**3.1 I can assess the effectiveness of different project management techniques** Comparisons & Guides

**What is the Difference Between Agile and Waterfall?**

Agile and waterfall are two distinctive methodologies of processes to complete projects or work items. Agile is an iterative methodology that incorporates a cyclic and collaborative process. Waterfall is a sequential methodology that can also be collaborative, but tasks are generally handled in a more linear process.

Following the agile methodology, your project will move through a series of cycles throughout the lifetime of the project. The development phase, review, feedback, and then approval of the work item – either yes or no. If yes, implement and complete the task. If no, record and make any necessary changes, track and adjust the backlog or prioritization to reflect the newly acquired knowledge, and then move onto the next task or sprint.

Following the waterfall methodology is a simpler process of moving tasks through the phases of defining requirements, designing the implementation, implementing the work item, verification of implementation and quality assurance, and then maintenance of the feature in the end.

Selecting the right methodology for your projects will depend on preference and the nature of each project. Some projects require a more iterative process and others require a more sequential approach.

**What is Waterfall methodology?**

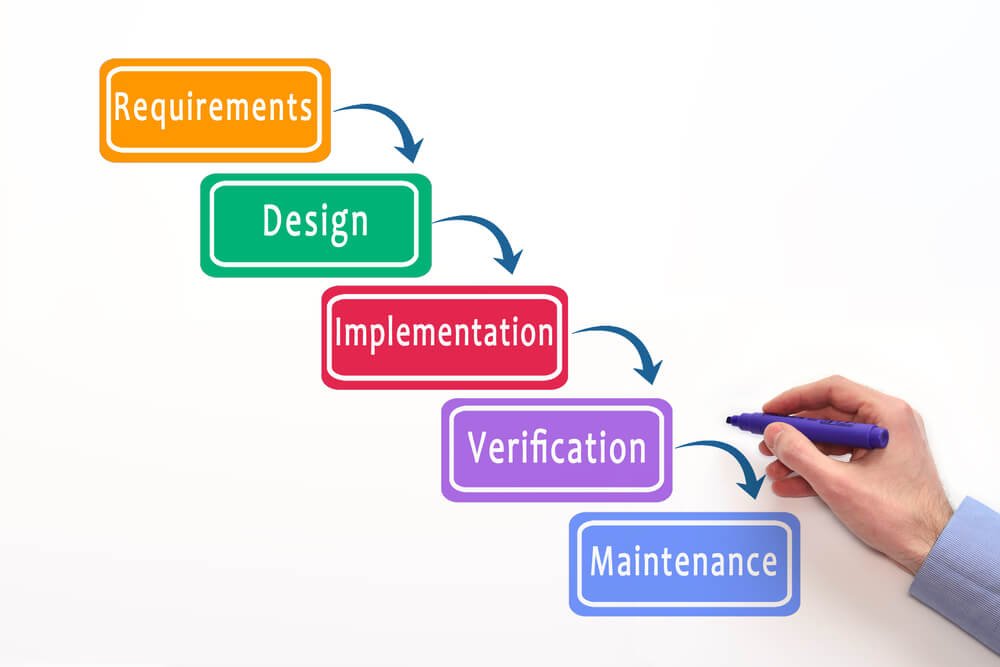
The traditional mindset of developing software is usually what is called the Waterfall model. Using the Waterfall model a project should not move from one phase to the next until the preceding phase is completely fulfilled and perfected. It is not possible to jump back to an earlier concluded phase.

The Waterfall development model is displayed in the figure below. One problem with this approach is that risk is pushed back into the development process and can (if the software is of any significant size) cause problems later in the process. This can lead to increased cost in the project or may even result in project termination (**[Kruchten, 2003](http://philippe.kruchten.com/books/" \t "_blank)**).

For any non-trivial project, it is hard to get one phase of a software [**project life cycle**](https://www.forecast.app/faqs/what-is-the-professional-services-project-life-cycle) perfected before moving on to the next phases and learning from them. For example, clients may not be aware of exactly what requirements they want before they see a working prototype and can comment upon it. They may change their requirements constantly, and program designers and implementers may have little control over this.

If clients change their requirements after a design is finished, that design must be modified to accommodate the new requirements that most likely will invalidate a part of the effort if overly large amounts of time have already been invested.

The figure below gives an overview of the typical Waterfall development model.



*The Waterfall model*

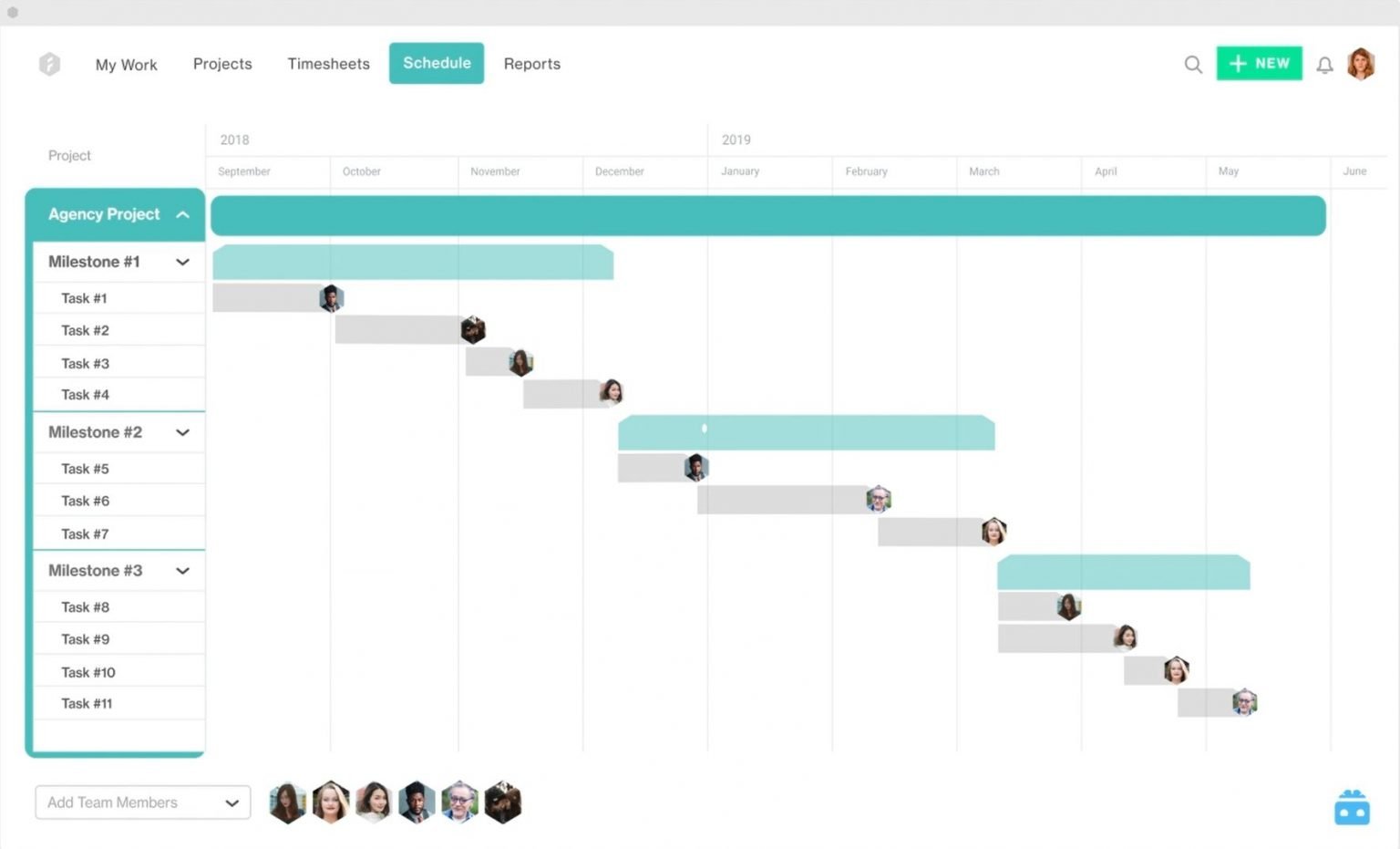
**The Waterfall development model**

Designers may not be aware of future implementation difficulties when writing a design for an unimplemented software product. It may become clear in the implementation phase that a particular area of program functionality is extraordinarily difficult to implement. If this is the case, it is better to revise the design than to persist in using a design that was made based on faulty predictions and that does not account for the newly discovered problem areas.

Unless those who specify requirements and those who design the software system in question are highly knowledgeable of the problem to solve, it is difficult to know exactly what is needed in each phase of the software process before some time is spent in the phase following it.

If you do not actively attack the risks in your project, they will actively attack you ([**Tom Gilb, 1988**](http://www.gilb.com/)).

Feedback from the following phases is usually needed to complete the preceding phases satisfactorily. It can be difficult to [**estimate time and cost**](https://www.forecast.app/blog/how-to-estimate-tasks) for each phase of the development process without doing some sort of "proof of concept" in that phase unless those estimating time and cost are highly experienced with the type of software product in question. It is worth mentioning that, with experience and learning from previous projects, it can be done with high accuracy though!



When thinking about Waterfall, it is often seen as a heavy-weight approach to project management with [**Big Design Up Front (BDUF)**](https://en.wikipedia.org/wiki/Big_Design_Up_Front) that does not have any checkpoints on the way, carried through without taking changing conditions into consideration. This is not necessarily how it must work and it does not mean you cannot assess your progress and change your course along the way. It does not mean you cannot measure where exactly you are or compare that against expectations which were set up front.

Waterfall can be far superior when the customer knows exactly what they want and nothing changes substantially, other than bug fixes. The Waterfall methodology gets a lot of discredit because the majority of customers do not know what they want (which is completely fine, as long as there is someone to guide them).

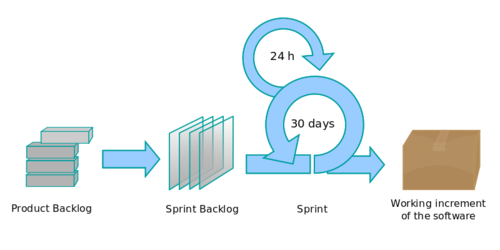
**What is Agile methodology?**

The basic idea behind any agile development process is that it is iterative or cyclic in nature, meaning that the implementation of the software happens incrementally. There are a number of different iterative approaches in existence today e.g. [**Scrum**](https://www.forecast.app/blog/beginners-guide-to-scrum), [**Kanban**](https://blog.forecast.it/6-tips-to-work-more-efficiently-with-kanban-boards-the-new-design-of-workflow?__hstc=123325972.80851da4220a66eb92061738465d4fab.1637574637784.1649758514169.1649763225299.228&__hssc=123325972.1.1649763225299&__hsfp=570208402), RUP, Extreme Programming, Rapid Application Development, etc. As mentioned earlier I will focus on Scrum since this has gained the most media popularity amongst the methods. 

**Agile development model**

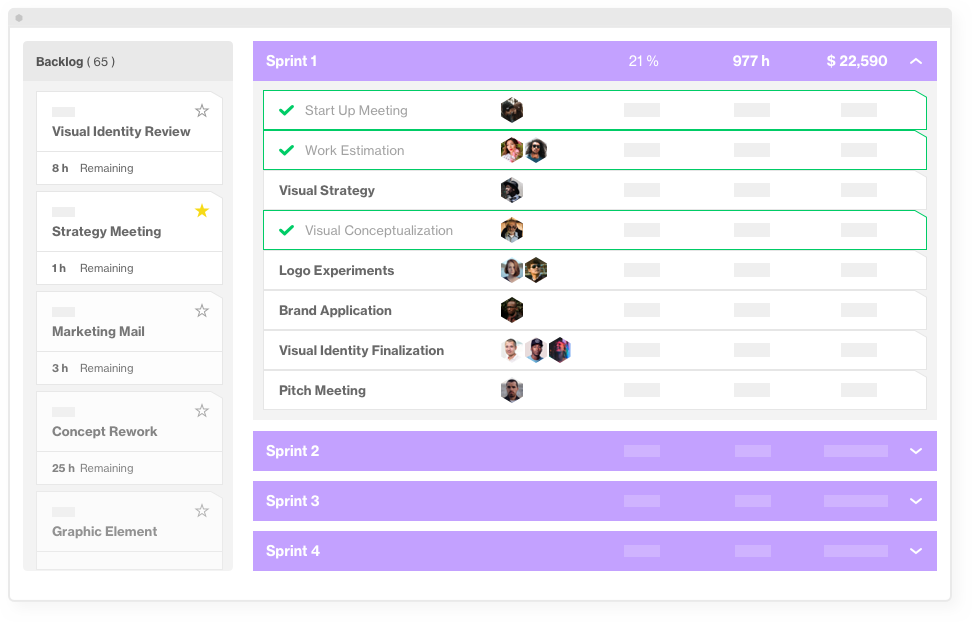
Scrum is a way for teams to work together to develop a product. Product development, using Scrum, occurs in small pieces, with each piece building upon previously created pieces. Building products one small piece at a time encourages creativity and enables teams to respond to feedback and change, to build exactly and only what is needed.

The figure below shows an example of the Scrum approach towards software development.

*Example of Agile with Scrum*

Rather than creating all tasks and schedules upfront, all time is time-boxed into phases called [**sprints**](https://www.forecast.app/blog/why-do-you-work-in-sprints). Each sprint has a defined duration, which is usually a few weeks, with a running list of deliverables, planned one sprint in advance. Deliverables are prioritized by business value as determined by the customer. If all planned work for the sprint cannot be completed, work is reprioritized and the information is used for future [**sprint planning**](https://www.forecast.app/blog/sprint-planning).

In the figure above, it is seen that work is taken from an ordered list, the backlog, prioritized into sprints, and then developed. Items at the top of the backlog are more refined than items further down. In order not to waste time and money, backlog items should only be expanded enough so that proper planning decisions can be made. For each backlog item, a user story, the implementation, and testing are done in the sprint.



*An example of a sprint dashboard in Forecast*

As work is completed during each sprint, it is continuously reviewed and evaluated by the customer. All work that is completed should be defined as shippable, meaning that it shall work as intended and have been through testing. As a result, agile relies on a very high level of customer involvement throughout the project since they need to validate the functionality and accept that it works as intended. This cycle goes on until the product is complete and ready to be deployed or deemed acceptable by the business.

Comparing this approach to the Waterfall development model, the business value is delivered constantly and at regular intervals, whereas the Waterfall model delivers all business value at the end of the project. By choosing the [**Scrum approach**](https://www.forecast.app/blog/implementation-of-scrum-7-steps) it can be easier to maintain control of the project with regards to cost, the return on investment (ROI), and overall business value. It makes the project much more transparent. In a Waterfall approach, it can often be extremely complex to see the total benefit of the project, until development is very close to being complete. This is not necessarily a bad thing about Waterfall though! More on that in the next posts.

**Cost, functionality, and time**

Like any human undertaking, projects need to be performed and delivered under certain constraints. Traditionally, these constraints have been listed as cost, functionality, and time/scope. These are also referred to as the [**Project Management Triangle (PMT)**](https://en.wikipedia.org/wiki/Project_management_triangle), where each side represents a constraint. One side of the triangle cannot be changed without affecting the others. This relationship is illustrated below.



The time/schedule constraint refers to the amount of time available to complete a project. The cost constraint refers to the budgeted amount available for the project. The functional/scope constraint refers to what must be done to produce the project's end result. These three constraints are often competing constraints: increased functionality typically means increased time and increased cost, a tight time constraint could mean increased costs and reduced scope, and a tight budget could mean increased time and reduced scope.

Depending on what type of company/project you are working for, two of these will be more important than the third. If it is a fixed price project, the cost is a constraint that cannot be changed. In that case, either functionality or time will be the one that suffers.

In other words, you have three options:

* Develop a product quickly with high quality or much functionality, but then it will not be low cost.
* Develop a product quickly and with a low cost, but then it will not be of high quality or have much functionality.
* Develop a product with high quality or much functionality and with a low cost, but then it will take a relatively long time.

Quality is not a part of the PMT, but it is the ultimate objective of every delivery. Hence, the PMT implies quality.

Many project managers are under the notion that 'high quality comes with high cost', which to some extent is true unless the amount of functionality is the parameter that suffers. In all cases, using low-quality resources to accomplish project deadlines often does not lead to the success of the overall project. Like with the scope, the quality will also be an important deliverable for the project.

In agile, functionality or cost are usually the ones that suffer. Time is usually less of a factor because iterations continue until the team/project/customer believes the product is ready to be released. It is more about continuously getting functionality out the door so the project can stop at any time and still have a working product.

**Pros and cons of choosing Waterfall**

There are advantages and disadvantages to any method, and Waterfall is not an exception. Here’s what you should take into consideration before deciding on one.

**The Positives of Waterfall**

Planning, designing, and architecture are more straightforward since the customer and the developers agree on what will be delivered early in the life cycle. Since the full scope of work is known, progress is also more easily measured.

Depending on the phase of the project all team members do not need to be locked down in the project. As an example business analysts (BAs) can proceed to work on other parts, or other projects entirely, after requirements are written and vice versa, developers do not need to be involved before the BAs actually know what needs to be delivered. Based on requirements from BAs, testers can prepare test scripts while developers are coding along.



After the requirements phase, there is no strong need for customer presence and thus they can focus on other more pertinent things. The customer basically only needs to be involved for status reviews and approvals.

If multiple systems/components/software packages need to be integrated and launched at the same time (internal and/or external) an early design is often required. This is thus suited perfectly to the staged approach of Waterfall.

Architecture, a trade often forgotten or neglected in modern software development, can also be designed better for scalability, overall coherence, and completeness since there is a more thorough understanding of all parts of the entire system. Code does not need to be rewritten over and over again, which sometimes can be the case in Scrum and which sometimes results in a piecemeal system.

Project resources are not as critical, since planning and documentation have already been done. Thus, if a developer drops out, it is easy for a new resource to fill the position and get up to speed quickly and follow the plan.

Because the Waterfall method requires upfront planning, software can be launched quickly after development. [**Estimation of complete timetables and budgets**](https://www.forecast.app/blog/whats-wrong-with-using-spreadsheets-for-project-cost-estimation) can be done more accurately, which definitely tends to please customers.

**The Negatives of Waterfall**

Gathering and documenting requirements so the customer understands what they mean is very difficult. Customers are often intimidated by details, and specific details must be provided early in the project before progress can be made. Customers are often not able to visualize an application from a requirements document. Wireframes and mockups will help, but most end users have difficulty understanding these elements in enough detail to clearly visualize the end product.

This problem means that customers might be dissatisfied with their delivered product once it is complete. As all deliverables are based upon documented requirements, customers may not see what will be delivered until large parts of the project are finished. At that point in time, changes are difficult and costly.

Feedback and testing are deferred until bigger parts of the project are complete. This means that problems or inadequacies can be harder to change without substantial amounts of time and effort.

**When to choose Waterfall over Agile Scrum**

Some things to consider when to use Waterfall vs Agile:

* Project size and complexity is large, e.g. large system integration involving many different systems
* Customer cannot commit to extensive involvement
* External system integrations are numerous involving many points (incl. legacy systems)
* Budget and/or schedule are fixed or difficult to modify
* Full-featured project/product required before launching
* Developers are in need of guidance, management, and supervision
* The customer does not expect major changes in the scope
* The customer knows exactly what they want

**Pros and cons of choosing Agile Scrum**

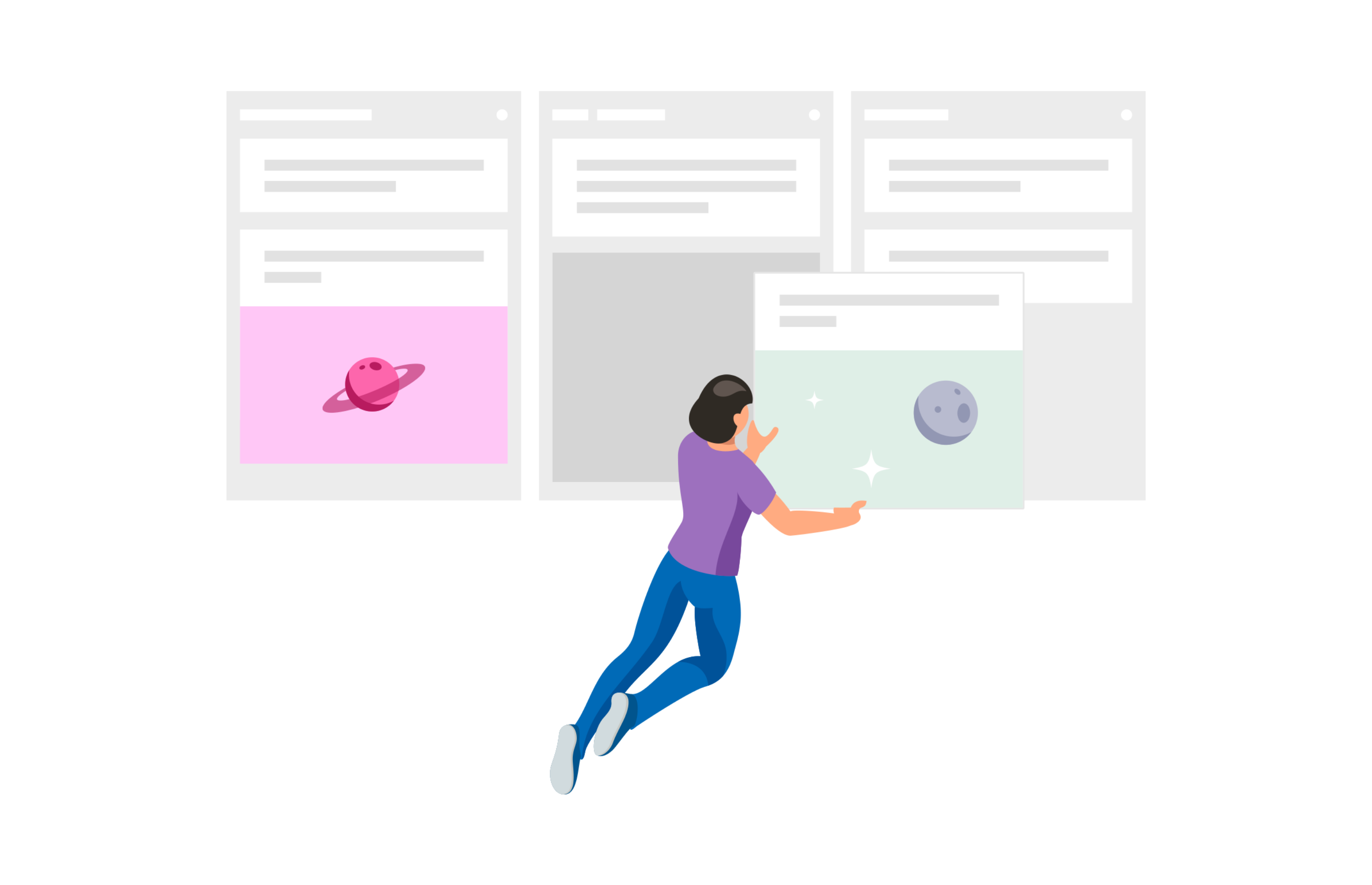
The ability to discern between Agile and Waterfall for different types of projects and clients can result in several business benefits. It could mean being able to quickly respond to new opportunities, react to change, and adapt to challenges in ways that you never imagined. So let’s look at the pros and cons of Agile.

**The Positives of Agile with Scrum**

Since the customer has frequent and early opportunities to see the work being delivered and make decisions and changes throughout the project, the customer gains a sense of ownership. The customer works extensively and directly with the project team throughout the project.

Scrum will produce a simple working base version faster than Waterfall, which for instance makes it suitable for prototyping.

Development is user-focused and changes due to problems or inadequacies can be remedied quickly and less costly. Since the design is flexible the project will run a more evolutionary and less stringent life. This has a number of advantages, especially in project environments where development needs to be able to respond to changes in requirements rapidly and effectively.



Scrum can be especially beneficial in situations where the end-goals of projects are not clearly defined. If it is not possible for the customer to provide a clear goal early on in the project Scrum will work well since this will be clarified as the project progresses.

Using Scrum there is typically more interaction and communication, which often takes precedence of design. Since stakeholders are more involved, it is especially conducive to teamwork-oriented environments. Different developers work on different modules throughout the development process and then work to integrate all of these modules together into a cohesive piece of software at the end of the project.

**The Negatives of Agile with Scrum**

Not all customers have either a wish or the time to be highly involved in the project. Scrum works best when all members of the team are completely dedicated to the project. Thus it is not generally a good idea to have less than 100% dedicated resources for this type of project.

Scrum focuses on time-boxed delivery and frequent reprioritization and thus some items might not be completed within the allotted project timeframe. Additional sprints might be needed to become feature-complete, which of course will add to the total cost. Also, the high degree of customer involvement often leads to additional features requested throughout the project, extending the time to complete.

Customers often have difficulty keeping up with Scrum’s rapid development iteration and testing cycle. This can put the project at risk, since the longer developers go without feedback, the harder it is to recover if they are not developing what the customer expects. Since Scrum is highly flexible, it simply does not have the structure that the Waterfall method has. Scrum projects tend to be hard to predict, from [**timelines**](https://www.forecast.app/blog/how-to-create-a-project-timeline) to [**budgets**](https://www.forecast.app/blog/how-to-create-a-project-budget). Without a concrete plan and complete requirement set, everything remains a bit vague.



The close teamwork in Scrum is easiest to manage when the team members are located in the same physical space. Furthermore, the intense collaboration and less than complete requirements result in catastrophic problems if key team members leave the project.

Scrum has less emphasis on understanding the finished system as a whole early in the project which can lead to a reduction in overall system quality and performance. This is especially true for larger-scale implementations, or with systems that include a high level of integration points.

This [**method of development**](https://www.forecast.app/blog/project-management-methods-for-software-development-teams) can be quite time consuming, much more time consuming than the Waterfall method. There is a lot of rewriting code due to changing requirements and bolted on functionality. Often the requirements are not clear from the beginning, which can result in a suboptimal architecture that is not flexible enough to meet the final requirements. Often there is not much time spent getting into the true details and really fleshing out what the users need before coding is commenced.

As the initial project does not have a definitive plan, the final product can be grossly different than what was initially intended. This also relates to the inability to give an exact launch date once a contract is signed.

**When to choose Agile with Scrum over Waterfall**

Some things to consider:

* Project size and complexity is small or not complex (can be used on larger projects under certain circumstances) e.g. smaller eCommerce, websites, app development
* The customer is available frequently through the duration of the project
* The customer is able to change the scope of the project during the progress
* External system integration is simple or not needed
* Budget and/or schedule are flexible and changes are welcomed by the customer
* Rapid prototyping and deployment required. Project/product can be launched without being feature complete
* No clear picture of what the final product should look like
* Developers are skilled, adaptable, and able to think independently without constant supervision
* When the product is intended for an industry with constantly changing standards

**Which method is better?**

As agile has become increasingly popular, organizations all over the world are scrambling to adopt this paradigm, mainly in the form of Scrum. There have also been realizations where using agile has proven to make the projects more expensive/take longer/have less features than if a sequential model like Waterfall had been chosen.

When it comes down to it, neither the agile Scrum method nor the Waterfall method is inherently better than the other. That being said, each method does have its merits.

Waterfall tends to be best for projects where it is not likely that many changes will be made throughout the development process or projects that cannot afford mistakes. Scrum tends to be a better option for smaller projects where changes are either rapid or likely to be made during the design process or with projects that can be modularized.

In Scrum, [**scope creep**](https://www.forecast.app/blog/scope-creep) is not a problem because it is all about getting functionality out the door, in the right order, based on business needs. But in practice, Scrum can become disjointed and directionless, an excuse for the absence of internal processes and organizational chaos. Scrum is often assumed to not need a strategic view, which is a complete and utter misconception.

If project teams are located across sites and timezones, coordination of work needs to be relatively detailed to avoid confusion and wasted time. In this instance Waterfall is likely to be the most appropriate method, offering clear deliverables and project milestones and dependencies.

When projects require unique specialist skills and these resources are not immediately available, good planning is required. If this is costly or difficult to organize, it’s important to ensure that the resource is fully utilised during its scheduled usage. For this, Waterfall is a better approach, as planning is much more reliable, to ensure maximum efficiency of use.

While each model has its strengths and shortcomings, misconceptions reign over both. It is either disorganization due to a lack of discipline or unnecessary complexity rooted in an irrational belief that one can eliminate risk. Efficient and mature enterprises understand when to apply each of these methods. In practice, organizations with successful development cycles appear to employ a hybrid approach, taking a little bit from each methodology.

**The best of both worlds**

The hybrid approach could be conceived as a Waterfall type analysis and design, initially, but then followed by Scrum as a production method. It gives you an overview of the complete project, but allows development with constant feedback based on the requirements of the users. It allows the team to gain insight into the large picture with upfront analysis and design while permitting incremental changes during the development and implementation phases.

The team can deliver complete, but focused functionality which comprises a small part of the whole. This will keep the business engaged and responsive, while at the same time ensuring that the BA and tester team(s) have a better feeling with the progress of the project as well. This ensures that proper planning can be done and realistic estimates set for the entire project.

Adoption of an overall architecture approach to minimize requirement gaps and to eliminate solution gaps is required. Solution workshops are to be conducted with all stakeholders e.g. business, marketing, IT, to ensure congruence in the actual requirements and final solution.

Change control and risk has to be strictly enforced, in order to control and manage scope creep. Major changes have to be taken to a review board with key stakeholders, especially for large or transformational projects.

And finally, executive sponsorship has to be strong for all but the smallest projects to succeed!

Really, when it comes to choosing a method there is not a right or wrong choice. You just need to understand which method is better suited to your project and your needs, be it Waterfall, Scrum or the hybrid approach.

Different situations obviously call for different approaches. But one thing that is for certain is that using either methodology vastly will increase success compared to not following a process at all or just going ad-hoc.

Blog by Forecast