An Overview of the Top 7 Software Development Methodologies

by Dennis Hammer

When you first started building your startup's product, you probably didn't follow a specific development methodology. You just built what you needed.

That's how most technology products begin, but it doesn't get you far. Eventually you'll have too many pieces (bug fixes, features to add, changes to test, improvements to make, etc.) to manage. And the whole thing gets a lot more complicated when you start hiring a team…

Part of building a quality product is managing the process. This seems simple, but poor project management can burn through your cash and your sanity, and even kill your startup.

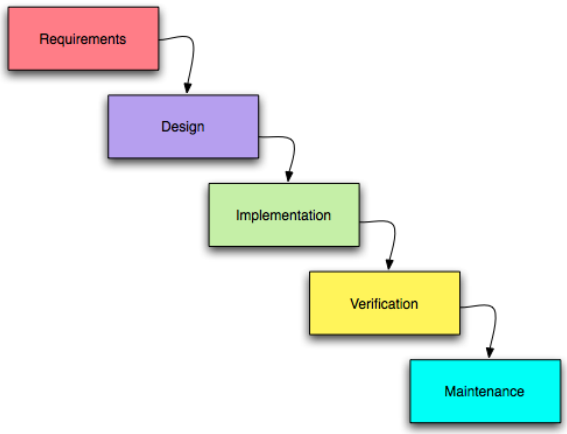
As a founder, *how* you and your team build a product is just as important as *what* you're building. You need a clear system in place to organize the process so you build a quality product efficiently.

In today's article, we'll give you an overview of the top software development methodologies.

1. Waterfall Development
2. Agile Development
3. Rapid Application Development
4. Scrum Framework
5. Extreme Programming
6. Prototype Methodology
7. Spiral Development

1. Waterfall Development

Waterfall is considered the traditional software development methodology. It calls for a linear system of phases (requirements, design, implementation, verification and maintenance). You must complete each phase before moving to the next phase. Previous phases are rarely revisited. For instance, once the project design is complete, the project's requirements (previous phase) are locked in and unchanging.



Pros

- Linear method is easy to understand.
- Great for projects with clear objectives and simple, unchanging requirements.
- Good for inexperienced project managers and teams whose composition changes frequently.

Cons

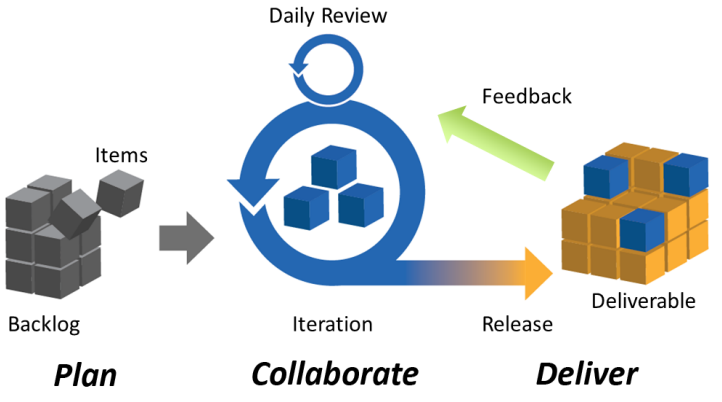
- Rigid structure makes it slow and costly (you don't get a usable product until the very end).

- Only matches precise needs. It can't adapt to new information or new ideas once the process has begun.
- Not useful for long, complex, or ongoing projects.

Learn more about the waterfall development methodology here.

2. Agile Development

Agile is the most common software development methodology among the makers of cloud-based applications. Its main goal is to minimize risk by developing software in "iterations" that last for a predetermined length of time (usually two to four weeks of development).



Agile Project Management: Iteration

Each iteration is an entirely unique mini-project with its own requirements, planning, design, implementation, and testing phases. Each phase ends with a "release" of something that satisfies the mini-project's requirements, though the result may not be something worth releasing to customers/users.

Pros

- Highly adaptive to change.
- Stakeholders and non-technical team members like to see a working product iterated over time.
- Allows for lots of communication between team members.
- Easy to find and fix bugs because testing happens so frequently.
- Excellent for environments where the project scope or requirements change frequently.
- One of the most popular methodologies in the startup scene.

Cons

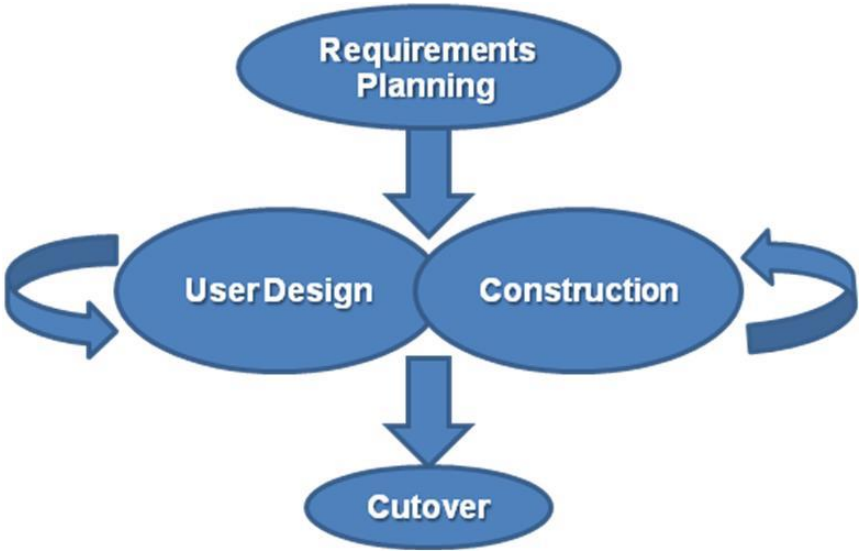
- Lacks documentation because so much is "fluid" during development. At some point someone has to write help notes and add code comments.
- Outcome isn't perfectly clear from the beginning, so it's easy to get off track.

- Requires constant communication, especially face-to-face contact, making it difficult for remote teams.

Learn more about the Agile development methodology here.

3. Rapid Application Development

RAD is a condensed development process designed to reduce the amount of engineering necessary to build a product. It aims to produce a working product with low investment costs in a short time frame by regularly soliciting customer feedback.



Scott Stiner, the CEO and President of UM Technologies, says RAD lets his team adjust to shifting requirements. “We've found that by using the RAD process, our team can adapt to a flexible process as the project evolves. The byproduct is the ability to continually incorporate knowledge from UX design process to rapidly iterate, prior to launching the software.”

The RAD process has four phases: Requirements, user design, construction and cutover. The design and construction phases repeat as long as necessary until the user says the requirements have been met.

Pros

- Most effective when there’s a well-defined business objective.
- Great for small or medium-sized time-sensitive projects.
- It encourages direct feedback from actual users continuously throughout development.

Cons

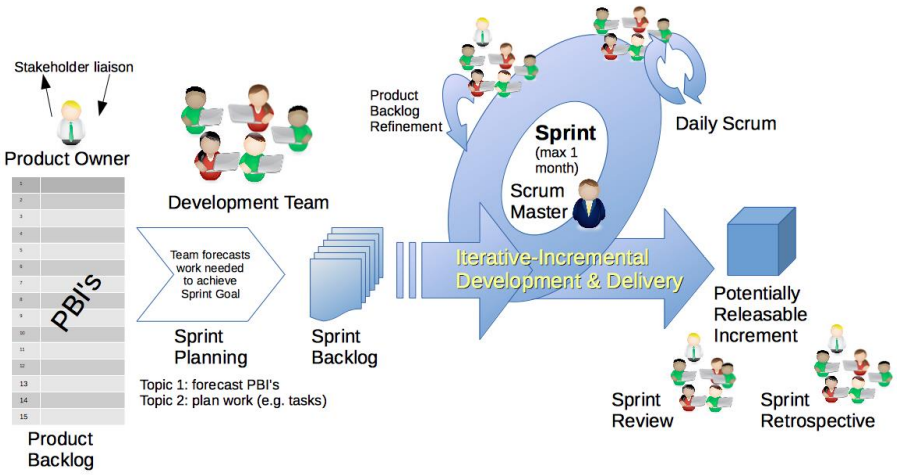
- Requires a development team with low turnover.
- Needs a clearly defined user group (you *must* know who you’re making the product for).
- Developers must be highly skilled with deep cross-sector knowledge.

- Not great for projects with small budgets.

[Lean more about Rapid Application Development here](#).

4. Scrum Framework

Scrum borrows many of its principles from Agile. It emphasizes customer feedback and fast increments. It's a flexible way for teams to react to changing requirements. Unlike RAD, it's useful for projects with long development times.



Scrum uses development cycles called "sprints" that last no more than 30 days. The goal is to develop and test a product increment every sprint.

People on a Scrum team play one of three roles: 1) The Scrum Master, who facilities the sprints, meetings, and engineering environment, 2) The Product Owner, who prioritizes decisions and organizes the sprints, and 3) The Development Team, made of self-organizing people with diverse skills who design, build, and test the product.

Pros

- The group self-manages itself, so people build the features that play to their skills.
- Great for long-run development projects with deep requirements.
- Short, daily meetings keep communication open and clear.
- Perfect for projects with frequently changing requirements.
- Product Owner makes sure that business requirements (broader goals, ROI, etc.) aren't forgotten.

Cons

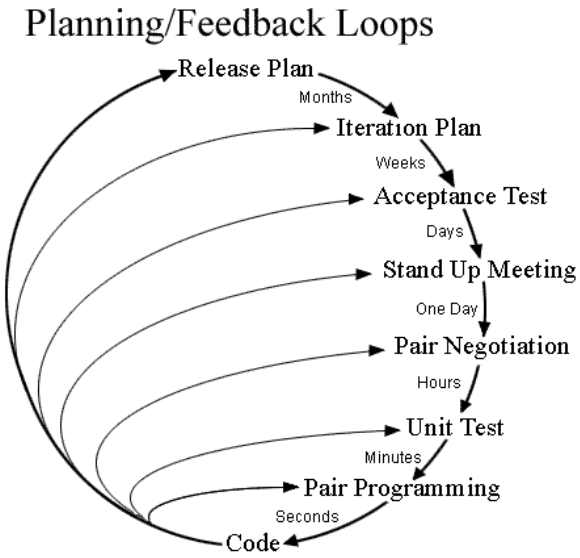
- Costs are not predictable.

- No room for novices on the team. Everyone must be comfortable working with a large stack of tools and technologies.

Learn more about Scrum here.

5. Extreme Programming

Extreme Programming (XP) is another methodology that borrows from Agile. Unlike Agile, however, XP is designed to create a high-quality product that doesn't consider any functions outside the project's scope. That is, you wouldn't build Feature X in a certain way because someday you hope to build Feature Y. You would just build Feature X in the simplest way possible to get the job done.



XP is "extreme" because it's twice as intense as other development methodologies. There are twice as many code reviews and unit tests. It also requires a lot of human resources because it requires pair-coding: Two engineers at the same workstation. One writes while the other reviews and navigates.

Pros

- Code is simple because it's written to only meet the needs of today's project.
- The best choice if you're building software under a deadline and/or for a client who doesn't care *how* it works, only that it does.
- Best for small teams, but it does work on large teams.
- Focuses heavily on the customer's needs and requirements.
- Has clear plans and schedules.

Cons

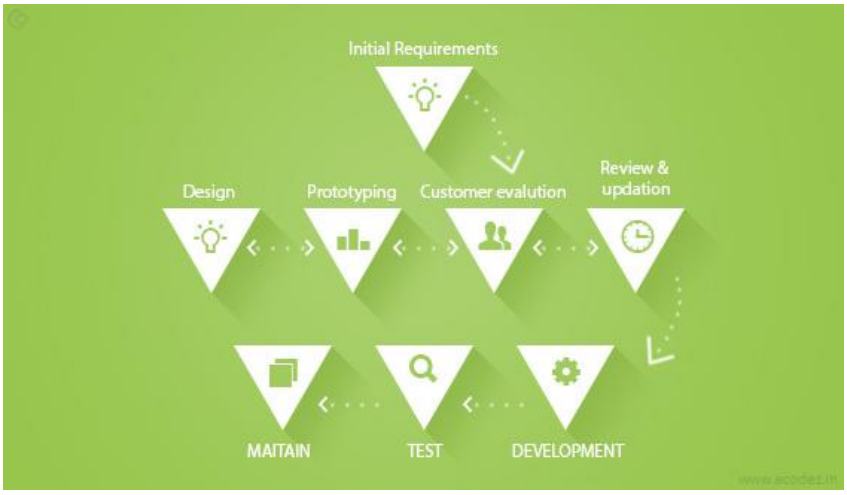
- It's not impossible to change project requirements, but the cost can be very high.

- It can be challenging to work so closely with a customer/user who's on-site.
- Simple code means the software isn't useful beyond the project.

Learn more about extreme programming here.

6. Prototype Methodology

Prototyping is fairly straightforward: It's when you build an early version of a product in order to get a better idea of the client's needs. A prototype gives the client (who might be your employer) something they can feel and use. It's not intended to be a working product, just a functional *example* of what the developers could create. It's often used before a developer commits to the waterfall methodology.



Source: acodez.in

Pros

- Customers, stakeholders, and executives like to have something they can touch before they invest a lot of time and money into a product.
- This methodology reduces the risk of failure or creating something unneeded.
- Helps you learn the true needs of the product (what the client *really* means).
- This method makes it easy to identify risks and obstacles before building.
- You can get important feedback from customer in the early stages.

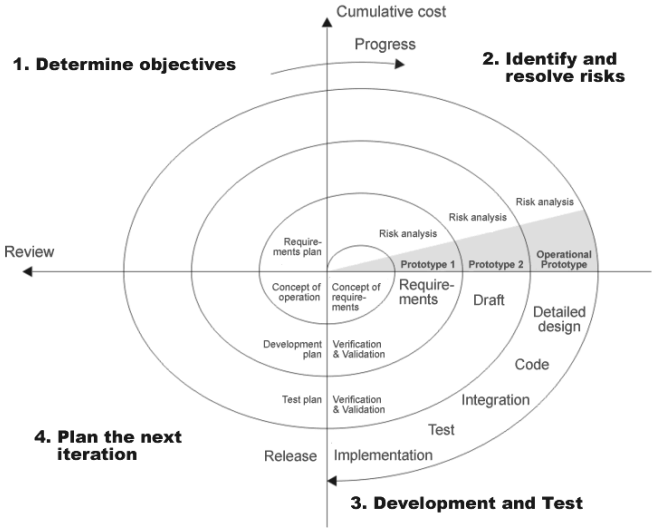
Cons

Adds additional development costs because you still have to build the product from scratch. Adds time to the whole development process.

Learn more about the prototype methodology here.

7. Spiral Development

The spiral methodology is similar to the waterfall methodology, but with a few changes. First, there's a built-in prototyping phase for a "big picture" look. (Remember, if you use waterfall, you can't go back and add something later, so a prototype helps define the requirements early.)



Second, spiral methodology places a lot of emphasis on identifying risk factors early to either account for them or abort the project. It uses loops of checks, reviews, and testing to identify risks as soon as possible.

Pros

- Better than the waterfall methodology if you have a big, complex project.
- Reduces the number of problems, bugs, and risks by identifying them early.
- Easy to add additional features and functions at any point.

Cons

- It's not always a cost-effective model. In some cases, it can be quite expensive.
- Without good management, developers could work in a spiral forever without completing the project.
- The risk analysis is key. If the analysis is performed incorrectly, the whole methodology is faulty.
- Not a useful methodology for projects without risks, obstacles, or potential for many bugs.

Learn more about the spiral development methodology here.

Choose the One That Works for You

As you can see, there's no "best" methodology. How you manage your engineering (whether you're directing a team or you're doing it alone) depends on your product, your customer, and even your preferences. Build your product in whatever manner works best for you.

Need help building your product? Check out the 10xU membership. We'll help you take your business to the next level. Learn more >