project costs are under-budgeted, and scope creep will occur if the initial project scope hasn't been adequately defined.
- Actual Performance against the Approved Baseline Plan. i.e. whether the actual performance tracks to the baseline plan.
- Management's Determination to Influence the final results. Final results for a project based on earned value projections can be modified based on management's commitment to take action as soon as deviations from the plan are observed.

Agile projects fail to meet the first two critical success factors. First, the quality of the original baselined plan is very low from a completeness and scheduling perspective. We could try to use EV against our continuously evolving plans, but the figures become meaningless if you keep changing the baselined plan. Each time you do so the performance indexes and variances change making tracking and forecasting extremely problematic.

For what regards the Resource allocation model here we are comparing the Putnam model and the Revised model with the real effort – time curve in *Figure 41*.
From the beginning of the project, I have collected data regarding the hours spent in the various meetings and lessons. By subtracting this latter from the total daily working hours, I was able to find the time the team dedicated to add value to the project. The speed of the team is related to the learning curve, in fact initially the team spent much more time understanding the project itself and knowing the working method of its colleagues than working in the project. For this reason, the tasks that are initially done are less than those that are done once the project is started and that, consequently, the learning curve increases. Now we can compare the Putnam model with the revised model and understand which of the two is more correct based on which one is closest to the real effort curve.

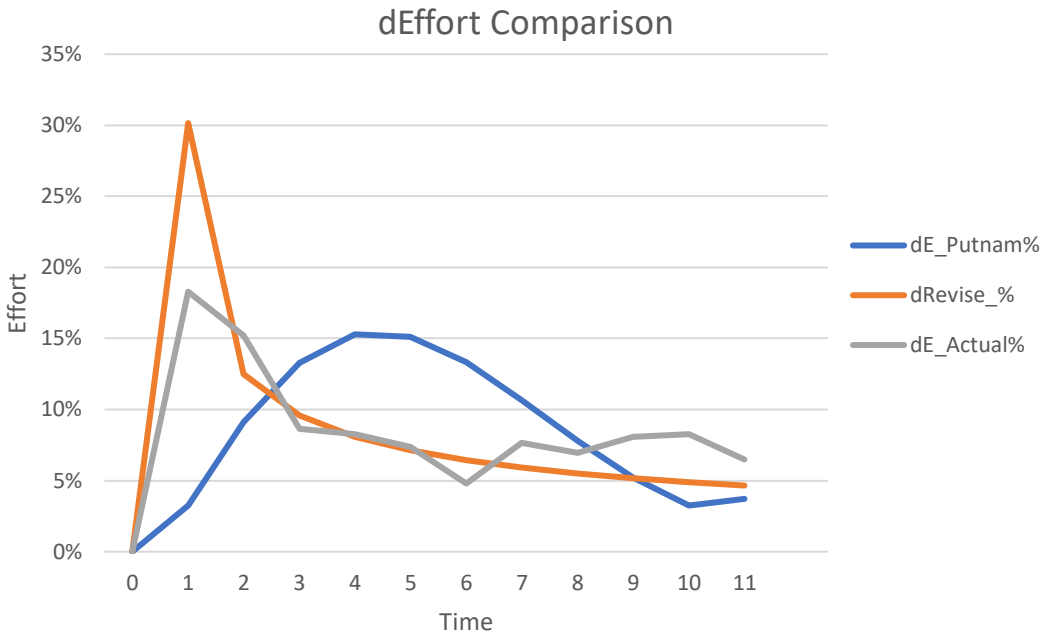

Figure 41. Effort in Putnam and Revised model compared to the real effort

From the comparison we can conclude that the Revise model is more in accordance with the actual data collected from the project and in particular seems to have an opposite behavior compared to the Putnam model. Whatever, the effort in the Revised model is higher than the Real effort spent. This is clear seeing the curves progress. In the Revised model the pick was at 30% while in the Actual model it is at 18%, anyway in both case the maximum effort is in the first sprint.

In the Revised model, the effort was considered as the actual time the team worked on the project divided by the points that were actually made in that sprint. In the first iteration, about 52 hours were spent on making only nine points, while in the second sprint, about the same number of hours were spent on making twelve points. Each member acquired knowledge as time went on, its learning curve increased and consequently the effort the team expended in carrying out the tasks decreased proportionally. Contrary to the Putnam allocation model what our theory suggests is to focus the attention on the previous phase of the life cycle of a project. So, it is better to allocate the majority of resources there as the actual data collected show.

In support of my demonstration, I would like to add that using the Putnam model to monitor int the Sprint 6 would suggest to increase resources but, being in the middle phase of the project, it would not lead to good performance of this latter for the

reasons I explain in the previous chapter. It would result in postponing the increase of the learning curve of the team members and consequently delaying the project. In the case of resource constrained we could gather them in initial phases, especially if they have experience and they can help the team to speed up the learning curve of the team.

Index of figures

Index of tables

Bibliography

- Firend Alan Rasch (2019). Methodologies in Project Management
- Project Management Institute (2021). A Guide to the Project Management Body of Knowledge (Seventh Edition).
- Project Management Institute (2017). A Guide to the project management body of knowledge (Sixth Edition).
- Dr. Scott J. Amos, PE (2004). Skills & Knowledge of Cost Engineering (Fifth Edition).
- Alberto de Marco (2018) - Project Management for Facility Construction (Second Edition)
- Joydeep Kundu, Tanmoy Kumar Bishoi, Manasija Bhattacharya and Anupam Chowdhury (2015). Project Management Software – an overview. (First Edition)
- Project Management Institute (2017). Agile Practice Guide (First Edition).
- Roger D.H. Warburton (1983). Managing and Predicting the Costs of Real-Time Software.
- Lawrence H. Putnan (1978). A General Empirical Solution to the Macro Software Sizing and Estimating Problem.
- Famuwagun, Olajide Samuel (2020). Project Management Methodologies and Bodies of Knowledge in Contemporary Global Projects.
- Izak Wilhelmus van der Merwe (2017) - How relevant are waterfall project management methodologies in today's modern project environment?
- Jordan Barlow (2011) - Overview and Guidance on Agile Development in Large Organizations.
- Ning Jingfeng, Jiang Yan and Yu Honglei (2014) - Research and application of estimation method for software cost estimation based on Putnam model.

Sitography

https://ensembleconsultinggroup.com/wp-content/uploads/2018/07/CCPM-Executive-Guide.pdf

file:///C:/Users/ealfieri/Downloads/Agile_Processes_and_Methodologies_A_Conceptual_Stu.pdf

https://www.certwise.com/wp-content/uploads/2017/03/CW_PMP_Reading-Sample.pdf

https://cs-633-team-8.github.io/files/Team-8-Content-Standards.pdf

https://scrum-league.org/tribune/autres-notions-scrum/le-burndown-chart/

https://sites.google.com/site/softwareestimation/-slim

https://opentextbc.ca/projectmanagement/chapter/chapter-3-the-project-life-cycle-phases-project-management/

https://www.certwise.com/wp-content/uploads/2017/03/CW_PMP_Reading-Sample.pdf

https://www.studypool.com/documents/12114258/a-project-management-primer

https://www.manage.gov.in/studymaterial/PM.pdf

https://pdf.sciencedirectassets.com/277811/1-s2.0-S1877042814X00133/1-s2.0-

https://www.projectsmart.co.uk/agile-project-management/the-blending-of-traditional-and-agile-project-management.php

https://activecollab.com/blog/project-management/critical-path-method-cpm

https://plaky.com/learn/project-management/critical-path-method/

https://asana.com/resources/critical-chain-project-management

https://www.appvizer.com/magazine/operations/project-management/critical-chain-vs-critical-path

https://www.lucidchart.com/blog/critical-chain-project-management

https://www.geniuserp.com/blog/what-you-need-to-know-about-critical-chain-project-management

https://www.projectcubicle.com/pert-method-definition-examples/

https://www.linkedin.com/pulse/what-pert-how-can-we-use-dave-fourie-pmp-prince2-/

https://keydifferences.com/difference-between-pert-and-cpm.html

https://www.teamgantt.com/waterfall-agile-guide/waterfall-methodology
https://assets.website-files.com/5a690960b80baa0001e05b0f/5beb3841fbcbc1a653052310_Waterfall%20Project%20Managment.pdf
https://www.taskade.com/blog/waterfall-project-management-guide/
https://www.ijraset.com/best-journal/project-construction-through-agile-method-and-analyzing-its-benefit-and-drawback-with-other-existing-methods
https://www.lucidchart.com/blog/waterfall-project-management-methodology
https://hochsolutions.com/2019/03/15/what-are-the-4ps-of-project-management/
https://www.productboard.com/glossary/agile-values/
https://www.atlassian.com/agile/kanban
https://scrumguides.org/scrum-guide.html
https://www.scrum.org/forum/scrum-forum/
https://airfocus.com/product-learn/scrum-framework/
https://www.scruminc.com/definition-of-done/
https://cs-633-team-8.github.io/files/Team-8-Content-Standards.pdf
https://premieragile.com/burn-down-chart-vs-burn-up-chart/
https://www.pmi.org/learning/library/earned-value-management-understand-agile-6567