



Applications Development and Emerging Technologies

Dr. Lumer Jude Doce

IT/Physics Dept.

MSU- GSC

Software Development Approaches



Software Development Methodology

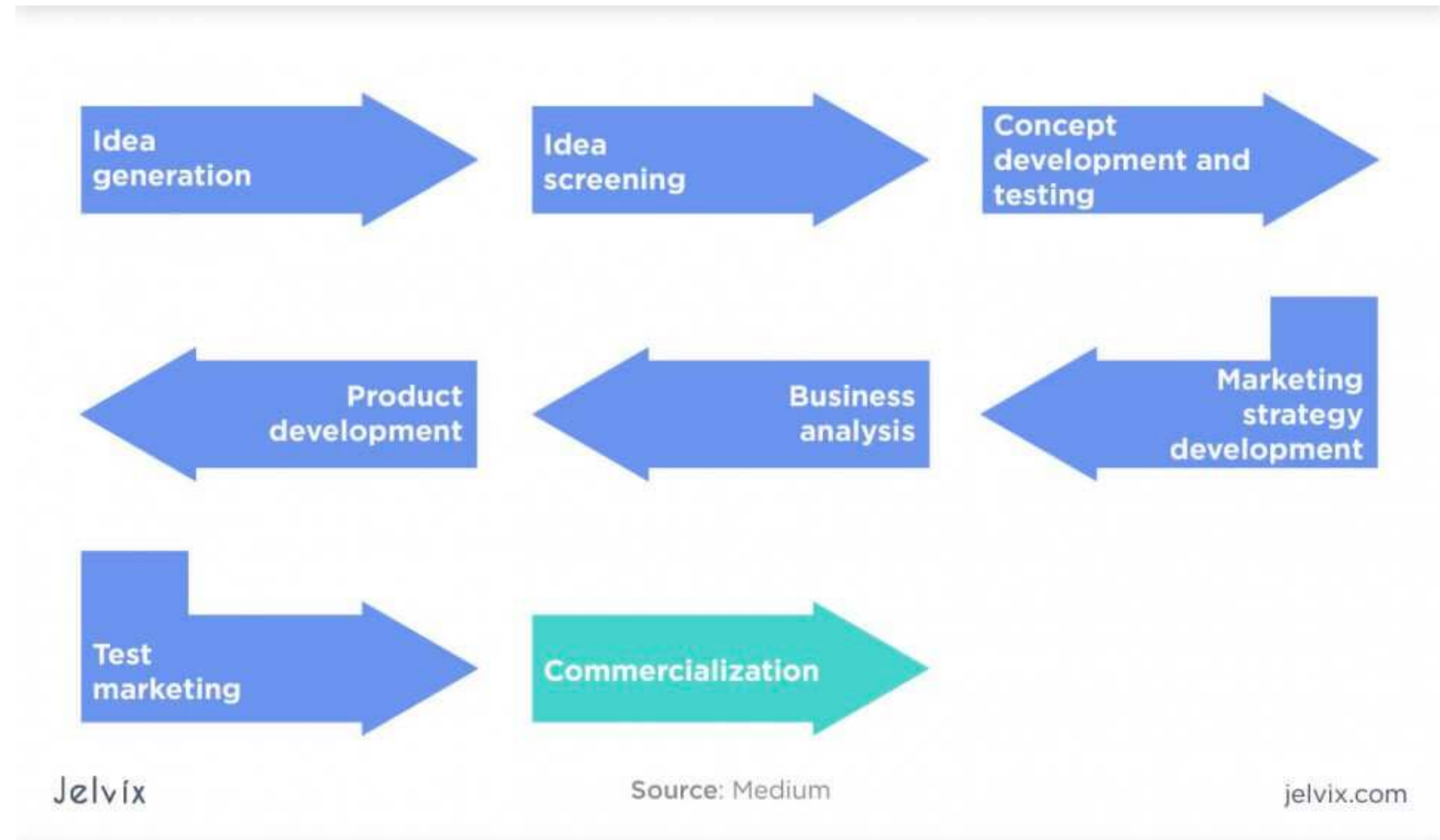
- Structured processes involved when working on a project.
- It is a blend of design philosophies and pragmatic realism that stretches back to the early days of computing.
- The goal is to provide a systematic approach to software development.
- Implies dividing the workload into specific segments for its faster management.

Software Development Methodology

- It also describes management methods used to increase the delivery efficiency, cooperation between development, design, testing, and other teams, as well as the product owner.
- Methodologies also regulate the participation of involved parties outside of the team, like clients and stakeholders. Project participants know how and when to control the project without interfering with the team.
- Each methodology features a set of metrics and critical thresholds that showcase the team's efficiency.
- Isn't just a set of instruments but a mindset for digital transformation.

Software Development Process

- Unites several stages of product conception and execution.



Why adhere to a methodology?

- It is crucial to choose a software development methodology and apply it with discipline throughout the project.
- Without structured guidance, developers can suffer from customers' ever-changing requests, and even more so when there are miscommunications. This leads to frequent revision in the software without considering the overall implications of the project.
- The result? Wastage in time, money, and effort with the risk of producing a subpar application that doesn't bring much to the table.

Why adhere to a methodology?

- Software development methodologies are developed to benefit both the development team and customers. Choosing the right one ensures that discussions are conducted on proper channels, and decisions are made after evaluating all factors.
- Using a software development methodology allows the team to cut down on inefficiency and provide a more accurate delivery timeline.
- It prevents the team from reacting to every input, but instead, allows them to be more organized and structured when dealing with spontaneous changes.

Choose the one that suits the project (or you)

- Most of the methodologies fall can be categorized as either a waterfall, iterative or continuous model.

Choose the one that suits the project (or you)

Waterfall

- A waterfall methodology follows a fixed sequence in implementation. Stages of development are defined in a rigid, sequential manner.
- This model was highly popular during the early days of programming due to the certainty in project scope. However, the rigidity of its structure also contributes to a high failure rate for many projects.

Choose the one that suits the project (or you)

Iterative

- The iterative model offers an alternative for software development that's less focused on rigid documentation but provides room for constant revisions.
- It uses multiple sprints to quickly build and test ideas to ensure they are relevant to users. As such, problems get fixed early on, and the team stays within the project's goals.
- Agile and Scrum are two of the most popular iterative software development methodologies.

Choose the one that suits the project (or you)

Continuous

- The continuous model is inspired by the Toyota Production System. It is about minimizing interruption or ensuring the flow between different phases of development.
- The goal of the continuous software development approach is to avoid wastage and improve the efficiency of the various phases.

Agile Methodology

- Agile focuses on how to satisfy the users instead of emphasizing documentation and rigid procedures.
- With Agile, tasks are broken into short sprints that take about 1 to 4 weeks to complete.
- It's an iterative model that involves multiple tests as the development progresses. Developers continuously seek feedback from customers and make changes to the software.
- Communication is the priority in Agile, particularly between developers, customers, and users.

Agile Methodology

- Agile is based on the following fundamental manifesto, which is defined by four values:
 - Priority is given to individual interactions over tools and processes – people come first;
 - Collaboration over negotiation: the team strives to find the compromise with the product owner, not to negotiate the best one-sided contract;
 - Changes over planning: long-term goals are respected but pushed aside if needed.
 - Product over complete documentation: developers should spend more time writing code than reports.

Agile Methodology



Agile Methodology



Pros & Cons of Agile Development Methodology

Pros:

- ✔ Software has minimal defects due to the iterative effort in testing and fine-tuning.
- ✔ Clarity between team members during development, thanks to the frequent and transparent development.
- ✔ Changes in project requirements are easily addressed and work on with little impact on the timeline.
- ✔ An overall improvement on the deliverable's quality.

Cons:

- ✘ The team can sometimes lose focus due to overwhelming change requests.
- ✘ Documentation takes a back seat in Agile, which can be a problem later on in development.
- ✘ Agile focuses on discussions and feedback, which can be too time-consuming for the team.
- ✘ Due to its non-structured approach, Agile requires experienced developers who can work independently.

Agile Methodology

- The Agile software development methodology is ideal for projects with fast-changing requirements. If you're building software in a new niche, you'll want to use Agile.
- It works best to implement the additional ideas as you learn more about the market needs.
- This assumes that your team of developers is highly-independent and comfortable working in a fast-paced, non-structured environment.

Waterfall Development Methodology

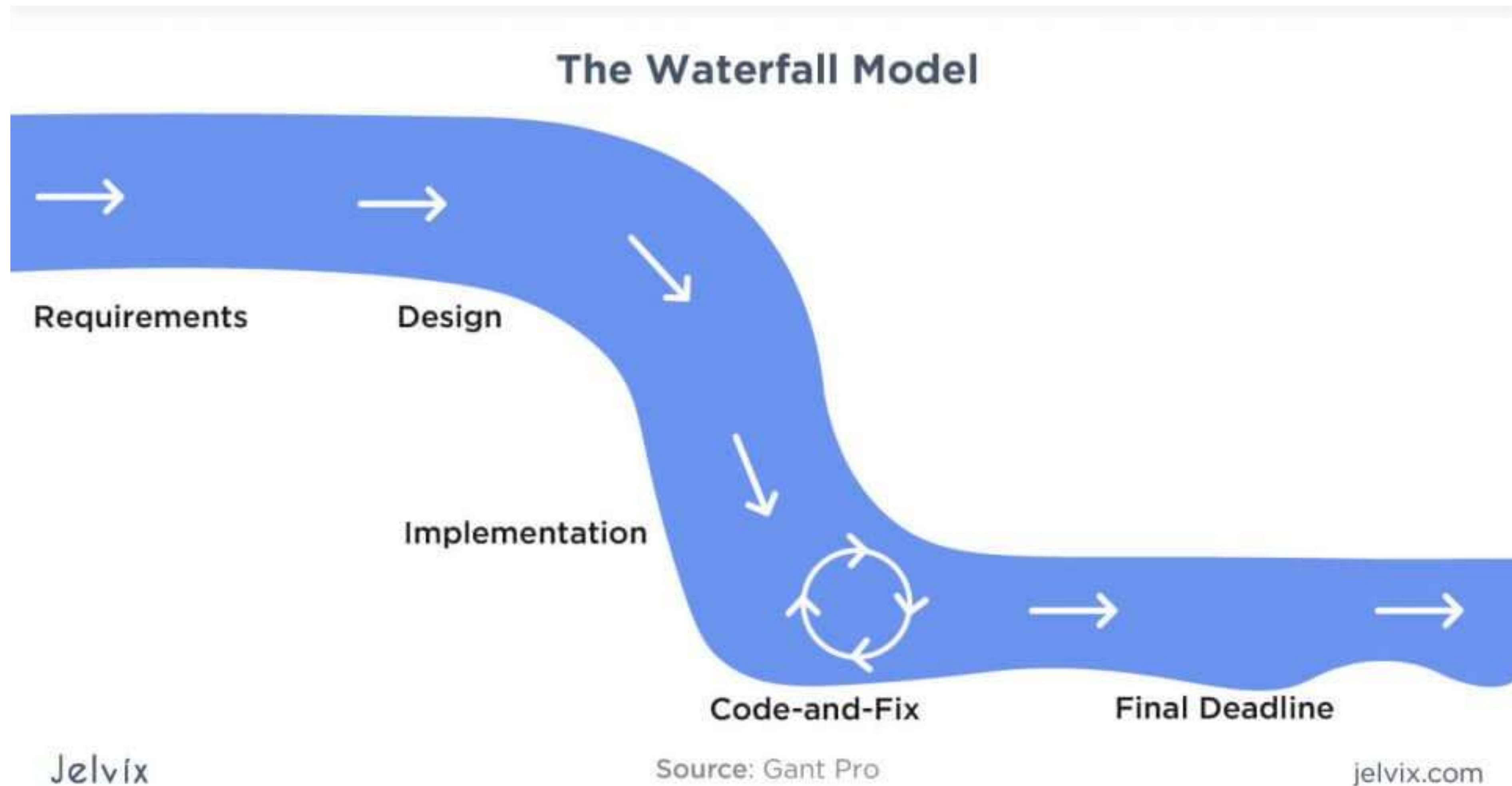
- Despite decades since it was first used, the waterfall methodology is still relevant in some projects today. It is a simple, linear method where development stages are arranged into sequential, cascading processes.
- The waterfall development methodology is easily understood, which makes it popular for teams with lesser design experience. Each stage must be completed before moving on to the next. For example, all the requirements must be established before design can commence.
- Just like how a waterfall flows in one direction, there's no going back in this approach. This makes waterfall a non-flexible method and to be avoided for projects with rapidly-changing requirements.

Waterfall Development Methodology

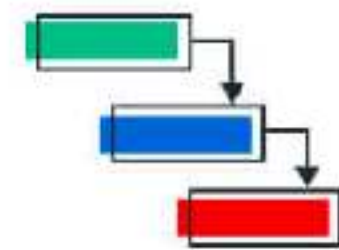
Typical Phases

- Gathering requirements: this is the stage where the team defines the product's functionality, interface, performance, checks its connection to business goals.
- Design: software development teams pick tech stacks, determine the databases that they prefer, and prepare the technology. UX and UI designers define user personas and create the interface.
- Build: developers write code for software functionality (user-end and server-end).
- Deployment: the code is tested and uploaded to the server.
- Maintenance: the team supports the final version of the application.

Waterfall Development Methodology



Waterfall Development Methodology



Pros & Cons of Waterfall Development Methodology

Pros:

- ✓ The linearity of the waterfall model makes it easy to understand, particularly for new developers.
- ✓ All specifications and deliverables are spelled out before the development commence.
- ✓ There's no room for miscommunicating information with the waterfall model as it's clearly defined in each stage.

Cons:

- ✗ It doesn't include customer feedback in the early phases, which increases the risk of the project veering off target.
- ✗ Testing is only executed at the end of the development. Some problems are harder to fix at a later stage.
- ✗ The rigidity of the waterfall model gives no room for changes, making it unsuitable for complex projects.
- ✗ The team can spend too much time on documentation instead of delivering solutions that solve the user's problems.

Waterfall Development Methodology

- Use waterfall only when you have a project with a clearly-defined scope.
- This software development methodology is not suitable for development that involves many unknowns.
- Waterfall is ideal for projects with predictable outcomes and when you have a team of inexperienced developers.

Lean Development

- Lean development is born out of the lean manufacturing principles by Toyota. It focuses on minimizing wastage and increasing productivity. With the guiding principles, developers avoid non-productive activities while delivering quality in their tasks.
- The Toyota-inspired methodology also emphasizes continuous learning and deferment of decision. It empowers teams to keep an open mind during the course of development and consider all factors before finalizing a decision.
- With the lean methodology, developers are tasked to identify bottlenecks that could hamper the process. The goal is to establish an efficient system that works flawlessly. The methodology also emphasizes human respect, which means communication is key to enhancing team collaboration.

Lean Development



Lean Development



Pros & Cons of Lean Development

Pros:

- ✓ Cuts down wastage in the project, such as redundant codes, unnecessary documentation, and repetitive tasks.
- ✓ The overall cost of development is reduced with the lean principles.
- ✓ Time-to-market for the software is shortened as lean development promotes efficiency.
- ✓ Increased motivation amongst team members as they are empowered with more decision-making authority.

Cons:

- ✗ For lean development to be successful, you'll need a team of highly-skilled developers, which isn't easy to put together.
- ✗ Less-skilled developers can be overwhelmed by the responsibilities and loss of focus on the project.
- ✗ Detailed documentation is needed, which places an enormous burden on the business analyst.

Lean Development

- With the Lean software development methodology, developers are tasked to identify bottlenecks that could hamper the process.
- Its principles of minimizing wastage and increasing efficiency mean you'll be leveraging a small team to achieve tremendous results.
- Lean development becomes less practical for larger projects, as you'll need a larger team to take on the tasks.

Prototype Model

- Instead of developing a full-fledged software, the prototype model allows developers to work on the prototype version of the final product. The prototype is then made available for customer testing, evaluation, and feedback.
- Based on the gathered feedback, the prototype goes through several iterations of refinement until it's deemed satisfactory by the customer. The appeal of the prototype approach is its rigorous evaluation that uncovers possible issues before actual development begins.
- This approach's success lies not only in the development team but also in how well they communicate with the customers in carrying out the test. It's also worth mentioning that the developers often bear the cost of building the prototype.

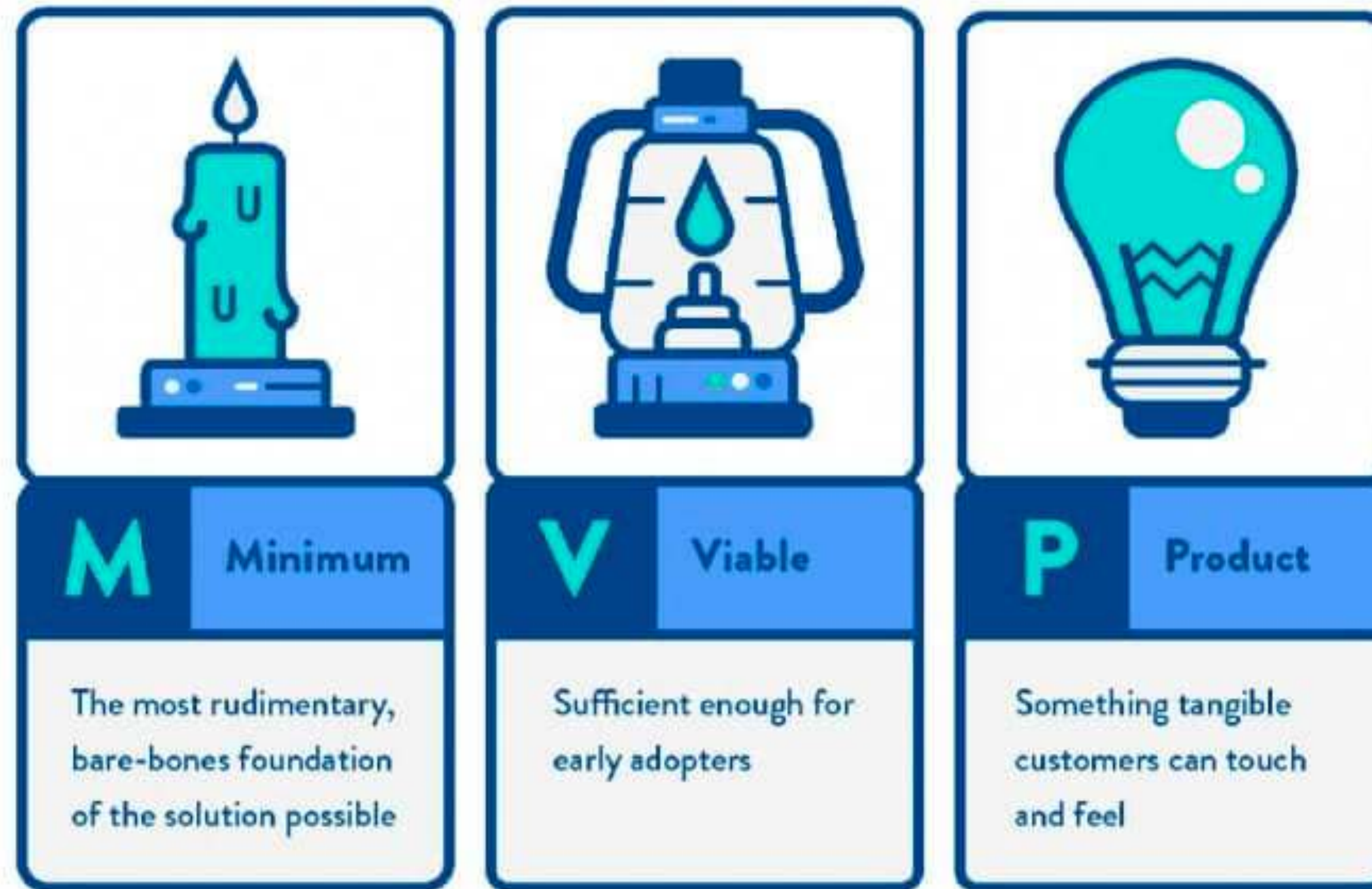
Prototype Model

Characteristics

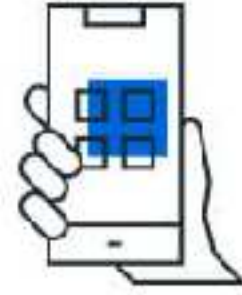
- The project starts with defining requirements for the final product and the ultimate business model. Then, they are stripped down to the minimal version, leaving only characteristics that are necessary for the execution of essential features.
- The minimal viable product is conceptualized and designed. UX designers analyze the target audience for the project, whereas UI designers select actual graphic elements.
- The build stage. Developers create code for the basic backend and frontend functionality.
- Beta-testing. The MPV is released to a small group of users who evaluate the functionality and detect bugs.
- After the prototype was evaluated by users, the team analyzes the feedback and creates the next iteration of the product.

Prototype Model

What is a minimum viable product?



Prototype Model



Pros & Cons of Prototype Model

Pros:

- ✔ Good in ironing out potential issues in the early development stage, which greatly reduces product failure risk.
- ✔ Ensures the customer is happy with the 'product', before real development works started.
- ✔ Build rapport with the customer early on with the discussions, which helps throughout the project.
- ✔ Gather detailed information with the prototype, which is later on used in building the final version.

Cons:

- ✘ Excessive to and fro in testing out the prototype with the customer can delay the development timeline.
- ✘ The customer's expectations of the actual product may not align with the prototype.
- ✘ There's a risk of cost overrun as the works on the prototype are often paid for by the developer.

Prototype Model

- The Prototype software development methodology is ideal when you're building software with many unknowns.
- For example, an online platform with intense user interaction. With the prototype methodology, you can find out what works best with users and reduce the risk of developing the actual product.

Rapid Application Development

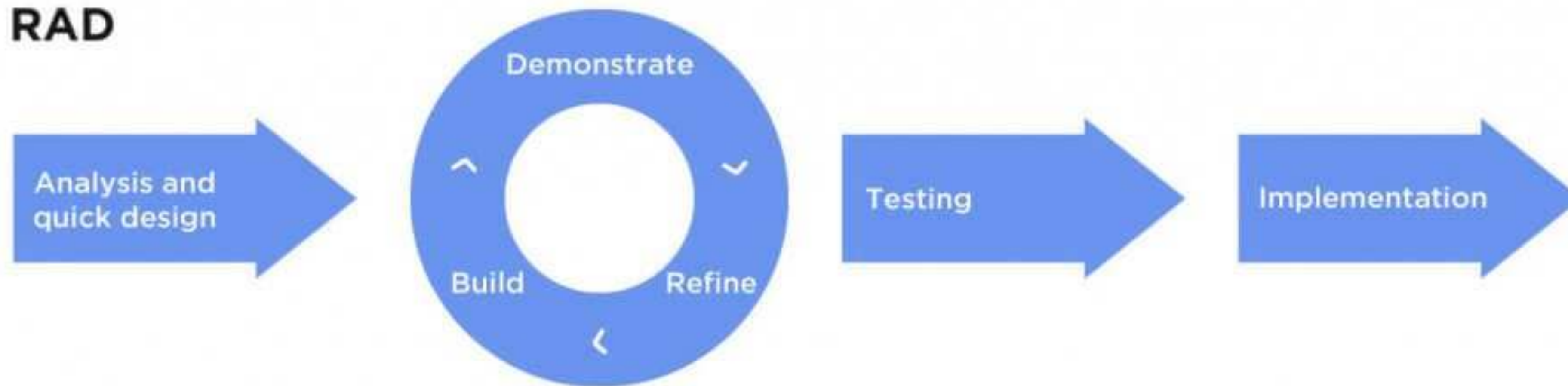
- The Rapid Application Development (RAD) model was introduced in 1991 and served as the foundation of modern iterative frameworks. It focuses on getting products built in a much shorter time frame without compromising the quality.
- RAD is a 4-step framework, which is defining project requirements, prototyping, testing, and implementation. Unlike linear models, RAD emphasizes building prototypes with the given requirements and testing them out with the customer. This is done through multiple iterations until the customer is happy with the results.
- Rigorous testing of the prototype results in valuable feedback, which helps to eliminate much of the product risk. Using RAD leads to high chances of successful product release within the stipulated timeline. RAD often uses development tools that could automate and simplify the development process.

Rapid Application Development

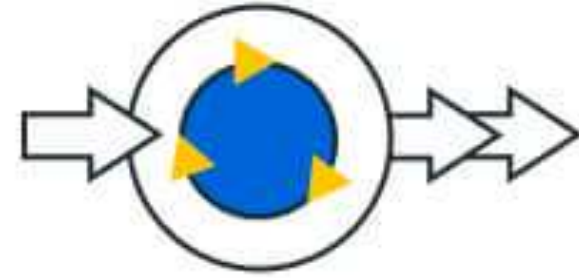
Traditional



RAD



Rapid Application Development



Pros & Cons of Rapid Application Development

Pros:

- ✓ Risks reduction through regular customer feedback.
- ✓ Increased-satisfaction for customers.
- ✓ Works well for small and medium applications.
- ✓ Reduces time-to-market.

Cons:

- ✗ Highly-dependent on a responsive customer.
- ✗ Requires a team of highly-skilled and experienced developers.
- ✗ Not suitable for projects with budget constraints.
- ✗ Lack of documentation for progress tracking.

Rapid Application Development

- You'll get the best results from Rapid Application Development if you have a team of experienced developers and customers who are equally involved in the project.
- Communication is key to realizing projects with the RAD method.
- You'll also need to invest in RAD tools like low-code/no-code applications to speed up the development.

Dynamic Systems Development Model

- The Dynamic System Development Model inherits much of its principles from the Rapid Application Development framework. This refined software development methodology is aimed at the prompt delivery and aligning project goals to business needs. It features four iterative phases of feasibility & business study, functional model, design & build, and implementation.
- Throughout the process, end users are greatly involved in providing feedback. This reduces the risk of straying from the project goals and requirements.
- The Dynamic Systems Model also features detailed documentation, which is lacking in most Agile frameworks.

Dynamic Systems Development Model

- DSDM don't have to be standalone. It can be combined with Agile, prototype development, or rapid development.
- However, the priorities are shifted towards analyzing business, getting a strategic understanding, and doing in-depth market research.

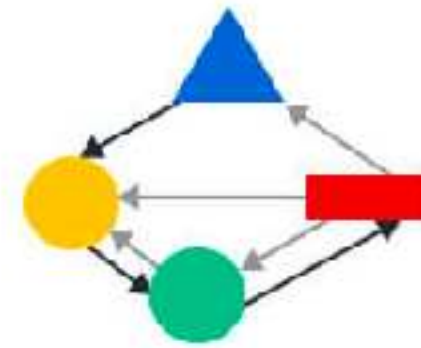
Dynamic Systems Development Model

- DSDM comprises five key stages: feasibility study, business research, functional model ideation, system design, and implementation.
- The majority of these stages fall in the first half of the product development process. DSDM emphasizes on making sure that the product can bring expected benefits to the company and market.

Dynamic Systems Development Model



Dynamic Systems Development Model



Pros & Cons of Dynamic Systems Model

Pros:

- ✓ The iterative approach ensures that basic software functionalities are delivered promptly.
- ✓ Developers have better control of the development timeline and budget.
- ✓ Necessary documentation is created throughout the development.
- ✓ Establish communication between end-users and developers, which keeps the team on the right track.

Cons:

- ✗ It can be quite expensive to execute. Heavy user's and developers' involvement is needed, and it takes a considerable amount of budget to train them.
- ✗ Smaller teams will have a hard time implementing this methodology.
- ✗ The concept and implementation of the model are quite complex.

Dynamic Systems Development Model

- Using the Dynamic Systems Model for small organizations is out of the question. Instead, it is a better fit for large organizations saddled by red-tapes, bottlenecks, and redundancies.
- It breaks down rigid processes into smaller iterative parts and streamlines communication between different teams.

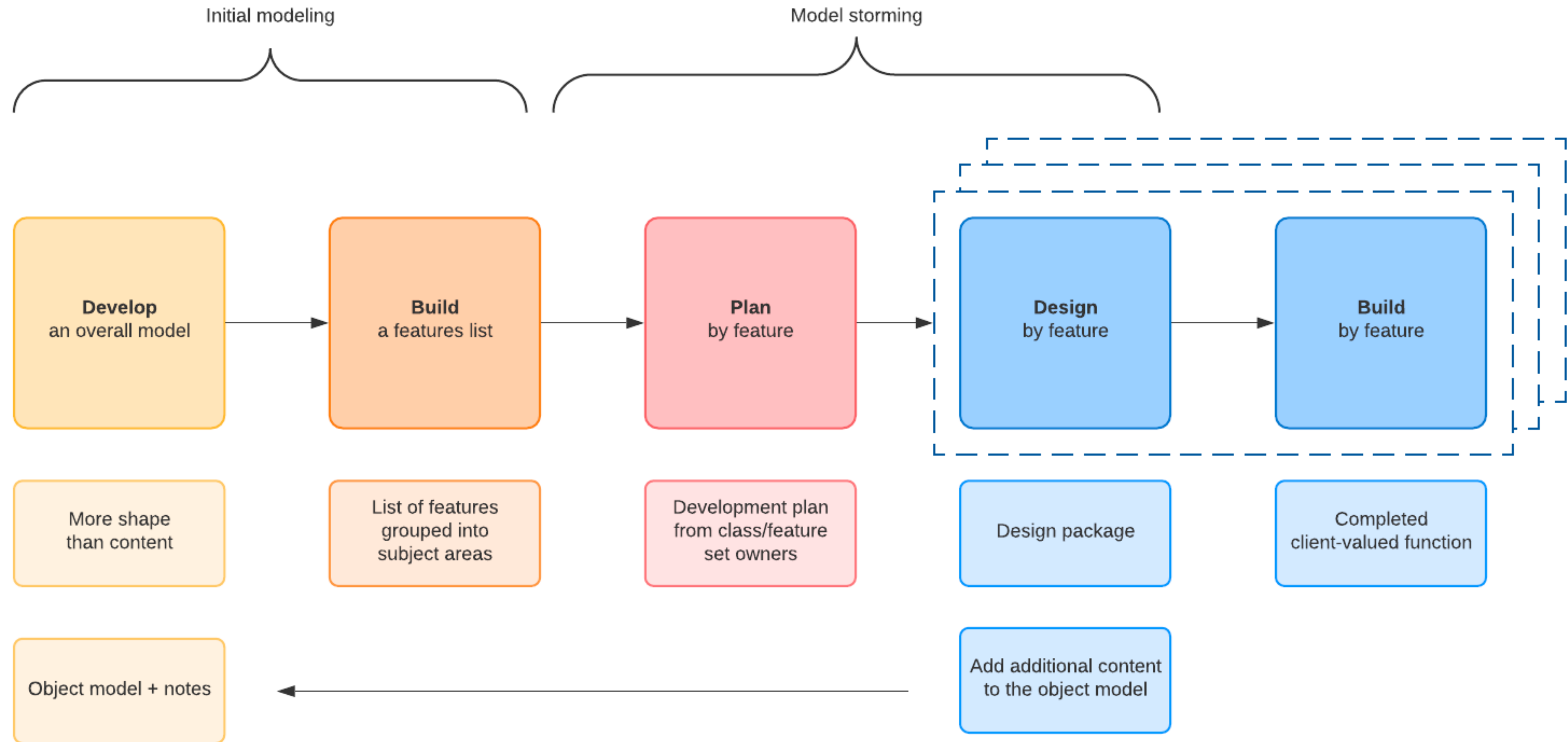
Feature Driven Development

- Feature Driven Development, or FDD is a software development methodology based on Agile. Its goal is simple, to prevent confusion that leads to costly rework.
- FDD is sometimes mistaken as focusing on each of the software features.- it is not.
- What Feature Driven Development does is break down development activities into a feature list on the overall model. For each of the features, developers go through an iteration of planning, designing, and building. Typically a feature should take no longer than two weeks to accomplish.

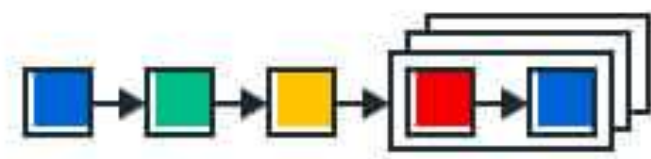
Feature Driven Development

- The outcome of FDD is quick, impactful results for each of the activities listed as features. This approach is meant for large teams, and information is communicated through detailed documentation.

Feature Driven Development



Feature Driven Development



Pros & Cons of Feature Driven Development

Pros:

- ✓ Breaks down complicated tasks into smaller activities, which fueled efficiency.
- ✓ Enable large teams to work on multiple tasks simultaneously.
- ✓ Based on predefined standards and best practices, which leads to predictable success.

Cons:

- ✗ Not suitable for smaller projects.
- ✗ Highly dependent on the lead developer, as he/she is in charge of coordinating the tasks between members.
- ✗ It can sometimes deviate from delivering value to the end-users as the model is driven by activities.

Feature Driven Development

- Feature Driven Development works best for larger teams working on complex projects. It offers the best of an iterative framework but with a more structured approach.
- Ideally, you'll want to have a competent lead developer taking charge of FDD.

Scrum Development

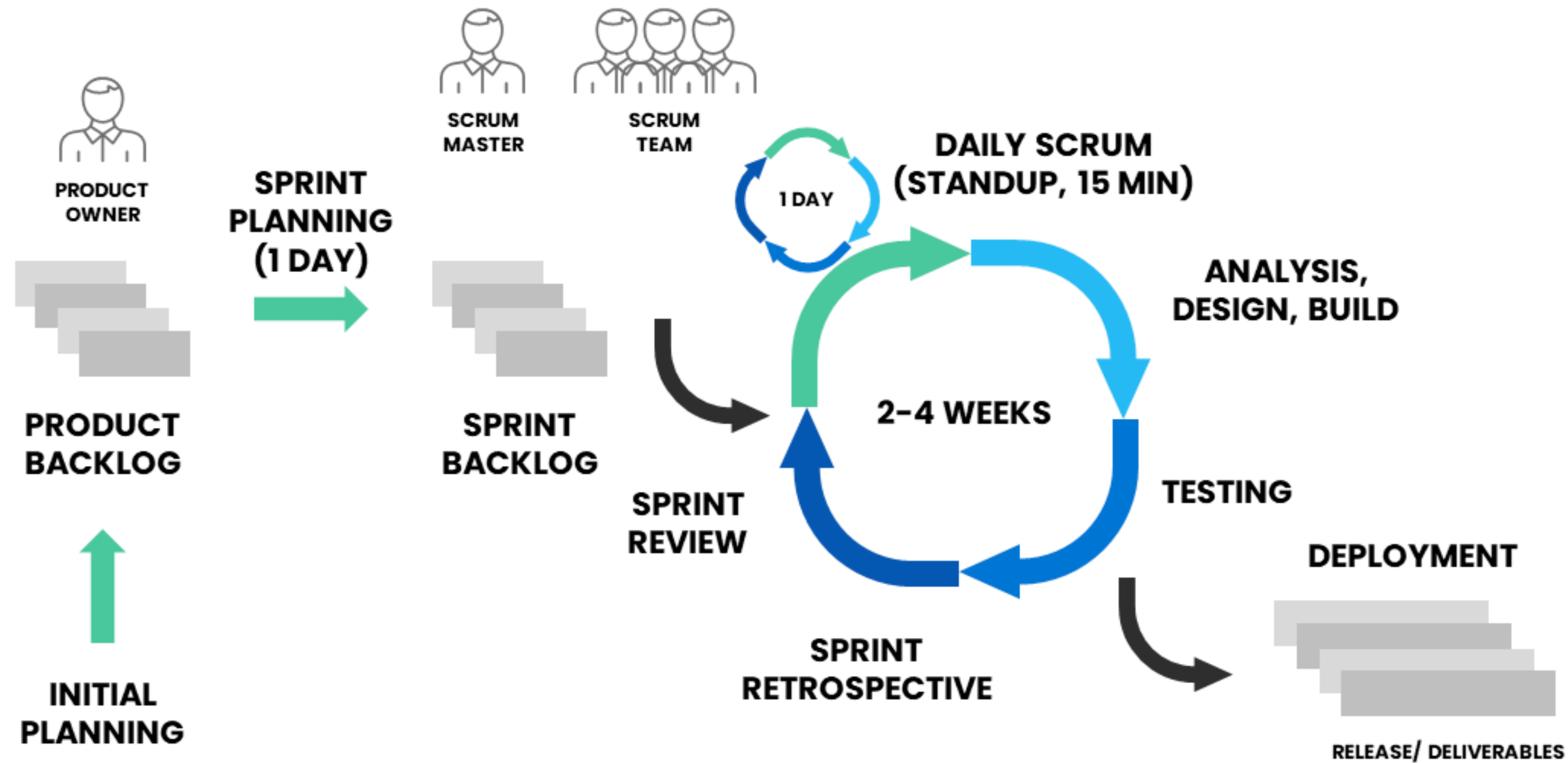
- Scrum is arguably one of the most flexible software development methodologies available.
- It is based on the Agile philosophy and is favored for its incremental and iterative approaches. The Scrum methodology involves the Product Owner, Scrum Master, and the Development Team.
- The product owner takes input from the client and ensures that the team is on track in fulfilling the client's requirements. Meanwhile, the Scrum Master acts as a facilitator and ensures that team members are familiar with the Scrum process. The team takes charge of executing the development.

Scrum Development

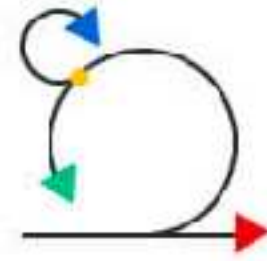
- What makes Scrum an ideal software development methodology in a fast-paced environment is how tasks are executed in sprints.
- Each sprint takes up to 4 weeks. The speedy execution allows teams to identify issues, introduce solutions, test, and gather feedback in a short period.
- It makes tackling fast-paced projects much easier.

Scrum Development

SCRUM BASICS



Scrum Development



Pros & Cons of Scrum Development

Pros:

- ✔ Short iterations allow quick resolutions to problems.
- ✔ Scrum is very responsive to changes as the process includes regular feedback.
- ✔ Scrum is economical and effective.
- ✔ Regular meetings ensure that team members are on the same page at all times.
- ✔ Contributions of individual members are noticed and appreciated through the Scrum meetings.

Cons:

- ✘ All team members must be equally skilled and committed for Scrum to work.
- ✘ The daily Scrum meetings can be draining for team members.
- ✘ May increase time-to-market if there's no strict control on the deadline.
- ✘ Not suitable for large projects.

Scrum Development

- Scrum is the go-to methodology if you have a project with vague requirements but needs to adapt to frequent changes.
- For example, you need to get an MVP built quickly and test it out amongst users.
- Remember that Scrum is only effective if you have a fully-committed and experienced team.

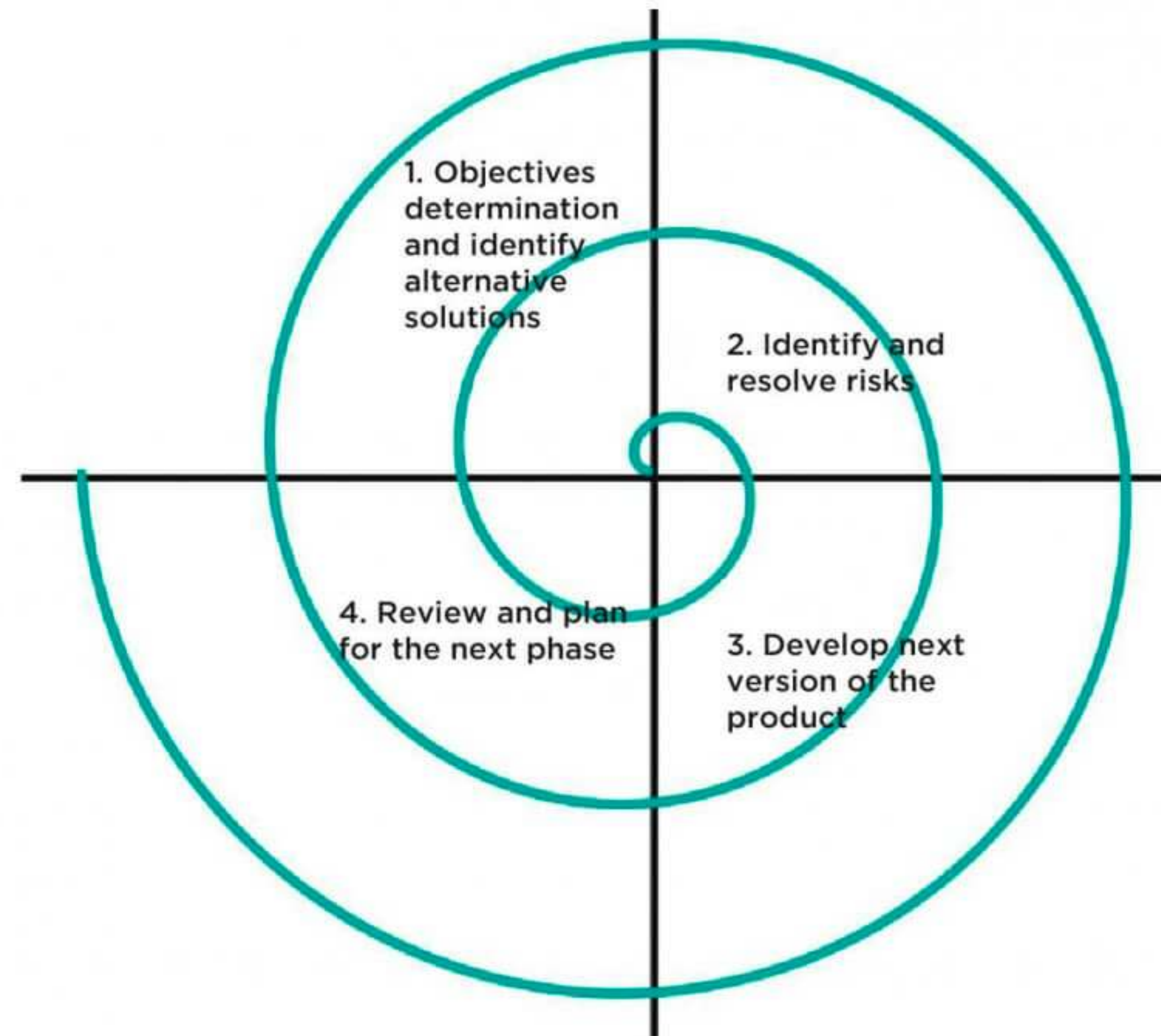
Spiral Methodology

- The spiral model isn't a standalone methodology – it can be used with Agile, Waterfall, Prototyping, RAD, and even DSDM.
- The model's purpose is to provide the visualization and detailed descriptions of all risks associated with the software products.
- Spiral software development methods follow the standard product development process – only for each stage, involved parties need to present partners and product owners with the rundown of likely risks with a probability evaluation.

Spiral Methodology

- Each project is completed in loops. Similar to sprints and test points, loops of spiral methodology define the task or a cluster of tasks that a team will work on during a given period. The more complex the project is, the more loops it needs.
- The riskier the project is, the higher will be the number of loops. Putting too many risks in a single loop is not recommended.
- The process of risk identification never stops. At each stage of product development and for each new product iteration, the team has to identify and resolve potential issues and plan the subsequent steps.

Spiral Methodology



Spiral Methodology

Advantages

- Security assurance: product developers and owners understand possible difficulties and have contingency plans;
- Facilitated management of large projects: when the stakes are high (a lot of stakeholders are involved in the project, and there's high potential reputation damage), thorough risk assessment during the entire product development helps prevent an unexpected crisis.
- Flexibility: the risks are analyzed at each development stage, which helps to keep in touch with economic trends.

Spiral Methodology

Disadvantages

- Complicated: the methodology requires highly-developed analytical skills;
- Expensive: determining risks on each development stage takes a lot of productive time, which increases the final cost of the product;
- Difficult management-wise: the project manager can have trouble evaluating and monitoring the efficiency of risk assessment since it's hard to quantify.

Extreme Programming

- Extreme programming is a test-driven software development methodology.
- The code can't be considered perfect until the team didn't run the tests of all system modules.
- Extreme programming heavily relies on unit testing: after a piece of code is developed, its quality needs to be checked immediately.

Extreme Programming

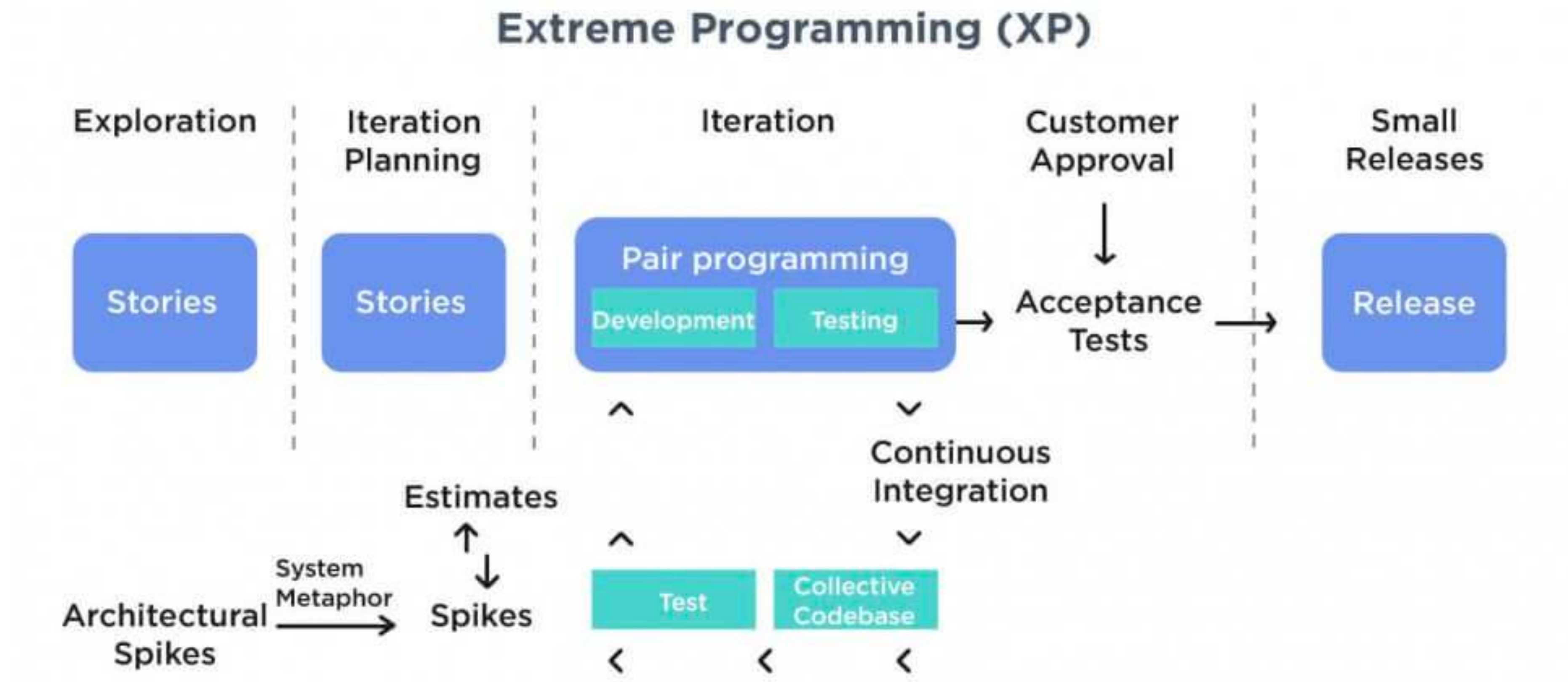
- It is evident that extreme programming requires QA and development teams to run thousands of tests every day.
- To make this work, the team has to adopt automated testing. The priority should be to invest the time at early project stages to automate functional, performance, and unit tests, and save time later on during the QA and maintenance stages.

Extreme Programming

Fundamental Principles

- Software requirements and metrics are constantly evolving;
- Developers and testers invest a lot of time in automation and metrics;
- The development team is small – the emphasis is placed on testing and operations;
- Automation is a priority: everything that can be automated shouldn't be done manually.

Extreme Programming



Extreme Programming

Advantages

- Time and cost-efficient: the team doesn't waste time on rewriting functionality and eliminating bugs or technical debt;
- No need to refactor: the code is evaluated as it's being written;
- Better performances: users don't see functionality or interface bugs in the ready project;
- Reduced risks: due to constant monitoring, the team minimizes the risk of sudden system failures.

Extreme Programming

Disadvantages

- The lack of universality: a strict approach to quality assurance can limit developers' creativity, pushing them to develop error-free solutions rather than innovative ones;
- Investment in testing: product owners need to be ready to invest additional budgets into testing execution and management.

Systems Development Lifecycle

- The systems development life cycle is a conservative system that heavily bases on the stages of the product development process.
- It is similar to Waterfall – you can't jump from one step to another or return to the completed stage.
- The main difference is that SDLC is more regimented: while Waterfall allows multiple variations of the names and structure of the development process, SDLC follows an organized approach.
- It is not just the set of values – like the majority of methodologies here – but a set algorithm.

Systems Development Lifecycle

Main stages

- **Analytics:** the product owner and development team evaluate the existing tech solutions, define issues, and identify solutions.
- **Planning:** the development process is planned in detailed, with precise estimates on time and budget;
- **Design:** the system is evaluated by UX designers who create user personas, define the scope of the product, and refine main characteristics. UI designers implement this vision in interfaces and graphic elements;
- **Development:** the software is developed, firstly the essential functionality, later – additional features;
- **Quality Assurance:** testers verify the performance of the software and compare it to requirements, set in the beginning;
- **Deployment:** the ready product is uploaded to the server where it's available to end-users;
- **Maintenance:** after the product was released, the team maintains the codebase and prepares new releases.

Systems Development Lifecycle



Systems Development Lifecycle

Advantages

- Defined algorithm: project managers and team members know exactly which system development methods to follow;
- Easy to measure: project managers can assign metrics to each development stage in advance, knowing that the structure of the workflow won't be much affected in the process;
- Developers know exactly what their responsibilities entail: there's less confusion with unrealistic expectations.

Systems Development Lifecycle

Disadvantages

- No flexibility: even though some projects can require the introduction of innovative development stages, the methodology doesn't take it into account.
- Tech debt and stockpiled complications: code errors and hardware problems tend to accumulate, making maintenance a lot harder.

How to choose?

- It should be clear by now that software methodologies are very different.
- Each methodology works within a particular system of priorities, varying degrees of flexibility, and specific deliverables.
- There's no such thing as good or bad software development methodology types – it all depends on the project.

How to choose?

- Consider the product's scope
 - You need to understand the goals of the project: what is the main task that the final solution will accomplish?
 - What audience does the product target? The software can require constant updates – if it's a community-based marketplace or a software-as-a-service.
 - Small websites and landing pages, on the other hand, don't depend on users as much – so they can be developed with more conservative and less flexible methodologies.

How to choose?

- Define product requirements
 - The crucial part of selecting an application development methodology is determining what stages it will entail.
 - For that, you need to create a list of software requirements – describe the ideal functionality, interface, positioning, and results (number of visits, downloads, and others).

How to choose?

- Identify the target audience
 - Some audiences are more conservative and don't like it when their solutions are updated too often. We've all seen complaints on App Store or Google Play that went on about unsolicited updates and uncomfortable new interfaces. Some users don't like to accommodate new features and design – this should be taken into account when you plan the development process.
 - Other users (usually the tech-savvier ones) prefer working with a regularly updated solution. They like to see that the product is improving and know what they are paying for. In this case, it's better to go with Agile or prototyping – flexible methodologies that allow releasing multiple product iterations.

How to choose?

- Consider your team's location and previous experience
 - If your organization is remote, you need to consider the communication practices that they used before. Usually, remote developers work with Agile, Waterfall, and prototyping – these methodologies are the most common ones, and they are also easy to implement.
 - If the team doesn't have enough experience with your type of projects or tech stack, don't complicate the process by choosing rare mythology (like extreme programming). If the project is challenging, it's better to stick with the methods of the team's choice.

A software methodology is not just an algorithm of actions or a collection of tools. It's a system of beliefs and values that will be applied on a long-term basis.

Choosing one is an important decision that sets your project up for success. It's worth putting investing your time into researching each methodology in the context of your project's requirements.