Estimation and Prioritization Process

Creating an Epic Backlog, Roadmap and Release Plan

written by Borjan Petreski

Working on projects in the “Complex-Chaotic” quadrant intersection of the Cynefin diagram, where today’s projects are mostly found in, can be quite challenging. Trying to gather every valuable input, feedback and requirement from a stakeholder group that is large and dispersed, lacking in communication and collaboration between its representatives, product, or feature owners, and not having a single point of decision making, can easily discourage and demotivate.

However, there is a simple structure, process; an approach that has been proven time and time again effective, categorised under different names and found in many certification programmes, it follows four phases to build a prioritised delivery plan, a roadmap on an “Epic” level of abstraction.

The Approach

The goal behind this holistic approach is to clarify the path one needs to take in building a detailed, prioritised, and approved delivery plan. Certain steps are one-off, meaning one needs to make only once before continuing on the next one, such are the elicitation and approval phases, where the requirements are broad and highly abstracted, later approved in the same state; conversely some are iterative and incremental such as the estimation and prioritisation phase, as well as keeping the delivery plan up to date in track with any affecting changes.

This approach also incorporates in itself the practice of adding buffers once at the highest level of abstraction for each item, carefully adjusting the delivery rate to follow the cadence of the team as it is averaged out after some successful iterations. Therefore, the importance of iterative and incremental adjustments of the estimation and delivery can be naturally seen here.

Bringing value to the product is the main objective of creating any delivery plan, and its careful prioritisation. A large portion of this approach concentrates on the topic of faster return of investment and the author gives one best practice, i.e. Karl Wieger’s relative benefit and penalty, that has been proven useful in many projects in referenced literature, as well as in real life experiences. The statement that high-value and high-risk items should be developed first; low-value and high risk should be avoided completely will be analysed in greater detail.

Wrapping things up with committing to the proposed delivery plan with the theory of setting up delivery date ranges that we can follow with 50% to 90% certainty will create a delivery plan that the stakeholders can approve and follow closely with the delivery managers, making only small adjustments inside the proposed and agreed upon limitations.

Elicitation of Stakeholder Requirements

The aim of this step is to get feedback from every stakeholder in terms of new features and existing functionality improvements that have been identified by their teams, either coming from end users using the system or a suggestion shared by the clients.

In this phase an invitation is sent to all stakeholders, but engagement by the chief decision makers is crucial; they would advise everyone to elicit feedback from the teams and provide a list of suggested improvements in the format: title, description, key benefits, and requester/owner (see Table 1.1). *Note: The owner can be later changed to the one responsible for a timely decision making.*

In terms of the process for collecting feedback, many techniques can be found in the BABOK guide (Business Analysis Body of Knowledge). The most challenging step in the collection of feedback from dispersed stakeholders is to gather them in one place at one time and ask for a few hours of their time. The streamlined version of this process proposes sending a document in the type of “Survey or Questionnaire”.

Initially each group of stakeholders can produce separate backlog documents that will include the template table. At a later stage, these documents should be merged and diffed to create a single proposal for a Backlog, that will later be used in succeeding phases to be built upon to create the final document.

Note: There should be a time limit on waiting for the response from the stakeholders. Typical duration should not exceed 2 weeks. This time limit should be clearly communicated, and it should be clear to everyone that after this period, any request will not be considered valid until the next time this process is restarted.

Table 1.1: Elicitation of requirements

No	Title	Description	Key Benefit	Requester
1	Please specify the title that briefly describes the feature here. *(Ex. Custom landing pages, Custom website views, etc.)*	Please provide a brief description of the functionality. *(Ex. Update the foo functionality to include the option to customise the website links used in the foobar feature using the variables that are available for the user, i.e., add Hi [First_Name] message in the invitation email and if the customer is from the IT industry pre-filter the report page to display the matching industry results.)*	Feel free to just drop your view on the benefit that you feel this feature will provide. *(Ex. Primarily this feature should increase engagement by providing a unique customised experience. This feature would also be available for our clients and would provide additional value to our existing service and help them bring unique experiences to their customers as well.)*	Stakeholder Requester/Owner name *Ex. Jane Smith*
2				
3				

Estimation and Prioritization

Once the first phase is completed and the information is collected from all stakeholders, the next step is to schedule a short session (depending on the size of the backlog it can usually vary between 30 minutes and 2 hours; anything longer than that should be split between multiple sessions) with the key decision makers to define the business value of each feature and to prioritise by their desirability.

Prior to this meeting we will go through the process of collating the feedback from the different stakeholders and identify any overlaps so that we can refine the final list of features that we will review in this session. Prioritisation would be based on estimated Business Value, Risk and Desirability.

The next step is for the affected projects developer leads, supported by the product chief architect, will estimate the effort for every requirement starting from the top of the backlog, which will provide us the total score and rank.

Estimation is a topic on its own, but some main guidelines can be followed when estimating large undertakings with still ambiguous details. One of the simplest epics can be analysed in greater depth by everyone, perhaps roughly broken down into its sub-components and estimated as usually done during Sprint Planning or Refinement ceremonies. After the estimation of all sub-components, their effort should be summed up as estimated, and be given a buffer only at its highest level (giving a buffer on each component can result in overestimating). More about buffering will be mentioned in later phases, but as a rule of thumb the following values can be used.

Certainty and suggested buffer sizes.

Certainty	Buffer Size
100%	15%
90%	25%–30%
80%	50%
50–70%	100%
<50%	200%

Certainty is the percentage likelihood that a task will be completed on time. The 100% certainty implies that similar tasks always complete on time. If only one in every two tasks tends to complete on time, then the certainty would be 50%. Certainty is the gut feeling of the developers who are on the spot. If they feel confident, they may estimate 90% certainty. If they've never seen something before and have no idea how to do it, they are more likely to estimate <50% certainty. For example, if a developer estimates a job as 1 month, with a 40% certainty of completion, then the estimate should ideally be inflated by 2 to 3 months.

After a better understanding of what estimation points mean, relative to the epic estimated in greater depth, estimations should be given to other items in the backlog. Many stakeholders struggle with the Story Point estimation system and usually they might ask for a team-day or team-months translation. When providing these values, historical values can be helpful if the teams have previously worked on a project together, but even among teams under the same program, story points scale can vary. Therefore, another (but not additional) buffer can be introduced here, and the delivery date be given as a range instead of a single date. The range can follow the buffer given for the epic, the earliest date as the original story point estimation and the latest date as the buffered one.

Prioritisation on Business Value and Risk

We often look at four factors that determine the value of the business decision:

The financial value of having the features. How much money will the organisation make or save by having the new features included in the product? Often, an ideal way to determine the value of a feature is to estimate its financial impact over a period—usually the next few months, quarters, or possibly years. This usually involves estimating the number of new sales, the average value of a sale (including follow-on sales and maintenance agreements), the timing of sales increases, and so on.

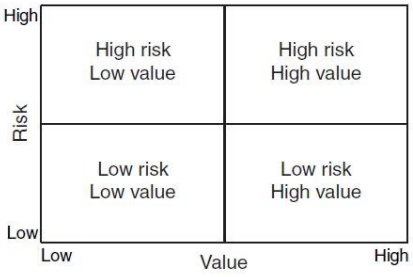
The cost of developing (and supporting) the new features. The duration of fully implementing each feature directly reflects on its cost. The longer one feature takes, the more man-hours would need to be invested. The cost rises as more experienced engineers and SMEs are engaged in its specification and development. The best way to reduce the cost of a change is to implement a feature as late as possible—effectively when there is no more time for change.

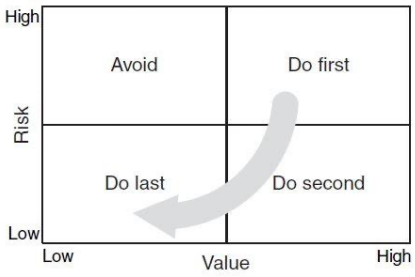
Learning and the knowledge gained by developing the features. On many projects, much of the overall effort is spent in the pursuit of new knowledge. It is important that this effort be acknowledged and considered fundamental to the project. The flip side of acquiring knowledge is reducing uncertainty. At the start of a project there is some amount of uncertainty about what features the new product should contain. There is also uncertainty about how we will build the product. We can also split this knowledge into two categories.

- **Project knowledge** is the knowledge about how the product will be created. Examples include knowledge about the technologies that will be used, about the skills of the developers, about how well the team functions together, and so on.
- **Product knowledge** is knowledge about what will be developed. It is knowledge about the features that will be included and about those that will not be included. The more product knowledge a team has, the better able they will be to make decisions about the nature and features of the product.

The amount of risk removed by developing the features. Almost all projects contain tremendous amounts of risk. A risk is anything that has not yet happened but might, and that would jeopardise or limit the success of the project. There are many different types of risk on projects, including: Schedule, Cost, Functionality, Technology and Business risks. Prioritisation to reduce or mitigate risks is reviewed in greater detail later in this post.

Consider the figures on the right which maps the relationship between the risk and value of a feature into four quadrants. At the top right are high-risk, high-value features. These features are highly desirable to the customer but possess significant development risk. At the bottom right are features that are equally desirable but that are less risky. Whereas features in the right half of the figure are highly desirable, features falling in the left half are of lower value.





The high-value, high-risk features should be developed first. These features deliver the most value and working on them eliminates significant risks. Next are the high-value, low-risk features. These features offer as much value as the first set, but they are less risky. Therefore, they can be done later in the schedule. Next are the low-value, low-risk features. These are sequenced third because they will have less impact on the total value of the product if they are dropped, and because they are low risk. Finally, features that deliver

low value, but are high risk, are best avoided. Defer work on all low-value features, especially those that are also high risk.

Prioritisation on Desirability

Separating product features into "Must Haves", "Should Haves", "Could Haves" and "Would Haves" (commonly known as MoSCoW) can provide a great deal of information about how to prioritise work for a new product release. This approach gives us a way to separate features into three categories:

- ***Threshold*** *features* are those that must be present in the product for it to be successful.
- A ***linear*** *feature* is one for which "the more, the better" holds true. These are called linear features because customer satisfaction is correlated linearly with the quantity of the feature. The better one of these features performs (or the more of it there is), the more satisfied the customers will be.
- ***Exciters*** and ***delighters*** are those features that provide great satisfaction, often adding a price premium to a product. However, the lack of an exciter or delighter will not decrease customer satisfaction below neutral.

Because must-have features are required for a product to play in its market segment, emphasis should be placed on prioritising the development of all threshold features. A product's must-have features do not need to be developed in the first iterations of a release. However, because users consider these features to be mandatory, they need to be available before the product is released. Keep in mind, though, that partial implementation of must-have features may be adequate, because gains in customer satisfaction drop off quickly after a base level of support for threshold features has been established.

Secondary emphasis should be placed on completing as many linear features as possible. Because each of these features leads directly to greater customer satisfaction, the more of these features that can be included, the better. Finally, and with time permitting, at least a few delighters should be prioritised such that they are included in the release plan.

Karl Wiegers Relative Weighting Technique

The Karl Wiegers' approach considers both the positive benefit of the presence of a feature and the negative impact of its absence. This approach relies on expert judgement. Collaboratively, but led by the product owner, the team assesses each feature being considered for prioritisation. Each feature is assessed in terms of the ***benefits it will bring if implemented***, as well as the ***penalty that will be incurred if it is not implemented***. As with estimates of story points and ideal time, the estimates of benefits and penalties are relative.

By cross-referencing the answer to the functional question with the answer to the dysfunctional question, a prospective user's responses can be reduced to a single meaning. So, if a user says she expects to be able to graph event times and that she would dislike it if it did not, we cross-reference those answers and get that she considers the feature to be mandatory. In her mind, graphing event times is a must-have feature.

To rank the features, we need to assign values for the Relative Benefit and Relative Penalty from 1 to 10, with 1 being the lowest and 10 being the highest value. For each feature, the relative benefit and penalty are summed and entered in the Total Value column. The sum of the Total Value column (in this case, 33) represents the total value of delivering all features. To calculate the relative contribution (Value %) of each feature, divide its total value by the sum of the Total Value column. For example, feature 1 has a value of 14. The total value of all features is 33. Dividing 14 by 33 results in 0.42, or 42%. This means that Feature 1 represents 42% of the total value of all listed features.

The next column, Estimate, holds the story-point or ideal-days estimate. The preference is for story points. Exactly as with Total Value, the Estimate column is summed (61, in this case) and the percentage of each estimate is calculated in the Cost % column. In this example, feature 1 is estimated at 32 story points. The three combined are 61 story points. So, feature 1 represents 53% of the total cost of these three stories (32 / 61 = 0.53 or 53%).

The final column, Priority, is calculated by dividing the Value % by the Cost %. Higher numbers represent higher priorities because they will create more value for the effort invested in them.

The method for categorising the responses is shown in the following table.

Feature	Relative Benefit	Relative Penalty	Total Value	Value %	Estimate	Cost %	Priority
Feature 1	8	6	14	42	32	53	0.79
Feature 2	9	2	11	33	21	34	0.97
Feature 3	3	5	8	25	8	13	1.92
Total	**20**	**13**	**33**	**100**	**61**	**100**	

You can see this in the table by looking at the Feature 3. This feature generates a little more than half the value of feature 1 (a Total Value of 8 compared with 14), but it does so at one-fourth the cost (an estimate of 8 compared with 32). Because of its high value-to-cost ratio, feature 3 is the highest-priority feature in this analysis.

The goal of this exam is to provide the ordering that will satisfy each stakeholder's expectations. In reality, this is a very difficult task to achieve. Most stakeholders' expectations can be conflicting but an agreement and commitment by all stakeholders is paramount. Very rarely the ordering is entirely accepted at first glance; although most of the items are ordered properly, some can be found misaligned. Certain stakeholders have given Benefit and Penalty values hoping that they are sufficient to raise the certain item higher in the backlog. However, after the initial ordering, tweaks in these two categories can be proposed and they should be allowed if stakeholder reasons are clear and accepted by all other concerned parties.

Note: This process is iterative and should be seen as producing a living document (a backlog), i.e., it can change in time as expected with any software engineering project. The revision can happen in a predetermined reoccurrence specified and agreed at the first prioritisation meeting, or per need as the items in the backlog are being completed.

Table 2.1: Prioritised Backlog

No	Title – Description - Key Benefits – Requester *(hidden for this table)*	Estimated Effort (story points) With Buffer	Translated Effort (team-days) Range with Buffer	Relative Benefit (1-10) 10 as highest benefit	Relative Penalty (1-10) 10 as highest penalty	Total Value	Value %	Cost %	Priority Points	Priority
1	See table 1.1	Story Points: 1,2,3,5,8,13,21, 40,80,100	Number of Weeks: ex. 16-20	1-10	1-10	Relative Benefit + Relative Penalty	Total Value / Sum of Total Values	Estimated Effort / Sum of Total Effort	Value % / Cost %	Rank (Current Line Priority Point against All Lines in Backlog)
2										
3										

Delivery Timetable

Within this stage we will start working on specifying the top priorities in more detail and schedule estimating sessions with the project development leads and chief architects that will enable us to define estimated release dates that would be ideally defined monthly or quarterly depending on the scope of the feature. This would again be a big picture estimate and would include the design (both architectural and UX/UI), implementation (development and testing), and feedback from User Acceptance Testing (UAT). Initially everything will be provisionally estimated, although in greater depth than in previous phases, something like the referent epic discussed in phase two; all dependencies considered; revised continuously to provide a more accurate delivery plan as the iterative development progresses.

Even if you have a reasonable understanding of the requirements and know who will compose the team, the risk of being wrong is significant. In these cases, there is either greater uncertainty or greater implication to being wrong about a delivery schedule. Because of this, it is useful to include a buffer in the determination of the schedule. A buffer is a margin for error around an estimate. In cases where there is significant uncertainty or the cost of being wrong is significant, including a buffer is wise. The buffer helps protect the project against the impact of the uncertainty. In this way, buffering a project schedule becomes an appropriate risk management strategy.

Creating a feature buffer is simple to do on an agile project. First, the customer selects all the mandatory work. This represents the minimum that can be released. The customer then selects another 25% to 40% more work, selecting toward the higher end of the range for projects with more uncertainty or less tolerance for schedule risk. The estimates for this work are added to the original estimate, resulting in a total estimate for the project. The project is then planned as normal for delivery of the entire set of functionalities.

To protect a project schedule against uncertainty, we need a way to quantify the uncertainty. When we estimate and assign a single value to a user story, we pretend that a single number reflects our expectations about the amount of time developing the feature will take. More realistically, though, we know the work may be completed within a range of durations. A team may estimate a particular user story as three-story points, knowing that three story points typically represents four or five elapsed days of work. If the story takes six days to complete, however, no one will be shocked; things sometimes take longer than planned. There normally is not much that can be done to accelerate the completion of a task, but there are an indefinite number of things that can go wrong and delay the completion of a task.

Suppose we want to be 90% confident in the schedule we commit to. One initial approach to doing this might be to estimate the 90% likely duration for each user story in the project and then use those estimates. However, if we do this, the project schedule will almost certainly be too long. The first number for each task is the 50% estimate of how long the task should take. I expect tasks to take longer half the time and shorter half the time. The second number is the additional amount of time to reach the 90% estimate. The additional time between the 50% and the 90% estimate is called local safety.

Adding up the 50% numbers gives us an expected duration of ex. an hour and ten minutes. On the other end, adding up the 90% estimates gives a total of ex. two hours and fifty minutes. The ideal plan is built using the 50% estimates and then adding a project buffer. This type of plan makes much more sense than one built entirely from summing the 50% or the 90% estimates. The plan protects the only deadline that matters: the overall project deadline.

Table 3.1: Delivery Timetable

No	Title	Description	Dependency	50% estimate	90% estimate	Estimated Date of Launch
1	Please specify the title that briefly describes the feature here.	Please provide a brief description of the functionality.	*Marketing team / SH providing feedback for a redesign*	*8SP*	*13SP*	*2-3 weeks*
2						
3						

Because our estimates are at the 50% and 90% points for each item, this means that the difference between these two estimates is about two standard deviations. The standard deviation for each item is then $\frac{w_i - a_i}{2}$, where w_i represents the worst case (the 90% estimate) for the story and a_i represents the average case (the 50% estimate) for the same story. We would like the project buffer to protect the overall project to the same 90% level that each task was protected by its own 90% estimate. This means our project buffer should be two standard deviations and can be determined from this formula:

$$2\sigma = \sqrt{(w_1 - a_1)^2 + (w_2 - a_2)^2 + \cdots + (w_n - a_n)^2}$$

Review and Final Approval

Within the final stage, the backlog with provisionally estimated dates will be presented to the Stakeholders or Stakeholder representatives by the Delivery Manager, with support from the Project Development Leads and the Software Architect for providing answers for any technical concerns. The goal of this session is to cover any potential questions and to get formal approval on the prioritised backlog in order to create the roadmap and begin the development process.

Collecting an approval for each item in the list might be a tedious process, but it will create an initial roadmap that can give a starting point and a general direction. However, certain items might be deemed unacceptable by the stakeholders, for which we need to go back to phase 2 where we will refine or reduce the scope for each requirement and continue the process as described. Some items might even go back as far as phase 1 after some time when more information is known, and the product knowledge has increased. The process is complete when the list and projects are acceptable to every stakeholder, but the backlog lives as long as the product is being developed. So, the process iterates.

Table 4.1: Final Approval

No	Title	Description	Estimated Date of Launch	Approved (Yes/No)
1	*See Table 1.1*	*Exp. Update the website widget functionality to include the option to customise the website links used in the prospecting emails using the variables that are available for the prospect.*		
2				
3				

Conclusion

Every start of a project is difficult, even more so when there are different stakeholders with conflicting requirements, which is usually the case. Finding oneself in these situations can be demotivating and even terrifying, but it should not be so. There is a certain satisfaction in knowing how much lies ahead and how much can be done. Until there are requirements and change requests for a product, that product lives, and it can be managed. Even with an existing and ongoing product, there comes a point where a certain refreshment is needed, and this process can quench that thirst.

Keeping in mind that any agile process is iterative, and one can come back to a phase at any point in time, can be both a relief and a frustration. There should be a clear set of objectives, but also boundaries set up front that will guide the execution of any process to its completion. In the end, the goal of the process is to bring a return to its investors, provide higher value to its customers and to increase the quality of the service it offers.

However, there is one greater goal in creating and standardising processes, and that is to serve the profession. Our duty to our work, to our every good endeavour, is to leave it better for the ones that will come after us. It is said that there are three traits to every good manager, the “three always of leadership”: always be deciding, always be scaling, and **always be leaving**. For anything to be ready to be handed over, it should be created with the purpose to be changed, to be improved, or to be discarded and replaced with something better - an egoless state of existence for the sole purpose of serving the current needs of its generation.