How to Run Schedule and Cost Estimations

A sophisticated example of how to apply the techniques of schedule and cost estimations in order to create realistic forecasts.

Image source: www.freepik.com

Foreword

The fact is that the more unique a project is, the harder it is to produce accurate estimations. Properly estimating the project budget, costs and schedule might be the difference between project success and failure. This Ebook is created with the purpose of implementing the several methods of schedule and cost estimation, including building the work breakdown structure, identifying the work packages and the relationships between them, as I refer to them as "Preliminaries to the Estimation Process."

Later, in the book we move on to understanding the basics of estimations and building realistic cost and schedule estimations.

The concepts and examples discussed in this book make the principles of schedule and cost estimation easier to understand.

Several tools are discussed, which should be observed in every estimation procedure.

Image source: www.freepik.com

The Contents of the Ebook

Image source: www.freepik.com

Preliminaries to the Estimation Process

Image source: www.freepik.com

Defining the Project

The general project to be implemented here is the development of a software. In order to keep things simple, I did not specify the type of software we are developing and just considered the common steps in the development of each and every program. Despite being somewhat general, this example has a very good amount of tasks that will allow us to explore in details the concepts of cost and schedule estimation. As any software development project, ours will have 5 phases:

1. Defining Requirements Phase
2. Design Phase
3. Coding Phase
4. Testing Phase
5. Deployment Phase

1. Defining Requirements Phase

2. Design Phase

3. Coding Phase

4. Testing Phase

5. Deployment Phase

Image source: www.freepik.com

1. Defining Requirements Phase	The requirements phase is when the owners of the project will define the stakeholders, hold meetings to understand their requirements and analyze whether the project is feasible or not.
2. Design Phase	The design phase is when the structure of the software is created. The output of the design phase is a document containing the outline of the software, as well as user interaction points, system and hardware requirements.
3. Coding Phase	The coding phase is when the work is actually done. The software is divided into modules or units, and each member of the project team is assigned one specific task. Different versions - alpha, beta, release candidates - are developed through the course of the coding phase.
4. Testing Phase	The testing phase runs tests on different release candidates to identify and fix critical bugs. During this phase the testing team will also identify opportunities for further enhancements in future versions.
5. Deployment Phase	The deployment phase is when the final customer will receive and test the software. The project team is still involved in the software development as it has to collect the feedback from the customer and implement the required changes.

Building the Work Breakdown Structure (WBS)

What is a Work Breakdown Structure

Before we start building the WBS, it is important to understand what should and should not be in the document. If you still did not read the article about the Work Breakdown Structure, I highly recommend you take 10 minutes to go through it before going further in this Ebook. Here is a summary of what the WBS is and which kind of information should be presented there. A WBS is a structured list of every task involved in the project. It takes the complex project and breaks it down into manageable work elements. These can be of two types: summary elements and work packages. Summary work elements involve several subordinate activities, each one called a work package. In our example, the five main phases of the software development are our summary tasks. Therefore, the higher levels of our Work Breakdown Structure will be the five labels (Figure 01):

1. Defining Requirements Phase
2. Design Phase
3. Coding Phase
4. Testing Phase
5. Deployment Phase

A summary task can be broken down into several work packages, which are practical tasks that can be objectively marked as *completed*.

Image source: www.freepik.com

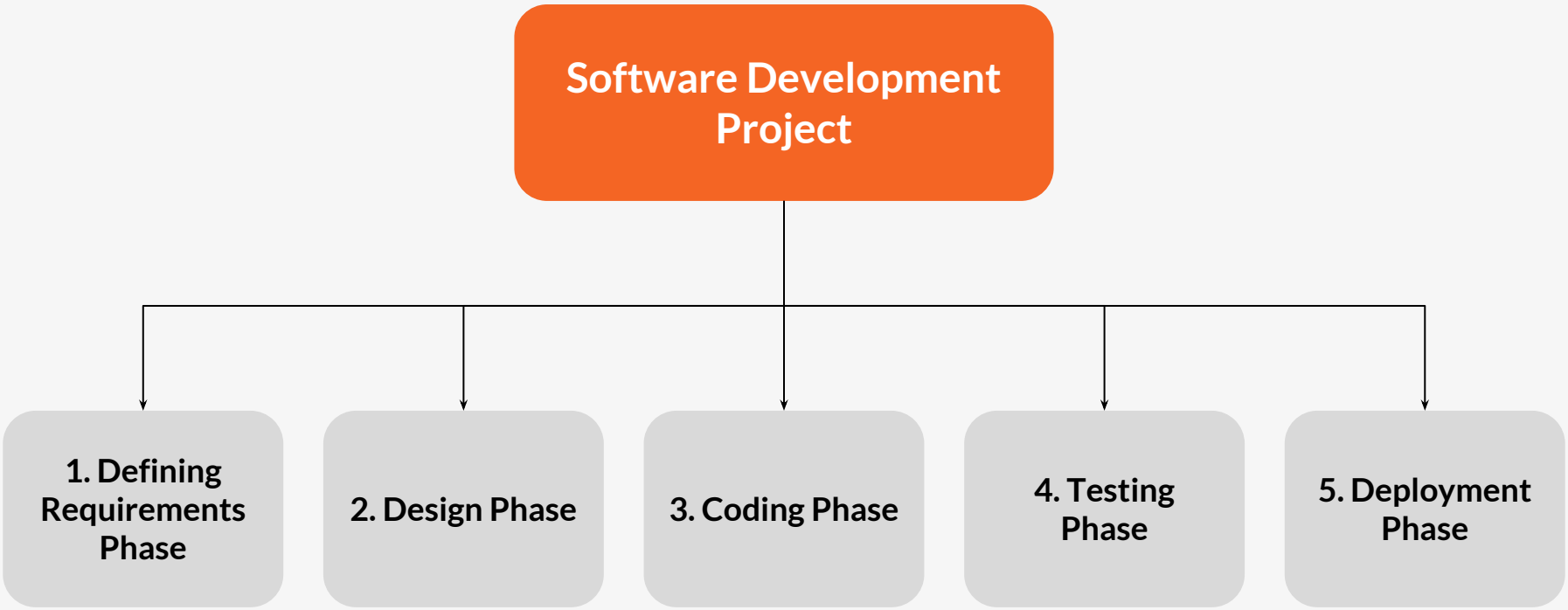


Figure 01:

Hierarchical representation of the five main phases of the software development project.

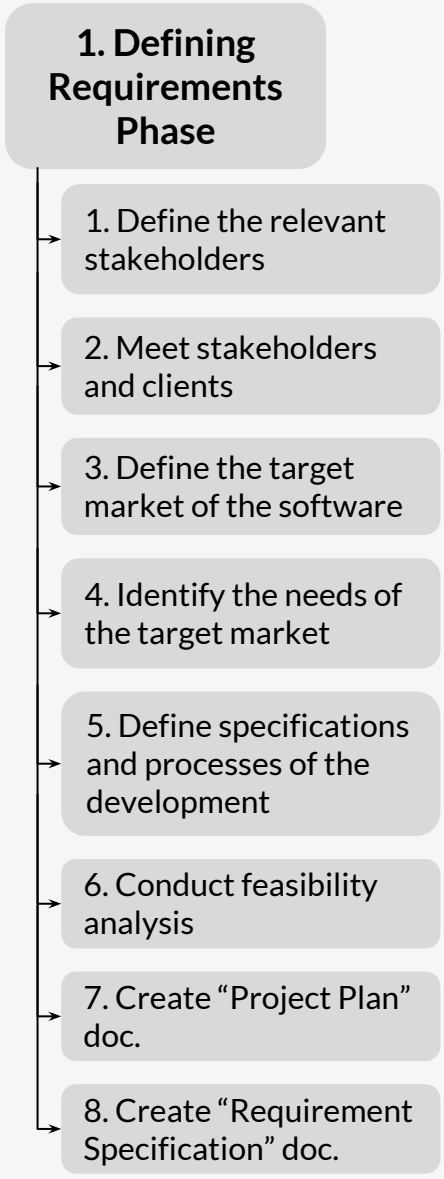


Figure 02: branch for Defining Requirements Phase

As you can see, the Work Breakdown Structure does not incorporate time constraints into its design: the objective of the WBS is to describe the relationships between the summary tasks and the work packages, as well as to link a work package to **a single summary task**.

Now that we know the five main phases of the project, we can identify the work packages related to each one.

Work Packages for the Defining Requirements Phase

The Defining Requirements Phase includes the following work packages:

1. Define the relevant stakeholders
2. Meet stakeholders and clients
3. Define the target market of the software
4. Identify the specific needs of the target market
5. Define the specifications and the processes of the development
6. Conduct feasibility analysis
7. Create a Project Plan document for the development
8. Create a Requirement Specification document

Therefore, these activities will be listed under the label “Defining Requirements Phase” in the WBS. Figure 02 shows how the branch should look like. It is important to highlight that it might be the case that these work packages contain further relevant subtasks. Should this happen, you will execute exactly the same process for the relevant node.

Work Packages for the Design Phase

The design phase can be broken down into the following work packages (Figure 03):

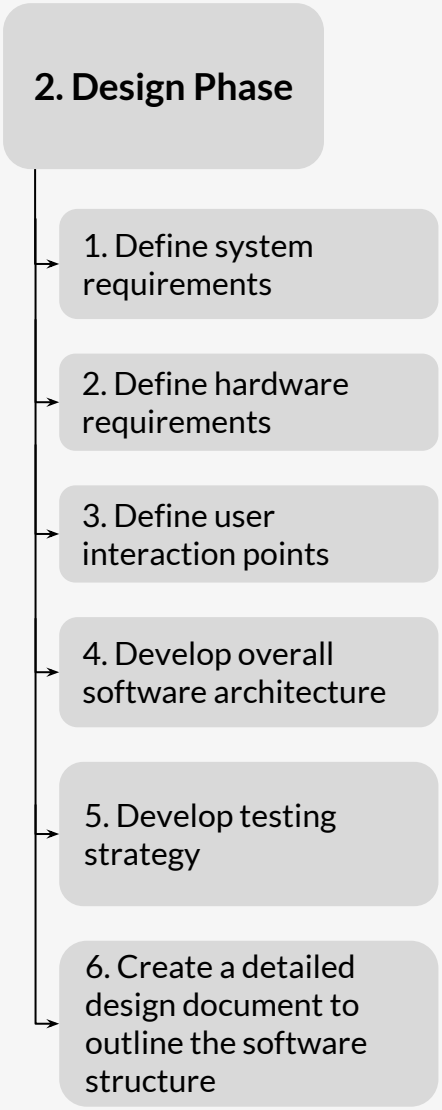


Figure 03: branch for Design Phase

1. Define system requirements
2. Define hardware requirements
3. Define user interaction points
4. Develop overall software architecture
5. Develop testing strategy
6. Create a detailed design document to outline the software structure

Work Packages for the Coding Phase

The coding phase can be broken down into the following work packages (Figure 04):

1. Divide software into modules\units
2. Assign specific modules to development team
3. Create a development tracking document
4. Develop an alpha version with main functionalities
5. Develop a beta version for internal testing
6. Develop different versions of release candidates

Work Packages for the Testing Phase

The coding phase can be broken down into the following work packages (Figure 04):

1. Test release candidates for system specifications and bugs
2. Fix critical bugs
3. Define possible enhancements for future versions

Work Packages for the Deployment Phase

The coding phase can be broken down into the following work packages (Figure 04):

1. Deploy the beta version of the software for the final user

2. Collect feedback and implement changes
3. Develop a Production Build version for final release
4. Deploy the Production Build version

Building the Final Version of the Work Breakdown Structure

Now that we have all the information about the work packages involved in each summary task, we can define the entire WBS for our project. Figure 05 shows the final look of our WBS. As you can see, each work package is linked to **only one** summary task. One limitation this WBS has is that it does not divide the tasks related to development into further work packages. I prevented myself from doing it for two reasons: it would make the Ebook tediously long, and this phase varies considerably for different types of softwares.

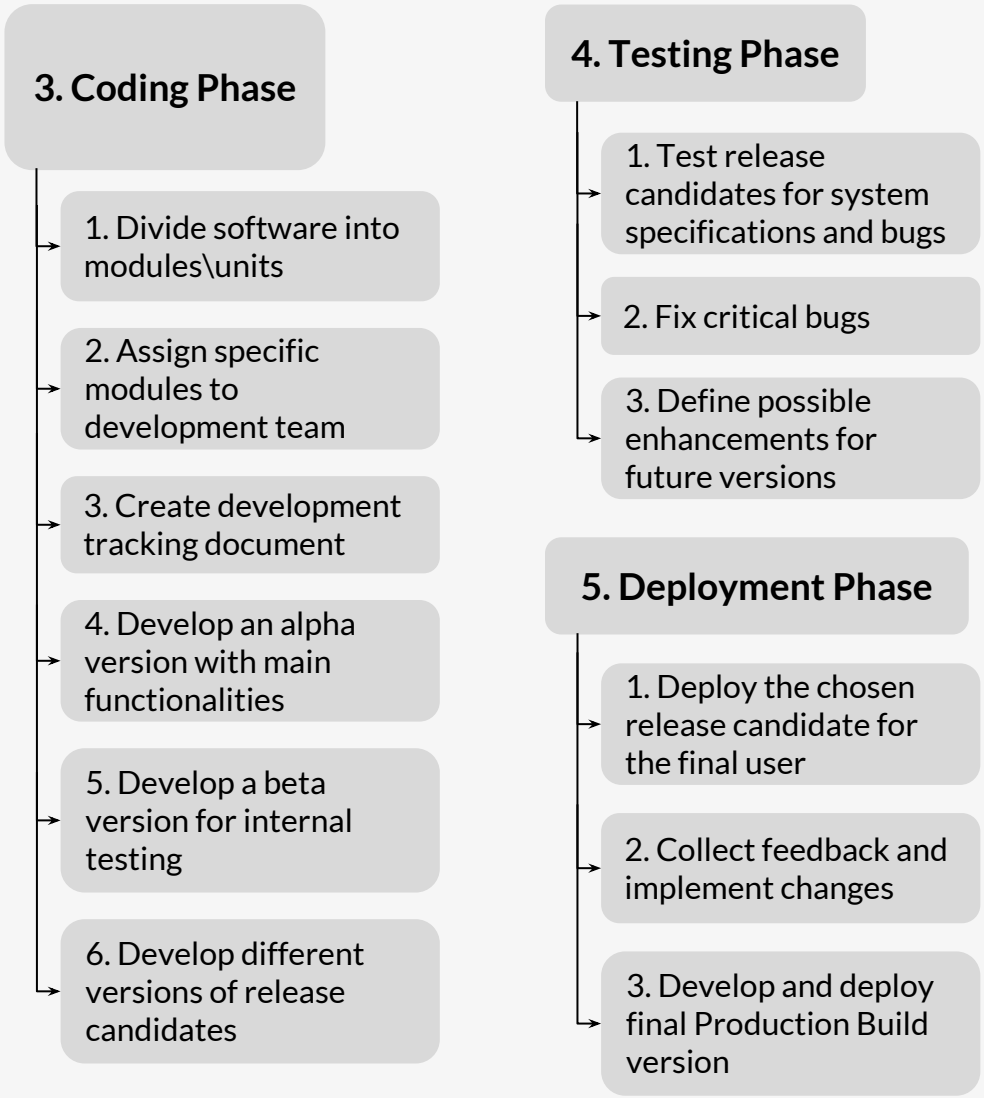


Figure 04: branches for Coding, Testing and Deployment phases

Figure 05: Complete Work Breakdown Structure of the Software Development Project

Identifying the Relationship between the Work Packages

Once the Work Breakdown Structure is ready, the next step is to identify the relationships among tasks. The output of this phase is a Network Diagram that identifies how each task is related to each other following a “requirement” logic. We are still not worried about the time or labor constraints; instead, we want to identify which tasks are requisites for posterior work packages and which tasks can be carried simultaneously. Figure 06 puts our WBS in the form of a table, which makes the task of identifying predecessors much easier. In fact, the five phases of our project already create four big sequence constraints: the design phase must be executed after the defining requirements phase, and we can only deploy our software after coding and testing it. Evidently, there are many ways in which the execution of work packages under different summary tasks can be carried simultaneously. However, these tweaks deal with increasing the efficiency of the project and drastically vary from project to project and from organization to organization. As our goal here is to create a general example, we will focus more on the implementation side than on the optimization side of project management. Therefore, we will follow the division of our WBS and stick to five project phases in our Network Diagram. There will be some overlapping of summary tasks, but nothing too complicated to halt the understanding of the processes being discussed.

Image source: www.freepik.com

Implementing a table version of your WBS is highly recommended for writing more detailed information about each work package. The table version is also required for creating the Gantt Chart and the cost estimation for the project. Once you have identified the "requirement" constraints between your work packages, you can create your network diagram. Figure 06 gives us the correct diagram considering our restrictions in Table 01.

Table 01:

Complete table version of the project's Work Breakdown Structure

Task ID	Description	Predecessors
1.	**Requirement gathering and analysis**	
1.1.	Define the stakeholders	-
1.2.	Meet stakeholders and clients	1.1
1.3.	Define the target market of the software	1.1
1.4.	Identify the specific needs of the target market	1.2, 1.3
1.5.	Define the specifications and the processes of the development	-
1.6.	Conduct feasibility analysis	1.4, 1.5
1.7.	Create a Project Plan document for the development	1.6
1.8.	Create a Requirement Specification document	1.6
2.	**Design**	
2.1.	Define system requirements	1.7, 1.8
2.2.	Define hardware requirements	1.7, 1.8
2.3.	Define user interaction points	1.7, 1.8
2.4.	Develop overall software architecture	2.1, 2.2, 2.3
2.5.	Develop testing strategy	2.1, 2.2, 2.3
2.6.	Create a detailed design document to outline the software structure	2.4, 2.5
3.	**Implementation or coding**	
3.1.	Divide software into modules or units	2.6
3.2.	Assign specific modules to development team	3.1
3.3.	Create a development tracking document	2.6
3.4.	Develop an alpha version with main functionalities	3.2
3.5.	Develop a beta version for internal testing	3.4
3.6.	Develop different versions of release candidates	3.5
4.	**Testing**	
4.1.	Test release candidates for system specifications and bugs	3.3, 3.6
4.2.	Fix critical bugs	4.1
4.3.	Define possible enhancements for future versions	4.1
5.	**Deployment**	
5.1.	Deploy the beta version of the software for the final user	4.2
5.2.	Collect feedback and implement changes	5.1
5.3.	Develop a Production Build version for final release	5.2
5.4.	Deploy the Production Build version	5.3

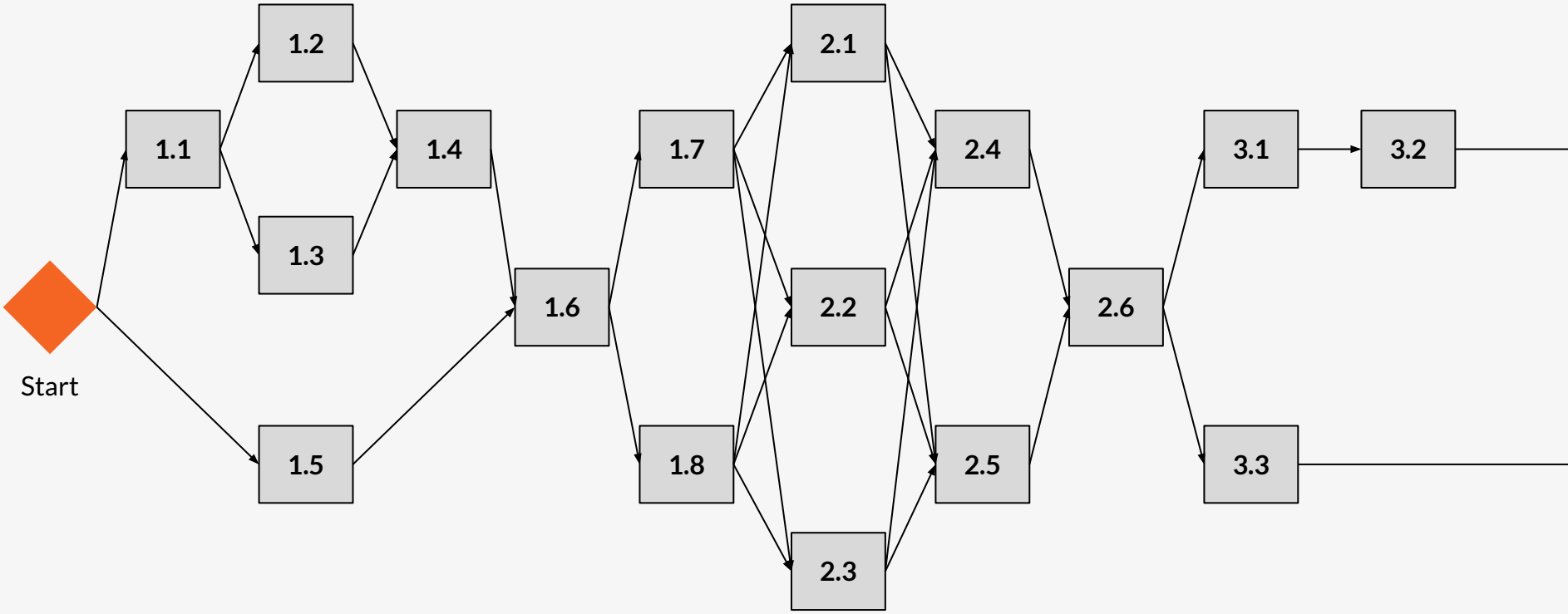


Figure 06.a: Complete Network Diagram for Software Development Project - Part 01

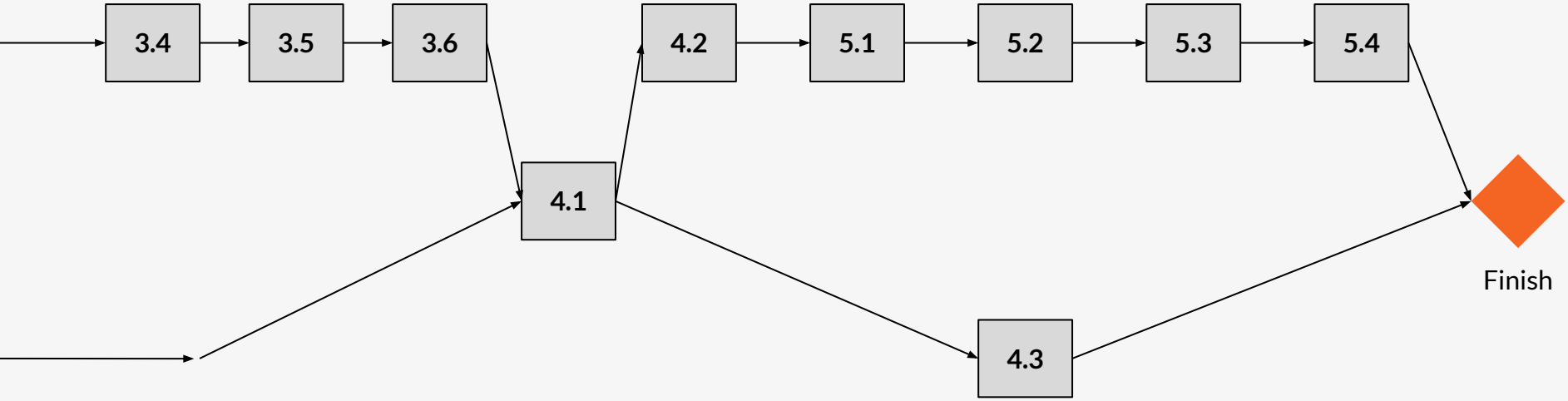


Figure 06.b: Complete Network Diagram for Software Development Project - Part 02

Estimating the Cost and Duration of Work Packages

Image source: www.freepik.com

The Basics of Cost Estimation

With our Network Diagram in hands, now it is time to estimate the costs and the duration of each work package in the Work Breakdown Structure. We will visit different estimation methods and discuss the particularities, benefits and setbacks of each one.

Before we start discussing the methods themselves, a few things must be understood. First of all, there is a trade-off between costs and schedule. A tighter schedule will usually result in a higher cost, and to reduce the cost we will probably have to extend the schedule (by removing, for example, overtime work).

Another important topic are the sources of data for your estimation. The estimation processes are not extremely complex themselves; it is collecting and organizing the data for your estimations that makes this process so difficult sometimes. While it is not possible to predict all the future costs of your project at the planning phase, we can focus in four major sources of costs for your project.

Internal Labor Costs

The internal labor cost refers to the amount of labor required from people employed by your company. An accurate amount of labor comes from having accurate amounts for the work packages in your breakdown structure. Since the hourly salary of your employees might differ among each other, the best way to come up with realistic

Image source: www.freepik.com

numbers is to use the **burdened rate**. The **burdened rate** is nothing more than the average hourly total cost of an employee to the company. Using just the hourly payment of each employee is not good enough because people cost more than just the salaries they receive. Often there are benefits they are paid, absences we have to account to, overhead costs (such as fixed costs mitigated among several projects which are not from one specific project, so they can't be assigned to it) and so on. Assessing the burdened rate is fairly simple: the financial department of your company must have this information. You really don't need to spend efforts trying to figure it out, just go there and ask them.

Internal Equipment Costs

These are the costs of equipments that are not normally available for your staff, in other words, that are specific to

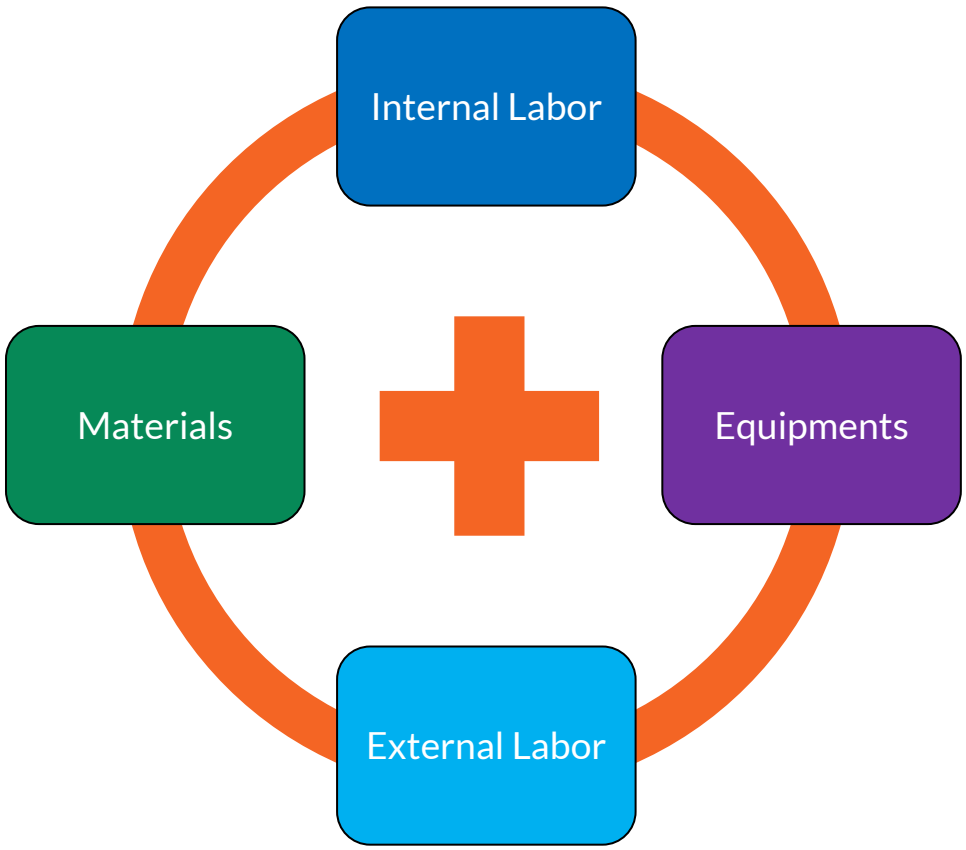


Figure 7:

The four sources of costs: internal labor, equipments, external labor and materials

the project or that are acquired for the project and subsequently used in other activities. For example, consider a new construction project that required a deeper, stronger foundation than all the project executed by the company before. As a consequence, the business needs to rent (or buy) a special bulldozer for the task. The cost of renting this bulldozer should be included in your project since it is not an equipment your company own and it will be rented specifically for this project.

Now consider the situation where the company is considering to buy the bulldozer instead of renting it. This might be an option if, for example, there are positive prospects of future large contracts that will require the special bulldozer for the constructions. How should you include it in the budget? It doesn't really make sense to include all its costs under one single cost estimation, as this would drastically increase the costs of your project and decrease the chances of getting it approved. In order to overcome this problem, the best way is to estimate the total life cycle of the equipment and to divide the initial cost in smaller units. You can, for example, estimate the useful life of the bulldozer as around 50,000 hours of work, so a simple approach would be to divide the cost of the bulldozer by 50,000 and multiply the cost per unit of time by the estimated amount of hours the machine will be employed in a specific project. Although this estimates based on estimations, it is a good way to avoid attributing all costs of equipments to a single point in time.

Costs of External Labor and Equipment Use

Contracting an external party to execute some of the tasks is not uncommon in the world of project management. The key aspect here is to understand how to include the estimation for the external costs in your project. If the contract determines a rate used by the vendor to bill labor, equipment and materials, you will use the rates and the estimated amount of work to calculate the estimated cost of that service for your company. If, however, your contract states a fixed fee, the work of estimating everything will be done by the vendor and you will just need to worry about including a single final number in your project.

Material Costs

Finally, material costs are the last major source of costs in the execution of a project. This component of the budget is one of the few that is not calculated based on the Work Breakdown Structure. Although the WBS can be of some help, the product or service specification is the best place to look for information to estimate the cost of materials.

Estimating the Cost and the Labor Hours of the Summary Tasks

I could simply show you a final table (like Table 02, for example) with a bunch of numbers and say "well here are the final estimates for the labor hours of each task". But I will not do that for two reasons: (1) it is unrealistic to estimate the amount of labor in every phase of the project right at the beginning, and (2) it would reduce our possibilities of exploring different methods of estimation.

What I will do, instead, is to combine **phased estimation, apportioning, Order of Magnitude estimation and bottom-up estimation.** Here is the logic behind it:

At the start of a project, we hardly ever have a precise guess of the detailed estimates for each of its stages. Therefore, we need a more general number (the Order of Magnitude) to approximate the total cost of the project. In order to obtain such number, we will combine the bottom-up estimation of the Defining Requirements Phase with the apportioning method. Finally, the phased estimation will be used to stick to one phase at a time and avoid guessing too much at the beginning. We will revise a bit of theory for each process and at the same time see how they can be applied to our project.

Order of Magnitude Estimation

The Order of Magnitude is a quick way to provide a rough estimate of how much our project will cost us. The number is calculated based on the information from

Image source: www.freepik.com

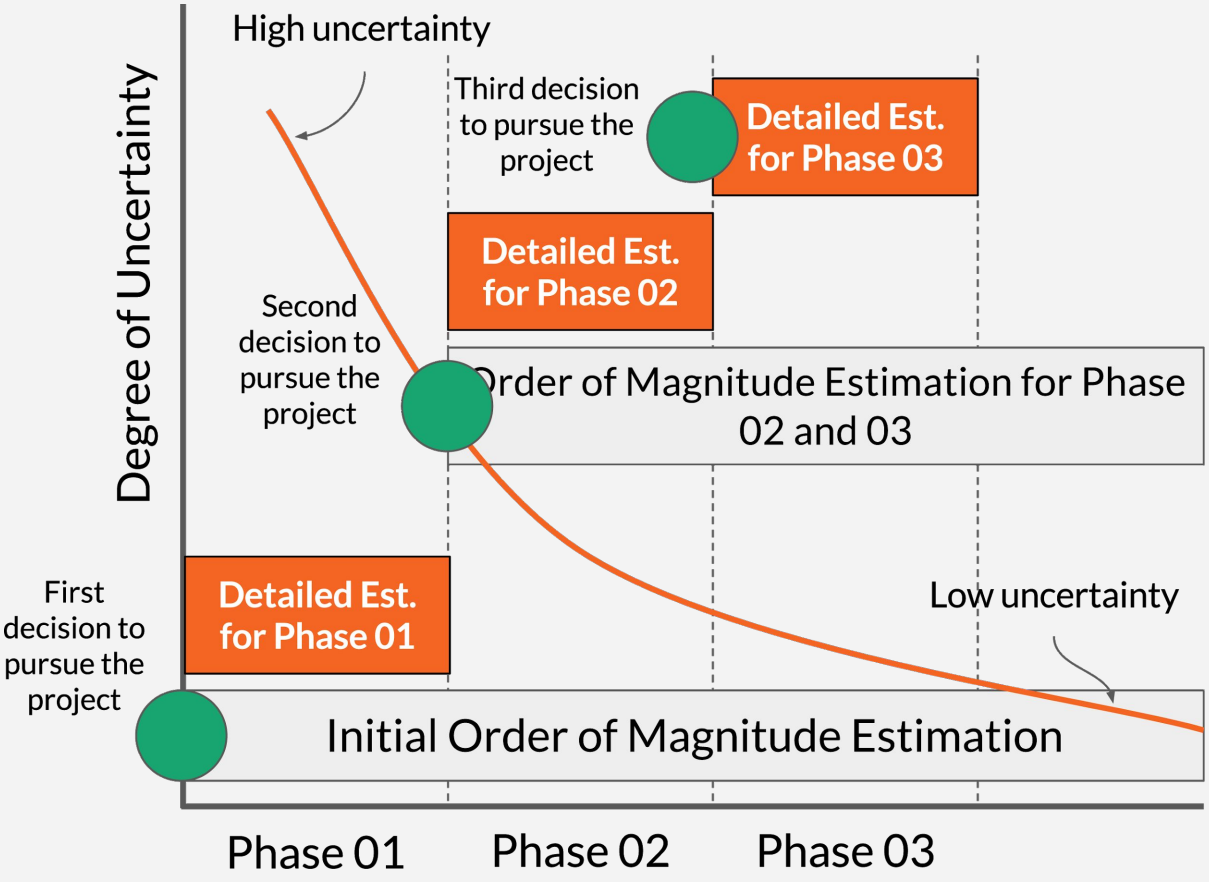


Figure 08:

Stages of Phased Estimation

previous projects, and it tries to find similarities or relations between past projects and the current estimation. Naturally, such figure is considerably variable and it does not serve the purpose of providing a detailed cost estimation for each phase. Its sole purpose is to quickly generate a number to broadly check the feasibility of a project.

Phased Estimation

Phased Estimation starts to incorporate some complexities in the cost estimation process. Figure 08 shows how the process works in practice. As you can see, the initial phase starts with an Order of Magnitude estimation for the entire project and a detailed estimate for the first phase. Once the first phase is over, the cost estimation tasks starts again and we produce a second

detailed estimate for the second phase of the project. In addition to the detailed estimate, we update our Order of Magnitude estimation to incorporate the data we collected during the first phase. Finally, in the last phase we just need to provide a detailed estimate (there is no need for a new Order of Magnitude number, as it would be exactly the same as the detailed estimate). We will use the process of bottom-up estimation later to give the detailed forecasts required by the Phased Estimation.

Apportioning

Apportioning, also known as top-down estimating, starts from the top with a total figure for the project and then assigns percentages of that budget to different phases and tasks. As you can imagine, the Work Breakdown Structure is essential for you to have a

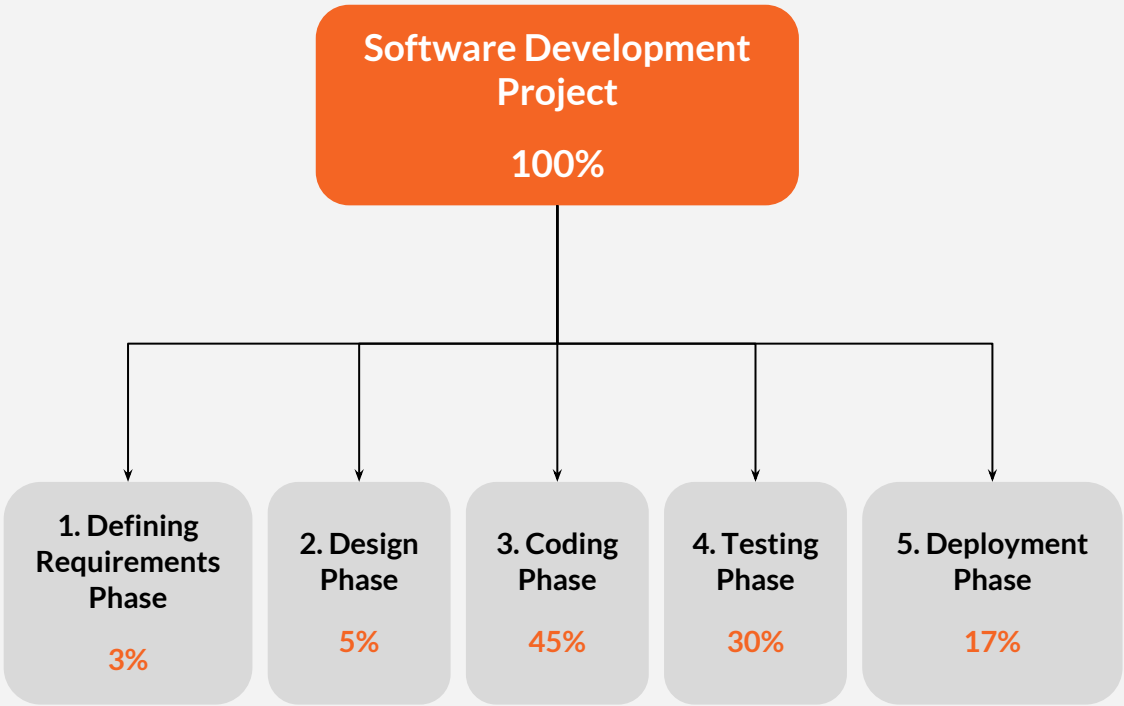


Figure 09:

Apportioning method for the five summary tasks

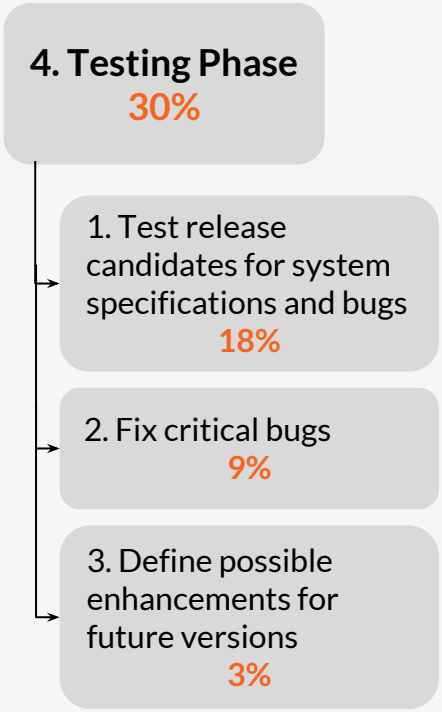


Figure 10:

Apportioning method for one specific summary task

clear picture of which tasks will be involved in the project, as well as their complexity. As the top-down estimate derives the smaller budgets from the top, we must be careful to provide a good number at the beginning. If the overall estimate is not valid, the smaller shares of the budget will also deviate from reality.

Let's use apportioning to define the share of each stage in our complete project. Imagine that, from previous experience, you know that the approximate cost shares of each phase of the project are as follows:

- Defining Requirements Phase: 3%
- Design Phase: 5%;
- Coding Phase: 45%;
- Testing Phase: 30%;
- Deployment phase: 17%.

Figure 09 shows how our WBS can be adapted to include the percentages. As is the case generally with software development, coding and testing phases usually take the most of your time, and the first phases are quicker (yet very relevant for developing a detailed strategy and ensure the efficiency of the coding and testing tasks). The deployment phase involves collecting feedback and adapting the software, and this might take a while. But again, it might be the case your company works differently. The important thing is to look at your previous data and try to find some similarities between your past projects and the current one.

The apportioning method can also be used to decompose each of the five major steps of the project. Figure 10 presents an example of how you can break down the 30% attributed to the testing phase. While breaking down each summary task into further percentages (what we would consider a "pure" apportioning approach)

could be done, I particularly don't think this is the best approach to running accurate cost and schedule estimations. The fact is that we need these percentages to divide the whole project into major slices of cost, but the more we try to go deeper into percentage estimations, the more likely we are to be wrong. Apportioning is a good approach for the high level summary tasks because you can use your past data with a good degree of preciseness. However, for lower levels we will change the approach and use the bottom-up estimation. Finally, we will see how to bring the bottom-up, apportioning and Order of Magnitude processes together to generate your final initial number for the project.

Bottom-up Estimation

The bottom-up process is the most complex one because it requires actual estimations of how many labor hours are used in each work package, as well as how much they will cost you. We already discussed that the **burdened rate** (the hourly total cost of an employee) is the best measure to use here.

One important thing to discuss is the trade-off between schedule and cost. It is fairly intuitive that a tighter schedule requires a higher budget and vice-versa, so the true question is: which one should be your priority? Personally, I observe that budget is a more sensitive topic than schedule. While an increase of a week on schedule does not hurt much (if the deadline is not critical), an increase of $1000 looks pretty bad. Therefore, my suggestion is for you to first define the budget and then adapt the schedule accordingly.

While you should look at your previous projects to estimate how much labor is required to complete each work package, I will use fictional (yet realistic) figures in

the example. Table 02 shows the estimation of labor hours for each of our work packages in the Defining Requirements phase. The column "Resources" establishes who will be doing the respective task. The minimum duration assumed for a task is one day, even though a task might actually take less than one day to be completed (for example, "Define the stakeholders" takes only 4 hours, which is half of a full working day - assumed to last 8 hours here. Also, we will assume our project team has 3 member plus the project manager, and each member of the team is fully capable of doing all the tasks. In reality, this might not be the case and your project might have know-how

Task ID	Description	Duration	Labor Hours	Resources
1.	Requirement gathering and analysis			
1.1.	Define the stakeholders	1 day	4	Project manager
1.2.	Meet stakeholders and clients	3 days	20	Project manager
1.3.	Define the target market of the software	1 day	4	Project manager
1.4.	Identify the specific needs of the target market	1 day	4	Project manager
1.5.	Define the specifications and the processes of the development	1 day	8	Project manager Project team [1]
1.6.	Conduct feasibility analysis	1 day	4	Project team [1]
1.7.	Create a Project Plan document for the development	2 days	16	Project team [1]
1.8.	Create a Requirement Specification document	2 days	16	Project team [1]

Table 02:

Bottom-up estimation for Defining Requirements Phase

restrictions that you have to deal with. However, for the sake of our example, this is a reasonable assumption. Furthermore, if there are two or more people working on the same task we will assume the work is split equally. For example, in the task "Define the specifications and the processes of the development" the project manager and the member of the project team will work 4 hours each. Finally, the number in brackets identifies how many members of the project team will be working on that task. When we move further into the example and assign more people to the same task you will see a [2] or [3] to represent that two or three people will be working simultaneously.

In some tasks, like "Create a Requirement Specification document", the resources assigned to that work package will be fully used; in others, they will partly employed.

Finally, we must assume the burdened rate for all the resources involved in the project. This might vary considerably from business to business, but let's define the hourly costs:

- Project team member: $40.00
- Project manager: $55.00

Generating the Order of Magnitude Estimate

Now that we have some numbers we can look at how our first estimation can be integrated with the apportioning method to generate the Order of Magnitude figure. As the table below shows, the total hours estimated for the first phase is 76, and the forecasted labor cost is $3,580.00. Since these numbers represent a stage that we believe will account for approximately 3% of our efforts, we can conclude that the Order of Magnitude estimate should be around:

- Hours: approx. 2600
- Cost: approx. $120,000.00

	Project Manager	Project Team Member	Total
Total Hours	36	40	76
Cost per Hour	$ 55.00	$ 40.00	$ 47.11
Total Cost	$ 1,980.00	$ 1,600.00	$ 3,580.00

Now that we have an approximate number for the total duration and cost of the project, we can break them down into the different phases. Let's be realistic and slightly round the numbers up while we divide the 2600 hours and the $120,000.00. As you can see, our total amount of hours is now 2676, a slight increase of 3%. The new estimation for the costs is $120,580.00, or 0.48% increase over our initial number. The costs are likely to decrease a bit because the burdened cost of the project members (who will carry the coding, testing and deployment phases) is lower than the burdened cost of the project manager. However, let's leave things as they are now and update our estimations in a later stage.

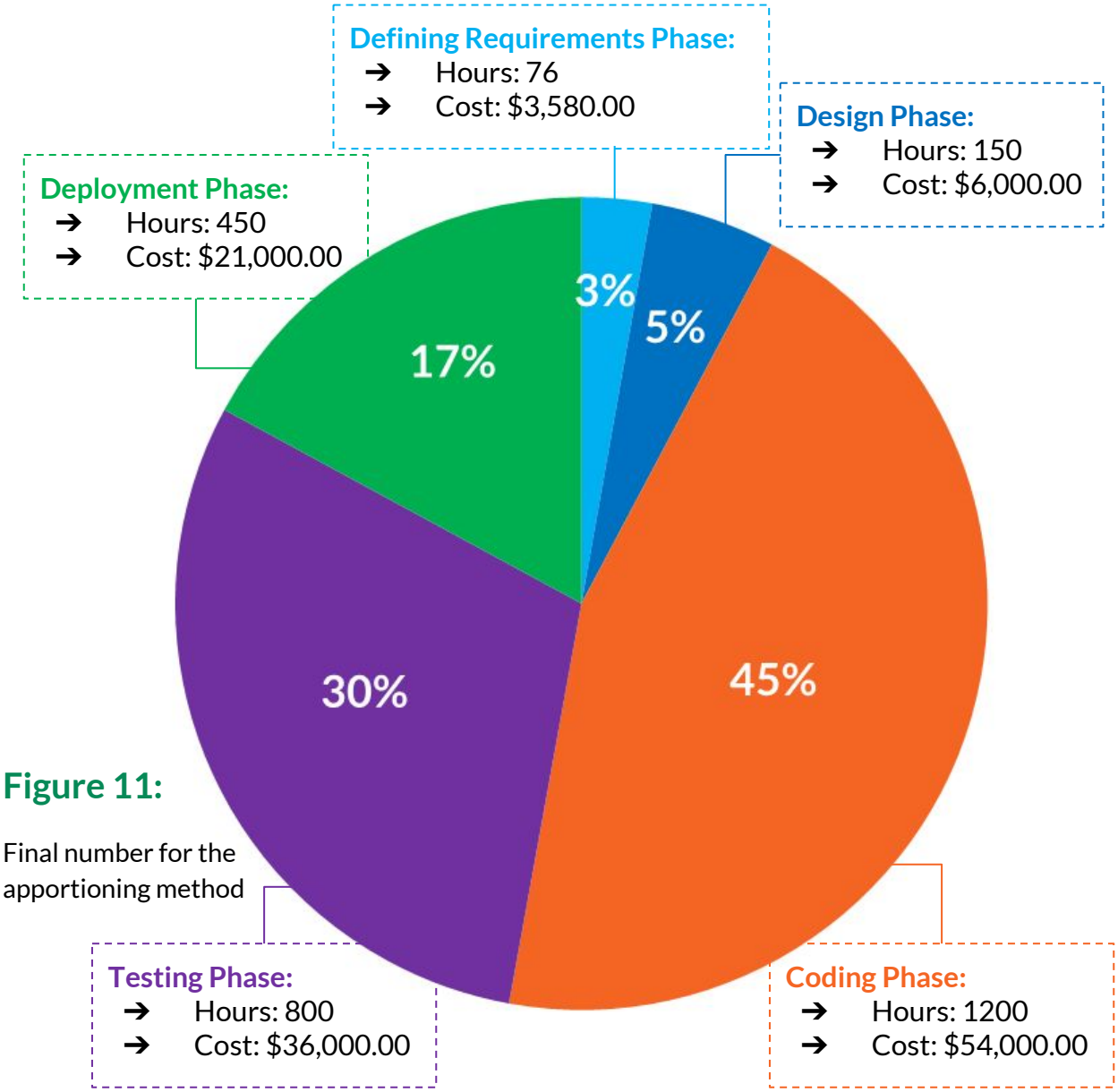


Figure 11:

Final number for the apportioning method

Calculating the Schedule of the Defining Requirements Phase

Once the labor hours and the number of days for each work package are defined, the next step is to calculate the schedule of the project. We will do it in two steps: first we will stick to our previous estimation number and we will build a detailed schedule only for phase one. Then, once the principles are understood, we will simulate a detailed estimation for the entire project and build our complete budget and schedule estimations. I think it is relevant to cover both possibilities since in reality the company you work in might require either of them. The process to calculate them, however, is the same (it just gets a bit more challenging when we are dealing with the entire project).

To calculate the schedule of first phase we will have to look at our Network Diagram and implement some changes based on our estimation for the duration of each task. We will include the following information in the Network Diagram:

- Early Start Date: is the earliest date a task can begin;
- Late Start Date: is the latest date a task can begin without increasing the overall duration of the project;
- Early Finish Date: is the earliest date a task can finish;
- Late Finish Date: is the latest date a task can finish without increasing the overall duration of the project;
- Float: the flexibility on the start date of a task. It corresponds to [Late Start Date] minus [Early Start Date].

Image source: www.freepik.com

In order to calculate these numbers, we will use three processes.

Forward Pass

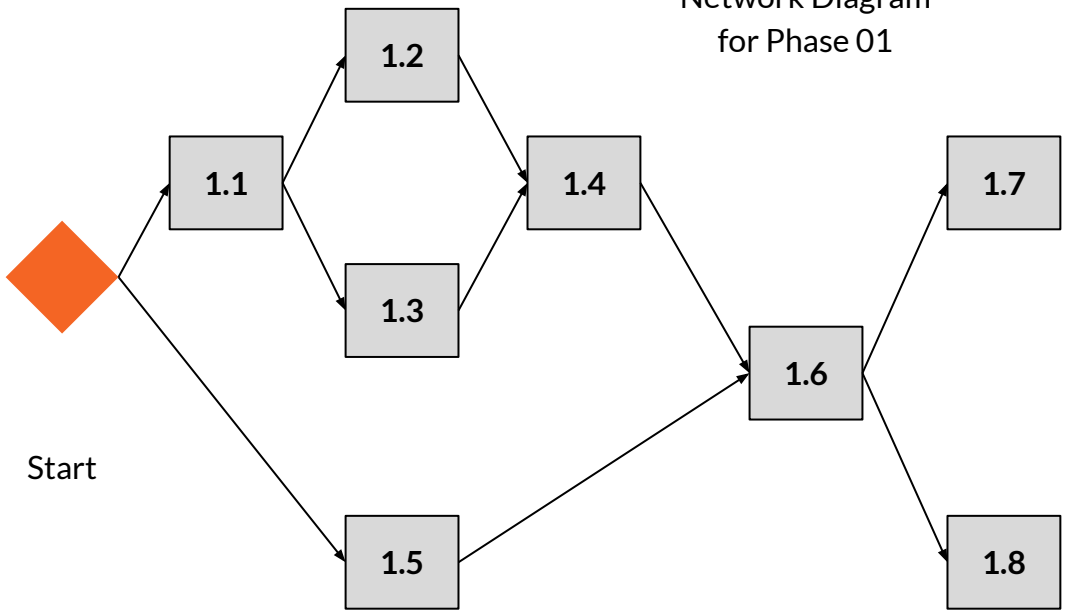
This step determines the early start and finish dates of the tasks in your schedule. It is calculated in two steps:

1. Set the beginning of your project as “day 01” and place the activities that have no predecessors there.
2. Add the estimated duration to your primary tasks and place the following activities right after the end of their predecessors (if there are two or more predecessors, you should consider the longest one as the finishing point). Keep doing so until all your tasks are placed in your schedule.

The process itself is not complicated, although it can get considerably boring if there are a lot of tasks.

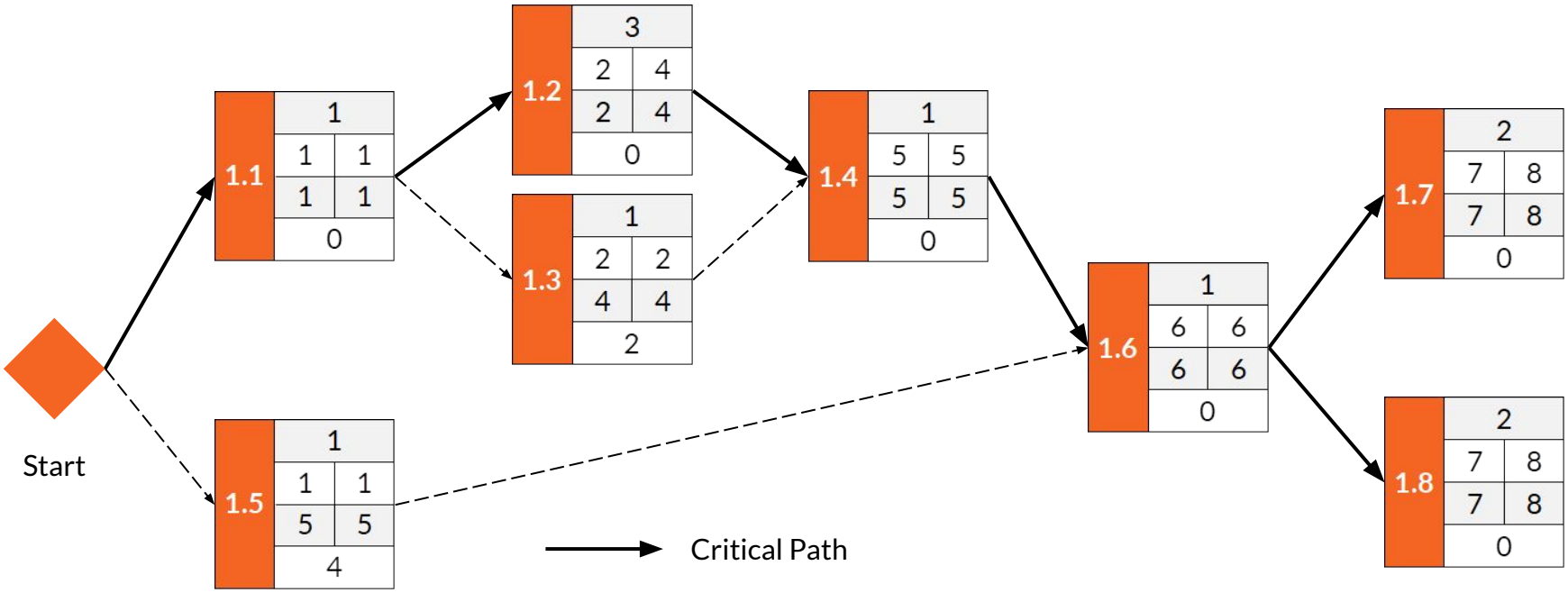
Figure 12:

Network Diagram for Phase 01



Above you find the Network Diagram to the Defining Requirements Phase of the project, and on the right there is a typical table that will replace each node of the Network Diagram. Figure 13 presents the complete diagram with the updated nodes.

Task ID	Duration	
	Early Start	Early Finish
	Late Start	Late Finish
	Float	



Backward Pass

The backward pass is used to calculate the late start and late end date of each work package. In order to calculate them, we start at the end of the project and place the final tasks at the latest possible date of completion. Then, in backwards logic, we start by deducting the duration of the tasks to obtain the latest start date for each of them. Keep doing so until you list all the tasks in your project. In the end, you will obtain a similar diagram to the one created by means of the forward pass method, but now with the late early and start dates.

Figure 13:

Updated diagram for Phase 01

Calculating the Float

The float of a task gives information about how flexible the start date of that task is. In order to calculate it, you just need to subtract the Early Start Date from the Late Start Date. If a task has a float of zero, it means that any delay on the start of that task will cause delays on the final date of delivery of the project. Since it is critical to keep those tasks on schedule, the sequence of all tasks with zero float is called **Critical Path**. If a task has a positive float (task ID 1.5, for example), its start can be delayed by the amount of days the float indicates. The float in task 1.5 indicates that it can be postponed for 4 days (it must invariably start until the 5th day or our schedule will be delayed. Once we have the information about the start and finish dates, as well as the float of each work package, we can build a Gantt Chart to illustrate how the schedule looks like. Figure 14 depicts the Gantt Chart for the Defining Requirements Phase.

Building a Gantt Chart

The Gantt Chart represents graphically the sequence constraints of your work packages. It shows which tasks can be carried together, which one have to wait for their predecessors, as well as the flexibility of each task.

The second part of Figure 14 shows the amount of hours that each member of the project team will have to work if all tasks start at the early start date. As you can see, the Project Manager will have to work overtime if this schedule is followed. This, however, may not be desirable since it may cause unwanted raises in the cost of the project. In order to solve the issue, we can apply the tools of **resource leveling**.

Resource Leveling

Resource leveling aims at optimizing personnel and equipment usage during the execution of the project. By assuming that a

Task ID	Description	Week of July 3rd					Week of July 10th				
		M	T	W	T	F	M	T	W	T	F
1.1	Define the stakeholders										
1.2	Meet stakeholders and clients										
1.3	Define the target market of the software										
1.4	Identify the specific needs of the target market										
1.5	Define the specifications and the processes of the development										
1.6	Conduct feasibility analysis										
1.7	Create a Project Plan document for the development										
1.8	Create a Requirement Specification document										

Non-critical path Critical path Float

Resources	Week of July 3rd					Week of July 10th				
	M	T	W	T	F	M	T	W	T	F
Project Manager	8	12	8	4	4					
Project Team [3]	4					4	16	16		

Figure 14: Gantt Chart and Resource Usage Table

continuous effort is more efficient than a fragmented one, it tries to reduce repetitions and unnecessary additions of tasks during the execution of the tasks. Resource leveling removes most of the superfluous overtime from the project and keeps the workers employed for a longer period at a

Resources	Week of July 3rd					Week of July 10th				
	M	T	W	T	F	M	T	W	T	F
Project Manager	8	8	8	8	4					
Project Team [3]	4					4	16	16		

Figure 15:

Updated Resource Table after the process of resource leveling.

steady rate. The idea of resource leveling is to adjust the schedule so we avoid over and under-allocation of work. Since having many simultaneous tasks might require more work than available, we might have to alter the schedule so it matches the needs of a project to the availability of resources.

Resource leveling involves four basic steps:

Forecast how much resources of each type will be needed throughout the project. The best way to do it is to build a resource spreadsheet. The spreadsheet can then be correlated to the diagram of the schedule and you can better visualize how much of each resource is needed each day. However, it might be the case that using an early start schedule will cause over-allocation of resources, so we have to take a few more steps to make sure our resources are best allocated. This can be simplified by use of a project management software that automatically gives you the resource workload.

Identify the peaks of usage of resources. By using the resource spreadsheet and building a resource histogram (if you really need to), you can identify the periods when there will be more demand than supply of resources.

Delay non critical tasks within their floats. In order to solve the over-allocation problem during resource peaks, the best strategy is to extend or delay non critical tasks within their floats. In other words, you will be redistributing the tasks within your schedule and moving them from periods where there is too much work for periods where there is too little work.

Reevaluate the work package estimates to eliminate any further resource peaks. If, after leveling the resources, you still have resource peaks of over-allocation, you might

want to revise the estimates of resources in your work packages. Maybe you can assign less workers for longer periods of time to a task and still obtain the same predicted outcome. These changes, however, will alter the time flexibility of the tasks being modified, so you should come back to recalculating your initial schedule and carrying a resource leveling once you're done with redefining the estimates for the work packages.

In our case, the Resource Diagram shows that the Project Manager has too much work on Tuesday but too little on Friday. We can immediately solve this problem by reallocating four hours of work from the former day to the latter (by moving task ID 1.3 within its float). This way, the Project Manager has eight hours of work for every day of the entire week.

Regarding the project team itself, the sixteen hours of work on Tuesday and Wednesday of the second week are not a problem since we have three members working simultaneously. Since our estimation of labor hours (Table 02) forecasts one member of the team for each work package (IDs 1.7 and 1.8), it is perfectly natural to have a workload of 16 hours per day for two workers. There is no work overload here.

Updating the Order of Magnitude Estimate after Phase 01

Before we move on to the last part of the example (where we will build estimations, construct the Gantt Chart and carry the process of resource leveling for the entire project), I would like to quickly show a simple technique for you to update your Order of Magnitude Estimate once the phase Defining Requirements is over. This is important from the perspective of Phased estimation, as a new Order of Magnitude Estimate might determine whether the company will continue the project or suspend its execution.

Suppose you have finished Phase 01 and now you have to update your estimates. You look at your data and you see that instead of the initial number of 76 hours, you needed a bit more: 84 hours of work. This amount was divided as follows: 40 hours of work for the Project Manager and 44 hours of work for the members of the project. Your updated cost (based on the burdened rate for the Manager and Team Members) is $3,960.00.

You know that this information accounts for an expected 3% of number of hours and costs for the entire project, so now your work is to provide an updated amount for the remaining 97%.

If you take the 84 hours and the $3,960.00 and divide them by 0.03, you will obtain updated numbers for the entire project based on your historical data from the first phase. The results will be: 2800 hours and a total cost of $132,000.00. Now all you have

Image source: www.freepik.com

to do is to deduct the actual amount spent in phase 01 to obtain the figures for the remaining 97%.

Costs: $132,000.00 - $3,960.00 = $128,040.00

Hours: 2800 - 84 = 2716 hours

With these numbers in hand, you can update the apportioning values for the following phases and present a better estimate for the rest of the project. Figure 16 shows the process graphically.

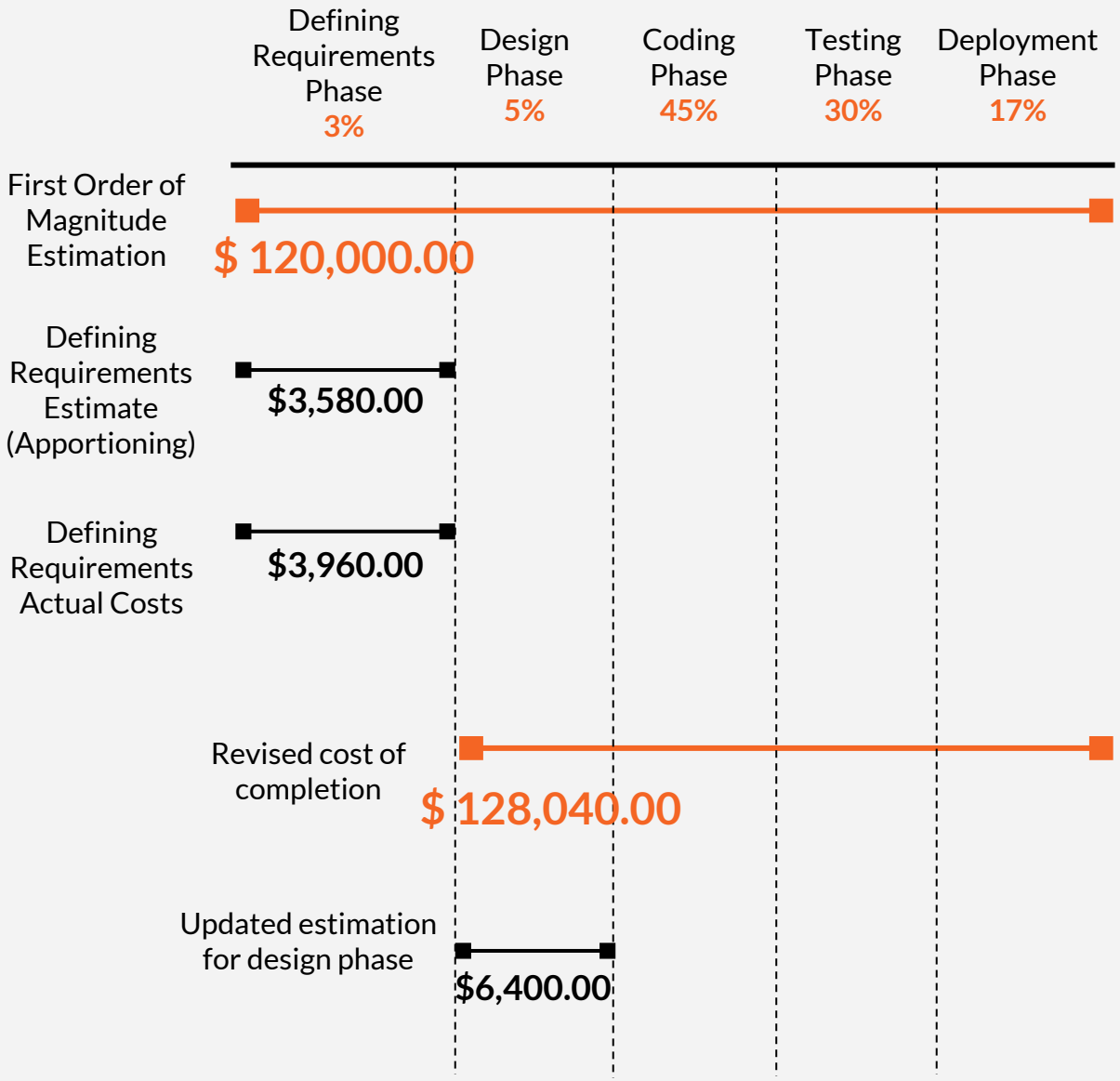
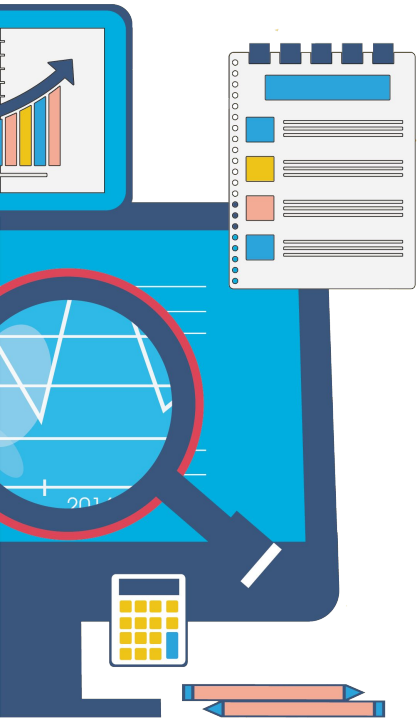


Figure 16:

Updating the Order of Magnitude Estimate



Creating a Gantt Chart for the Entire Project

Finally, I would like to explore a bit more the concept of the Gantt Chart by providing a complete schedule for our project. The figures I will present are based on our initial Order of Magnitude Estimates, and will serve the sole purpose of illustrating how to build more sophisticated Network Diagrams, Gantt Charts and Resource Leveling Tables. Table 03 shows a detailed bottom-up estimation for the entire Software Development project. As you can see, the total amount of hours adds up to something close to our initial prediction, and the cost is lower because the burdened rate of team members is lower than the burdened rate of the Project Manager (when we used the bottom-up estimation from the Defining Requirements Phase to calculate our Order of Magnitude estimate we did not adjust the numbers for the fact that the Project Manager will be working less during the execution and testing phase). In any case, Table 03 is a rough estimate of how we could produce a detailed bottom-up estimation, but it is still far from reality (as the Project Manager, for example, is not included in any task starting from the second phase on). In any case, it will be a useful resource for us as an example.

Figure 17 shows a Network Diagram with the early start, early finish, late start, late finish and float information based on Table 03. The calculation of these numbers follows the principles already discussed,

Image source: www.freepik.com

but now we need more attention to make sure no mistakes are made. The use of a Project Management software is the best way to guarantee a precise calculation of start and finish dates, as well as the float of each work package. As you can see, there are not so many tasks with positive float, meaning that our project is quite tight on the deadlines. Whether this is something good or bad depends on how your company approaches the issue, but usually tight deadlines mean additional and unwanted delays.

Table 03:

Detailed estimates for the entire project

Task ID	Description	Duration	Labor Hours	Resources	Costs
1.	Requirement gathering and analysis				
1.1.	Define the stakeholders	1 day	4	Project manager	$ 220.00
1.2.	Meet stakeholders and clients	3 days	20	Project manager	$ 1,100.00
1.3.	Define the target market of the software	1 day	4	Project manager	$ 220.00
1.4.	Identify the specific needs of the target market	1 day	4	Project manager	$ 220.00
1.5.	Define the specifications and the processes of the development	1 day	8	Project manager Project team [1]	$ 380.00
1.6.	Conduct feasibility analysis	1 day	4	Project team [1]	$ 160.00
1.7.	Create a Project Plan document for the development	2 days	16	Project team [1]	$ 640.00
1.8.	Create a Requirement Specification document	2 days	16	Project team [1]	$ 640.00
2.	Design				
2.1.	Define system requirements	1 day	8	Project team [1]	$ 320.00
2.2.	Define hardware requirements	1 day	8	Project team [1]	$ 320.00
2.3.	Define user interaction points	1 day	8	Project team [1]	$ 320.00
2.4.	Develop overall software architecture	2 days	32	Project team [2]	$ 1,280.00
2.5.	Develop testing strategy	2 days	16	Project team [1]	$ 640.00
2.6.	Create a detailed design document to outline the software structure	1 day	48	Project team [2]	$ 1,920.00
3.	Implementation or coding				
3.1.	Divide software into modules or units	2 days	12	Project team [1]	$ 480.00
3.2.	Assign specific modules to development team	1 day	4	Project team [1]	$ 160.00
3.3.	Create a development tracking document	1 day	8	Project team [1]	$ 320.00
3.4.	Develop an alpha version with main functionalities	20 days	480	Project team [3]	$ 19,200.00
3.5.	Develop a beta version for internal testing	15 days	360	Project team [3]	$ 14,400.00
3.6.	Develop different versions of release candidates	15 days	360	Project team [3]	$ 14,400.00
4.	Testing				
4.1.	Test release candidates for system specifications and bugs	20 days	480	Project team [3]	$ 19,200.00
4.2.	Fix critical bugs	10 days	240	Project team [3]	$ 9,600.00
4.3.	Define possible enhancements for future versions	4 days	96	Project team [2]	$ 3,840.00
5.	Deployment				
5.1.	Deploy the beta version of the software for the final user	2 days	16	Project team [2]	$ 640.00
5.2.	Collect feedback and implement changes	10 days	160	Project team [2]	$ 6,400.00
5.3.	Develop a Production Build version for final release	15 days	240	Project team [2]	$ 9,600.00
5.4.	Deploy the Production Build version	3 days	48	Project team [2]	$ 1,920.00
		Total	**2700**		**$108,540.00**

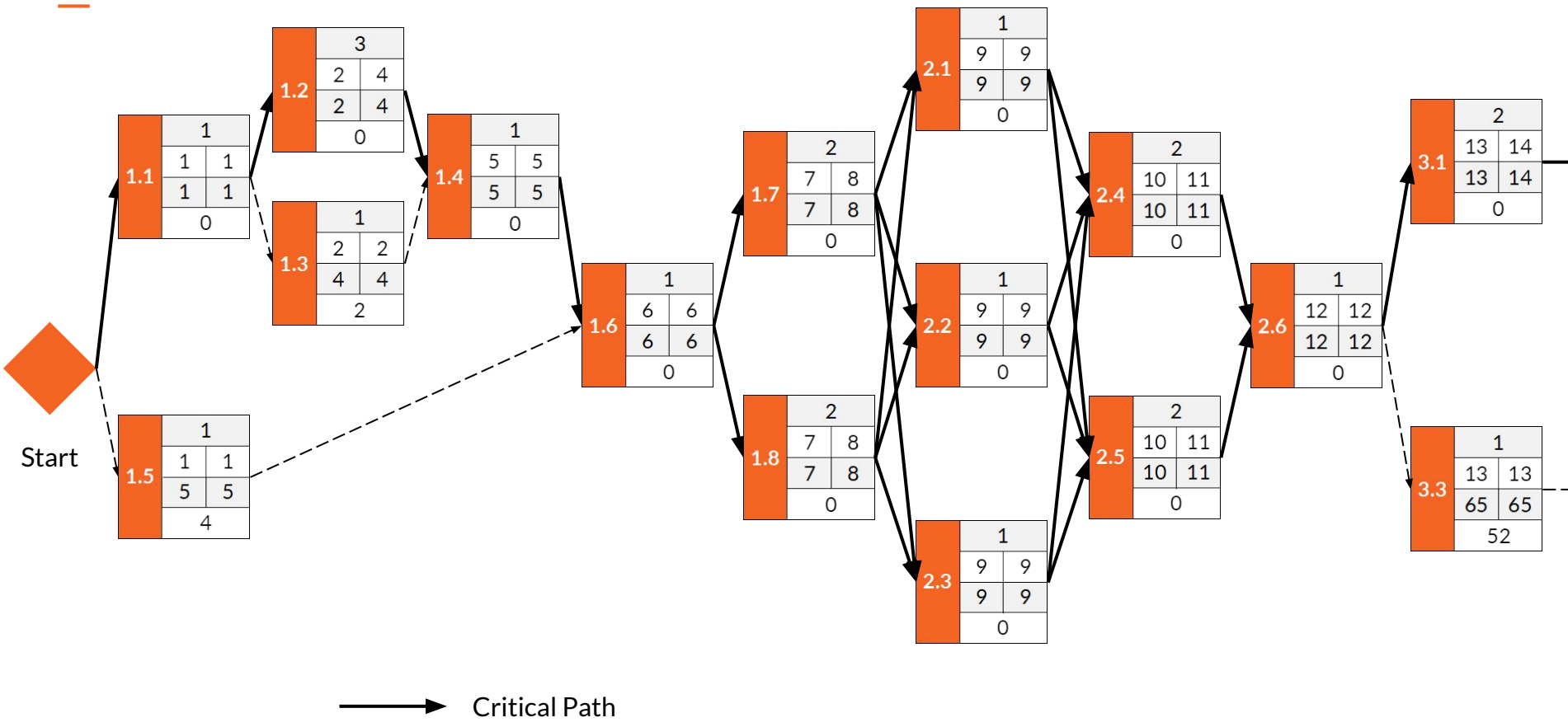


Figure 17.a: Updated Network Diagram for the entire project

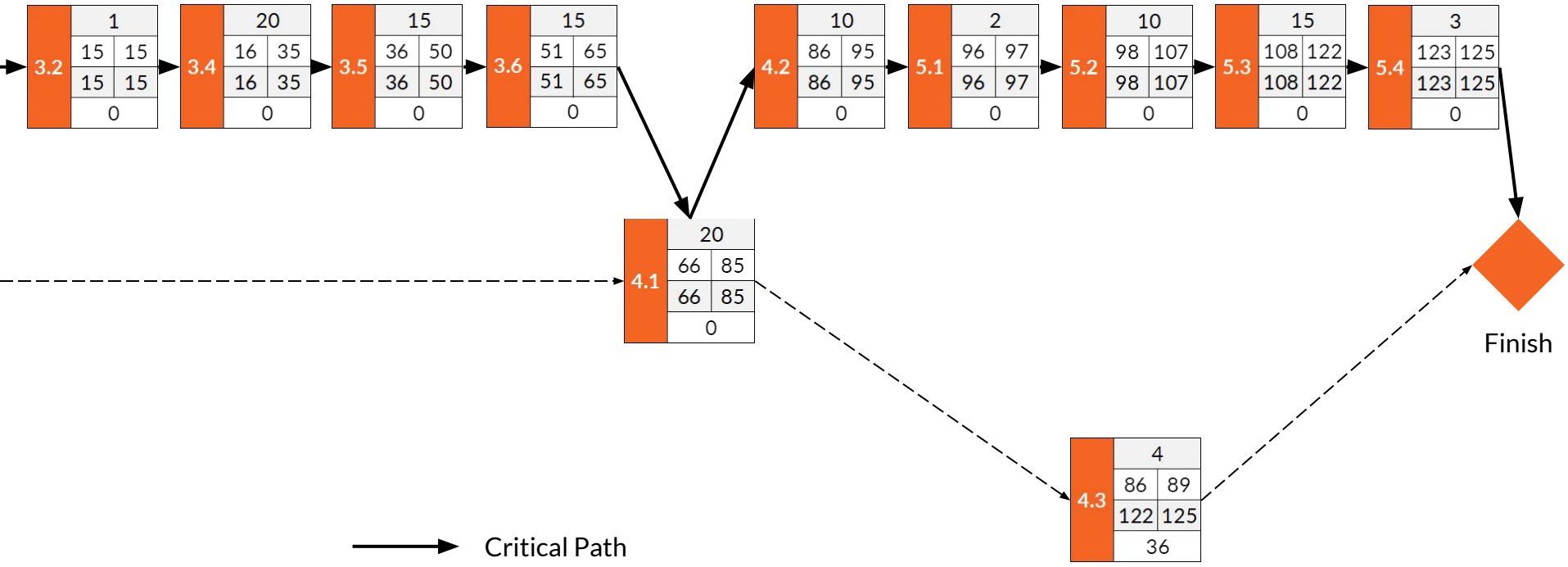
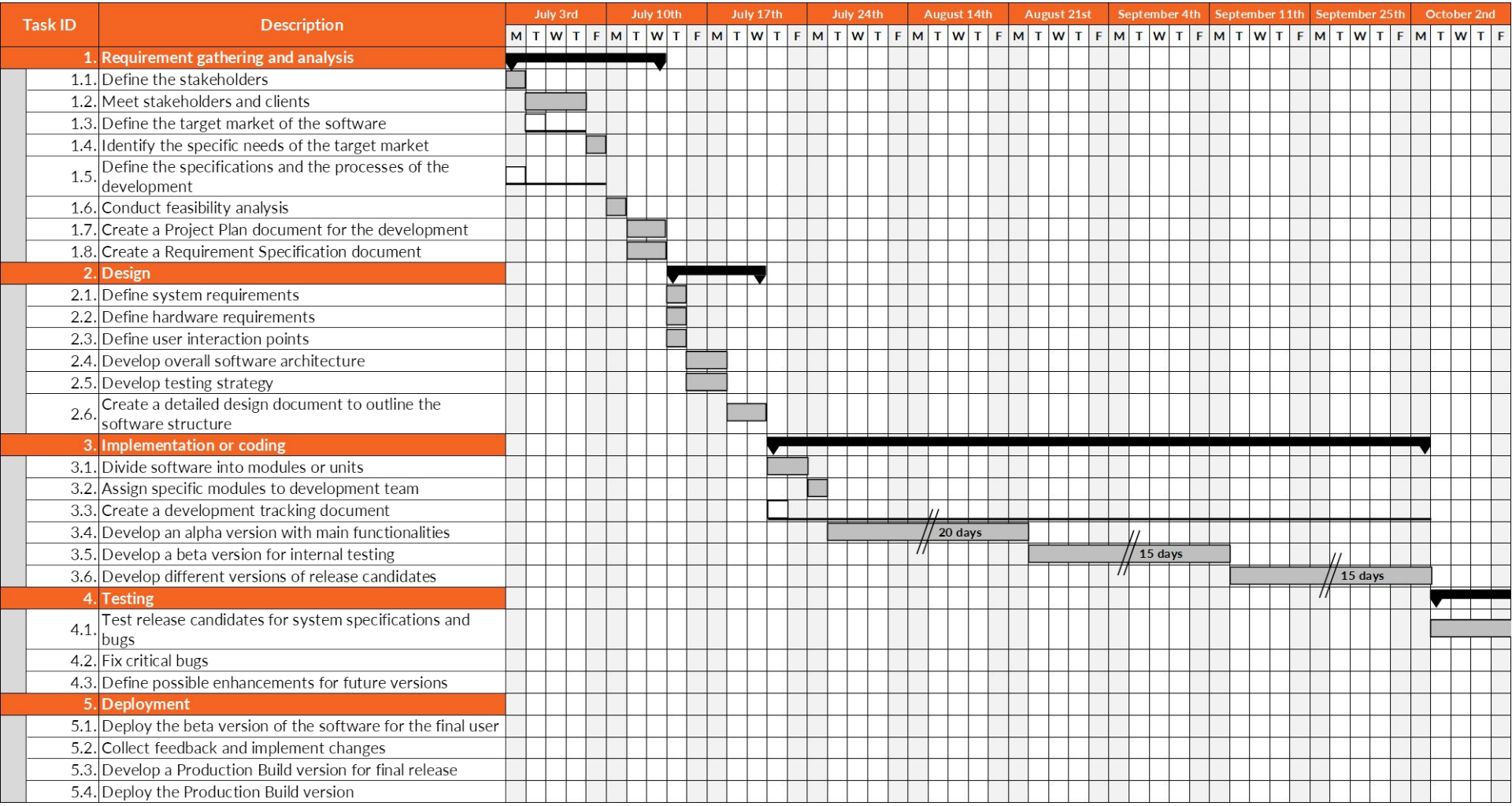


Figure 17.b: Updated Network Diagram for the entire project

With our Network Diagram in hands, we can move to the next step: the Gantt Chart. Again, no big issues here. Figure 18 shows how the Gantt Chart for our project would look like. I also included the summary tasks as they simplify the visualization of the start and end of each phase. Doing the Gantt Chart by yourself can be quite boring (own experience), but for smaller projects it shouldn't take long.

Figure 18.a:

Gantt Chart for the entire project



Task ID	Description
1.	**Requirement gathering and analysis**
1.1.	Define the stakeholders
1.2.	Meet stakeholders and clients
1.3.	Define the target market of the software
1.4.	Identify the specific needs of the target market
1.5.	Define the specifications and the processes of the development
1.6.	Conduct feasibility analysis
1.7.	Create a Project Plan document for the development
1.8.	Create a Requirement Specification document
2.	**Design**
2.1.	Define system requirements
2.2.	Define hardware requirements
2.3.	Define user interaction points
2.4.	Develop overall software architecture
2.5.	Develop testing strategy
2.6.	Create a detailed design document to outline the software structure
3.	**Implementation or coding**
3.1.	Divide software into modules or units
3.2.	Assign specific modules to development team
3.3.	Create a development tracking document
3.4.	Develop an alpha version with main functionalities
3.5.	Develop a beta version for internal testing
3.6.	Develop different versions of release candidates
4.	**Testing**
4.1.	Test release candidates for system specifications and bugs
4.2.	Fix critical bugs
4.3.	Define possible enhancements for future versions
5.	**Deployment**
5.1.	Deploy the beta version of the software for the final user
5.2.	Collect feedback and implement changes
5.3.	Develop a Production Build version for final release
5.4.	Deploy the Production Build version

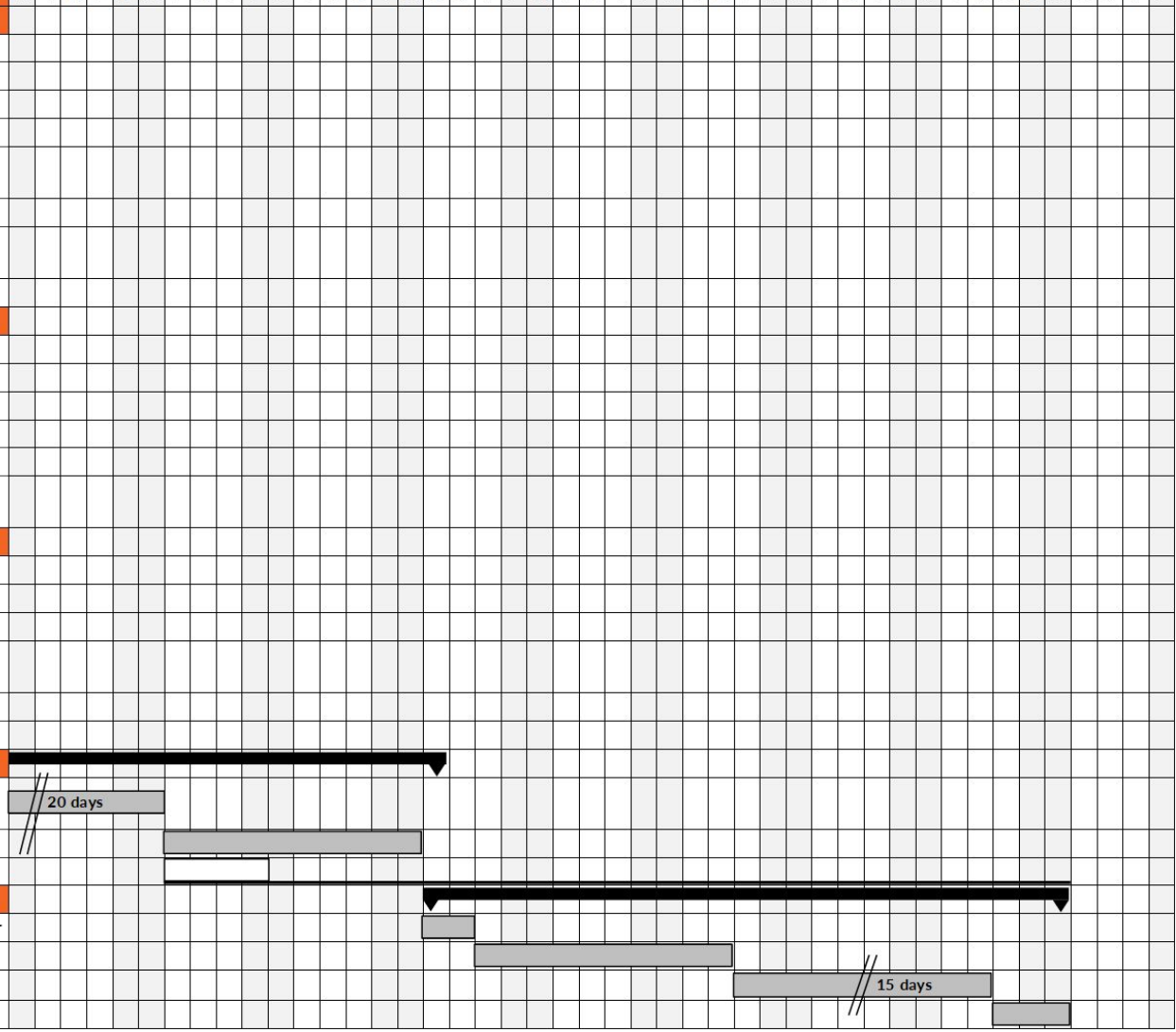


Figure 18.a: Gantt Chart for the entire project

Resources	July 3rd					July 10th					July 17th					October 30th				November 6th					November 13th					November 20th					November 27th				
	M	T	W	T	F	M	T	W	T	F	M	T	W	T	F	T	W	T	F	M	T	W	T	F	M	T	W	T	F	M	T	W	T	F	M	T	W	T	F
Project Manager	8	12	8	4	4																																		
Project Team [3]	4					4	16	16	24	24	24	24	24	14	6	48	48	48	48	24	24	24	24	24	24	8	8	16	16	16	16	16	16	16	16	16	16	16	16

Table 04:

Resource Leveling table

Finally, the last step is to carry the resource leveling process. Luckily for us, since our project is fairly linear, there are not so many opportunities for over allocation of work. Still, Table 04 shows two points where the Project Manager and the Project Team are requested to work more than 8 hours per day. The solution is to identify other days when they are requested to work less (squares in green) and transfer some amount of hours from the over- to the underallocated days. Here it is possible to do this without extending the total duration of the project because both tasks that cause the overallocation have positive floats. This allows us to play around with their flexibility and accommodate their work in days that are less busy to our workers.

Table 05:

Updated Resource Leveling table

Resources	July 3rd					July 10th					July 17th					October 30th				November 6th					November 13th					November 20th					November 27th				
	M	T	W	T	F	M	T	W	T	F	M	T	W	T	F	T	W	T	F	M	T	W	T	F	M	T	W	T	F	M	T	W	T	F	M	T	W	T	F
Project Manager	8	8	8	8	4																																		
Project Team [3]	4					4	16	16	24	24	24	24	24	14	6	24	24	24	24	24	24	24	24	24	24	24	24	24	24	24	24	24	24	24	24	24	16	16	16	16

Final Words

Hopefully this Ebook was helpful for you to understand the different components of running a schedule and cost estimation. We dealt mainly with labor estimation here, but there are other sources (briefly discussed) that must be considered when estimating your costs and schedule.

If we were to discuss every single detail and possibility extensively, this Ebook would become extremely boring for you and for me. Therefore, I opted for a general discussion of the most important topics about the subject.

I must say it was quite a challenge to come up with such an example and to build all the material to make it available, but I am glad I was able to do so. In any case, I hope I was able to make the principles of schedule and cost estimation easier to understand.

For more tips and articles on project management do follow our blog.