Software Engineering

LAB Manual

Dr. M. Mahmudul Hasan

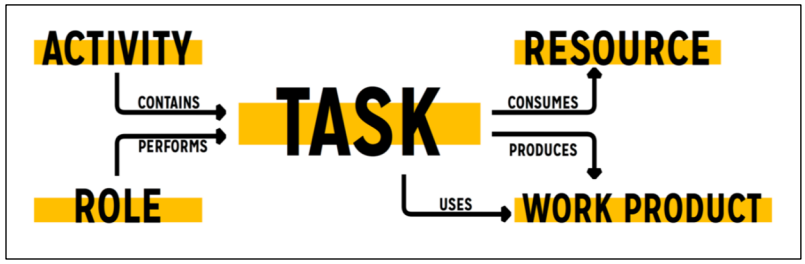
Table of Contents

Chapter 1: Software Engineering Process

1.1 Software Process Lifecycle

A key factor behind every successful software development project is to establish a process for development. Process organizes the development of a software product into multiple distinct phases or stages, for example, Specification, Design and Implementation, Verification and Validation phases. A process can be the entire life cycle of a software product or can be a sub-process within a life cycle process. Each phase is composed of associated activities. An activity is a group of related tasks (low-level atomic actions). Tasks are the small, manageable units of work for the project. They are the building blocks for completing an activity, finishing a phase, and ultimately completing a process [1].

Every task relates to roles, work products, and resources. A role is job-related activity assigned to a person (i.e., a role performs a task). Each role on a team describes specific skills that are needed to accomplish any given responsibility or task. For example, a programmer may write code for a certain feature. And a tester will be assigned to test that feature, and this role may be filled by a Junior Developer. A work product is an output produced by completing a specific task. Work products include the final product and they also include other intermediate outputs such as design documentation, requirements documents, source code, test cases, and project management artifacts such as status reports and sign-off forms. Since work products are outputs of tasks, we can say that a task produces a work product. Not only does a task produce work products as output, it also uses work products from another task as input. Completing any given task requires resources to advance or fund it, such as time, people, technology, knowledge, and money, equipment, etc. [1].



Practices are strategies used to execute processes smoothly. Practices contribute to the effectiveness of a development process. For instance, a manager can follow proven management practices for: scheduling and organizing tasks, minimizing resource waste, and tracking completed work. Additionally, estimating tasks, holding daily meetings, improving communication, and doing periodic reviews of the project are all practices that can help a process run well. A developer can follow established development practices for: preparing effective tests, maintaining simplicity, and conducting technical reviews. This ensures a quality product with low

defects. Practices are often gathered into a methodology i.e. a methodology is a set of practices. Methodologies based on the philosophy outlined in the Agile Manifesto are known as Agile methodologies. Scrum is an example of an Agile methodology [1].

Adopting good practices reduces wasted resources. Effective planning and communication prevent duplicating work or producing a feature that is not needed. Good practices then help to keep the project on schedule and on budget. Processes and practices are flexible and customizable. Combine various aspects of processes and practices to create something customized to your development needs. Finally, be open to making adjustments that refine the processes and practices to best fit your project and product [1].

Creating software is more like authoring a cookbook than following a specific recipe. There is no prescribed formula to writing a cookbook; recipes must be collected, organized, and tested. Recipe testing may lead to further refinements, experiments, and improvisations. Rejection of some recipes may require backtracking to the collection phase. And, if successful, a cookbook may have a long-life cycle involving multiple editions in which corrections are made, and new recipes are added. Similarly, software development requires continuous evaluation and decision making while moving through each phase of a given project. This continues well past the initial development project. A software product, just as with a cookbook, can have a long-life cycle involving multiple versions with improved features and resolved defects [1].

EXERCISE - 1: Problem Analysis

Objective: Develop a Project Proposal

Tools/ Device: Using Mind Mapping tools can be very useful to express your ideas and discussion with other group members. Here are some useful online mind mapping tools

1. Coggle: https://coggle.it
2. Mindmester: https://www.mindmeister.com
3. Lucidchart (a business process design modeling tool): https://www.lucidchart.com

Procedure:

1. **Self-Study:** open an appropriate software engineering guide and study the software development life cycle and related topics. Study the needs of the software engineering.

2. **Formulating Team:** A team is up to four students. In addition to each team, other people are allowed to help. A team can have a mentor, who advises on the project. A team can also have associates, who must be eligible students that can help the team with things, with business model planning, marketing, and so on. These people are not members of the team as such but it's good to have help sometimes.

3. **Identify Problem Domain:** Each team is required to come up with an idea of software development (either Problem domain: *Category A* or *Category B*). It can be either purely software or a combination of software and hardware. You must choose at least one of the options for *technology* (Web, Desktop, Mobile Phone/Handheld Devices), but if any team is optimistic enough: can choose all of the available options.

   Category A: Find a real-life problem, even in your own life or community, and then work to solve it. Build a project that could change lives. The next big thing could come from you. Facebook and Twitter started as student projects. Your ideas could be next. Come up with an innovative application idea and proceed accordingly.

   Category B: There might be dozens of software to solve our real-life problems and provide benefit to business. Come up with an idea that will extend the current version of those existing software.

4. **Prepare a Project Proposal:** write the background description that helps putting your project into the right context of a problem domain and gives everyone involved a common view of the project. What is the root cause of this problem? why is this problem is so important to consider? What is your project objective? What solution are you going to provide to solve the above-mentioned problem. Who are the target group of users of your solution? And how they will be benefited by your proposed solution to the problem? What are the basic functionalities of your proposed solution?

Evaluation: Your Project Proposal must address the following questions:

1. Does the team demonstrate a thorough understanding of the need, problem, or opportunity, including evidence of research into the need, problem, or opportunity?
2. Does the project have a clear objective including relevant benefits and target market or audience of the product?
3. What are the solutions you are going to propose to deal with the problem? why is this solution is particularly appropriate to solve the problem? Is the solution feasible to the meet the business objective?
4. Is the project's purpose and basic functionality easily understood?

1.2 Software Process Models

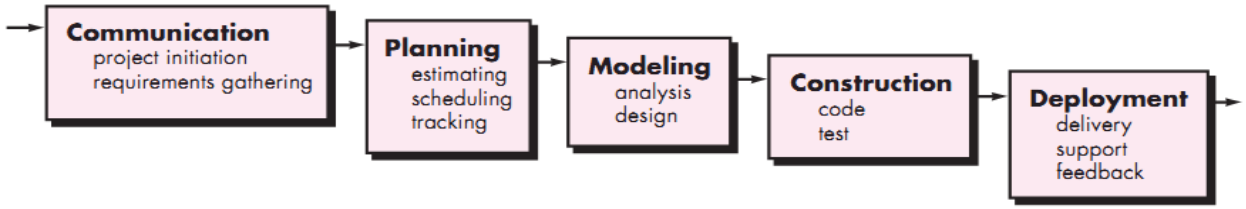
Every process model will have phases; however, there are major differences between models in terms of what activities constitute their phases and the sequencing of the phases. Three types of process models are [1]:

- **Linear process models** – phases that happen sequentially, one after another
- **Iterative process models** – phases that are repeated in cycles
- **Parallel process models** – activities that occur concurrently

Linear process models (The Waterfall model, the V model, and the Sawtooth model) follow a series of phases, one by one, with no return to prior phases. The product is essentially specified, designed, developed, and released in that order, with no opportunity to revisit earlier phases. Linear processes are best suited to projects where very little feedback or refinement is expected. Successful projects following a linear process model are well understood from the start, with little to no room for change [1].

1.2.1 Waterfall Model

The Waterfall model is a linear process model, in which each phase produces an approved work product that is then used by the next phase. The work product flows into the next phase, and this continues until the end of the process, like a waterfall. Based on the waterfall process shown, at the end of the requirements phase, there is an output work product: a requirements document. This document is formally approved and then fed into the next phase: design. This passing of work continues from phase to phase until the software product is the final output [1-2].



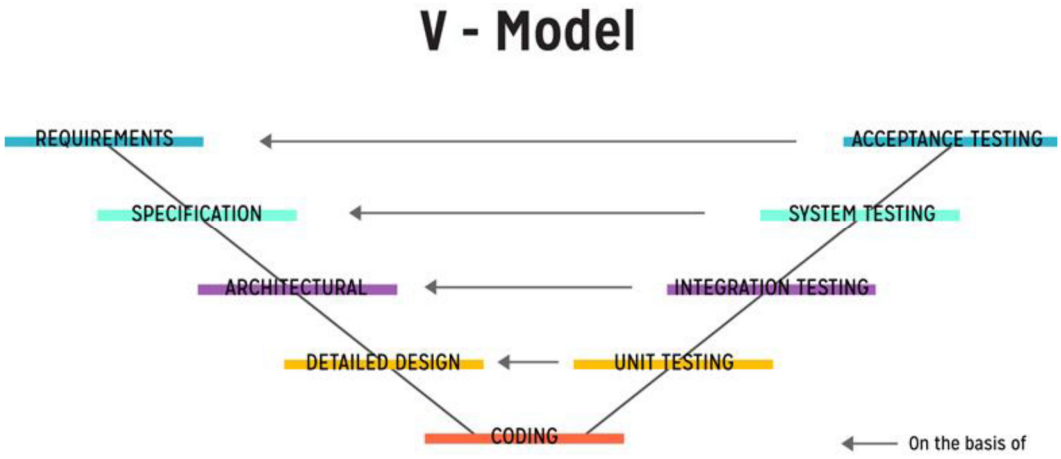
Benefits and Limitation of Waterfall Model

It may be primitive, but the Waterfall model does have benefits - even in today's development projects: This process is easy to understand; It clearly defines deliverables and milestones; It emphasizes the importance of analysis before design, and design before implementation; and it can be adopted within teams using more sophisticated processes, for well-specified parts of the product that can be outsourced [1-2].

The Waterfall model faces one major flaw: It is not very adaptable to change. That is, it is not amenable to Agile principles. Modern software development is often very dynamic, flexibility with respect to changing needs and emerging technologies is valued. The Waterfall model focuses on knowing all the requirements up front. It is simply not designed to address mid-stream changes easily or without great cost, because it would require revisiting earlier phases. In addition, testing occurs late in the process, and defects found late tend to be more difficult and costly to fix. Under the Waterfall model, the client does not see the product until close to the end of development, and so the developed product may not match what the client had envisioned [1-2].

1.2.2 V-Model

The V-model is another linear process model. It was created in response to what was identified as a problem of the Waterfall model. It adds a more complete focus on testing the product. The distinguishing feature of the V-model are the two branches that turn a linear structure into a "V" shape. Like the Waterfall model, the V-model begins with analysis and the identification of requirements, which feeds product information into the design and implementation phases. The analysis and design phases comprise the left branch of the V. The right branch represents integration and testing activities. As testing work proceeds up the right branch, some level of the product (on the right side) is actively verified against the corresponding design or requirements (on the left side) [1-2].

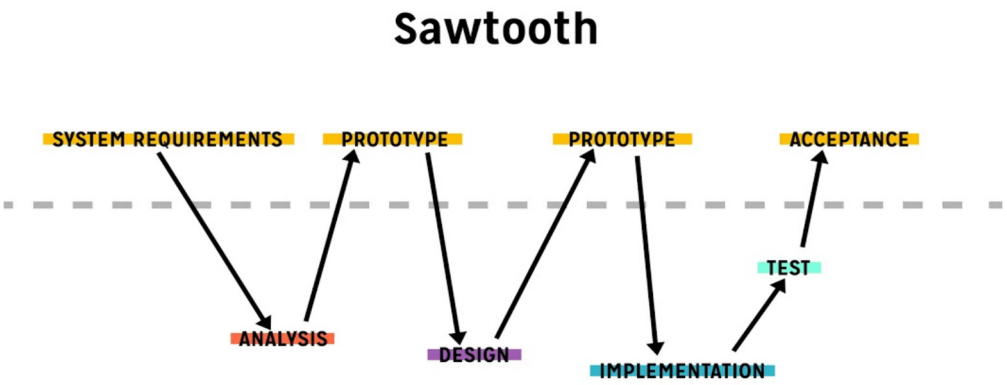


Benefits and Limitation of V-Model

The V-model has the same advantages and disadvantages as the Waterfall model in that it is straightforward to understand, but it does not accommodate change well. However, allowing the development team to verify the product at multiple levels in explicit phases is an improvement over the Waterfall model. In both the Waterfall model and the V-model, the client does not see the finished product until everything is virtually complete. The next evolution in linear processes is devising a way to involve the client in the development process [1-2].

1.2.3 Sawtooth Model

The Sawtooth model is also a linear process model, but unlike the Waterfall and V models, the client is involved to review intermediate prototypes of the product during the process. As the process proceeds linearly, there are phases that involve the client in the top section of the diagram and phases that involve only the development team in the bottom section, which creates a jagged "sawtooth" pattern [1].



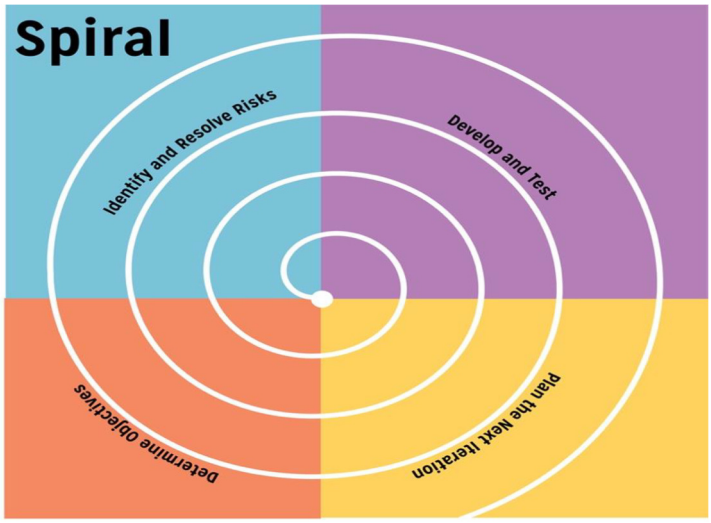
Having the client involved earlier in the process increases the likelihood that the product will meet the client's needs; this is a definite advantage of the Sawtooth model compared the Waterfall model and the V-model. However, like them, the Sawtooth model is still a linear process, so there is a limit to incorporating anything more than incremental changes [1].

Iterative Models

Iterative process models differ from linear process models in that they are designed for repeating stages of the process. That is, they are iterative or cyclical in structure. The advantage of iterative processes is the ability to loop and revisit previous phases (and their activities) as part of the process. Each "loop back" is an iteration, hence the name "iterative process." Iterations allow for client feedback to be incorporated within the process as being the norm, not the exception. Iterative process models are readily amenable to Agile practices, yet they also embody sequential portions reminiscent of linear process models [1-2].

1.2.4 Spiral Model

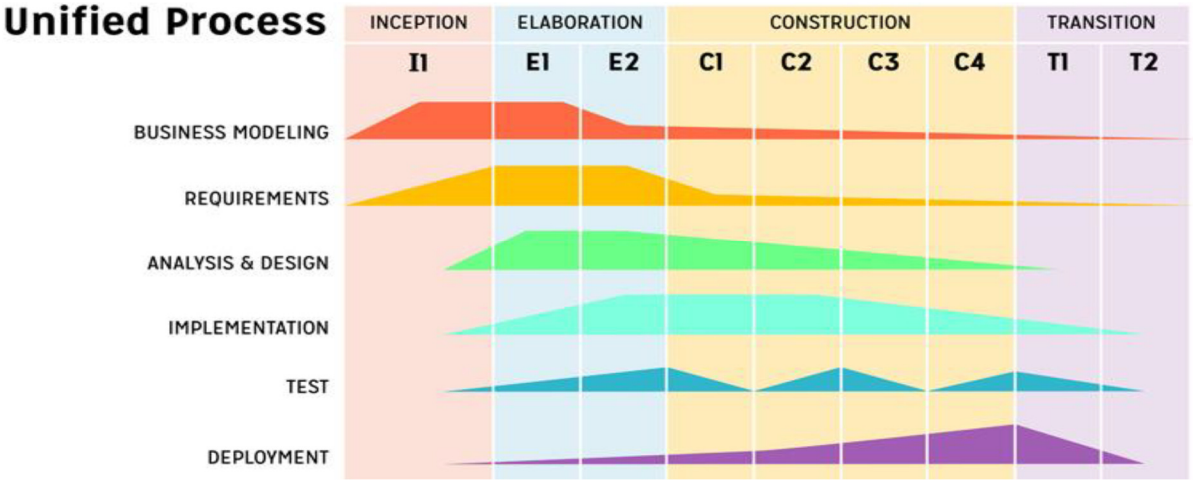
The Spiral model was introduced by Barry Boehm in 1986. The model outlined a basic process in which development teams could design and successfully implement a software system by revisiting phases of the process after they had been previously completed. A simplified version of the Spiral model has four phases, which have associated goals: determine objectives, identify and resolve risks, develop and test, and plan the next iteration. Taken in order, the four phases represent one full iteration. Each subsequent iteration has the sequence of the four phases revisited, and each iteration results in a product prototype. This allows the development team to review their product with the client to gather feedback and improve the product. Early iterations lead to product ideas and concepts, while later iterations lead to working software prototypes. The Spiral model is commonly charted with the four phases appearing as quadrants, and an outward growing spiral to indicate progression through the phases [1-2].



Estimating work can be more difficult, depending on the duration of the iteration cycle in the Spiral model. The longer the iteration cycle, the further into the future one needs to plan and estimate for; lengthy iterations can introduce more uncertainty in estimates. It is easier to estimate the effort on small things and plan for two weeks ahead than to estimate the effort needed on many big things for several weeks ahead. Also, the Spiral model requires much analytical expertise to assess risks. These extensive risk-assessment tasks can consume a great deal of resources to be done properly. Not all organizations will have the years of experience, data, and technical expertise available for estimation and risk assessment [1-2].

1.2.5 Unified Process Model

The Unified Process model is an iterative and also parallel model of software development. Parallel processes use a similar style of iteration—products are built in iterations. However, parallel processes allow for more concurrency of activities and tasks. Its basic structure has sequential phases within a repeatable cycle. Within most of the Unified Process model's phases, work happens in small iterations until the phase is deemed complete. Usually, phases are deemed complete when a milestone, a specific and identifiable point in a project, is reached [1].



While the general structure of the Unified Process is iterative, the model allows for tasks done in one phase to also occur in another. So, a requirements task or activity can happen throughout the phases instead of in just one. This also means that, for example, a requirements task, an architecture design task, and a test development task can happen in parallel with the same phase. To contrast this, in the Waterfall model, these tasks would be organized into specific, separate phases, with no parallelism for those tasks [1-2].

Inception phase

This first phase is meant to be short time to establish a strong enough business case and financial reason to continue on to the next phases and make the product. To do this, the inception phase typically calls for the creation of basic use cases, which outline the main user interactions with the product. Also define the project's scope and potential project risks. The inception phase is the only phase in Unified Process model that does not happen in iterations. If the inception phase is long, this might suggest wasted time over analyzing the requirements. The completion of the inception phase is marked by a lifecycle objective milestone. The work product for achieving this milestone is a reasonable description of product viability and the readiness to move on to the next phase of the process [1-2].

Elaboration phase

The Unified Process focuses on the importance of refining the product's architecture over time. Architecture is a set of designs upon which the software product is built. So, the goal of the elaboration phase is to create design models and prototypes of the product as well as to address risks. This phase is also the first of the phases to incorporate small iterations within the phase. Besides defining the system architecture, developers refine the requirements conceived earlier in the inception phase. They also develop key requirements and architecture documentation, such as use case diagrams, and high-level class diagrams. This phase gives the foundation on which actual development will be built. Building a prototype will likely require several iterations before the requirements and architecture models are deemed complete enough to move on. At the end of the elaboration phase, developers deliver a plan for development in the next phase.

This plan basically builds on what was developed during the inception phase; it integrates everything learned during the elaboration phase so that construction can happen effectively [1-2].

Construction phase

The construction phase has also iterations and focuses on building upon the previous work so far. This is where the software product begins to take shape. Since the Unified Process model is a parallel process, when the construction phase begins, elaboration phase work will still continue. The only difference is that the emphasis on the work may change. Testing and programming activities may have been important in the elaboration phase (for technical feasibility studies or to set up the development environment), but they become even more important in the construction phase. Similarly, assessing risks is important in the inception phase, but it's less important in the construction phase. In the construction phase, full use cases are developed to drive product development. These improvements upon the basic versions developed in the inception phase consist of a set of possible sequential interactions between users and systems. They help to identify, clarify, and organize functionalities of a product. These use cases are more robust than the ones initiated in the inception phase and offer more specific insights into how the product should support end-user tasks. The product is built iteratively throughout the construction phase until it is ready to be released. At that point, the development team begins transitioning the product to the client and/or the end users [1-2].

Transition phase

During this phase, the development team receives feedback from users. The transition phase reveals how well the product design measures up against users' needs. By gathering user feedback, the development team can make improvements to the product, like bug fixes and developing future updates to be released. Upon completing the transition phase, it is possible to cycle back through the phases of the Unified Process again. For example, there may be a need to create further releases of the product or to incorporate user feedback as a means of influencing the plans for later development [1-2].

1.2.6 Prototypes

Developing software products through a series of intermediate prototypes was a theme in both the Spiral and Unified Process models. There are five types of prototypes.

Illustrative prototype

This is the most basic of all prototypes. They could be drawings on a napkin, a set of slideshow slides, or even a couple index cards with components drawn out. The essence of an illustrative prototype is to get a basic idea down. Illustrative prototypes can give the development team a common idea of which to base their work, or to help them get a system's "look and feel" right without investing much time or money into developing a product. Use illustrative prototypes to

weed out problematic ideas or as a guide for later development. Illustrative prototyping can involve storyboarding, or wireframes. They are all meant to show how a system will work or illustrate a concept using only diagrams and pictures [1-2].

Exploratory Prototype

Exploratory prototyping puts the focus on more than just the product's look and feel. By building working code, the development team can explore what is feasible—fully expecting to throw the work out after learning from the prototype. With more time, a more comprehensive look at what the product will look like can emerge, as well as an understanding of the effort needed to build that product. Exploratory prototyping is expensive in terms of consuming resources, but it is used because it is better and less costly than finding out later in the process that a product solution just cannot work. The usual motivation behind exploratory prototyping is that the product developers want to study the feasibility of a product idea. Unlike illustrative prototyping, which is only concerned with what the product looks like, exploratory prototyping is about how realizable it is to develop the product or how useful the product may be, before committing further effort to the idea [1-2].

Throwaway Prototype

The first version of almost any product is bound to have various problems. So, why not just build a second, better, version from scratch and toss away the first? These first versions of products are known as throwaway prototypes – you build a functioning product that will get tossed out. Throwaway prototypes allow the opportunity to learn from past mistakes. There could be many useful lessons and problems revealed in the first version that can be avoided in a second version. This affords the chance to build the software product on a more robust second-generation architecture, rather than a first-generation system with patches and fixes [1-2].

Incremental Prototype

When a product is built and released in increments, it is an incremental prototype. Incremental prototyping works in stages, based on a triage system. "Triaging" means assessing each of the system's components and assigning a priority. Based on that priority, a product's components are built iteratively in increments from "most important" to "least important." In this way, incremental prototypes make use of the process philosophies behind iterative processes models. Priorities for a software product's features are based on three categories: "must do," "should do," and what "could do." Core features are assigned the highest priority—must do. All the non-critical supporting features are "should do" items. Remaining extraneous features are the lowest priority—could do. Based on these priorities, the initial iterations of incremental development focus on delivering "must do" priorities. The resulting software product with the core features could be released as a complete first-generation incremental prototype. As resources permit, features under the "should do" priority can be developed for a similar second-generation incremental release, followed by a third release with features from the "could do" category. The

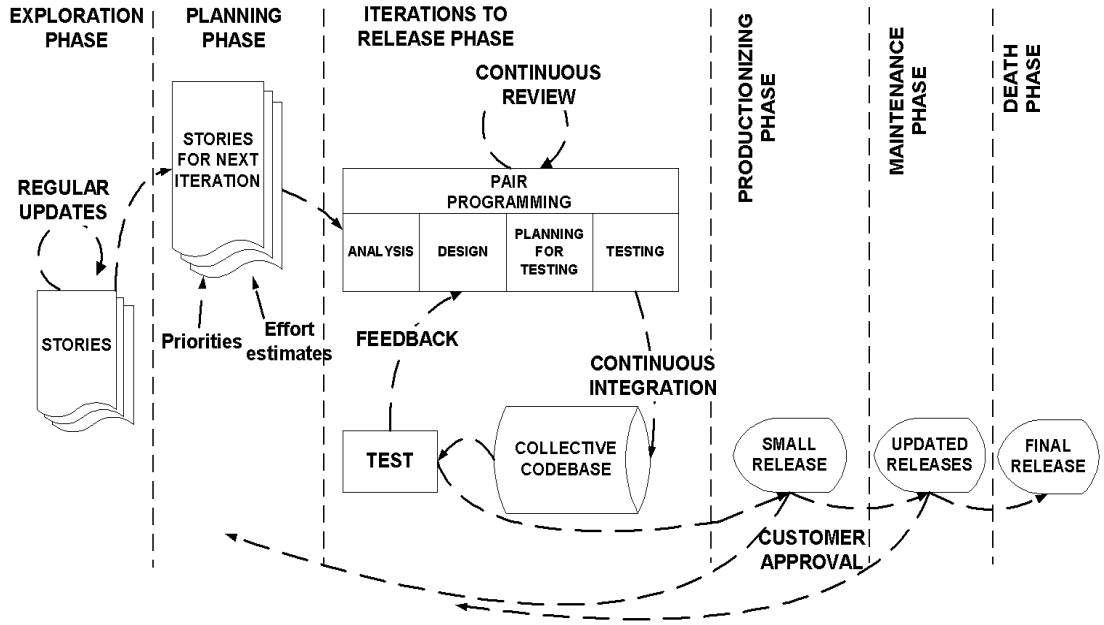
end result of these iterations is a series of incremental prototypes from essential basic features to fully featured [1-2].

Evolutionary Prototype

The final type of prototype is the evolutionary prototype. This prototype approach is very similar to incremental prototypes, but the difference is found in the first-generation prototype. In evolutionary prototyping, the first-generation prototype has all the features of the product, even though some features may need to "evolve" or be refined. A comparable first-generation incremental prototype has only a basic set of core features. Both incremental and evolutionary prototyping are ways to make working software that can be shown at regular intervals to gain further feedback. In practice, both approaches can be blended. A major benefit is the morale boost for the development team as they see the working product early and are able to contribute improvements over time [1-2].

1.2.7 Extreme Programming (XP)

Extreme Programming (XP) is an Agile methodology consisting of effective development practices to achieve client satisfaction. To apply XP, there are 12 practices to follow in extreme level.



Practice 1: The Planning

The client and development team work together in planning the product. This entails a larger planning session at project initiation and smaller sessions at each iteration. The larger session

determines the product's required features, their priorities, and when they should be released over time. A smaller session focuses on the features to be completed in an iteration and determines the development tasks needed [1-2].

Practice 2: Small Releases

Releases must occur as frequent as possible to gain plenty of feedback, which is best achieved if releases are small in terms of required new functionality. The required features should be prioritized to deliver value to the client early. The client and development team need to strike the right balance between what the client wants completed first and what can be developed early. Smaller releases also allow estimates to be more certain; it is easier to look a week or two ahead, rather than months. So, in the extreme, keep the iterations very short [1-2].

Practice 3: System Metaphor

A system metaphor makes it easier to explain the product to someone else; it can be an analogy to describe the product to someone who is not technical. For example, the desktop metaphor in graphical user interfaces helps to explain computer interactions in terms of something more familiar in the real world. Another example is the shopping cart metaphor in online shopping, which builds upon a concept from real-world shopping. A metaphor can also describe the product implementation to someone more technical. For example, the pipes and filters metaphor for software architecture helps to explain how a product implementation is structured around connecting and driving computing elements with information flows [1-2].

Practice 4: Simple Design

Focus on the simplest design that works to satisfy the client's needs. Requirements change, so it would be wasteful in making elaborate designs for something that will change. Design what is needed to make the required features work. Do not over-engineer the design for a future that may not come. So, in the extreme, make the simplest thing that works [1-2].

Practice 5: Continuous Testing

In XP, tests are prepared for a required feature or functionality before its corresponding source code is written. This focuses efforts first on understanding what is required, making the user or programmatic interface to it simple, and preparing suitable tests to verify that the required behavior has been achieved. If the test was difficult to write, it is a sign to rework the interface or original requirement. The implementation of the required feature or functionality comes afterwards. This practice is referred to as Test-Driven Development or TDD. Automating the tests will allow them to be executed continuously. So, in the extreme, write tests first and run them all the time. There are two main types of tests involved: acceptance tests and unit tests. An *acceptance test* is for the client to verify whether a product feature or requirement works as specified and thus acceptable. An acceptance test typically involves something an end user does, touching a large part of the product. Such a test can be automated or can be a set of instructions

and expected results that a human follow. A *unit test* is written and run by developers to verify whether low-level functionality works correctly. A unit test is typically specific, like testing an algorithm or data structure in some module of the software implementation. Consider a social media application with a required feature that an end user can post a message. An acceptance test for this feature would involve actions that end users do, like initiating a post, entering a message, and submitting the post. The test would check that the actions basically work for some trial data. Underlying this feature, and the product in general, is low-level functionality to store posts. Unit tests would more thoroughly check that this functionality works, such as dealing with international characters or very long texts [1-2].

Practice 6: Refactoring

Refactoring is a practice to restructure the internal design of the code without changing its behavior. The aim is to improve the design in small steps, as needed, to allow new product requirements to be added more easily, thus adapting to change. For example, refactoring could gradually reorganize the code to allow easier extension. A large, unwieldy module can be decomposed into smaller, more cohesive, more understandable pieces. Unnecessary code can be removed. As refactoring proceeds, the unit tests are run repeatedly to ensure that the behavior of the code stays the same. If refactoring is deferred or not done, then changes become more and more difficult. The undone work is like incurring an obligation, known as technical debt, to be "paid" in the future. A small amount may be fine to keep advancing the product, but a crisis could happen if a sufficiently large amount accumulates [1-2].

Practice 7: Pair Programming

To ensure high quality, XP employs pair programming, where two developers work side-by-side at one computer to work on a single task, like writing the code for a required feature. In regular code reviews, a developer writes some code, and after completion, another developer reviews it. Instead, in pair programming, the code is reviewed all the time by always having another pair of eyes to consider each edit. This brings together complementary skills and perspectives when solving a problem or generating alternatives. Riskier but more innovative ideas can be pursued, aided by the courage brought on by having a partner. Pairing a senior and junior developer can foster learning, where the senior one imparts strategic advice and the junior one offers a fresh perspective. In general, the pairs are not static but change dynamically. So, in the extreme, do code reviews all the time [1-2].

Practice 8: Collective Code Ownership

Although a specific pair of developers might initially implement a particular piece of the product, they do not "own" it. To encourage contributions, other team members can add to it or any other part of the product. Thus, the produced work is the result of the team, not individuals. So, project success and failure reside with the team, not on specific developers [1-2].

Practice 9: Continuous Integration and Daily Build

In XP, developers combine their code often to catch integration issues early. This should be at least once daily, but it can be much more frequent. With tests written first, the tests can also be run frequently to highlight pending work or unforeseen issues. For the Microsoft Daily Build, each "iteration" of the construction phase is laid out in a day, hence the "daily build." The point of the Microsoft Daily Build is to ensure that the programmers are all in sync at the beginning of each build activity. By following a process that makes the developers integrate their code into the larger system at the end of every day, incompatibilities are detected sooner rather than later. To control this, Microsoft uses a system of continuous integration. When a developer writes a piece of code and wants to share that code remotely with anyone else on the team, the code must first be put through an automatic testing process, which ensures that it will work with the project as a whole. All the developers can easily see how their work fits into the product as a whole, and how their work affects other members of the team. Continuous integration not only keeps developer morale up, but it also increases the quality of the product. The daily build does this by giving developers the ability to catch errors quickly before they become a real problem. A successful build allows overnight testing to proceed, with test results reported in the morning. At Microsoft, as an incentive for successful daily builds, a developer whose code breaks the build must monitor the build process until the next developer to break the build comes along [1-2].

Practice 10: 40-Hour Work Week

XP aims to be programmer friendly. It respects the developer's balance of work and life. At crunch time, up to one week of overtime is allowed, but multiple weeks of overtime would be a sign of poor management or estimation [1-2].

Practice 11: On-Site Customer

The client is situated near the development team throughout the project to clarify and answer questions [1-2].

Practice 12: Coding Standards

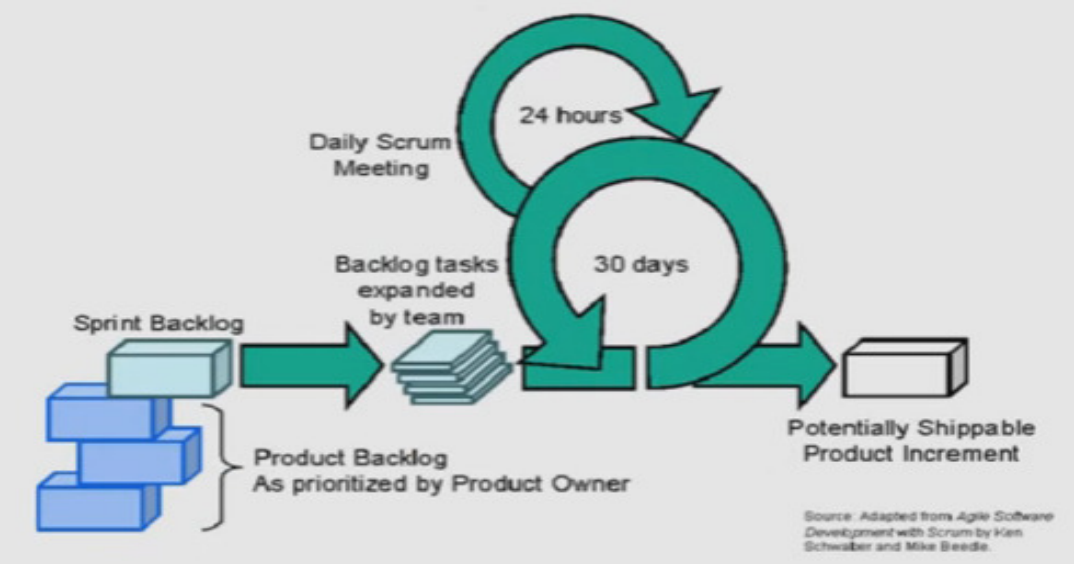
All the developers agree to and follow a coding standard that specifies conventions on code style, formatting, and usage. This makes the code easier to read, and it encourages the practice of *collective ownership* [1-2].

Limitation of XP Practices

One limitation is that XP is intended and suited for small development teams, for example, no more than ten people. As well, having a client available on-site with the development team may not be possible to arrange. Also, XP does not offer practices to define the architectural design of the software. Instead, this design emerges in response to changing requirements and refactoring, using the system metaphor as a guide. This approach would produce working software early, but it might involve more rework than if some effort were spent to plan the architecture [1-2].

1.2.8 SCRUM

Scrum is an Agile methodology consisting of lightweight management practices that have relatively little overhead. Its practices are simple to understand but very difficult to master in their entirety. Scrum uses an approach that is both iterative and incremental. There are frequent assessments of the project, which enhances predictability and mitigates risk [1-2].



Scrum is based on three pillars: Transparency, Inspection, and Adaptation . With transparency, everyone can see every part of the project, from inside the team and outside the team. Scrum encourages frequent inspection of work products and progress to detect undesirable deviations from expectations. The inspections, however, do not happen so frequently to impede development. Ken Schwaber, a Scrum trainer, says “Scrum is like a mother-in-law, it points out all your flaws.” Developers sometimes are hesitant to adopt scrum because it points out everything they are NOT doing right. When a deviation is detected, the team must adapt and adjust to correct the problem. Important is agreeing on standards for the project such common terminology and definitions such as the meaning of “done” [1-2].

SCRUM Process and Practices

The project timeline consists of a sequence of consecutive sprints. A sprint is an iteration, whereby an increment of working software is delivered to the client at the end. Each sprint is “time-boxed” to a consistent and fixed duration. That fixed duration is chosen at the start of the project and is typically one or two weeks. To enable the three pillars, Scrum outlines four required techniques or events that happen within a sprint [1-2]:

1. **Sprint planning** (to set expectations for the sprint)

2. **Daily scrums** (to ensure work is aligned with these expectations)
3. **Sprint review** (to gain feedback on completed work)
4. **Sprint retrospective** (to identify what to do better in the future)

Spring planning occurs at the beginning of a sprint to determine what will be completed in that sprint. This planning sets a sprint goal, a high-level view of what will be done. The plan also identifies the particular requirements to complete and the associated developer tasks. To keep the team focused on the plan, *daily scrums* are brief meetings held early each day, when each team member outlines what they did previously, what they will work on, and what may be blocking their progress. To encourage brevity and avoid long meetings, the daily scrums are conducted with the team standing up. So, the daily scrum is also known as the daily stand-up meeting. A *sprint review* happens at the end of a sprint to show the working software to the client and gain feedback. A sprint retrospective gives an opportunity to reveal lessons learned that can be addressed to improve future work. Once sprint planning is done, work proceeds according to the sprint goal, determined requirements, and identified tasks. To avoid disturbing work in progress during the sprint, major new requirements changes are noted for later [1-2].

SCRUM Roles

Scrum defines some key roles for a scrum team. Depending on the situation, there are various mappings for a client and software product manager to these roles. In a large organization or a large product, the client, software product manager, product owner, and scrum master can be all different people. In a small start-up company, there may be significant overlap [1-2].

- **Product owner** is the one person who is responsible for the product. All product requirements for the developers must come through this role. The list of requirements for the product is gathered in a product backlog, which the product owner is responsible for. The product owner also determines the priorities of these requirements and ensures the requirements are clearly defined [1-2].

  Naturally, the client could be the product owner, who determines the requirements and accepts the completed work. However, with clients inexperienced with defining clear requirements for developers, a software product manager may represent the client and take on the product owner role instead. If there is not an actual client, as in a mass-market product, then the software product manager may represent the end users and serve in the product owner role [1-2].

- **Scrum master** makes sure the scrum team adheres to scrum practices, by assisting the product owner and the development team. For the product owner, this help includes suggesting techniques to manage the product backlog, to attain clear requirements, and to prioritize for maximum value. For the development team members, this help includes coaching the team to self-organize and remove roadblocks. The scrum master also facilitates the four events listed above within a sprint [1-2].

A team lead could be the scrum master to facilitate scrum practices by the team. In a small company, a software product manager with leadership qualities may need to take on the scrum master role. In any case of involvement, it is important for software product managers to understand the scrum roles and their responsibilities [1-2].

- **Scrum Development Team** are the developers on a scrum team, also known as the scrum development team, must be self-organizing. No one outside the development team, not even the scrum master or product owner, tells them how to turn the backlog of requirements for a sprint into developer tasks to produce an increment of working software. Scrum development teams are small, ideally between three and nine people. They are self-contained, consisting of everyone and everything needed to complete the product. Also, each member generally takes on mixed tasks, like doing both coding and testing, rather than having dedicated coders and testers. There are no special sub-teams. Everyone on the team is responsible for the team's work products. Accountability rests with the entire team [1-2].

1.2.9 Lean Software Development

The origin of Lean software development is inspired by the manufacturing industry where Lean production grew out of the "Toyota Production System." At Toyota, their approach was used effectively to reduce waste in the production process and increase the quality of their vehicles. Lean is based on seven principles which can be effective for both small and large projects [1].

Principle 1: Eliminate Waste

How can software development be wasteful? Consider the potential waste of time and effort that can arise from: unclear requirements, process bottlenecks, product defects, and unnecessary meetings. These are deficiency wastes. Waste can also be disguised as efficiency because being "busy" does not always equate with being "productive." A busy team not focused on developing core, required features can easily produce "extra" features that interfere with the core functionality of the end product. Coding can be wasteful if it is not planned to be a released. Anything that does not add value to the product is considered waste and should be identified and eliminated [1].

Principle 2: Amplify Learning

Explore all ideas sufficiently before proceeding with actions. Do not settle and focus on a single idea before fully exploring other options. This principle seeks to find the best solution from a number of alternatives. Lateral thinking generates alternative approaches, and the more alternatives that are considered, the greater the likelihood that a quality solution will emerge. The biological mechanism of natural selection is a good metaphor for amplified learning. The most successful solutions emerge from the widest variety of alternatives. This exploration will lead to building the right product. The principle also encourages running tests after each build of the product. Consider failures as opportunities to amplify learning on how to improve[1].

Principle 3: Decide as Late as Possible

Unless you actually need to make a decision *right now*, don't make the decision yet. Deciding as late as possible allows the exploration of many alternatives (amplify learning) and selection of the best solution based on the available data. Early, premature decisions sacrifice the potential for a better solution and the "right product" [1].

Principle 4: Deliver as Fast as Possible

There is a relationship between Lean principles. Eliminating waste requires thinking about what you are doing; therefore, you must amplify learning. Amplified learning requires adequate time to consider alternatives; therefore, you must decide as late as possible. Deciding as late as possible requires focused productive work; therefore, you must deliver as fast as possible. Delivering as fast as possible is chiefly concerned with evolving a working product through a series of rapid iterations. Each release can focus on core product features, so time and effort are not wasted on non-essential features. Frequent releases provide opportunities for the client to give feedback for further refinements [1].

Principle 5: Empower the Team

The first four Lean principles deal with the process of developing the right product. The remaining three principles describe *how* the process can be made efficient [1].

The best executive is one who has sense enough to pick good people to do what he wants done, and self-restraint enough to keep from meddling with them while they do it.
— Theodore Roosevelt

Lean aims to empower teams that follow its practices. It encourages managers to listen to their developers, instead of telling the developers how to do their work. An effective manager lets the developers figure out how to make the software [1].

Principle 6: Build Quality In

Aim to build a quality product. Ways to build quality software include conducting reviews of the work products, preparing and running automated tests, refactoring the code to simplify its design, providing useful documentation, and giving meaningful code comments. Developing a quality solution reduces the amount of time developers must devote to fixing defects. By proactively applying this principle to build quality into the product, the development team eliminates waste in time and effort [1].

Principle 6: See the Whole

Maintain focus on the end-user experience. A quality product is cohesive and well designed as a whole. Individual components must complement the entire user experience [1].

1.2.10 Kanban

Kanban involves a technique to organize and track project progress visually, which is widely used even outside Agile or Lean software development. *Kanban* is a Japanese word, which loosely translated means board or signboard. So, the Kanban technique uses a board, with a set of columns labeled by the stages of completion [1].

Tracking Tasks

A Kanban board can track the status of tasks completed by a team. So, for simple tasks, suitable column labels for the stages of completion would be "To Do", "Doing", and "Done." Initially, all the required tasks are written on sticky notes and placed in the "To Do" column. As work proceeds and a task to do enters the "doing state," its sticky note is pulled from the "To Do" column to the "Doing" column. When that task is done, the sticky note then moves from the "Doing" column to the "Done" column. For the required tasks, the board easily shows their states at a glance to the entire team. The team has full visibility of what needs to be done, what is being done, and what has been done. It is motivating to see the tasks march steadily across the board and useful to see when a task appears blocked. Typically, a requirement for a feature is considered done when it is coded, tested, documented, and accepted [1].

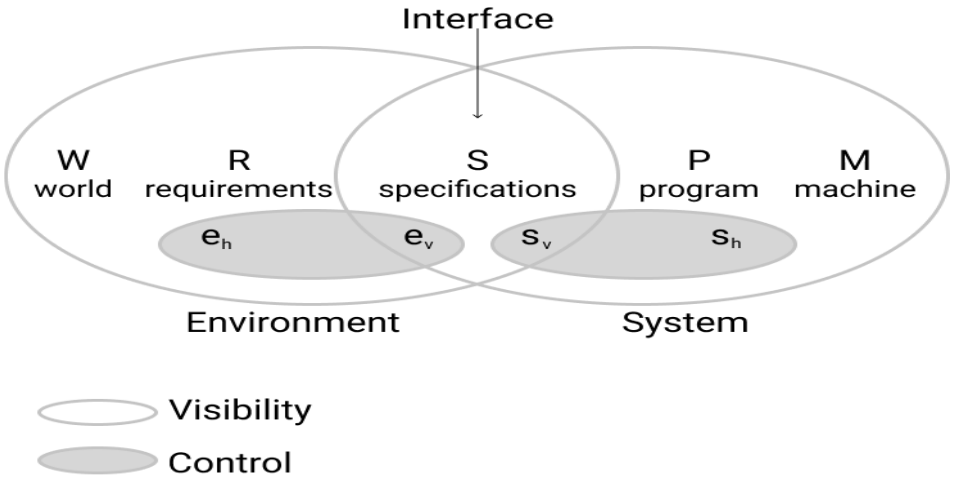
Tracking Product Requirements

A Kanban board can be used effectively with Scrum. For example, the board above can track the status of the developer tasks that are planned to be done within a sprint. As well, a different Kanban board can track the status of individual requirements on the product backlog through the stages of being considered "done" in Scrum. Here, example column labels for the stages of completion could be: "Backlog," "Analysis," "Design," "Tests," "Coding," "Review," "Release," and "Done," following the phases of some software development process. Initially, the requirements on the backlog are written on sticky notes and placed in the "Backlog" column. As work proceeds, the requirements march across the board, eventually landing in the "Done" column. The requirements will move at different rates across the board toward completion [1].

In general, a requirement stays in a certain intermediate column until the team actually has the capacity to start the next stage for it. That is, a requirement is generally "pulled" into the next column. However, one tactic is to have a special "Next" column just after "Backlog", so that a client or product owner can take a high-priority requirement off the backlog and explicitly "push" it into the development process to be worked on in a sprint. In this way, Kanban can be used to organize work as well as tracking work [1].

1.2.11 WRSPM Reference Model

WRSPM Model is a reference model of how we understand problems in the real world. It helps to understand the difference between the requirement (user's needs) and the specification (developer's understanding). Requirements are always in the problem domain. It is all about what the users want to do in order to solve some problems that they have. Our job is to take those requirements and then determine what software specifications we need in order to constrain the solution (what we are going to do with our computer, our system to solve the problem). Software specification is in the solution (computer) domain. And there are several layers of abstraction that exist in between requirements, user's goals and the software specifications (Sofia, 2020).



W — is the world assumptions, that we know are true. They have an impact on our system and on our problem domain. These things everyone takes for granted, and they are one of the most difficult parts to capture.

R — is the requirements. This is the **user's** language understanding of what the user wants from the solution. For example, users want to withdraw money. The ATM is the solution.

S — is the specification. It is the interface between how the system will meet those requirements. It is written in system language that says in plain English what the system will do. The specification is how the system meets the requirements. For example, in order to withdraw money from the ATM, you have to insert your card, insert PIN-number, etc. Those are the things user doesn't care about. The user wants to get money.

P — is the program. It is what software developments will write. The program will meet the specifications to provide the user goal for requirements. The program has all the code, underlying frameworks, etc.

M — the machine. It is the hardware specification. For example, it includes the roller for distributing money, the lock box etc.

Within these things we carry about 4 variables.

Eh — are the elements of the environment that are hidden from the system. These are parts of the environment that the user wants. For example, the card.

Ev — are the elements of the environment that are visible to the system. For example, the data generated, when you read a mag strip on the card and the entered PIN number.

Sv — are the elements of the system that are visible in the environment. For example, the button, the information on the screen.
Sh — are the elements of the system that are hidden from the environment. For example, the roller inside the machinery that the user can't actually see, making sure that the machinery gets the approval number from the bank before distributing money.

EXERCISE - 2: Selection of Process Model

Objective: Selection of appropriate process model for the software development

Tools/ Device: Protégé Ontology Development tool to check rational and consistency of your process model selection.

Procedure:

1. Study the Software Engineering process model to appropriate process model to develop your proposed system solution.
2. Present your arguments based on your analysis about why your selected method(s) is the best choice among all other methods to develop your proposed software.
3. Identify all the roles in the project management activities in software development. Describes the responsibilities of the role in the software development.
4. Your Design Specification must address the evaluation rubrics mentioned in the software engineering course outline.

Chapter 2: Software Requirements Analysis

EXERCISE - 3: Requirement Analysis

Objective: Develop a Specification Document of the Proposed Solution

Tools/ Device: SRS Template

Procedure:

1. Prepare a standard Software Requirements Specification (SRS) document that specifies the System's Functional Requirements, Non-Functional Requirements, and Project Development Constraints.
2. Your Specification document must address the following questions:

Evaluation: Your specification document must address the following questions:

1. Do the functional requirements demonstrate the functionality of the software? Do the functional requirements have been leveled accordingly? Do the functional requirements have been prioritized?
2. Do the non-functional requirements demonstrate the qualities of the software? Do the non-functional or quality requirements are quantifiable measure? Do the non-functional requirements have been prioritized?
3. Do the project requirements demonstrate the system development constraints, also prioritized, and leveled appropriately?

Example: Functional Requirements

1. **Software Login**
   Functional Requirements
   1.1 The software shall allow users to login with their given username and password.
   1.2 The login credentials (username and password) will be verified with database records.
   1.3 If the login successful the home page of the user account will be displayed.
   1.4 If the username and/or password has been inserted wrong, the random verification code will be generated and sent to the user's email address by the system to retry login.
   1.5 If the number of login attempt exceed its limit (3 times), the system shall block the user account login for one hour *[optional function]*

   Priority Level: High
   Precondition: user have valid user id and password
   Cross-references: 4.1, 7.2 (example)

Example: Non-Functional Requirements

Usability: *A trained user shall be able to submit a complete request for a chemical selected from a vendor catalog in an average of four and a maximum of six minutes.*

Example: Project Requirements

Tools: *The system developer needs selenium tools in perform testing activities in week 6*

Chapter 3: Software Design

a. Modularity

b. Coupling

Coupling is all about how tightly coupled one module is to another. One of the key components of modularity is the idea of decomposability, separating complexity. One way we can control the effects of change on a design is to enforce this separation. When the requirements are changed, and they will be, maybe halfway through our process, we do not want those changes to have massive impacts across the entirety of our system. By enforcing low coupling, what we are hoping to accomplish is that changes do not cross the boundaries of our modules. Ideally, when a requirement changes, and the changes in our code should be contained within a single module. The module tasked with completing that function that is changed. When you produce effective low coupling, changes in one module should not affect the other modules or should do so as minimally as possible. Even though the monitor is low coupling, high cohesion, we talk about the levels of coupling in terms of loose and tight coupling, those terms make more sense in isolation. The worst, strongest, highest forms of coupling are listed here.

In the TIGHT coupling there are:

(a) **Content coupling** occur when two modules rely on the same underlying information. Content coupling happens when module A directly relies on the local data members of module B rather than relying on some access or a method.

(b) **Common coupling** also occurs when two modules rely on the same underlying information. But it happens when module A and module B both rely on some global data or global variable.

(c) **External coupling** is a reliance on an externally imposed format, protocol, or interface. In some cases, this cannot be avoided, but it does represent tight coupling, which means that changes here could affect a large number of modules, which is probably not ideal. You might consider, for example, creating some abstraction to deal with the externally imposed format, allowing the various modules to maintain their own format, and delegating the format to the external but into a single entity, depending on whether or not the external format or the internal data will change more often.

In the MEDIUM coupling there are:

(a) **Control coupling** happens when a module can control the logical flow of another by passing in information on what to do or the order in which to do it, a what-to-do flag. Changing the process may then necessitate changes to any module which controlled that part of the process.

(b) **Data structure coupling** occurs when two modules rely on the same composite data structure, especially if the parts the modules rely on are distinct. Changing the data structure could adversely affect the other module, even when the parts of the data structure that were changed are not necessarily those that were relied on by that other module.

In the LOOSE coupling there are:

(a) **Data coupling** is when only parameters are shared. This includes elementary pieces of data like when you pass an integer to a function to compute the square root.

(b) **Message coupling** is then the loosest type of coupling. It is primarily achieved through state decentralization, and component communication is only accomplished either through parameters or message passing.

(c) **No coupling**, but this is usually the trivial case and is not really of that much interest to us. In any sufficiently complex design, there is going to be multiple modules. We care mostly about how tightly coupled are the modules that do communicate, and not really about the modules that do not and should not communicate anyway, we do not really care about that. The ones that do communicate, those are the ones we are going to focus on.

c. Cohesion

Cohesion is how well everything within a module fits together, how well it works towards a singular purpose. Technically, everything you do is cohesive to the idea that you are building a software. Every line of code works towards that singular goal. Technically inheritance weakens cohesion. There are various types of cohesion in different level. Generally, we can categorize cohesion into Weak, Medium, and Strong cohesion.

In the WEAKEST form of cohesion there are:

(a) **Coincidental cohesion** is effectively the idea that parts of the module are together just because they are in the same file. If you just throw all the code into one file, technically it is cohesive in that it resides in the exact same file location. It's in the same class, for example, in object-oriented programming.

(b) **Temporal cohesion** means that the code is activated at the same time. That is really the only connection. Being in a module because you are both called at the start is not a very effective way of looking at the solution.

(c) **Procedural cohesion** is similarly time-based and not very strong cohesion. Just because one comes after the other does noy really tie them together, not necessarily.

(d) **Logical cohesion** is the idea that components which perform similar functions are grouped. The idea here is that at some level the components do similar, but separate or parallel things.

In the MEDIUM form of cohesion there are:

(a) **Communicational cohesion** means that all elements of the component operate on the same input or produce the same output. This is more than just doing a similar function. It is producing identical types of output or working from a singular input.

(b) **Sequential cohesion** is the stronger form of procedural cohesion. Instead of merely following the other in time, sequential cohesion is achieved when one part of the component is the input to another part of the component. It is a direct handoff and a cohesive identity.

In the STRONGEST form of cohesion there are:

(a) **Object cohesion**, each operation in a module is provided to allow the object attributes to be modified or inspected. Every single operation in the module. Each part is specifically designed for purpose within the object itself.

(b) **Functional cohesion** goes above and beyond sequential cohesion to assure that every part of the component is necessary for the execution of a single well-defined function or behavior. So, it is not just input to output, it is everything together is functionally cohesive.

EXERCISE - 4: System Design Specification

Objective: Perform the user's view analysis and Develop a Design Specification of the Proposed Solution (Use case diagram, Class diagram, Sequence diagram, State diagram, activity diagram)

Tools/ Device: UML tools

1. UMLet: https://www.umlet.com

Procedure:

2. Prepare a standard Software Design Specification document that specifies the System's functionalities that represents narrative of the design specification with diagram (Use case diagram, Class diagram, Activity diagram, and Sequence diagram, E-R diagram)

   (A) Identify and analyze various processes, use-cases, actors etc. of the system. And, use processes at various levels to draw the use-case diagram.
   (B) Identify various elements such as classes, member variables, member functions etc. of the class diagram. And, draw the class diagram as per the norms
   (C) Identify various elements such as controller class, objects, boundaries, messages etc. of the sequence diagram. And, Draw the sequence diagram as per the norms.
   (D) Identify various actions and corresponding events of triggering actions. And, draw the activity diagram as per the norms.
   (E) Identify various entities, attributes, and data dictionary of the ER diagram. And, draw the activity diagram as per the norms.

Evaluation: Your Design Specification must address the following questions:

1. Does the use case narrative represent the Scenario of the use case diagram? Does the Use Case diagram include the major use cases, actors who perform the use cases and the relationships among the use cases needed to deliver by the system?
2. Does the class narrative represent the Scenario of the class diagram? Does the Class diagram include the major classes (attributes, operations) and the relationship among the classes needed to deliver by the system?
3. Does the activity narrative represent the Scenario of the activity diagram? Does the Activity diagram include the major activities needed to deliver by the system?
4. Does the sequence narrative represent the Scenario of the sequence diagram? Does the Sequence diagram include the sequence of the major activities needed to deliver by the system functionalities?
5. Does the ER diagram include the major data entities mentioned in the user scenario? Does the data dictionary cover all the entities as mentioned in the ER diagram?

Chapter 4: User Interface and Experience (UI/UX)

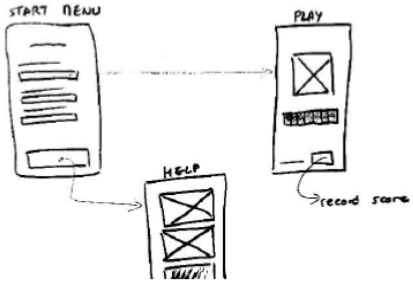
Have you ever seen the outcome of projects focused on action without at least a small amount of before-hand thinking and planning? Would you feel comfortable having your house built solely based on a quick oral description of what you are expecting? Chances are the final result would not match your expectations, especially if there were several people involved in the construction. In the long run, you will lose time and money swapping the upstairs kitchen and the master bedroom opening onto the garage [3].

That is the reason why you are encouraged to take some time before start programming, to think your application through, and maybe get feedback from perspective users, so you do not have to undo any of your future programming. There are different tools to help you keep your mind clear about your goal, to communicate within a team of developers (even if a priori you are not there yet) and to test user acceptance. These tools are wireframes, mockups, and prototypes. You will often find the terms used interchangeably. Assuming wireframes and prototypes are the same thing is like mistaking the blueprint of a house for a display house: they represent the same product, the actual final house you consider buying, yet they are not exactly the same and they serve different purposes. The objective is to clarify the composition and the goal of wireframes, mockups, and prototypes. It also provides a list of tools you could experiment with to design your application [3].

d. Wireframes

A wireframe offers a simplified, low fidelity (reliability), visual representation of the layout of each screen of the application: which content goes where, what are the possible interactions. Wireframes are prepared quickly, and anything which takes too long to draw (a carefully crafted icon, a proof-read text…) is represented in a simplified way, using placeholders (often gray boxes). Interactions are represented as lines/links between screens [3].

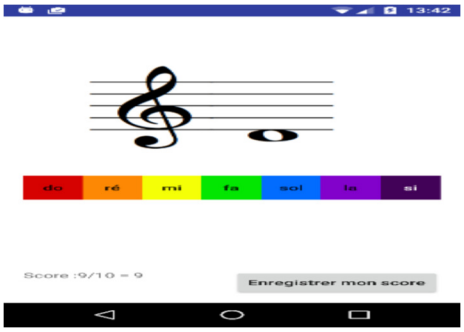
Wireframes are not interactive. A wireframe can be compared to the blueprint of a house: it specifies the architecture of the house but you cannot use it to judge the beauty of the house and unless you are an architect it is difficult to imagine how the final house will "feel" like. Wireframes are used in the earliest stages of a project when the design (layouts and workflow) still undergoes frequent changes because they are fast (thus inexpensive) to modify (you do not feel guilty if you trash them!). Wireframes are mostly adapted to the earlier iteration and documentation phase of development [3].



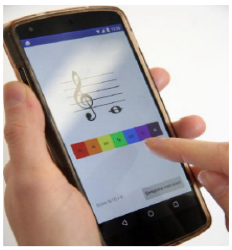
e. Mockups

A mockup offers a more refined, detailed version of the wireframe: it shows content like graphics and texts using a "look" (colors, fonts…) as close as possible to the one in the final application.

A mockup is built on top of a wireframe to realistically represent what the application will look like (Figure 12). Mockups are sometimes called "middle or high-fidelity (reliability) wireframes". But just like a wireframe, a mockup is static, non-interactive. It can be compared to the 3D model of a house. Mockups are useful to test the user acceptance of the visual side of the application (it can be a key factor of the application's popularity!) [3].



f. Prototypes



While wireframes and mockups model the user interface (UI), prototypes model the user experience (UX). A prototype is an interactive wireframe ("mid-fi prototyping") or interactive mockup ("hi-fi prototyping"). The workflow of the application is simulated by working links and transitions: areas of the graphical representation are clickable, enabling a journey through the user interface. The prototypes can be previewed on a computer and/or on your smartphone [3].

Prototypes are used in user testing before the development begins. It can be compared to a display house: it provides a drive test to future residents. The number of prototypes is usually limited because they are more time-consuming to create. Yet some tools are able to exploit your prototype to automatically generate part of the final code of your application. This optimizes your time but is often reserved to relatively simple projects [3].

g. What makes a good Android Application?

The rubric will help you identify what makes a project more or less successful, and why. It will help you to self-assess your own work and enable your peers to provide you more constructive feedback [3].

(A) **Workflow, navigation within the application**

- **Simplicity:** Natural, easy to grasp. Furthermore, in addition with this contextual help, help menu, hints in text input areas can be presented in the application.

- **Shortcuts:** The most common/frequent path in the application workflow requires as few clicks/actions as possible. For example, there are a "add item from toddler toys dpt." and a "add item from other dpt." buttons, the second button leading to the choice of dpt. Furthermore, the path most frequently chosen by the user gets associated with a shortcut. For example, if unlike other persons the user visits the "music dpt." More often than the "toddler toys dpt.", the short-cut button will be customized to link to the music dpt. Instead.

- **Acknowledgement:** Lets the user know if his actions where successful or not for instance, uses pop-up messages, bip or buzz sounds.

- **Error Message and Suggestions:** error messages are clearly visible and explain what is wrong in the operation and how the user can fix the problem. For example: "bad name: names start with a capital letter and contain only letters and single quote character", written in red under the problematic text area. Furthermore, the application also tries to fix the problem for the user. For example, if the user types "jOhn" for his name, the application proposes to replace by "John".

- **Redundancy:** Information is transferred from one part of the application to the next and not asked twice. For example, when the user is asked to provide an address to process his order, the zip code field is automatically filled using information provided in a previous page the user visited earlier. Furthermore, whenever possible, the application "guesses" information to avoid that the user types them. For example, if the user provided his city, look up the zip code in a data base instead of asking it to the user.

- **Confirmation and Fault Tolerance:** displays confirmation dialog before performing important actions (such as deleting all data or placing a shopping order). In addition, provide a way to restore deleted data.

(B) **Respect of GUI standard practices:** Conforms to standard GUI practices, for example,

- Vertical scrollbars are located on the right side
- Confirmation button is called "OK" (and pairs with "cancel")
- Colored or underlined text is a clickable link buttons initiate actions
- In addition, adapt to user's country usages

(C) **Accessibility and Readability:**

- **Font & Color:** Reasonable font size and color contrast (i.e. color choices or font sizes allow the users for good readability)
- **Clickable area:** at least 57 px wide X 45 px height clickable areas so the clickable areas are not too small or to close to each other to allow precise clicking without zooming in
- In addition, option to allow the user to configure colors and fonts.

(D) **Layout:** Adapt to different display sizes, for example, place a scrollbar to reach a button at the bottom. In addition, adapt to both portrait and landscape orientation

(E) **Data Input:**

- **Default values:** Provides automatic completion or lists of pre-entered data to choose from (i.e. provides default values).
- **Data History:** In addition, save and propose previously entered data
- **Keyboard:** Display the appropriate soft keyboard depending on the expected kind of input. For example, it does not display the default a-z soft keyboard when the user clicks an Edit Text supposed to start with a number instead provide number keyboard.

(F) **Pauses or interruptions handling:**

- **Pause:** When incoming call is picked up or the back button is hit, music is paused, and data is saved.

- **Resume:** In addition, phone rotation is handled gracefully: the user does not restart from a fresh screen with empty Edit Text or unchecked radio buttons, instead the screen displays the same information as before the rotation.

(G) **Bugs in the Application:**

- No bug under normal condition (valid user inputs)
- In addition, user inputs are checked for correctness, for example, the date Feb. 29th, 2015 will be detected as non-valid and the user will be asked to correct it.

(H) **Code Maintainability**: Ease of future evolution by a different developer

- Code is indented according to standard conventions
- Code is commented for other developers to understand properly
- Objects have meaningful names/ids to recognize easily
- In addition, external documentation with mockups, flow map and textual descriptions, classes and sequence diagrams, tests suite are provided

(I) **Performances:** The app is careful with power and storage resources and with responsiveness. For example:

- Be aware that the app does not wastes battery power, or storage space, or is not responsive enough. For example, it makes a lot of useless computations, it requests localization in all its activities even though it is only really necessary in a particular one, operations which take a while to complete freeze the user interface until they are accomplished
- It lets the user remove some stored information (for example, the user can decide to delete the profile picture he provided at first)
- Time-consuming operations (such as network transfers) are performed in the background and the GUI keeps responsive

(J) **Securities of the Application**

EXERCISE - 5: UI/UX Design

Objective: Design User Interface and User Experience of the Proposed Solution

Tools/ Device:

1. **Pencil Project:** This is an Open Source interactive wireframe tools. It runs inside Firefox or as a standalone version available for all platforms. It is available at https://pencil.evolus.vn, Git source: https://github.com/prikhi/pencil/blob/develop/README.md

Some other available prototyping tools are:

2. Moqups: https://moqups.com
3. UXPin: https://www.uxpin.com
4. Axure RP: http://www.axure.com
5. Fluid UI: https://www.fluidui.com
6. Proto.io: https://proto.io
7. Xiffe: https://xiffe.com
8. Flinto: https://www.flinto.com
9. InVision: http://www.invisionapp.com

Procedure:

1. Select and appropriate tools for prototyping
2. Design a prototype of your proposed solution using the selected tools based on the above-mentioned UI/UX design principles.

Evaluation: Your Prototype must address the above-mentioned design criteria

Chapter 5: Test Planning

The question "What is software quality?" evokes many different answers. Quality is a complex concept—it means different things to different people, and it is highly context dependent. The research has analyzed how software quality is perceived in different ways in different domains, such as philosophy, economics, marketing, and management. Software Quality (as per ISO/ IEC 9126): The totality of functionality and features of a software product that contributes to its ability to satisfy stated or implied needs. It can be customized for organizations. Software Quality (as IEEE Std 610): The degree to which a component, system or process meets specified requirements and/or user/customer needs and expectations [4].

Software Quality Assurance (SQA): It develops an effective plan for software development. It represents the function of software quality that assures that the standards, processes and procedures are appropriate for the project and are correctly implemented. It is also defined as a planned and systematic approach to the evaluation of the quality of and adherence to software product standards, processes and procedures. In SQA, there is a set of activities designed to ensure that the development and/or maintenance process is adequate to ensure a system will meet its objectives. It is an umbrella activity that is applied throughout the software process which consists of a means of monitoring the software engineering processes and methods used to ensure quality. An effective approach to produce high quality software [4].

Software Quality Control (SQC): involves in executing the software development plan effectively. It is the function of software quality that checks that the project follows its standards processes,

and procedures, and that the project produces the required internal and external (deliverable) products. In SQC a set of activities designed to evaluate a developed work product. It includes the following activities: Requirement Review, Design Review, Code Review, Deployment Plan Review, Test Plan Review, and Test Cases Review [4].

5.1 Testing Activities and Complete Testing

It is not unusual to find people making claims such as "I have exhaustively tested the program." Complete, or exhaustive, testing means *there are no undiscovered faults at the end of the test phase*. All problems must be known at the end of complete testing. For most of the systems, complete testing is near impossible because of the following reasons [4]:

- The domain of possible inputs of a program is too large to be completely used in testing a system. There are both valid inputs and invalid inputs. The program may have a large number of states. There may be timing constraints on the inputs, that is, an input may be valid at a certain time and invalid at other times. An input value which is valid but is not properly timed is called an *inopportune* input. The input domain of a system can be very large to be completely used in testing a program.
- The design issues may be too complex to completely test. The design may have included implicit design decisions and assumptions. For example, a programmer may use a global variable or a *static* variable to control program execution.
- It may not be possible to create all possible execution environments of the system. This becomes more significant when the behavior of the software system depends on the real, outside world, such as weather, temperature, altitude, pressure, and so on.

We must realize that though the outcome of complete testing, that is, discovering all faults, is highly desirable, it is a near-impossible task, and it may not be attempted. The next best thing is to select a subset of the input domain to test a program.

5.2 Testing Levels

Testing is performed at different levels involving the complete system or parts of it throughout the life cycle of a software product. A software system goes through four stages of testing before it is actually deployed. These four stages are known as *unit*, *integration*, *system*, and *acceptance* level testing. The first three levels of testing are performed by a number of different stakeholders in the development organization, whereas acceptance testing is performed by the customers [4].

In unit testing, programmers test individual program units, such as a procedures, functions, methods, or classes, in isolation. After ensuring that individual units work to a satisfactory extent, modules are assembled to construct larger subsystems by following integration testing techniques. Integration testing is jointly performed by software developers and integration test engineers. The objective of integration testing is to construct a reasonably stable system that can withstand the rigor of system-level testing. System-level testing includes a wide spectrum of testing, such as functionality testing, security testing, robustness testing, load testing, stability

testing, stress testing, performance testing, and reliability testing. System testing is a critical phase in a software development process because of the need to meet a tight schedule close to delivery date, to discover most of the faults, and to verify that fixes are working and have not resulted in new faults. System testing comprises a number of distinct activities: creating a test plan, designing a test suite, preparing test environments, executing the tests by following a clear strategy, and monitoring the process of test execution [4].

5.3 White-Box and Black-Box Testing

Test cases need to be designed by considering information from several sources, such as the specification, source code, and special properties of the program's input and output domains. This is because all those sources provide complementary information to test designers. Two broad concepts in testing, based on the sources of information for test design, are *white-box* and *black-box* testing. White-box testing techniques are also called *structural testing t*echniques, whereas black-box testing techniques are called *functional testing* techniques [4].

White-Box Testing: In structural testing, one primarily examines *source code* with a focus on control flow and data flow. Control flow refers to flow of control from one instruction to another. Control passes from one instruction to another instruction in a number of ways, such as one instruction appearing after another, function call, message passing, and interrupts. Conditional statements alter the normal, sequential flow of control in a program. Data flow refers to the propagation of values from one variable or constant to another variable. Definitions and uses of variables determine the data flow aspect in a program [4].

Black-Box Testing: In functional testing, one does not have access to the internal details of a program and the program is treated as a black box. A test engineer is concerned only with the part that is accessible outside the program, that is, just the input and the externally visible outcome. A test engineer applies input to a program, observes the externally visible outcome of the program, and determines whether or not the program outcome is the expected outcome. Inputs are selected from the program's requirements specification and properties of the program's input and output domains. A test engineer is concerned only with the functionality and the features found in the program's specification [4].

5.4 Test Automation

Test automation aims to automate some manual tasks with the use of some software tools. The demand for test automation is strong, because purely manual testing from start to finish can be tedious and error prone. On the other hand, long standing theoretical results tell us that no fully automated testing is possible. However, some level of automation for individual activities is possible, and can be supported by various commercial tools or tools developed within large organizations. The key in the use of test automation to relieve people of tedious and repetitive tasks and to improve overall testing productivity is to first examine what is possible, feasible, and

economical, and then to set the right expectations and goals. Various issues related to test automation include [4]:

Project Name:			Test Designed by:	
Test Case ID: FR_1			Test Designed date:	
Test Priority (Low, Medium, High): Medium			Test Executed by:	
Module Name: Login Session			Test Execution date:	
Test Title: verify login with valid username and password				
Description: Test website login page				
Precondition (If any): User must have valid username and password				
Test Steps	Test Data	Expected Results	Actual Results	Status (Pass/Fail)
1. Go to the website 2. Enter username 3. Enter password 4. Click submit	Username: 99999999999 Password: 321	User should login into the application	As expected,	Pass
Post Condition: User is validated with database and successfully login to account. The account session details are logged in the database.				

- specific needs and potential for automation
- selection of existing testing tools, if available
- possibility and cost of constructing specific test automation tools
- availability of user training for these tools and time/effort needed
- overall cost, including costs for tool acquisition, support, training, and usage
- impact on resource, schedule, and project management.

5.5 Test cases/test items

EXERCISE - 6: Project Test Planning

Objective: Perform various techniques for testing using the testing tool : unit testing, integration testing, Blackbox testing, Whitebox testing, etc.

Tools/ Apparatus: Selenium

Procedure:

1. Select a particular system (Web/Desktop/Mobile/Device)
2. Identify various modules of the system so that they can be tested stand alone.
3. Prepare test cases of testing the selected elements of your identified software.
4. Perform the test according to your generated test case and produce a bug report which will helpful for the system developer to modify the system for improve system's quality.

Chapter 6: Project Planning

This chapter will discuss what is planning, why do we need planning, what types of planning we need prepare, and how do we prepare each type of planning in a project development. According to Peter Drucker (2006), an action plan defines the following things [5-6]:

- It is a statement of intentions, but it is not a commitment
- It is not a straitjacket as it is not intended to restrict our movement or to restrict the operation of our business.
- A plan is meant to be revised often and continuously.
- It should anticipate the need to be flexible
- Needs a system for checking the results against expectations. Allow us to measure how we are progressing against the plan.
- Becomes the basis for executive time management. It informs the way we plan, the evolution of our executives, and the time that they spend in operating our business.

A project plan is very specific which includes the flowing phases:

1. **Scope planning:** Identify all the scope of the project
2. **Activity or Schedule planning:** Identifying all the tasks, the dependencies among them, and the schedule in order to come up with an actual duration and time frame for the completion of our project
3. **Resources planning:** Identify the types of resources need in the project. And, how many resources we need and when do we need them in the project development lifecycle
4. **Decision making plan:** Clarifying the tradeoffs that we are going to be making in terms of cost, resources versus time.
5. **Risk plan:** Develop a robust and a useful risk management plan that will allow the project team members to cope with the changes tin future throughout the life of the project.

One of the critical reasons why we plan projects is to help us bridge the gap between the initiation phase and the execution phase. The plan allows us to reflect on the objectives that we set in the initiation phase and to realize their feasibility. It also allows us to come up with ideas around how we plan to execute our project and what kind of mechanisms need to be in place. For example, the project management office, how it is going to look like, where it is physically going to be located, what type of meetings will occur, how often, and the plan gives us some indication as to what is needed and when. What are the pressure points of the project, milestones that have to be hit, and how we can plan accordingly [5-6]?

In general, there no prescribes rule to tell you how long you should spend planning. It is a work in progress, and it is a continuous improvement on our expectation. The plan is only there to start the process off. One of the most important factors, is to realize that, while you do plan, even if you have moved to the execution phase, and even if you want to progress the project, you might have to go back and modify your plan [5-6].

6.1 Scope Management

The first step in planning a project is scoping the project properly. And, yet this step is probably the most neglected step in the entire process of planning a project as we rush too quickly to look at timing and Gantt charts.

Work Breakdown Structure (WBS)

In order to scope the project properly, we often use a tool called the Work Breakdown Structure (WBS). It is a visual tool to break down the project into working components. Not only is it visual but also it is detailed. It allows us for an opportunity to really reflect on how and what is going to be part of our project. It is as simple as thinking about a hierarchy from the high-level project to sub-projects that it might include, to work packages and to actual activities and tasks. However, sometimes it can get very messy and complex. But in the process of coming up with the list and even looking at the visuals, there is huge amount of insight that we can gain around the nature of our project. And learn new aspects of it that we have not thought of before [5-6].

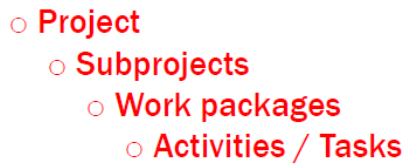


Figure: Hierarchy of a Work Break Down Structure (WBS)

Consider the project associated with manufacturing of a new product, a new cold press organic juice. From an empty facility all the way to having it on the shelf in a supermarket. When we started the project and we start thinking about this project, there are many different aspects of it. We need to think about the facility, we need to think about the manufacturing process. We need to think about the recipe that goes into the product and getting certified. Listing these out,

even on a blackboard in an old-fashioned sense, allows us to get an indication of what exactly are we going to have to do over the next few months in order to get our product to where we would like it to be. Once we brainstorm, either in the form of a blackboard, sticky notes, or on a white board. Then we can use certain tools and software in order to organize these thoughts in a more accessible way [5-6].

A high-level work breakdown structure for this specific Lumi juice plant example in given figure 10. At the high level, we see that the Lumi project will entail many different categories. Activities to do with packaging, activities related to the liquid itself, activities that are associated with the financing, or the information technology that the plant will need to operate, etc. The WBS will be breaking down even further to the level of the task itself (low-level WBS) to map all the associated tasks and project outcomes involved in the project management. It also tells us which of the categories have more activities associated with them. And where we might need to think about cutting down some of our scope or adding certain activities. Furthermore, maybe realizing that we neglected to think consciously around the bottling and the caps that have to come, have to be purchased and outsourced as part of our project. Therefore, visualizing the project and looking at what the activities are, and what is their nature, helps us gain clarity on the exact scope of our project [5-6].

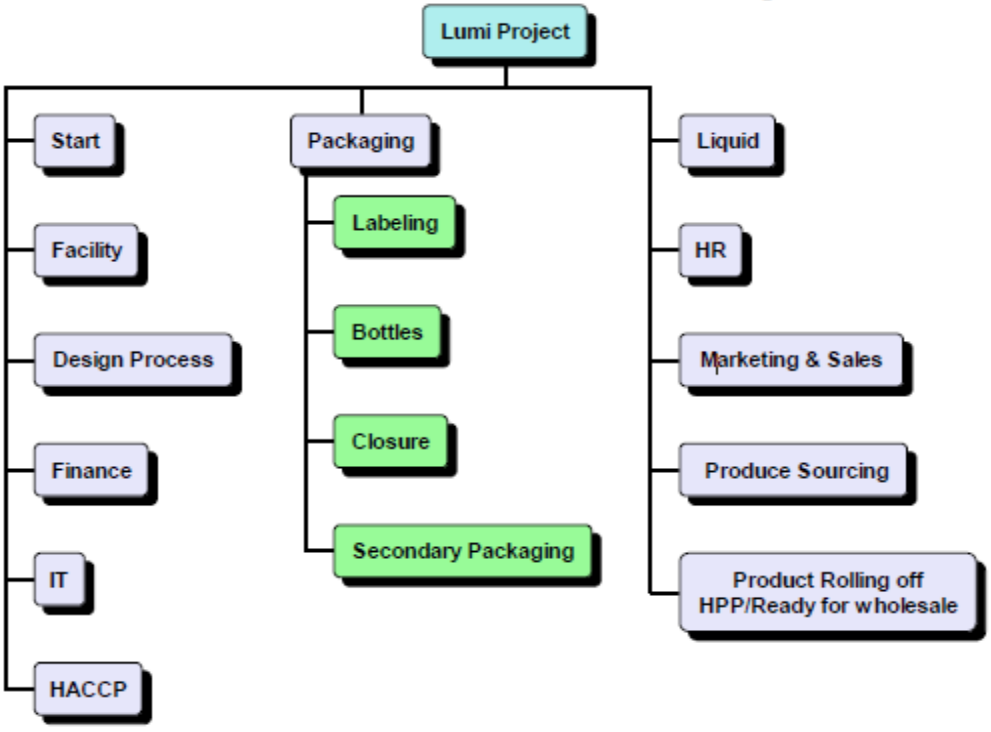


Figure: WBS of Lumi Juice production project

Now there are software tools that will allow you to automate and allow you to develop, work breakdown structure fairly simple. But the brainstorming process of going through and identifying what is part of our project and what we might want to leave out of this specific

project is vital before we even start looking at the schedule. So, a few suggestions are given below when you think about a work breakdown structure [5-6].

Brainstorm: start messy i.e. start with something that you can easily modify. Either by placing sticky notes on the board, reorganizing them.

Group discussion: use a team of people to brainstorm, and work together with individuals, including those who are actually going to be doing the work or your clients, who you might need to deliver the product to in order to make sure the that the scope is well defined.

Use Software Tools: Also, you can think about and find out more information about automated tools. Microsoft Project has a work breakdown structure ability within it. WBSPro is another add-in that might be helpful for you as you start working on your work breakdown structure.

How details: there is some guidance that we can use, and some rules of thumb associated with how detailed we need to be. However, you should consider each task at the end their tree, must be executed by a specific resource. Some like to think about the actual tasks taking no more than 10 to 20 days of work. However, as a proportion of the broader project each task in the big WBS hierarchy tree should be somewhere between 5 to 10% of the work.

Now, if you cannot get to the level of detail that is required, it implies that you might not be talking to the right people. And you might need to find out more information about the scope of your project.

EXERCISE - 7: WBS and Effort Estimation

Objective: Perform project management activities: effort estimation, WBS, activity planning, resource allocation

Tools/ Apparatus: Microsoft project

Procedure:

1. Identify all the micro tasks related to project management and categorize them within the WBS structure
2. Perform detailed effort estimation correspond with the WBS and schedule
3. Draw a Gantt chart of the identified tasks from WBS based on the precedence of each tasks you've identified

Evaluation: Your work must address the following questions:

1. Does the provided WBS correspond with the user scenario and the estimations made at the Feasibility study?
2. Does the detailed effort estimation correspond with the WBS and schedule?

3. Does the detailed effort estimation compliment with the estimation at Feasibility Study?

6.2 Time Schedule Management

The first step toward activity planning is to identify the dependencies among the tasks identified in the work break down structure. And then planning with the activities accordingly.

6.2.1 Activity Dependencies

Once you have you work breakdown structure, the next step is to think about the dependencies among the actual activities. Which tasks are dependent on others? And how are they dependent on them? The dependencies will give us the logical flow of the project and ultimately, they will allow you to find out the duration of our project [5-6].

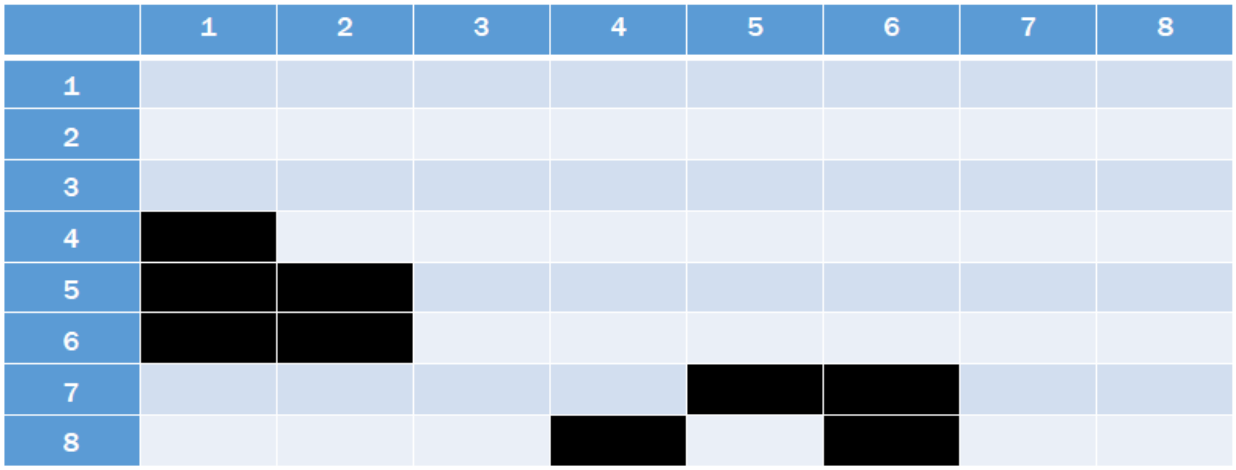
So, how might you visualize the dependencies among the activities?

To identify the dependency, start with the work breakdown structure of a simplistic project and just walk through how the dependencies might emerge. For example, in a startup project, there is a simplistic high-level view of a project gives us the basics which has the following activities: creative step, a strategy, information technology, fundraising, marketing and sales, finance and HR. So, what is the dependency among them and what is the sequence in which you can execute these specific activities?

The dependency in the project activities can be identified by three different techniques which are Design matrix, Dependency table, and Network model.

(A) Design Matrix to identify Dependency:

One useful way to think about the dependencies is in the form of a design matrix. The design matrix typically has the list of activities in the column and the list of the activities in the rows. And, for instance, in the design matrix we have the following task four is dependent on task one. Similarly, we see from this design matrix that task five depends on task one and task two. Task six depends on task one and task two. Task seven depends on five and six. And finally, eight depends on four and six. Each row highlights the tasks on which the task labeled at the beginning of the row is dependent on [5-6].



There is a lot of information in this design matrix. For example, we can easily spot whether we have circular relationships, meaning two tasks that are continuously dependent upon each other. That will imply that we are going to do some rework along the way. Also, from the design matrix we can spot how complex a project is. For example, if many of the activities are dependent on previous activities, there is going to be a lot of iteration and our project is going to be more complex, and we will need to plan accordingly [5-6].

(B) Dependency Table to Identify Dependency:

There is a table format to present the dependencies where you can list each activity with the set of their predecessors. Similar information, but in a different way of presentation. In Table 1, fundraising depends on the creative. And, the useful thing is that the individuals who are going to execute these tasks can agree that indeed they depend on others before them. They can also make sure that they understand who depends on the output from their tasks in order to ensure proper flow of the project activities [5-6].

Table: Dependency Table

Activity #	Description	Predecessors
1	Creative	-
2	Strategy	-
3	IT	-
4	Fundraising	1
5	Marketing	1,2
6	Sales	1,2
7	Finance	5,6
8	HR	4,6

(C) Network Diagram to Identify Dependency:

It is a visual representation of the project, and the flow of the information, and the physical aspect of the tasks and the materials that we need to accomplish along the way. In this case, we see, once again, the same links exist, in which creative feeds into marketing, sales, and fundraising [5-6].

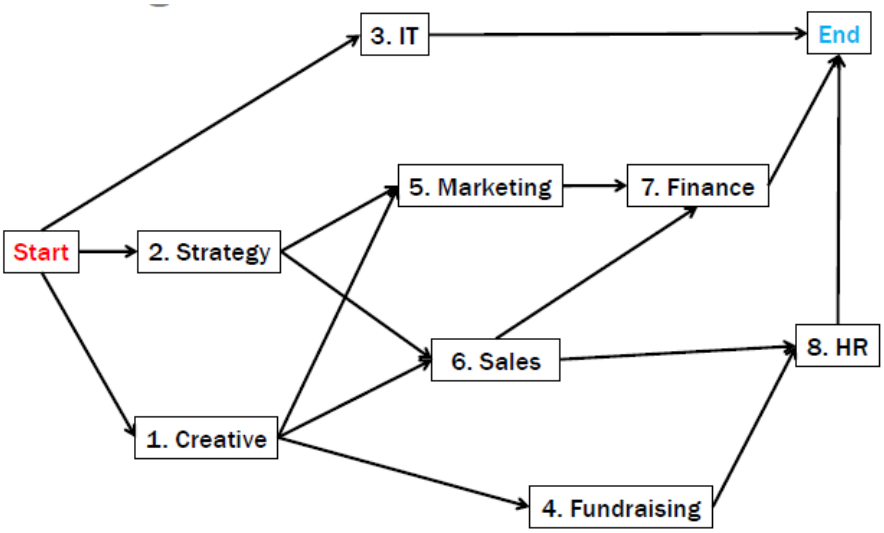


Figure: Network Diagram

Now you might notice, in this network diagram, that IT is left hanging. It does not depend on anything, nor does anything depend on it. But that is technically not exactly correct, because the process cannot be completed until the information technology is installed and operates properly. And, one recommendation that you should be thinking about is that as you look at your network diagram, make sure that at the end of the day there are no hanging tasks. You have one endpoint for your project. Similarly, you probably have one starting point that kicks everything off. And while I could do IT, strategy and creative at the same time, they are all going to be dependent on the start of our project. And so, we are going to add two more nodes to our project. The start of the project and the end, and we no longer have any hanging tasks in our network diagram [5-6].

<u>Types of Dependency:</u>

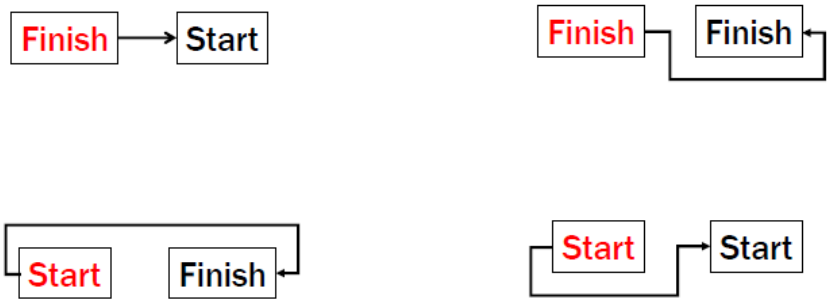
Furthermore, as we think about our dependencies, there are different types of dependencies as presented in the following diagram.

(A) Finish to Start Dependency: it the most basic one is a simple finish to start relationship. You need to finish Task A in order start Task B. Typical example of that might be in software development. You need to finish the coding in order to start the quality assurance.

(B) Finish to Finish Dependency: Another type of relationship could be a finish to finish dependency. Two tasks that need to finish together. For example, during a wedding, the bride needs to get ready and she needs to finish her preparation at the same time that she arrives, for instance, with the minister or the rabbi. In some case the rabbi was late, and the bride finished sooner.

(C) Start to Start Dependency: Another type of relationship that we might see is a start to start. Two activities that might need to start and work simultaneously together. Typical examples are construction projects, where we might need to cement and pave (concrete) at the same time.

(D) Start to Finish: a slightly more complex relationship might exist where we have a start implying a finish. We cannot technically finish an activity until we start the next. typically holds in a situation where we have physical materials and we need to hold together for instance a skeleton by starting another activity.



6.2.2 Activity Effort Estimation

In order to establish how long our project will take, what is the schedule, and when is the completion date for our project? We need to think very carefully about the durations of each of the individual tasks or activities. For instance, in the case of the startup project example, we might come up with the following estimated duration in weeks for each one of our tasks. We assume that creative might take us five weeks. Strategy will take us two weeks to develop. The IT might take four weeks to put in place. Fundraising is a long activity that will take us four weeks. Marketing will take us five, sales will take us two weeks. And hopefully, with finance and HR, we will be done within two and one weeks [5-6].

Table: Estimated Duration of Project Activities

Activity #	Description	Predecessors	Duration (Weeks)
1	Creative	-	5
2	Strategy	-	2
3	IT	-	4
4	Fundraising	1	4
5	Marketing	1,2	2
6	Sales	1,2	5
7	Finance	5,6	2
8	HR	4,6	1

But while thinking about these durations and coming up with these weekly estimates is not an easy task. It takes a lot of practice, and there are a lot of problems and a lot of difficulties when we come to think about our estimates. Few of them are discussed here [5-6].

Problems in Effort Estimation

(A) **Parkinson's Law (Over Estimation):** One of the most notorious struggles that individuals have when they think about how long a task will take in a project is something called Parkinson's Law, made publicly known by work by Cyril Parkinson in the 1950s. The Parkinson's law tells us that if we plan for an activity to take a certain amount of weeks, be sure that the work will fill up that entire duration. If we give somebody four weeks to complete their task, they will take the entire four weeks to complete their task.

(B) **Student's Syndrome (Over Estimation):** It says that if someone give you four weeks to complete a task, you are like to not even start working on that task until the very last minute. And therefore, even if we think that a task might only take two days, but we give you two weeks to do it, you will actually only work towards the end of that duration. Again, it makes it harder for us to really get a good sense of how long a certain piece of work might need or might require when we plan up front.

(C) **High Confidence (Under or Over Estimation):** Sometimes, individuals are notorious for being overconfident. And, if you ask them how long something will take them, they are likely to be very sure of themselves. And actually, report something is probably longer or shorter than it will turn out to be in reality.

(D) **Anchoring and Confirmation (Under or Over Estimation):** other biases that individuals suffer from are anchoring and confirmation. I'll say the task take you two or three weeks and If I ask you how long a task will take? Here, I anchored you around those numbers. And, it is not likely that you suddenly tell me, oh, actually you're way off, it's going to take me eight, eight weeks to complete. And so, by having a conversation, we sometimes anchor ourselves and we sometimes like to confirm historical patterns. And so, if we estimated a task in the past to take on two weeks, we might report that going forward.

6.2.3 Activity Scheduling

A typical network diagram of activity scheduling might look like the following diagram. We have the exact same layout of our tasks previously we have discussed in a start project development. We have our start and we have our end. And we wrote underneath each one of the names of the tasks and the estimated duration that we came up with. Five weeks for creative, four weeks for IT, similar to what we saw in the table format before. But, how long will this actual project take, from start to end? What is our expected completion date [5-6]?

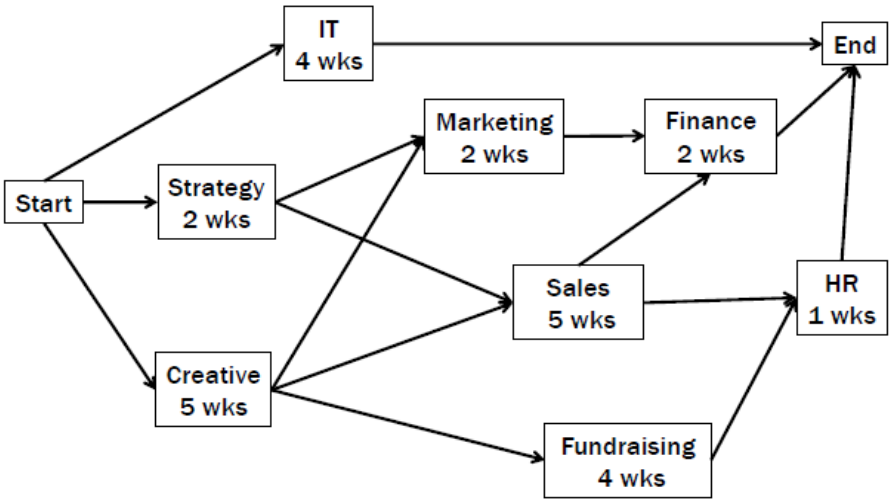


Figure: Estimated Duration in Network Diagram

It might not be too hard to find the longest path. If we look at all the different combinations and look at all the different possible paths along the tree, we might be able to see that the longest it will take us will be a path that goes from the start through the creative, through the sales, finance to the end of the project a total of 12 weeks. But imagine that you have a much more complex and a much larger project to analyze. With many tasks and many combinations of paths. What will be your project duration? Are you going to sit down and list all the possible paths, from start to finish, to come up with the longest one? There is a systematic process that allows us to identify the project completion date. How long the project will take and will allow us to identify more information about our project that will turn out to be very useful as we think about our project and plan it properly [5-6].

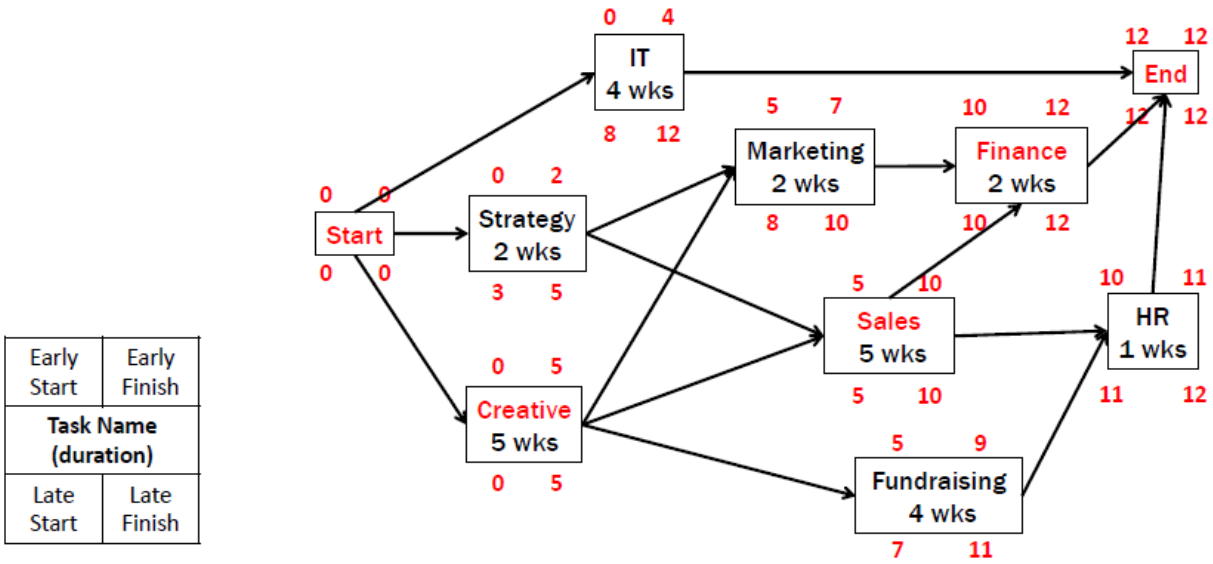


Figure: Activity Scheduling in Network Diagram

Earliest Start and Earliest Finish (Time moves from Start to End):

This step presents everything from left to right (i.e. Start to End). We can start at time zero. So, the start task, we know, is a dummy task and it does not take us long. It does not have any duration of its own. So, it finishes its start at time zero. If we are at time zero, we can actually immediately start the IT task, the strategy, and the creative because they do not depend on other activities to finish. So, the three of these tasks can start as soon as possible at time zero. Now if the IT started at times zero, since it takes me four weeks to complete, the earliest we can finish it will be at time four. Similarly, the task associated with the strategy, since it takes me two weeks to complete, it will end at time two. And the creative, in a similar way, we can add five weeks, which is its duration, and we will end at time five. Since strategy and the creative tasks are completed, we can start the marketing task. The two predecessors for marketing have actually completed. When had they been completed? The creative activity is the later of the two, and so the earliest that the marketing task can start will be at time five. Therefore, since it takes two weeks to complete, it will be finished at time seven. In a similar way, the sales activity can start at the earliest at time five and it will be completed at time ten. Finally, the fundraising activity only depends on the creative activity, and, therefore, it can start at time five, and since it takes four weeks to complete, it will be finished by time nine. The finance task depends on marketing, and on sales. Since sales is the later of the two, therefore, the earliest the finance can start will be in week ten. Adding two weeks for its duration, it will be completed at time 12. The HR activities depend on sales and fundraising and they can too start at time 10, and they will complete it will be completed at time 11. And therefore, if we look now at the end task, the earliest the project can be completed will be at time 12 weeks. If everything is scheduled as early as possible 12 is the duration of the project [5-6].

Latest Start and Lates Finish (Time moves from End to Start):

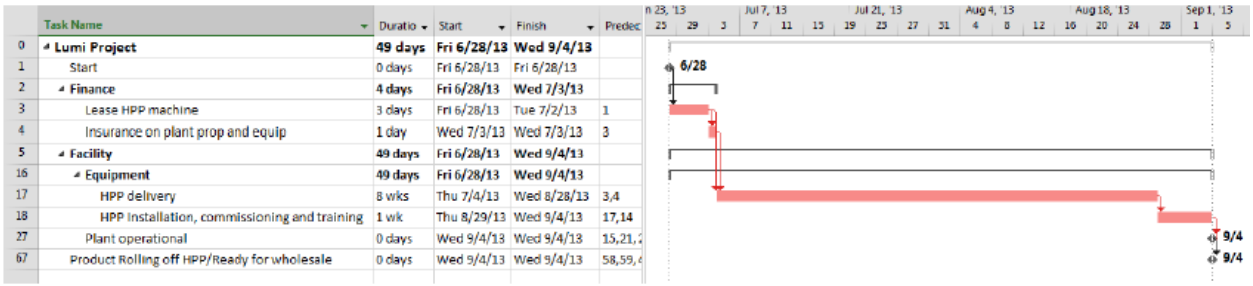
This step presents everything from right to left (i.e. End to Start). We are going to go back in time and discover what is the latest we can start? How much can we push some of the activities and still not impact that total duration of 12 weeks?

So, we have planned to complete the project by 12 weeks. This implies that, at the very latest, the HR task will be finished, we would not want it to be finish past 12 weeks. Which means at the latest it would need to start at time 11th weeks since it takes us one week to complete. Similarly, the finance, at the latest, we want to finish at time 12 weeks in order to not postpone our entire project. And therefore, it would at the latest have to start at time 10th week. Furthermore, we now know that the IT task at the very latest we would like it to finish at time 12 week since it is feeding into the final completion of the project. And therefore, at the very latest it was to start, that would have to be at time eight in order for the four weeks to be completed on time. The marketing task at the latest could finish at time ten, and therefore at the very latest, it would have to start at eight. The sales feeds into both HR and finance activity. In order to ensure that it delivers what it needs to deliver to each one of those tasks, it must complete by ten, since the finance task is dependent upon it, and therefore it had to start at week five, at the very latest. Similarly, the fundraising at the very latest, can finish at 11, therefore, it had to start at seven. Since Creative work feeds into marketing, sales, and fundraising, therefore, it must be completed by week five in order to support the sales activity, and therefore to start immediately at time zero. Strategy however, since it feeds into marketing and sales, it has to be completed at time five, therefore, it could start at the very latest at time three. And if we mark our start activity with 0 and 0 (i.e. latest start and latest finish), Since it is a dummy task to represent the start of the project, we have completed the entire set of calculations. We have early start times, and we have late start times for each one of our activities [5-6].

Critical Activities:

It is interesting to see here that some of the activities, there is a gap between Earliest Start (ES) and Latest Start (LS) similarly in between Earliest Finish (EF) and Latest Finish (LF) called *slack* or *float* (i.e. the amount of time in gap can be delayed in some task without delaying the final project deadline 12 weeks). And there are some activities which do not have slack where we cannot delay in starting and completing the activities. These activities, in which the early start equals the late start, or similarly, the early finish equals late finish, these are the critical activities. They must start at the time designated to them and similarly they must end at the time that designated to them. Otherwise there will be a delay to the project. The other activities in which there is slack, those activities will be leverage where we can postpone starting them. And, the critical path contains all the critical activities connected from start node to end node. There will be at least one path in the network diagram which is critical path. The critical path is the longest path of activities chain in the network. The project duration, defined by the length of the critical path, is also known as *makespan*. A delay in any activity along the critical path will cause a delay in the project. The method was developed by Engineers at DuPont Corporation in the 1950s [5-6].

There are plenty, plenty of tools to calculate the critical path for you. They will run through the calculations and identify the early start and the early finish, the late start, and the late finish, depending on the precedence relations and duration of the activities you have defined. One of those tools is Microsoft Project which will allow you to see is the critical path highlighted in red color in the Gantt chart is a critical path [5-6].



	Task Name	Duratio	Start	Finish	Predec
0	**Lumi Project**	**49 days**	**Fri 6/28/13**	**Wed 9/4/13**	
1	Start	0 days	Fri 6/28/13	Fri 6/28/13	
2	**Finance**	**4 days**	**Fri 6/28/13**	**Wed 7/3/13**	
3	Lease HPP machine	3 days	Fri 6/28/13	Tue 7/2/13	1
4	Insurance on plant prop and equip	1 day	Wed 7/3/13	Wed 7/3/13	3
5	**Facility**	**49 days**	**Fri 6/28/13**	**Wed 9/4/13**	
16	**Equipment**	**49 days**	**Fri 6/28/13**	**Wed 9/4/13**	
17	HPP delivery	8 wks	Thu 7/4/13	Wed 8/28/13	3,4
18	HPP installation, commissioning and training	1 wk	Thu 8/29/13	Wed 9/4/13	17,14
27	Plant operational	0 days	Wed 9/4/13	Wed 9/4/13	15,21,
67	Product Rolling off HPP/Ready for wholesale	0 days	Wed 9/4/13	Wed 9/4/13	58,59,

Figure: Activity Scheduling in Microsoft Project

Non-Critical Activities and its Priority:

In the startup project example, there are several activities which are non-critical and have some slack associated with them. Let's look at the IT project. The IT project at the earliest we can start at zero and finish at four. However, we know that we can postpone it to start as late as week eight. So, we have plenty of slack associated with this task. So, when should we do the work? Should we schedule it to be performed as soon as we can, or should we postpone it, and have it done later [5-6]?

Depending on your project priorities and your project objectives, you might want to go with one schedule versus the others. Let's think about the benefits of an “as soon as possible” schedule versus “as late as possible” schedule. Are we constrained by cost or by time? And let's think about how those two dimensions interact with our schedule. If we schedule activities as soon as possible we are incurring earlier costs and using our resources immediately. We are however, on in terms of time, giving ourselves some slack. We are starting tasks as soon as possible even though we know that we have some slack in our schedule. And so, as soon as possible schedule has benefits on the time dimension, but it does have downsides on our cost. We need to invest cost now in order to gain slack and buffer in terms of our timing. And so, if we are constrained by time, and we are planning a project in which the objectives are determined by the time and we are going to compromise on our costs the as soon as possible schedule might be better. On the flip side, if we are constrained by costs, and we are working with a limited budget and we know that maybe time is something that we can compromise on. We are better off pushing our tasks as late as possible which delays the commitment of cost and delays the commitment of resources until a time that we know we must use it. Costs further down in the future are cheaper. However, it eats up all the slack. Postponing the tasks mean that we leave yourself with very little slack towards the end of our project and if we need extra time,

we will not have it. And, if our priority is cost, you are better off scheduling the non-critical activities as late as possible [5-6].

	Cost	Time
As Soon as Possible	✗	✓
As Late as Possible	✓	✗

Figure: Priorities of Non-critical Activities

Reduce Project Duration:

After gone through all the steps such as we have scoped our project and identified all the different activities that need to take place. We have identified the precedence relationships. And we have estimated the task durations to find the overall project expected completion time, and overall duration. But now, you realize that it is way too long. You do not have 12 weeks available to complete your project. Hence, you need to shorten your project to meet a deadline. What can you do? Some suggestions are given here [5-6].

(A) By Shorting the Duration of Critical Paths: The critical path is the primary way to focus our attention and to shorten our project duration (i.e. this is the longest path of completing a project). We start by identifying those tasks on the critical path that are candidate to shortening, or to crashing. We look and we think about what we need to spend in order to reduce their duration. Do we need to include additional resources? Do we need to spend more money? Once we have identified the least expensive task among the critical tasks that we are going to be able to shorten, we start experimenting with shortening the duration. But pay very careful attention. As you start shortening task duration, a new critical path, a new longest path in your project might emerge. And therefore, you might need to take your attention elsewhere and start crashing other activities from other parallel paths [5-6].

So, we need to conduct this crashing activity or reducing the project duration in a cycle. We identify our critical tasks. We find the least expensive task to reduce its duration. We reduce its duration, and we identify the new critical path that might have emerged. All the critical paths, even if there are multiple of them, will need to be shortened in order for our overall project duration to be reduced. In the process of doing this, what's going to happen? Overall, our critical path within our project is going to be shorter. It might change. We might introduce a new critical path and a new set of critical activities. And more of them, a higher proportion of our project, will be on the critical path. And therefore, there will be less slack overall in our project. And we might actually be opening the door to more risk. But if we need to meet that external deadline, we might be willing to take upon ourselves that risk, reduce the slack in

our project. Have more activities shortened and tightened in their duration in order to meet the overall deliverable date [5-6].

(B) By Reducing the Project Scope: If it is not possible to invest additional resources in order to reduce some of the critical tasks, another alternative is to go back to scope. Reflect on what is necessary and look at the work breakdown structure. Can we eliminate some unnecessary tasks? Can we reduce the functionality of our product, and perhaps, postpone some functionality to a later version or to a later phase? If we do that, and maybe we consider some areas where we can cut corners, or perhaps, outsource some of the work. We might be able to reduce the project completion time by looking at the scope, reducing the scope, and not necessarily accelerating our tasks. Depending on the tradeoffs among our objectives, if we are constrained by the scope or by time, or by cost. We can choose how to investigate reducing the overall project duration [5-6].

Reflecting on Common Mistakes in Project Planning

As we progress through the planning process, it might be worthwhile pausing and reflecting on some common mistakes that project development organization makes throughout the planning process. A research identified that following are some common mistakes made in the project planning from the large aerospace companies to small independent entrepreneurs. Reflecting on this list might help you shed some light on what you have done along the way that you might want to improve on as you develop your plans [5-6].

- **No Work Break Down Structure (WBS):** Many firms dig into the plan, and they neglect to stop and come up with a proper work breakdown structure. It is the most crucial step of the plan. No work breakdown structure will imply missing tasks, complete pieces of work that are neglected to be considered as part of your plan. It is all too simple to forget, or to neglect a component of your project and therefore, conducting a proper brainstorming session, or working out your work breakdown structure, will ensure that you do not find yourself without a proper scope. If you don't have a proper scope, then you won't have the proper project estimation, the duration estimates for instance. And, you won't have a proper network diagram or critical path, and your cost estimates will be way off, so spend the time working through the work breakdown structure [5-6].

- **Task Duration specified in terms of Work (24/7):** Another typical mistake is the task durations are specified in terms of work and not in terms of actual durations. Neglecting to realize that individuals have to go home for the weekend, or they have to take some time off once in a while in order to come back the next day refreshed. And so, think about durations and not only work [5-6].

- **No Network Diagram:** The network diagram is a visual tool. It is appealing in the sense that it gives us an overview in a very visual way of the flow within our project. Make sure to spend some time on mapping a network diagram. Look at your network diagram and make sure that

there is a logical flow of your chain from start to finish. That will allow you to spot your critical path and your entire set of precedence [5-6].

- **Poor Precedence Relationships:** the precedence relationships can often be very tricky to identify. And so, as we think through our plan and we work typically looking at a Gantt chart, we might rush in and we might neglect to identify proper precedence relationships between predecessors and successors. And, do not use fixed date to mimic a relationship. Do not expect, that if a task, if you fix it to start at the first of August, it represents a link, or a flow from other tasks that are feeding into it. The first of August is arbitrary. If you postpone the entire project, what you really want to be saying is that task b, depends upon task a. And so, represent it with the proper dependency. If you happen to have tasks in your project that are scheduled in the right sequence but don't have the relationship, you might not realize in the future as things change, that they are actually linked by some information flow or from some even material flow from one to the next. And, if you are working with a work breakdown structure with a complicated project and you have summary tasks, do not link the summary tasks. Those tasks include the detailed individual level tasks, and the relationships should be formed only at that level [5-6].

Table 3: Reflecting on Common Mistakes in Project Planning [5]

Issues	Consequences	Impact
No WBS	Missing tasks	Incorrect scope No responsibility assigned Incorrect project duration, critical path, cost estimates, and resource loads
Task durations specified in terms of work	Underestimate task durations	Underestimated project duration Incorrect critical path
Excessive task durations	Dominate plan	Incorrect project duration and critical path
No Network diagrams	Missing dependencies	Incorrect project duration and critical path
Tasks without predecessors or successors	Missing dependencies	No project critical path
Fixed start dates to mimic dependencies	Missing dependencies	Plan prone to errors No project critical path
No dependency defined between activities in the correct sequence		Plans update cumbersome and prone to errors
Dependencies between summary tasks	Unnecessary dependencies	Time wasted, unclear network diagram

Therefore, following these common mistakes and reflecting and ensuring that you do not exhibit them within your project plan, will allow you to put your best step forward, and have more confidence in your plan [5].

A check list is also discussed here to provide you a set of tasks to walk through in your project plan and ensure that you are on the right track. The checklist takes you through the steps and allows you to reflect on the project planning. These are very important to identify upfront in the project planning [5-6].

- **Work Break Down Structure (WBS)**
  - ✓ Is WBS available? Is it complete?
  - ✓ Did we show it to enough individuals within the organization to ensure that we have confidence and that everybody agrees on it?

- **Network Diagram**
  - ✓ Did we look at the network diagram? Do we have one start and one finish?
  - ✓ Do we have successors and predecessors to all of our tasks?

- **Milestones**
  - ✓ Did we consider milestones one start milestone and one project completion milestone?
  - ✓ Are there natural points in our project that have important deliverables? Or that many different paths meet and come together.
  - ✓ Are there Intermediate milestones? Does each one has a critical path?

- **Gantt Chart**
  - ✓ Did we actually spend some time looking at the Gantt chart and making sure that it reflects everything that we would like it to reflect?
  - ✓ Are the dependencies properly presented? And no missing dependencies between activities in correct sequence?
  - ✓ Are we not making use of any dates in a wrong way?
  - ✓ Are we making sure that we don't link any of the summary tasks and not any specific task takes way longer than the rest of our project which might mean that it actually should be broken down into sub-tasks?
  - ✓ Check generalized dependencies (overlaps, time lags)
  - ✓ And do we have a clear critical path from start to finish, and does it make sense?
  - ✓ Does it fit our intuition around what is going to take us the longest in our project?

This checklist will allow us to make sure that our planning process so far has followed the steps and has been checked for sanity and make sure that we are detailed enough and that we are following the guidelines to ensure that we have the most confident in our plan [5-6].

Summary of Project Planning

Let's summarize and wrap up the key steps in planning a project. First and foremost, acknowledge that it is the crucial part of our project planning. If we spend the time planning properly and not rushing to execution, we will have a higher chance of success. We know that successful projects have gone through the planning process properly. We also want to remind ourselves, as we set out to do the planning, that the plan is not a straitjacket. Rather it is a guideline, it is supposed to get us started. But we will allow for flexibility, and for changes, as time goes by. We will also be following our steps, through word breakdown structure, and identifying the scope, through a complete project schedule. We will make sure that the project goal is clear to us and that we know where the trade-offs are going to take place among the different objectives and making sure that we have a complete scope. Starting a project without identifying the complete scope is almost a guaranteed failure [5-6].

A few more points as we think about wrapping up our planning phase. Make sure that we estimate the individual activity durations. We know that there are some challenges associated with estimating the task durations, so spend some time reflecting on some of these individual biases such as overconfidence and anchoring. Decide who's going to own each one of your tasks and communicate with them. That they understand whether they're critical or not and what tasks depend on them further down in the project, and who they depend on upstream. Identify your critical tasks in terms of your duration, not necessarily in terms of the importance of the scope. Prevent conflicts up front and add dependencies if need to represent conflicts. And finally, know that you have, in your back pocket, the tools to decide to accelerate the project if need and to tradeoff your cost, time, and scope associated with your priorities of your project. Putting this all together will ensure that you are on your way to a successful implementation of your specific project [5-6].

EXERCISE - 8: Activity Scheduling and Resource Allocation

Objective: Perform project management activities: effort estimation, WBS, activity planning, resource allocation, risk analysis

Tools/ Apparatus: Microsoft project

Procedure:

1. Make a schedule planning of tasks identified in WBS
2. Describe the available resources and their allocation in performing the project tasks

Evaluation: Your work must address the following questions:

1. Does the network diagram correspond with the provided WBS?
2. Does the resources are identified and allocated in different activities in the projects?

6.3 Risk Management

We have planned our project by coming up with the correct scope. and the dependencies among the tasks. We have calculated the completion time of our project. Before we start executing the project plan. There is a big uncertainty or unknown, associated with how the project will actually take place. And so, part of our planning is going to pause after the initial first steps. And think about what risks do we foresee in the execution of our project. And can we plan and modify our plans accordingly in order to better face those risks as they manifest themselves? Before we start thinking about the tools that we are going to use in order to incorporate risks and uncertainty into a project plan. Let's try and identify what is the source of the uncertainty and where the risk's coming from as we think about projects and the execution of projects [5-6].

6.3.1 What are the Sources of Risks in a Project?

Using some terminology defined by Christoph Loch and his colleagues when they were working together at INSEAD, we are going to be thinking about the risks or in other words the uncertainties in a project on several dimensions [5-6].

(A) Foreseeable Uncertainty: the risks that we anticipate in our project will face that we can plan for and conceptualize, and we know they are going to occur. For example, Natural variation in the task durations, conditions associated with the weather that might change and might impact the duration and the execution of our project. We may want to think about alternative paths that we may want to take as we conceptualize and think about the future of our project. These are foreseeable uncertainties. Some can be positive uncertainties, and some can be negative. But we can think about them. We can articulate them. And we can conceptualize how we will deal with them.

(B) Complexity Uncertainty: Another source of uncertainty comes with the complexity of a project. When we have multiple tasks that interact with each other and sometimes in not obvious ways. Or we have stakeholders that might interact with each other and might lead to unknown outcomes, we may have a level of uncertainty that we cannot necessarily plan for it, but we know it exists. We know it is out there, but we do not exactly know what the setup outcomes might look like, and how likely they are to occur. And so, we cannot plan a mitigation plan, for instance, if we do not know exactly how an outcome will turn out. But we know that the project is complex, and so we can think about it in that context.

(C) Unforeseeable Uncertainties: And, finally on the most extreme level, we have unforeseeable uncertainties. We have unknowns that are far beyond what we can conceptualize at the start of the project, or the initiation and the planning phase. We are dealing with a very novel technology or a brand-new market that we have never experience before. And therefore, we cannot even conceptualize the range of outcomes or where they are likely to occur. And therefore, it is called unknown unknowns.

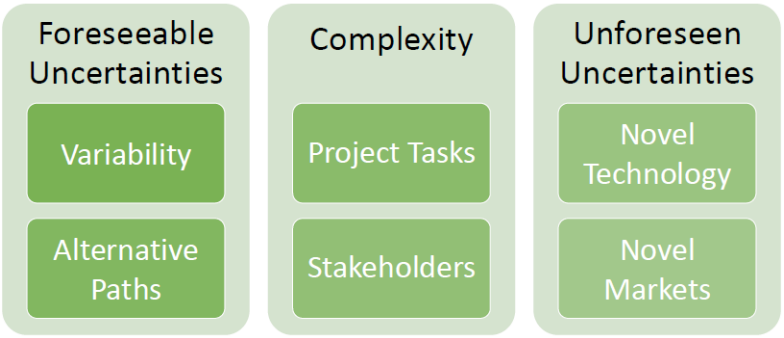


Figure: Source of Uncertainties in a Project [5]

Sometimes we can use the following terms to define these degrees of uncertainty. We can think about uncertainty, risks, and ambiguity at the extreme. Ambiguity characterizes those unknown unknowns. What's important to highlight is that risks, while they exist in projects, and typically, we think about them quite negatively. Risks and uncertainties, or many uncertainties, can offer big promise. If tasks are variable and the outcomes are variable, they can end up in a very positive place, not only on a negative and on the downside. Most of our planning, however, will focus on how we deal with the downside [5-6].

6.3.2 Risk Identification Matrix

In order to think about where your firm is, and your specific project on the spectrum of foreseeable uncertainties versus unforeseeable uncertainties, and how complex your project is, you may want to think about a mapping (Figure 19). Consider for instance where you fall in terms of technology uncertainty and demand uncertainty. Technology, identifying the technology of your actual project and what it requires to execute. And demand identifies your clients, your revenue stream, and the typical firm turnover in your space [5-6].

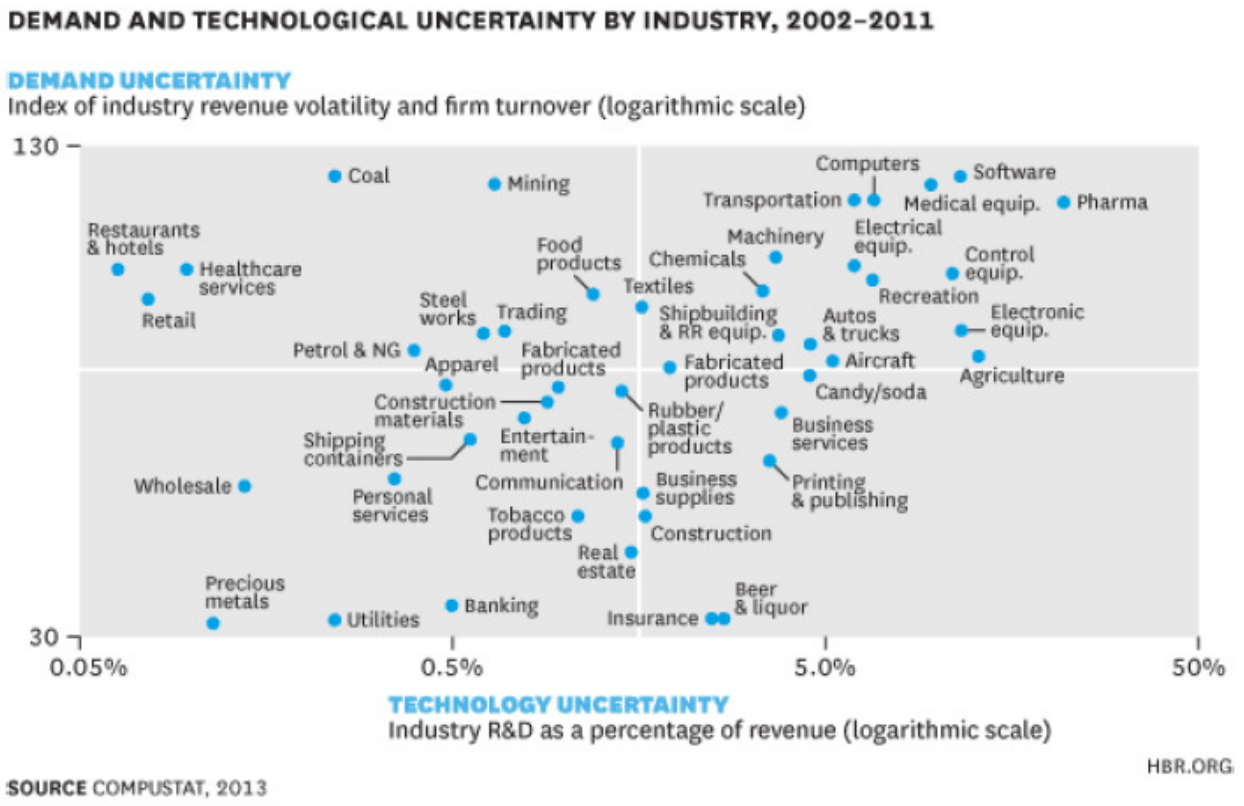


Figure: Mapping of Uncertainties [5]

And so, companies that operate in this base of pharma, computer, software industries. These projects are typically considered high uncertainty or high degree of unforeseeable uncertainty, both in terms of technology and in terms of demand. On the other side, projects that have to do with wholesale, utility, banking, or construction might sometimes be considered lower risk. We can plan better, and we can anticipate what is ahead of us. There is a low level of uncertainty associated with both the demand side and with the technology itself. And so, using such a mapping will help us identify where we are in terms of the spectrum of risks that our project is likely to face. The reason it is so important is that the degree of uncertainty that you face in your project will also indicate how useful some of the planning tools are for you [5-6].

Furthermore, if you think about your project, you think about your industry, and you think about the activities that you are about to engage in. Mapping them on the two axes of ambiguity and complexity will allow you to identify the best way forward (Figure 20). And, if your project has few interactions, it is not highly complex. Tasks are either very sequential, or not many stakeholders involved at the same time with your project. Then perhaps you can use a more straightforward planning tool. And, when there is low variation in your project, low uncertainty, and few interactions, just straightforward planning will suffice. As we introduce more interactions, more complexity, and there is a high degree of risk introduced from the complexity that our project requires, then we would like to add additional ways to planning and executing our project. We would like to encourage fast responses and allow our decision-makers and those

who execute in the field to take more authority and react quicker to changes in their working environment [5-6].

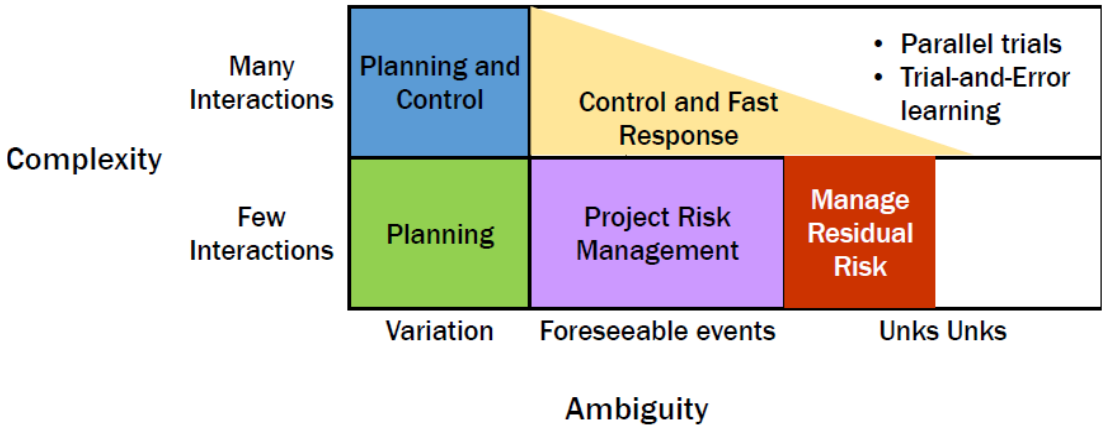


Figure: Mapping Ambiguity and Complexity

If our ambiguity goes up, and we are dealing with still foreseeable events, and we have few interactions, we can come up with a risk management plan that allows us to plan mitigation, contingencies, and reactions to uncertainties as they unfold during the life cycle of the project. The more we move into the domain of highly complex and highly ambiguous projects with unknown unknowns or void a novelty that we cannot articulate and plan for and scope out upfront. Then we must include an opportunity for trial and error, for parallel exploration, in order to be able to learn and to gain information about the area of our expertise. And so, the traditional project planning tools that we have been focused on so far are going to be less useful in those environments. And scoping will require a different mindset. And the execution that will follow will allow for parallel experimentation. And for mistakes to be made and information to be gained. As we move forward, and look, and conceptualize how to plan our risk or how to plan for the risk and uncertainties in our project, remember this mapping matrix. And remember to identify where your project falls in order to come up with the best mitigation plan and the best set of execution rules for your project [5-6].

6.3.3 Risk Management Framework

The first step in properly managing risks in a project is to come up with a risk management plan. This will allow us to identify the key drivers and the impact of the risk and come up with a mitigation or solution to dealing with those risks throughout the execution of our project. There are four steps are included in coming up with a risk management plan (a) Identification of the risk in a project (a) Assessment of the identified risk and prioritize them (c) Planning for the risk (d) Monitoring and Controlling the risk during the execution of the project [5-6].

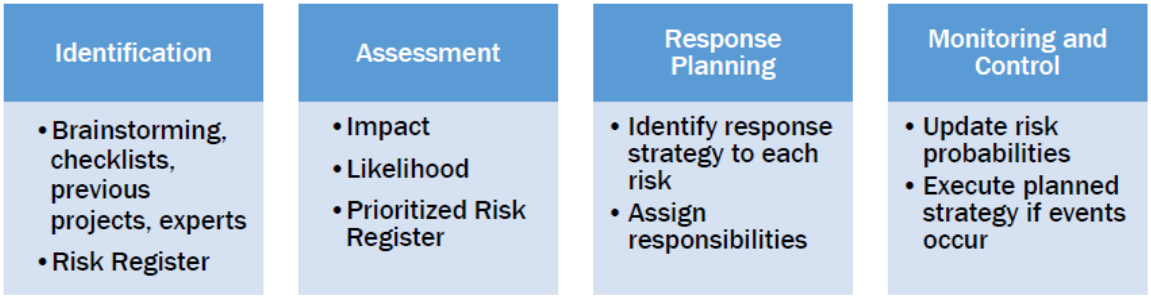


Figure: Project Risk Management Framework

(A) Risk Identification:

During identification, we want to bring together all the different stakeholders or representatives of those groups, individuals with a lot of expertise that have done similar projects or have been in different or inspiring industries before. They will help us come together and come up with a very detailed and thorough list. And identifying all the different risks that could occur throughout the life of our project. Furthermore, we can look at historical data to come up with actual statistics associated with those risks. Also, we can consult with experts and learn from their experience. Read postmortem reports, and come up with a risk register, or a list of possible risks that could occur to our project. These could be risks associated with the scope, associated with the duration, or associated with the cost of our project [5-6].

(B) Risk Assessment:

Next step is to assess these risks, how likely are they to occur, and what is the impact if they do occur? Is it going to end up being minor and not affect our project? Or will it be catastrophic, not only for our project, but perhaps to our entire company? The combination of the impact and the likelihood of a risk to occur gives us some sense of priority. The next question arises: how do we think about the impact and the likelihood of risk? And how do those come together to form a priority and to establish our strategy, in case of dealing with each one of our risks [5-6]?

One accessible way that many people like to think about the assessment component, is to think about a matrix. The matrix provides views of likelihood or the probability that a risk will occur on Y axis, and the X columns represent the impact. How significant will it be if the risk were to occur? The impact might be considered insignificant if it occurs and here might be catastrophic implications, not only to our project and to the success of the project, but to our business overall [5-6].

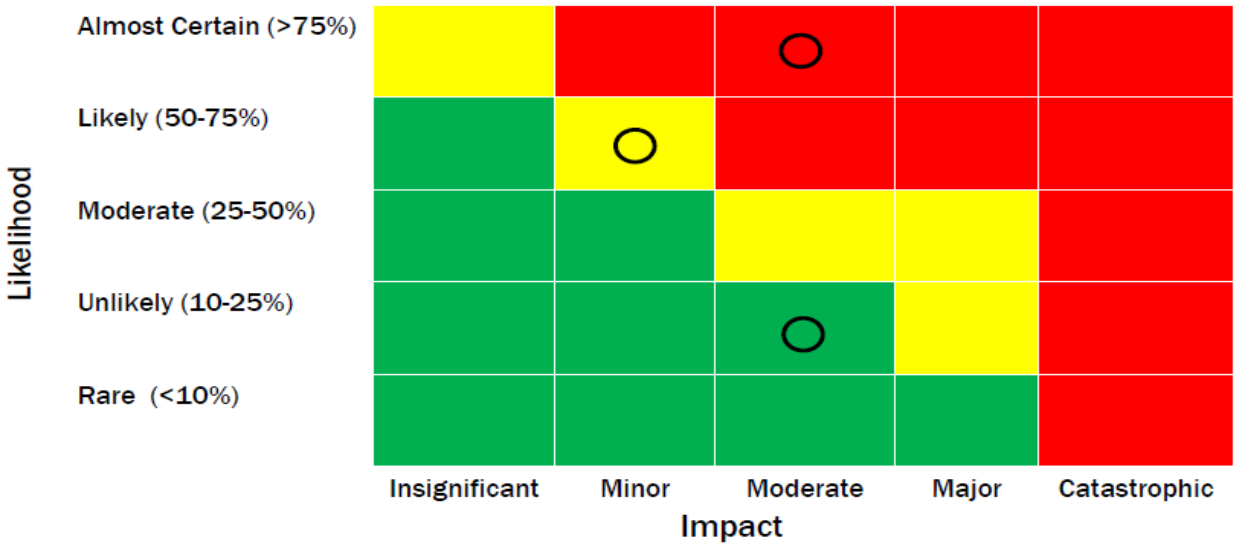


Figure: Risk Assessment Matrix

And so, we take the list of risks that we have identified and map them on to this metric. How likely are they to occur, and what is the impact if they do? We can identify, for instance, in the above example (Figure 22) three events that might occur. One is almost certain to occur with a moderate impact. One is likely to occur, let's say 50 to 70% chance of occurring with a minor impact. And finally, an event that is unlikely to occur, less than 25% chance of occurring with a moderate impact [5-6].

As we map our different risks and our different events onto this matrix, it will give us an indication on how we might want to plan the strategy to deal with those risks. Let's give you a few examples. If we are producing a, a movie, and we are filming in the months of the winter, December, January, February. We anticipate that there are going to be some harsh weather. Perhaps weather that might mean that we cannot film for that day. The likelihood of that occurring is moderate, and the impact to our project might be minor. Meaning, that while there could be delays, they won't be extreme in nature. They might be a couple of weeks, and we can adjust our plan accordingly. We can plan a contingency and add a buffer to our project and plan to film over a few extra weeks to ensure we get all the footage that we need. And to cope with the variable weather that we might face [5-6].

Another example might be constructing and opening an international airport, Denver International Airport, or even more recently London Heathrow Terminal 5. As part of constructing and developing a new brand-new airport, we put in place a new baggage handling system. One of the risks that we might identify is that the baggage handling system might not operate properly. There might be functionality issues with it, and we cannot afford for that to happen. The chance of that to happen, perhaps it is not certain, but it is highly likely given the complexity of the technology. And the impact that it will have on the success of our project might be major, a huge upset on behalf of the passengers and behalf of the airlines. And therefore, if we are faced with that kind of risk which is extreme in nature, we may want a mitigation plan. We

may want to find a way to avoid it completely and remove that event or that risk on the horizon from even occurring. And so, we might develop a backup system in case things go awry. We might consult with other firms to develop parallel attempts at coming up with a new baggage system, all in order to make sure that we've mitigated away that risk from occurring [5-6].

Contingency plans might be helpful in some cases. Mitigation plans could be helpful in other instances. We might also want to think about redundancy, meaning having two parallel paths to ensure that if one goes wrong, we have a backup in place. We could also think about protecting. If the risks are financial in nature and have cost implication, we can look at the financial markets and come up with a, a hedge plan. You know, to ensure that we do not find ourselves vulnerable on the cost frontier [5-6].

Many consulting firms like to take these two dimensions of likelihood and impact and push it a step further in order to come up with the priority score (Figure 23). In order to come up with a score of the risk that allows us to prioritize our risks in our risk register. And so, in order to prioritize, we need to take the impact factor, which we have previously discussed from insignificant to catastrophic and quantify it, assign a certain number. And then we can multiply together the probability of occurrence with the impact that it will cause [5-6].

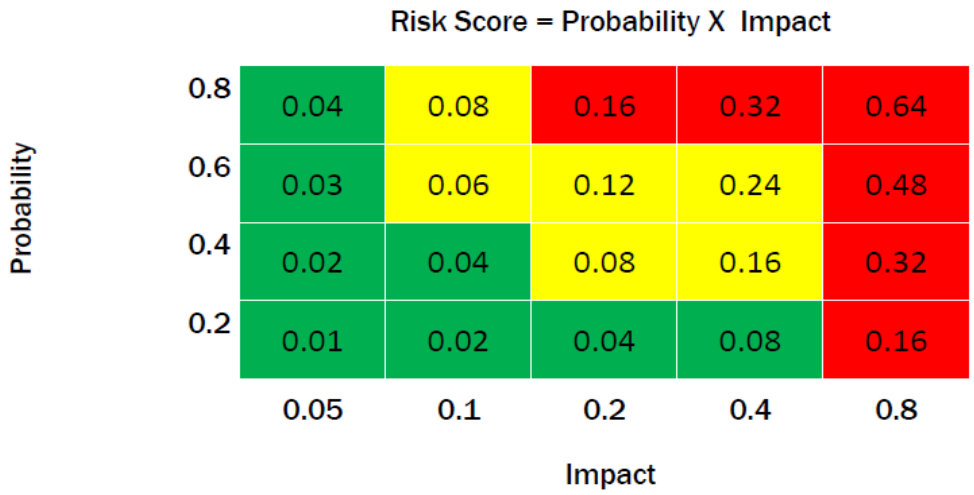


Figure: Risk Probability-Impact Matrix

These numbers naturally give us kind of ordering among the different risks. And therefore, we use the risk score to come up with the priority and prioritize the risks involved in the project. Which are the most extreme, that require attention immediately? And which can we have at the bottom of our list with the proper contingency in place, but they are not requiring ongoing attentions right now [5-6].

(C) Risk Mitigation Plan:

Once we have identified the set of risks in a project and prioritize them, then we can start planning a response. We can identify how we would like to deal with each one of those risks in our Risk Register [5-6].

(D) Risk Monitor and Control:

And perhaps most importantly, we can identify and assign a responsible individual in our firm that will monitor that source of risk. And that will raise the red flag and allow us to get plenty of alert and plenty of warning in advance of a risk occurring. That responsibility, that owner of a risk, will ensure that our project execution is impacted to the minimal degree, from the reality, as it unfolds. Once we move into the execution of the project and we monitor and control our different tasks, then we think carefully about updating our risk plan. The Risk Register is a living document (Table 4). As we progress through the project, we learn more about the nature of the work. We learn more about the world around us. We can update the probabilities and the impact that different risks impose on our project plan or our project itself. And therefore, as the project progresses, we might change the priorities in our Risk Register. And we might focus on the different set of risks later on in the project than we thought up front [5-6].

Table 4: Risk Register Table

	ID	Risk Area	Risk Type	Probability Level	Impact Level	Timeline	Risk Score	Risk Mitigation Plan/Owner
1	OP01	Operations	Schedule	4	4	1	16	
2	W01	Team	Scope	3	3	3	9	
3	E09	Engineering	Cost	3	3	2	9	

Summary of Risk Management Framework

Overall, we have gained a lot of information about risk management. We have identified the risks, we have conceptualized the likelihood that they will occur, the impact that they will have if they do occur. We have come up with a risk score, and we put it all together in the Risk Register. This is a living document that allows us to monitor the progress and to make sure that we are all aware of the sources of risk that we might face. And what is the strategy that we have put in place in case they occur? Most importantly, we identify an owner, an individual that is responsible for tracking how things are going, that will alert us if things have gone wrong. And that will raise the red flag or call for the additional resources when the time comes in order to deal with one of these risks that emerge. In reality, these risk registers might be very big and have many different tasks associated with them. They might require a lot of attention, but they are a vital part of our planning process. And they are crucial for the success of our project because they allow us to walk into the project. And start executing the tasks immediately, knowing that we have a plan to follow in case things go bad [5-6].

6.3.4 Time or Schedule Risk Analysis

In this section we are going to dig into a specific source of uncertainty, and we are going to think about the risk and the variability that we may face in the project schedule. We know that there is going to be variability to the duration of our project. How can we be more specific and learn more about it in order to plan properly [5-6]?

In order to do this let's look at the startup example as we have discussed in the project scheduling section. In the work breakdown structure of the startup project we have identified eight tasks that are part of our project. And, we know which tasks proceed which and we know an initial estimation for the duration. How long will each one of our tasks take? What is our best guess or our estimate for their duration [5-6]?

And, we were able to identify the overall expected project duration using network diagram and critical path analysis. What is the most likely time that we think our project will be completed by? We thought that it would, it was going to be 12 weeks. But what happens if the creative task, which is critical, takes longer for instance seven weeks? What if it finishes sooner, in three weeks? What is the actual impact to my overall project duration? And more than just the creative task, what about variability in the other tasks in my project? Each one might not be exactly the best guess, or likely scenario. It could go sooner, or it might take longer. What impact does it have on the overall project duration [5-6]?

We can actually model this and think about the ranges and see what impact each one of the tasks shifts in terms of optimistic scenario versus the pessimistic scenario. Let's take it further, in order to start, we are going to develop a list of minimum durations and maximum durations for each one of our tasks. In this case, we assumed that for instance the creative might range from three all the way to seven weeks. We think that perhaps the strategy is not going to be two weeks most likely expectation. But we think that it might vary between one, all the way up to the pessimistic case where it might take six weeks and so on and so forth [5-6].

Table: Schedule Risk Analysis

Activity #	Description	Predecessors	Min Duration	Likely Duration	Max Duration
1	Creative	-	3	5	7
2	Strategy	-	1	2	6
3	IT	-	3	4	5
4	Fundraising	1	2	4	8
5	Marketing	1,2	1	2	6
6	Sales	1,2	1	5	7
7	Finance	5,6	1	2	7
8	HR	4,6	1	1	1

We can identify for each one of our tasks what is the possible variability? What is the possible variation in terms of the length of time it will take us to execute? In some cases, for instance, in the case of the HR task, we might also believe that there is no variability whatsoever. If this is a task that we are outsourcing, or if we are contracting a company to deliver a specific piece of work for us, it might be that there is no variability in that task, and we know for certain that it's going to take one week to deliver [5-6].

Once we have come up with these estimates we can then explore, for each one of these tasks while holding the other at their most likely case. What is the swing on our project duration? And the best way to gain insight as to that impact is to look at the tornado diagram (chart). The tornado diagram is a visual representation of the variability in the overall project duration when we change each one of my tasks from its minimum to its maximum duration (Figure 24) [5-6].

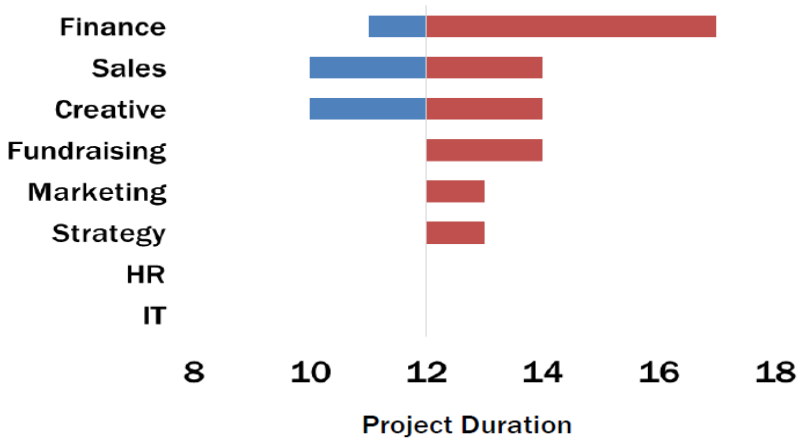


Figure 24: Tornado Diagram of Risk Evaluation

Let' take the finance task which appears on the top of the tornado diagram. The finance task shows the swing with the blue side going down from the line in the middle, which is on 12 weeks, it goes all the way down to 11 weeks. This is a project duration of 11 weeks and it goes all the way up to 17 weeks, in red color, the most pessimistic or the worst-case scenario. This implies that when the finance task swings from its earliest completion time to its maximum completion time the impact on the project duration can vary from 11 weeks to 17 weeks. There is a variability of six weeks and that is caused by the variability in the finance task itself [5-6].

Furthermore, the chart is called the tornado, because typically it is ordered from largest all the way down to the smallest and that gives us an indication as to the biggest ring versus the smallest ring. As we can see, HR and IT tasks do not have an impact on the overall product duration. And so, if we are trying to focus our attention on which tasks give us the most source of uncertainty, in terms of our overall project duration, we can focus our attention at the top of the chart, or those tasks that are closer to the top, as supposed to those closer to the bottom. So, the tornado diagram identifies and prioritizes some of these tasks for us in the context of managing

our project duration and getting a better sense and feeling more comfortable with any deadline that we set to our project [5-6].

Therefore, the first step associated with conducting an in-depth risk analysis for our project schedule is to first ask and collect three estimates for each duration of each task. Ask for the minimum duration, ask for the maximum duration, and try and identify the most likely. Sometimes it is useful to look at historical data to come up with that information. How quickly were tasks completed in the past when we have done something similar? Or what was the maximum it ever took us to get some sense of the maximum possible duration? You could ask multiple individuals who have experience in that area, how long did it take you in the past? What was the quickest you were able to achieve a certain task? Or what was the worst that ever happened to you? Or, we could ask individuals from different departments, with different expertise, to give us different perspectives and put that information together and combine it to come up with three points or three numbers to estimate a range. And over all we want to start thinking about our project completion as a range and not as a specific point. We know that our project is not likely to complete exactly in the 12 weeks that we identified up front. We know there is going to be a swing, it is going to be anywhere from maybe 11 to 17 weeks. And so, we start using those terms in order to describe our project and not guarantee a completion by, for instance, 12 weeks [5-6].

So, when we looked at the variability and tasks durations, we considered how much each task's variability affects the overall project duration when holding the others fixed, or constant at their most likely duration. But in reality, everything is going to be uncertain, and task's durations are going to move all over the place. And we're not going to have one source of variability with everything else staying constant. So, our next step is to move beyond changing each task separately. But we are going to throw all of our variability, and we are going to use tools to visualize and to conceptualize what the overall impact is going to be on our project duration. When we account for the interactions and the dependencies, the links, and different sources of variabilities that we have in our project [5-6].

We are going to walk through the startup project example and see what steps we might take in order to come up with a complete probability distribution for our overall project duration. In the startup project example, there are eight tasks, and we have our three data points of each task's estimate a duration. We have the minimum duration it could be, the maximum duration it could be, and some estimate of the most likely duration as well. Now we are going to allow our tasks to not only obtain their extreme durations, for instance, three or seven weeks for Creative task. But we are going to allow Creative task to move anywhere between three to seven weeks duration. And we are also going to articulate something around how likely we believe it is that Creative task will take a certain length of time. We are going to do this for all our tasks and assign a probability distribution for each one of the task durations allowing them all to vary simultaneously and preserving the precedence relationship will tell us something about the distribution of the overall project duration [5-6].

The first step to do this is to replace the estimates with an entire probability distribution. There are specialized tools out there, Crystal Ball or @RISK or many other alternatives. Using these specialized tools, we can replace the minimum and maximum durations with an estimated duration that is coming from an entire distribution. In this case, we use the triangular distribution where we need to assign a minimum and a maximum duration which we have collected for this specific task. And each one of the values in this range can be realized and It is a possible outcome that we might see (Figure 25) [5-6].

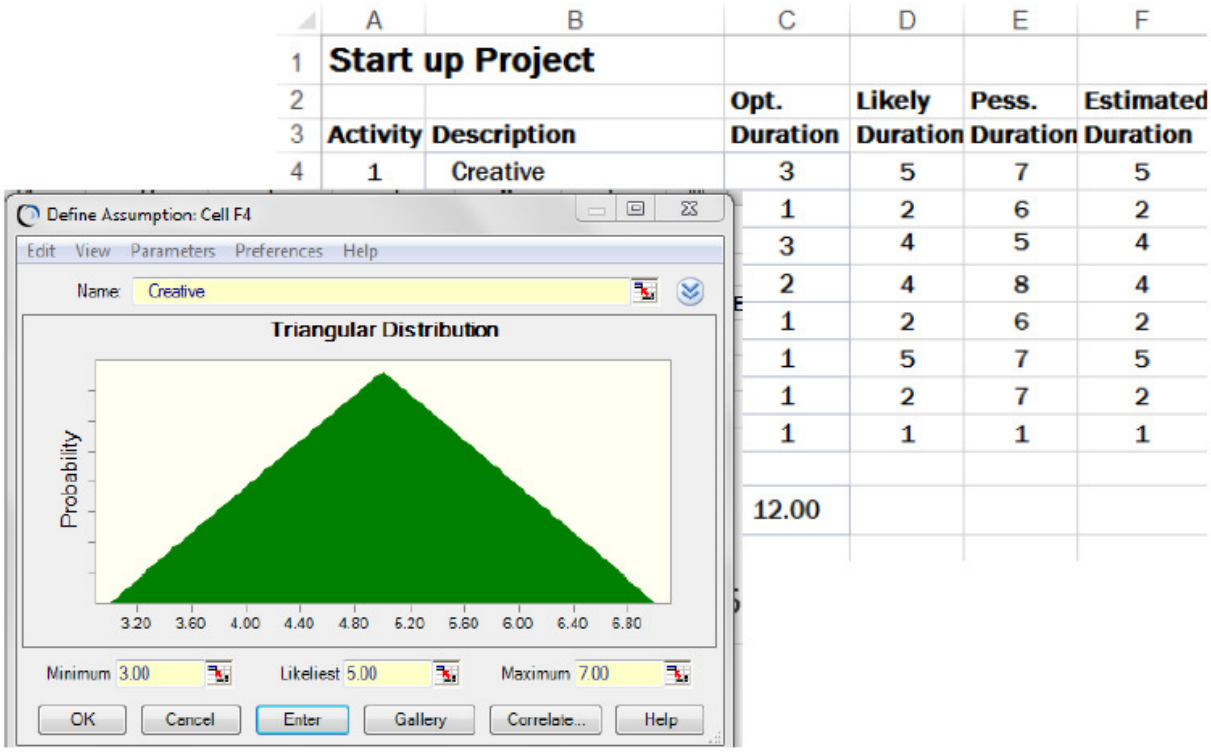


Figure 25: Triangular Distribution

But we are telling the tool that it is more likely to be centered. It is more likely in this case to be somewhere around five. So, there is a higher probability to obtain values somewhere in the middle. We could assign this distribution, a fairly intuitive and straightforward distribution, to each one of the tasks separately. And when press a button, we will see a different possible outcome for a future project. Doing this 10,000, 100,000, even a million times, will simulate a million possible realities that our project might face, a million instances of our future project. And if we look at the patterns that emerge, we will be able to conclude what the average, or expected duration would be. And also, we might be able to learn something about the likelihood of different values. And so, once we throw it all into the tool, and we assign a distribution for each one of my tasks, we run it a million times with a million potential futures. we see the cumulative distribution chart (Figure 26). It tells the likelihood that the overall project duration will be a certain number of weeks [5-6].

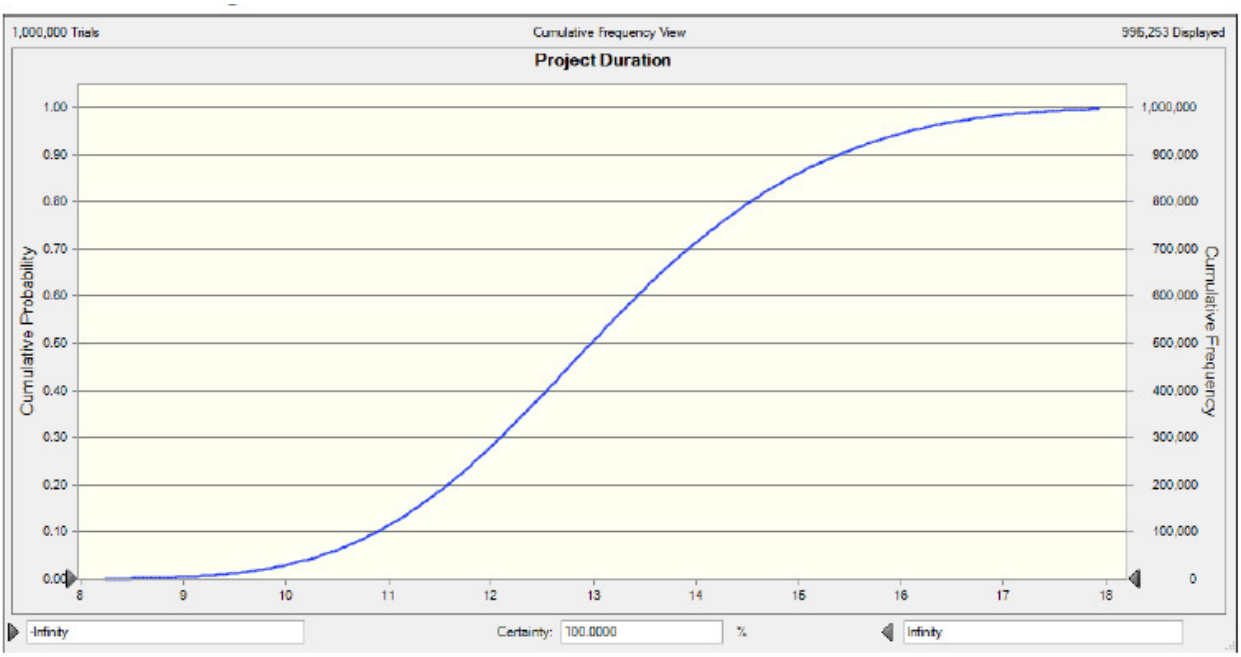


Figure 26: Probability of the Project Duration

The chart shows the 12 weeks which we initially calculated using our critical path analysis. there is somewhere around 25% chance that the project will be completed up to 12 weeks. There is closer to 75% chance that it will take us longer. And, if we go all the way up we estimate somewhere around 95% chance of completing this project within 16 weeks. And so, if we are looking to set a completion date or to report how long it might take us to complete a project, we can choose a confidence level. We can say, we are 95% certain that we will complete the project within approximately 16 weeks. Therefore, we can be more accurate, given the information that we have and given the variability that we know that exists in my project [5-6].

Another way to view it is looking at the frequency or looking at the probability. What is the chance that we will find ourselves within a certain range? Or what is the chance that the project duration, the project will be completed within a certain number of weeks? We can use the tool to come up with confidence ranges. In this case, we are 90% certain to finish between 10.5 to somewhere in the region of just over 16 weeks. we can convey our confidence level. We are 90% confidence that this is the range within which our project will be completed (Figure 27) [5-6].

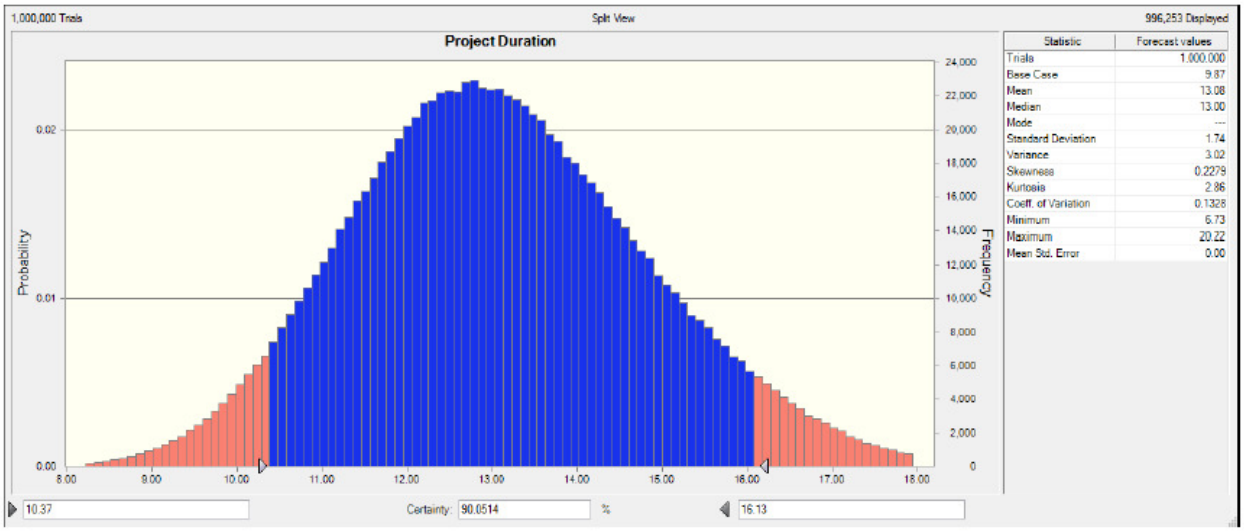


Figure 27: Confidence Level of Project Duration Estimation

What we also have to remind ourselves is that as we start and begin on a project we acknowledge the fact that the, and variability will occur and some of tasks will take shorter and some of them will take longer than our estimations we could find different critical path. We are no longer going to be talking about a specific single critical path. And so, we are no longer going to be asking which are the tasks that are on the critical path, because the critical path might vary in reality, and might vary as we move and progress through the project. Now, we have a tool that allows us to model the variability in each task, we can also ask the question, what is the chance, or how likely is it, that each one of our tasks is on the critical path? No longer certainty but a degree of likelihood, or some probabilistic statement [5-6].

And, when we simulate the startup project example in the tool, we can obtain the following information that there is 93% chance that the Creative task will be on the critical path. There is about a 70% chance that the Sales activity will be on the critical path, and there is virtually no chance that the IT will form part of our critical path (Figure 28). And so again we can use our critical index, or the likely that the project task is on the critical path, in order to prioritize our attention. And to tell us where to focus, and which task and which resources might require our attention to make sure that we complete our project within the range that we anticipate to complete it on [5-6].

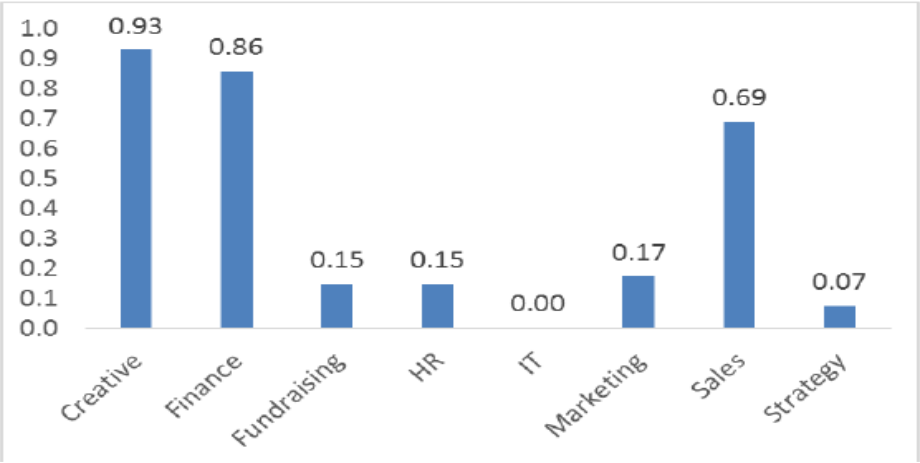


Figure 28: Probability that an Activity is Critical

6.3.5 Budget or Cost Risk Analysis

In a similar way to the fact that we thought about the schedule variability and the uncertainty that exists in our project completion time, there is also a certain, uncertainty related to the cost of our projects. Analyze the risks and variability associated with how much each task will cost us, and what is the overall cost of our project. The budget or cost suffer from the same type of variation. It could be external effects, such as weather or regulation that might imply that we will have to pay more to execute and to complete some of our tasks [5-6].

We might need extra resources. And therefore, we can think about the variability that exists. And we can start conceptualizing what kind of contingencies we would like in place. How are we going to ensure that we do not exceed our budget? And how do we plan accordingly to cope with any uncertainty that we find ourselves facing. A simulation analysis presents the uncertainty that exists in our project budget. But, just a word of caution where we put the contingencies? Do not put the contingencies on each and every task even though the tasks themselves could be variable in terms of the duration. Allow the project manager to control that contingency associated with the budget to give the project manager maximum control on where to spend the extra funds and when. And the project manager has that knowledge and the project manager can balance the objectives and the priorities associated with the project objectives in order to best utilize the contingency budget [5-6].

Furthermore, taking similar steps to the ones we took with the schedule, we can come up with a probabilistic forecast for how much the project would cost us? What is the budget that we need? and how confident are we that we will actually meet that budget? We can look at an entire distribution. What is the likelihood our project costing so much? We can talk about ranges.
In Figure 29, we see about a budget for a project that can vary between 346,000 to somewhere just under $354,000. This is our 90% confidence range. We are 90% confident that our project budget will be somewhere within that range [5-6].

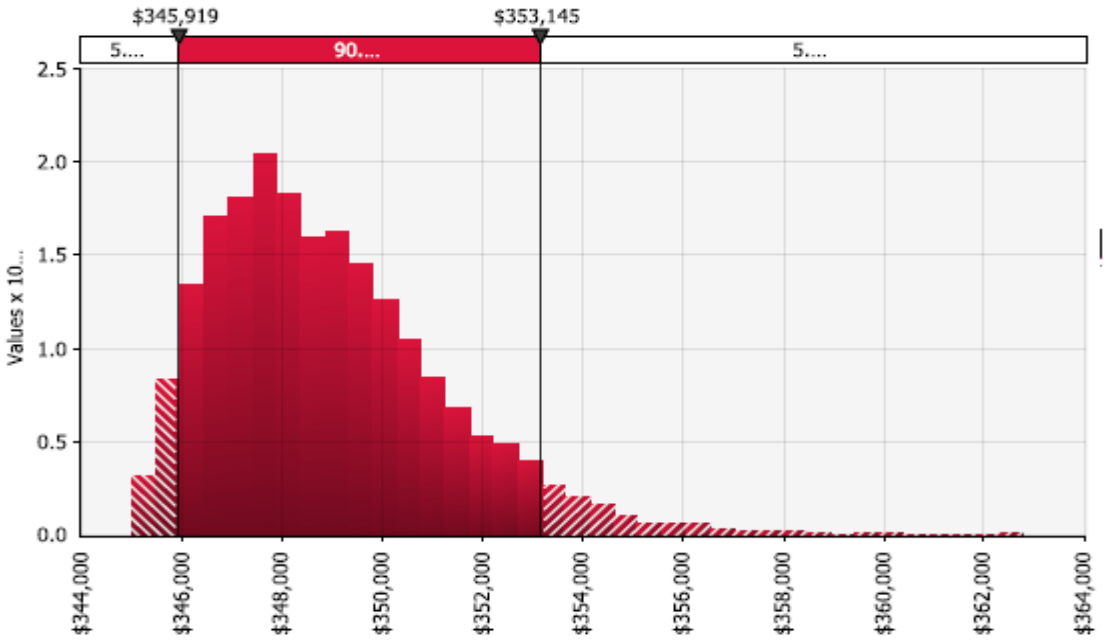


Figure 28: Budget Simulation Result

Therefore, we can choose our overall project budget, and allow for extra contingency. Perhaps, we set it somewhere in the region of $350,000. And we give ourselves 10 to 15% contingencies to make sure that we are highly confident that we are going to meet this project budget, and not exceed it. Perhaps, we will come in under budget, but we know that we have the available resources if we need them [5-6].

6.3.6 Ambiguity Risk Management

We have seen how we can consider uncertainty and variability that we expect in our project duration and our project cost. We have looked at a project plan and we have identified how variability in the individual task completion times might affect our overall project duration. But what if we do not have that information? What if we cannot estimate the durations at all, let alone three estimates for the durations? And what if we cannot even estimate which tasks to include in our project? What will it take to produce a product? What will the product look like? If we are in the sphere of highly ambiguous projects, an environment with many unknown unknowns to us, new technology, new customers, new market and new opportunities, we may not be able to come up with a plan, let alone conduct some sort of risk analysis that we have done so far. So, what do we do [5-6]?

Activity #	Description	Predecessors	Min Duration	Likely Duration	Max Duration
1	Creat				7
2	Strat				6
3	IT				5
4	Fundra				8
5	Marke				6
6	Sale				7
7	Finar				7
8	HR	4,6	1	1	1

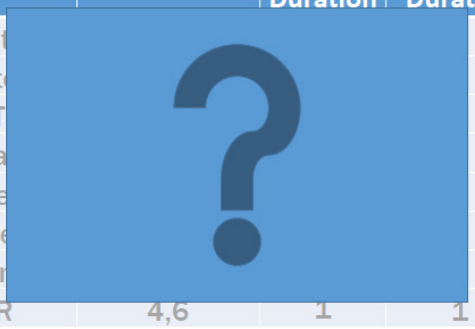


Figure 29: Unforeseeable Uncertainties in Projects

Well, luckily there are tools and there are concepts that we can introduce, and we will be discussing, that allow us to deal and to think about how to work in highly ambiguous project settings. Instead of limiting ourselves to a specific plan, and thinking about the steps in a very specific manner, we are going to open the door to conceptualizing a whole roadmap of opportunities, a whole swing of possible outcomes that we might find ourselves and our product might accomplish at the end of the day. And instead of limiting ourselves to very specific tasks and duration, we are going to allow our individual employees, the developers, the engineers to conduct the work themselves and to find out more information that will help us adjust our plans for the project [5-6].

We will progress in short and smaller durations on a near term basis. We will make little commitment as to how the future might unfold and we will be comfortable with the idea that we might find ourselves in many different final scenarios down the road a year and a half out. In the meantime, we will empower individual teams to do short iterations, trial and error, to make decisions for themselves and to come back to us and report successes and opportunities that failed, in order for us to shut some doors and open the new ones [5-6].

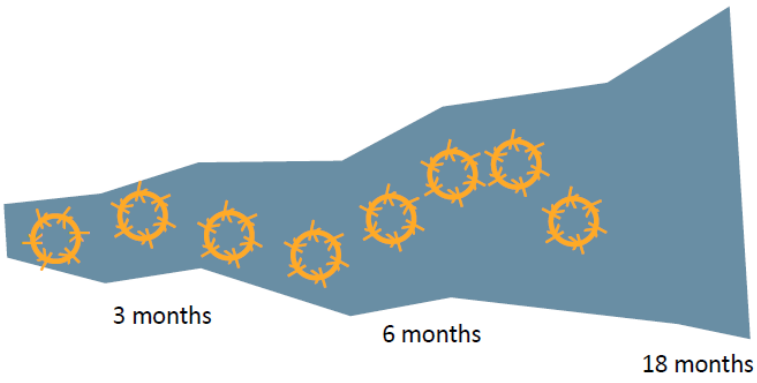


Figure: Roadmap instead of Plan [5]

This mindset moves away from the rigid work with the critical path. However, the critical path is important, and higher-level senior management might still like to look at it and plan accordingly a company vision and a company strategy. And, there needs to be a communication between the teams on the field that are doing short iterations and quick information gain on each sub-level task, to communicate back to the higher-level management what they have learned and how this fits with the vision of the progress of the project for the whole organization [5-6].

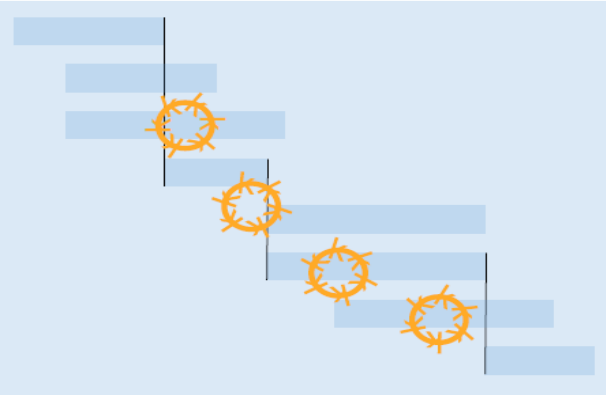


Figure: Trial and Error Learning through Iteration [5]

Working in highly ambiguous settings is important to recognize how much, how important it is to gain information, to have our developers and to have those engineers out there working and progressing and gaining knowledge all the time [5-6].

EXERCISE – 9 : Risk Analysis

Objective: Perform project management activities: effort estimation, WBS, activity planning, resource allocation, risk analysis

Tools/ Apparatus: Microsoft project

Procedure:

1. Identify all the potential risks in your project development and provide a mitigation plan

Evaluation: Your work must address the following questions:

1. Does the risk analysis identify the existing and possible threats that the project might encounter?
2. Does the detailed risk estimation compliment with the risk identification?
3. Does the risk management plan cover all the identified and unexpected risks?

Chapter 7: Execution of a Project

And so, in the lifecycle of a project, and focusing on the execution stage, we tend to think about the following dimensions. How during the execution are we going to monitor our progress? How will we communicate amongst the team and outside externally to the other stakeholders? Who will be doing the reporting? How will we correct and take action when something isn't progressing as planned? Here are some questions that will help us proactively think about the execution of our project to ensure or to increase the chances of a smooth execution [5-6]

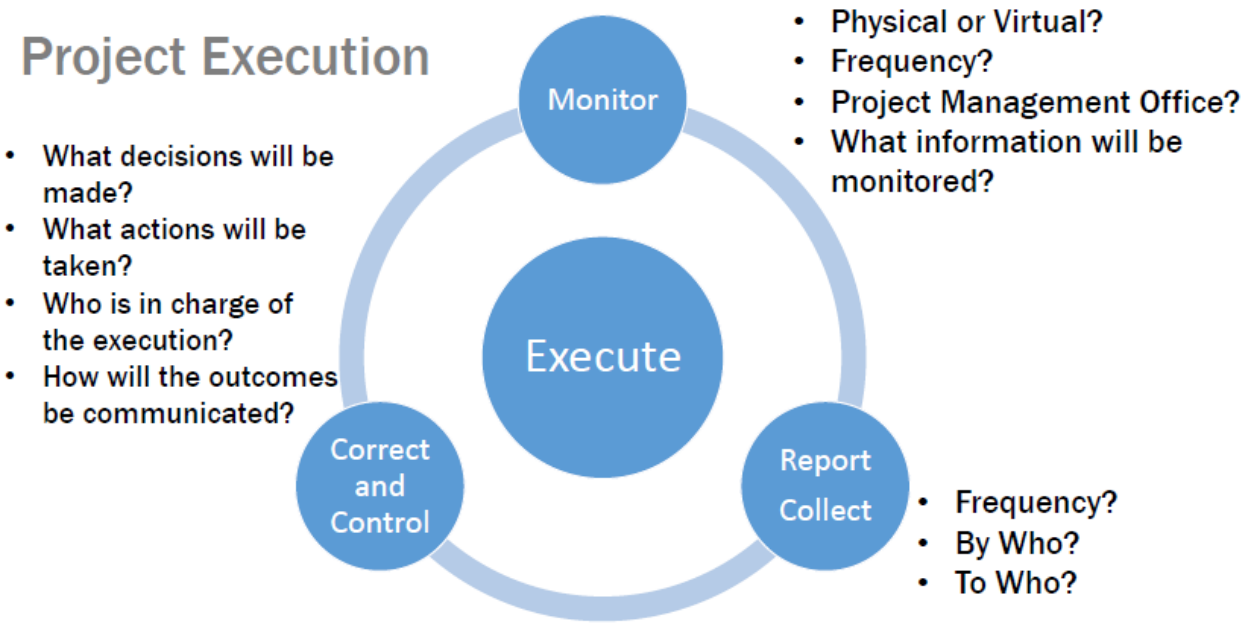


Figure: Project Execution

7.1 Project Execution Monitoring

As we think about the execution, we think about monitoring. How will the monitoring be done? Will it be physical or virtual? Is our project manager in the same location as the teams? Do they work together in such a way that they can actually view the work? Or perhaps is a virtual system for collaborative platform going to be used. And how often should we be looking at it? Are we going to look at our system every day and expect updates to be made in real time? Or are we going to be looking at it on a weekly basis or bi-weekly basis? Is there a central project management office? Does that office aggregate information from multiple projects in an organization, and do they share the same resources? Does the project manager get support from that office and have others on his team who can help him work through any problems that come up? What information will actually be monitored? Are we monitoring time? Are we monitoring

the actual costs? And how are we going to be monitoring functionality of the product as we execute and develop the product [5-6]?

7.2Project Reporting and Communication

In terms of reporting and collecting the information how often are we expecting updates reports? Are we expecting our team to call us every day and to tell us how, how they are doing? Are we scheduling a weekly meeting to talk about the execution and the progression of the project? Who is going to be in charge of these meetings? And who is going to attend them? Are only the individuals associated with the tasks going to show up? Or are all the stakeholders or related parties going to show up in order to share and learn from the execution of our project [5-6].

7.3Correct and Control

Once we have collected the information and the reporting has been done, then what? How do we correct action if something has gone wrong? If a red flag is raised and we are behind schedule, what do we do? So, it is important to identify what sort of decisions will be made. Do we have some flexibility in terms of the scope? And who can make the call on reducing the scope? Who has the ability to hire additional personnel or to outsource and to seek an expert to come and help us solve a problem? And assuming that we have reached a conclusion that we need to take action, and we have taken a corrective action and gone out and found an expert to help us. How will we communicate to the rest of the team where we are heading from here? How will updates be communicated in terms of change of directions and an alternative path that we are all going to have to follow from this point onwards [5-6]?

These are the types of questions that are on our mind as we work through the execution, as we start the execution, and throughout the entire execution of the project. And we will see several different ways in which this can be done [5-6].

7.4Earn Value Analysis (EVA)

The Earn Value Analysis provides us the information about how well we are doing in a project given the status update that we received. We are going to look at the type of information that we might want to consider reporting, and what type of analysis we can do in order to determine if we are in trouble or if we are on track. Consider the startup project example that includes eight tasks. For this project, we have come up with a set of precedents and with estimates associated with the duration. Given the analysis that we have conducted we know that we are expecting our project to be completed in 12 weeks. And that we have some cost associated with each task. In total, we expect the project to cost us in the region of $8,500. Specifically, we think that the creative task, identifying the creative features and writing up the creative part of our project will take five weeks and it will cost us $1,000. The strategy, developing a strategy in a business plan

for our firm will take us two weeks and will cost us about $500. And coming up with the correct IT system and putting it in place, will take us four weeks and will cost around $3000. The rest of the tasks as, as follows, we have information about them, but let's consider those three tasks and think about what happens at the beginning of this project. Assume we started off our project on the 19th of January. We have our Gantt chart. And we know that our critical tasks include the creative task, the sales, and the financing [5-6].

Table: Startup Project Total Duration 12 Weeks, $8500 Budget

Activity #	Description	Predecessors	Duration (Weeks)	Cost
1	Creative	-	5	$1000
2	Strategy	-	2	$500
3	IT	-	4	$3000
4	Fundraising	1	4	$500
5	Marketing	1,2	2	$2000
6	Sales	1,2	5	$500
7	Finance	5,6	2	$750
8	HR	4,6	1	$250

Let's assume that two weeks have passed. What happens two weeks later, on February 1st, we call up and we ask our individual who is executing the project, how are things going? Let's assume that all three tasks, creative, strategy, and IT have indeed started. The progress is as follows. Creative is about 25% complete and there are 75% of the work yet to be done. The strategy is 75% complete. And in terms of the IT, we are about a quarter of the way. We have three-quarters left of the work to do. We also know that we have actually spent some money. We have spent $500 on the creative activities. We have spent $300 on the strategy, and we have spent around $1000 in IT in hardware and in software [5-6].

Table: Startup Project Execution Reported on 1st February

Activity #	Description	Duration (Weeks)	Cost	Work Completed	Money Spent
1	Creative	5	$1000	25%	$500
2	Strategy	2	$500	75%	$300
3	IT	4	$3000	25%	$1000

So where did that put us? Are we in trouble? Are we on track? How can we tell? What are we going to measure in order to tell us how well we are doing and what we should do in case there is a problem? Well the first step is to update our plan. We can use many different project management software tools out there will allow you to update the progress in the plan [5-6].

Following Figure highlights how much we have actually completed the percent of the work done. For example, we have completed so far 25% of the creative 75% of strategy, and 25% of the IT.

Our Gantt chart can be updated accordingly to show us in a visual way that we are working through these three tasks shows in blue bar in this diagram and the red line gives us a progress update. The diagram also shows a zig, zagging back and forth. It shows us that we should have been further along. We should have been where the green line in the diagram at this moment and we are actually behind the schedule a little bit [5-6].

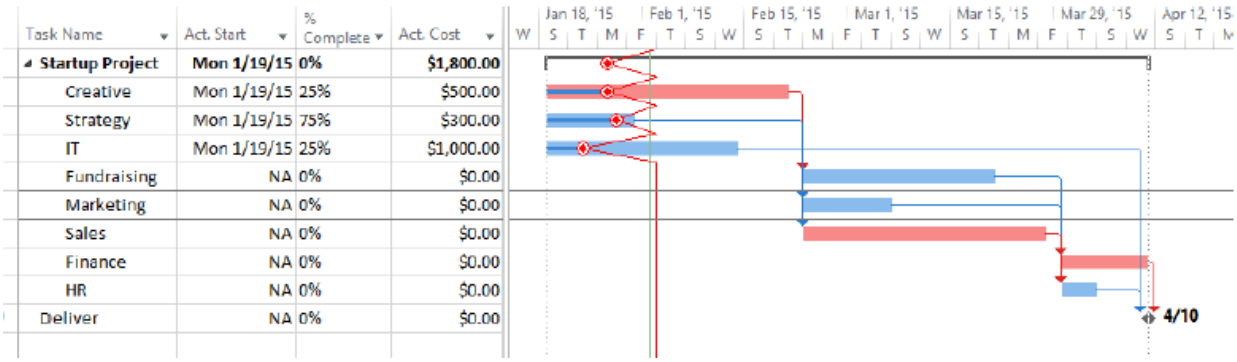


Figure: Startup Project Status Report

And so, what is the next step? How bad is it? We see that we are behind on the creative, which is critical. But we are also behind on the strategy and IT, which are, which neither of which are critical. So overall, how are we doing? Well in order to answer those questions, we're going to dig in a little bit deeper and do some calculations. Consider the following calculations [5-6].

First we know, and we are going to focus just on the creative task. We are 25% complete, and so far we've spent $500. We know that if things were going as planned, we are two weeks into the project. By now we expected to do two weeks of work out of a five-week task. We expect it to be about 40% done. We also know that in terms of the actual work, the progress that was made, we got a report that we are about 25% done. We had a budgeted cost for the entire task of $1000. So far, if we were progressing as planned, we would have completed 40% of the work, and therefore, we expect to have spent 40% of the budget, implying that our planned value, what we planned to spend at this time was $400. What we have earned, what our work has earned, what the work that we actually completed has earned us, 25% of the budget would imply that we have earned $250 worth of spending. But our actual spending was $500. And so, there seems to be some variance here. We can identify several sources of variance. We can identify that we are about 0.75, just under a week behind where we should have been at this point in time. We can also see that we are about $150 behind our scheduled work. And we can also spot, looking at the difference between the 250 and the 500, that we have about $250 worth of variance in terms of the actual spending that we have required in our project [5-6].

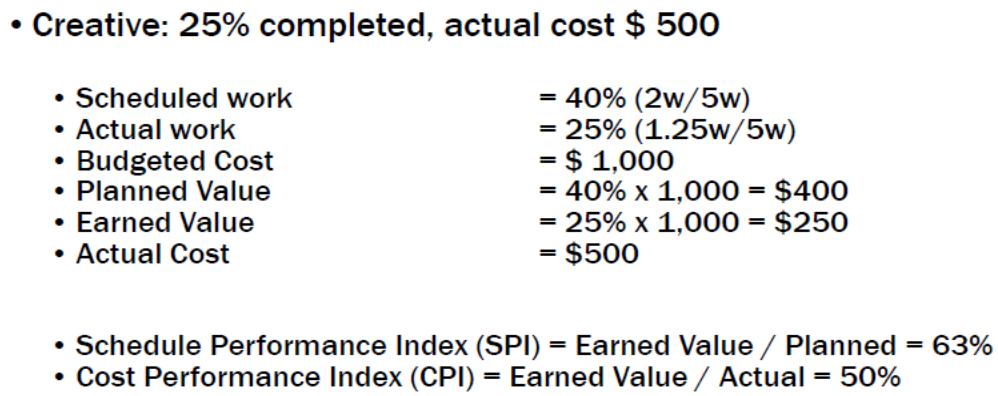


Figure: Earn Value Analysis of Creative Task

Schedule Performance Index and Cost Performance Index (SPI & CPI)

Here are two other measures that we can look at, two indices that tell us how we are performing on the schedule side of things and on the cost. From the schedule perspective we are going to be looking at the schedule performing index. We are going to be looking at how much we have earned for the work that we have done versus what we plan to do over this time. So, in the two weeks we planned to do 40% of the work we plan to spend $400. But we have actually earned from the work that we have completed $250. And so, our scheduled performance index is 0.63, or 63%. As a benchmark, 100% or 1 would have been on track, that means our work was performed on time. And right now, there is some indication that we are behind. We have not completed the amount of work that we thought we would at this point in time [5-6].

From the cost performance perspective, here we can also measure a similar component, or a similar index. We can look at our earned value, how much we have actually earned in terms of the work. And what was the actual spending that we have reported. And so, 250 of the work that we have actually achieved divided by 500 gives us around a 50% completion rate. Meaning our cost is about 50% over budget. Again, a 100% index would have indicated that we are on budget. And so, these two indices are helpful as we track our project and to check all the time, are we working as we planned and progressing on schedule, and are we spending the correct amount of money? We can look at it visually in a graph over time, and we can identify bands in which we need to raise some flags and alert individuals around the organization that we are either running behind in terms of the work or behind in terms of the cost and spending [5-6].

Measure	Equation	Comment
Earned Value (EV)	EV=% work completed × baseline cost	BCWP, Budgeted cost of work performed
Planned Value (PV)	PV=% work scheduled × baseline cost	BCWS, Budgeted cost of work scheduled
Actual Cost (AC)		ACWP, Actual cost of work performed
Schedule Variance (SV)	SV=EV-PV	
Cost Variance (CV)	CV=EV-AC	
Schedule Performance Index (SPI)	SPI = EV/PV	SPI < 1: Project is behind schedule SPI > 1: Project is ahead of schedule
Cost Performance Index (CPI)	CPI = EV/AC	CPI < 1: Project is over budget CPI > 1: Project is under budget
Cost-Schedule Index (CSI)	CSI = CPI × SPI	CSI < 1: Project is not healthy CSI > 1: Project is healthy

Figure: Earn Value Analysis Equations

And so, if again, an on-track measure is a measure of 100%, a ratio that is exactly 1. If we fall beneath the line, the dotted line which is 100%, we want to raise some flags. We are over budget and we are behind in terms of our schedule. The lower that number, the more severe the situation is. The lower the number, the more over budget we may be. Or the later we are, and we had not done as much work as we have hoped. But why might we want to flag situations in which our industries shoot up? What happens if we get some report around progressing 120%, meaning that we are ahead of schedule and that we are under budget. Here too we may want to raise some flags. Are we neglecting some component of the scope? Have we missed out some, some critical tasks, or some crucial tasks in terms of the functionality that are expected to be delivered at the end of the day? Are we not performing the tasks or executing properly, and are we may be cutting corners or rushing too quickly? And so, any deviation from that band of approximately 100% on both of these dimensions might raise some red flags [5-6].

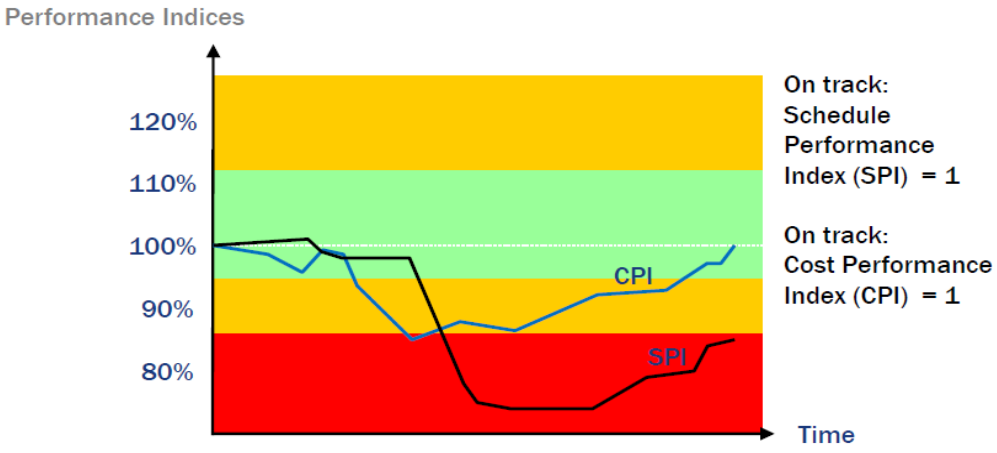


Figure: SPI and CPI Performance Indicator

This framework is used widely in the industry. It was developed by the Department of Defense. And today, in the government in the U.S. in particular, but internationally too, these types of measures are used in an ongoing basis. To report and to collaborate and to monitor the execution of projects. Individuals subcontractors or even within the same organization will typically talk about how well they are doing on the schedule performance index and on the cost performance index. The software tools that we use are helpful, as well. They will give us the analysis. So, they will do the calculation for us. And we can use its built-in functionality to look at how well we are doing. We can input the progress, and we can look at our variances and our performance indicators [5-6].

In summary, the three tasks that we have made some progress on, and we see that we can update our schedule to view how the three tasks, Creative, Strategy and IT, are doing in terms of our cost performance index and our schedule performance index. We see here, for instance, that specifically the strategy task. Seem to actually be under budget. So far we have not spent as much money as we thought we would for the amount of work that we have done (Table 8). Overall, as the project, are we in trouble? Do we need to make critical changes to our project to the execution? Do we need to find additional resources to help us get things back on track [5-6]?

Table: Earn Value Analysis of the Startup Project

	Task Name	Planned Value - PV (BCWS)	Earned Value - EV (BCWP)	AC (ACWP)	SV	CV	CPI	SPI
1	**Startup Project**	**$2,400.00**	**$1,375.00**	**$1,800.00**	**($1,025.00)**	**($425.00)**	**0.76**	**0.57**
2	Creative	$400.00	$250.00	$500.00	($150.00)	($250.00)	0.5	0.63
3	Strategy	$500.00	$375.00	$300.00	($125.00)	$75.00	1.25	0.75
4	IT	$1,500.00	$750.00	$1,000.00	($750.00)	($250.00)	0.75	0.5
5	Fundraising	$0.00	$0.00	$0.00	$0.00	$0.00	0	0
6	Marketing	$0.00	$0.00	$0.00	$0.00	$0.00	0	0
7	Sales	$0.00	$0.00	$0.00	$0.00	$0.00	0	0
8	Finance	$0.00	$0.00	$0.00	$0.00	$0.00	0	0
9	HR	$0.00	$0.00	$0.00	$0.00	$0.00	0	0
10	Deliver	$0.00	$0.00	$0.00	$0.00	$0.00	0	0

Using the tools and software packages, we can also update our project plan. We can assume that we are going to continue moving forward in the same pace that we have move so far. Giving us a re-calculation of our project, we can see that our critical path has stayed the same. Still the Creative task, the Sales task and the Finance task are still critical in this Gantt chart. We can see

the delay become real. We can see that overall, we have an additional, just under 0.75 week added to the expected completion of our project. And given the information that we have so far, we see that we will be going just a little bit over budget that we had anticipated. Instead of $8,500, we are looking more like just under $9,000. And so, what is our conclusion? Are we in trouble? Perhaps we can afford to be delayed by just under a week. Do we need to make sure that from now on everything takes place properly and everything is executed as planned? Any additional delays later in the project will just continue to push our deadlines further. And so, the fact that we have just started, we are two weeks into our project, and we are already a little bit delay, might require some attention. We know that we have an additional $425 in our expected spending. Perhaps we want to think about how we will find that already at this point in time or maybe find opportunities to save later on in, in the life of our project. But now we have a good idea on how things are progressing and where we stand in terms of our cost and the schedule of our product [5-6].

EVA Exercise: Assume you are a software project manager and you've been asked to compute earned value statistics for a small software project. The project has 56 planned work tasks that are estimated to require 582 person-days to complete. At the time that you've been asked to do the earned value analysis, 12 tasks have been completed. However, the project schedule indicates that 15 tasks should have been completed. The following scheduling data (in person days) are avalable [5-6].

Task	Planned Effort	Actual Effort
1	12.0	12.5
2	15.0	11.0
3	13.0	17.0
4	8.0	9.5
5	9.5	9.0
6	18.0	19.0
7	10.0	10.0
8	4.0	4.5
9	12.0	10.0
10	6.0	6.5
11	5.0	4.0
12	14.0	14.5
13	16.0	—
14	6.0	—
15	8.0	—

BCWP= 126.50 BCWS= 156.50 ACWP= 127.50

Given Total Task = 56; Effort Estimated= 582 Person Day

Figure 37: Progress Data of Project Execution

BAC = 582.00 person-day
SPI = BCWP/ BCWS = 126.5/ 156.5 = 0.808307
SV = BCWP - BCWS = 126.5 - 156.5 = -30 person-day
CPI = BCWP/ ACWP = 0.99
CV = BCWP – ACWP = -1 person-day

% schedule for completion = BCWS/ BAC = 156.5/ 582.00 = 26.89% [% of work scheduled to be done at this time]
% complete = BCWP/ BAC = 126.5/ 582.00 = 21.74% [% of work completed at this time]

EXERCISE – 10: Progress of Project Execution

Objective: Perform Monitoring and Control

Tools/ Apparatus: Microsoft project

Procedure:

1. Identify Schedule Performance Index and Cost Performance Index (SPI & CPI) at a certain point of time in the project execution time
2. Discuss effects of CPI and SPI in your project execution
3. Discuss how can you overcome the time and budget overrun without affecting the project outcome.

7.5Why still Project Fails?

Despite the effort given in planning and execution process, the reality is that still most projects fail. They are either too late, or they go and run over budget and sometimes both, and often the project functionalities suffer from user satisfaction. And so, what is it in the execution itself? What are the components that actually lead us astray? What are the reasons that projects do not progress as smoothly as we have hoped that they would be? And even our best risk management plans do not help us in cases of emergency. Well, we can identify several sources of project failure explain here. There are both technical factors and individual factors or human component involved in project complexities [5-6].

Technical Factors involved in Project Failure:

In accord of technical complexities, we have complex projects with many interactions between many different tasks that could lead to unknown, unexpected outcomes. Not only are the tasks linked and dependent upon each other, there are many of them. But sometimes they depend on external functionalities, and external parties, that all come together during the execution of the project. We have internal complexity, in our company, and we have external factors. Regulation that has changed, and weather emergencies or changes in the environment or perhaps suddenly a competitor has launched a product onto the market that we need to react to and takes us in a completely different direction. And, actually implies a cost overrun, for our project, and a delay to our schedule [5-6].

Human Factors involved in Project Failure:

On the other side the human factors have to do with the nature in which we operate and how we all like to work on a day to day basis. One of the key factors that ultimately leads to delays is the fact that we like to multi-task. If we can multitask throughout the day, throughout the week, and find ourselves moving from one item to the next. But multitasking can add a huge amount of extra time to our project. And why is that? Well, some obvious reasons like, set up cost. It always takes us a few minutes to rethink about what it was that we were doing or get back into the zone of a specific item that we were working on. If we step away physically, it could be demanding, and we could add some physical time into it by just multi-tasking and moving from one thing to the next. But there's also an obvious reason why multi-tasking is problematic and why it delays the delivery [5-6].

For example, imagine that we were trying to execute three tasks, a purple task, a yellow task and a green task. Each one of these tasks takes us pretty much the same amount of time, one unit of time. And so, if we did them sequentially, if we could devote all of our attention to completing the purple task, it would be completed in time period 1. The yellow task would be completed in time period 2, and the green task would be completed after 3-time units. Now what happens if we start to multi-task? in reality it implies that we will be delayed in transitioning each one of these tasks down the road if it is a best-case scenario. In reality when we start to multi-task, there will be additional transition time between the different activities. And so, it is very important that we realize that the multi-tasking activity that we might choose to engage in does have implications in terms of the schedule of our project [5-6].

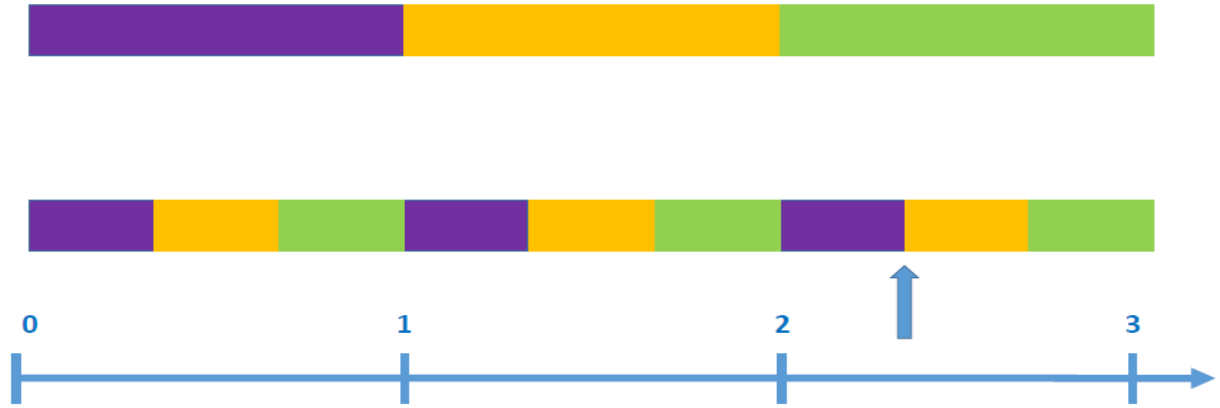


Figure: Example of Multitasking Activities in a Project

Two other factors we have mentioned in the past that have to do with how individuals will lead to some delays in our project. It implies that we might face some scheduling issues throughout the execution. Parkinson's Law and Student Syndrome. Parkinson's Law implies that if we are given an allotted amount of time, we will fill it up. We will work throughout the entire duration. Meaning if we were told that we have a week to complete a task, we will take a week to complete the task. And even if we are not that likely to go ahead and deliver a piece of work sooner. Student syndrome implies that if we have some slack, we will probably procrastinate. We will take

advantage of that slack and actually perform the task that we are expected to do at the very last minute of the assign deadline [5-6].

There are several methodologies that allow us to control the multitasking phenomena. For example, working in smaller teams, breaking up the work into smaller components. And allowing our team to be more independent in selecting those activities will encourage us all to work in a more efficient way. If we work on smaller pieces of work, we are less likely to multi-task because we can feel more satisfaction sooner by completing a task in front of us. If we work as a team, we might be more inclined to actually report that we have completed early because we know that in that case if we become available, we can help a team member and help them solve their problem because we have completed early. And next time, when we're delayed, we know that somebody will come and help us out as well. Instead of working with individual, tasked units and individual buffers on each task that we are assigned, think about the buffer on a project level. Step back and combine forces. And say, we will try best to complete a task as soon as we can. And we'll allow ourselves some room to maneuver on a project level. Again, that will encourage us to actually speed our work up, report that we are done, and go ahead and help our teamwork towards that overall goal of meeting our project's completion time. And so small changes to the way that we divide up the work and the way that we organize our workflow, might help us overcome some of the phenomena that induce extra time and extra budget in our projects [5-6].

REFERENCES

1. Wong, K. (2015).*Software Processes and Agile Practices*. University of Alberta
2. Pressman, R.S (2005). *Software Engineering: A Practitioner's Approach.*
3. Galtier V. & Ianotto, M. (2020). Build Your First Android App (Project-Centered Course). CentraleSupélec.
4. Naik, K., & Tripathy, P. (2011). *Software testing and quality assurance: theory and practice*. John Wiley & Sons.
5. Yael Grushka-Cockayne (2020). Fundamentals of Project Planning and Management. University of Virginia.
6. Hughes, B. & Cotterel, M. (1999). *Software Project Management* (Second Edition).

Sofia (2020). WRSPM Reference Model or The World Machine Model. Available at https://medium.com/@sofiia./wrspm-reference-model-or-the-world-machine-model-54f6436f31e7