**Brief history of DevOps**

If DevOps had a birth certificate, the father’s name would be penned in as Patrick Debois. Patrick was interested in learning IT from every perspective, and in 2007, he began working on a large data center migration where he was in charge of testing. During this project, he realized that the frustrations experienced in projects such as these are from the constant switching back and forth between the development side of the problem and the silo of operations on the other side of the fence. He recognized that a lot of time and effort was wasted navigating the project between these two worlds, but the divide between them seemed too wide to bridge.

Later, in 2008 during an Agile conference held in Toronto, Canada, a man by the name of Andrew Shafer tried to put together a meetup session entitled “Agile Infrastructure.” When Patrick showed up for the session, he was the only one there. Andrew had received so much negative feedback from his posting that not even he showed up to his own session. However, Patrick was so excited to learn of a like-minded person that he hunted him down at the conference and had that talk in the hallway. They formed a discussion group for other people to post their ideas for how to solve this divide between development and operations later that year.

Initially, the interest was pretty tame and not a whole lot came of it. In June of 2009, John Allspaw and Paul Hammond gave a talk entitled “10+ Deploys a Day: Dev and Ops Cooperation at Flickr.” Our friend Patrick happened to watch the streaming video of that presentation at his home in Belgium, and it instantly resonated with him. He realized this was exactly the solution for which he had been looking. Emboldened by this presentation, he put out a call to have a gathering of developers and system administrators to get together and discuss the best ways to start bridging the gap between the two disparate fields.

He named the event DevOpsDays, occurring in the last days of October in 2009. This event garnered a fair amount of attention from experts in both fields and sparked lively debates over Twitter where the hashtag was soon shortened to simply DevOps. It wasn’t long before some of the smaller tech enterprises were attempting to put together DevOps practices as well as tools built to aid these newly forming teams. DevOps had managed to achieve a grassroots following that was starting to put their ideas to use.

Finally, in March of 2011, Cameron Haight of Gartner presented his predictions for the trajectory of DevOps over the next few years. His positive outlook on its impact on the industry lead to more attention for the DevOps movement, and it wasn’t long before enterprises of all sizes were beginning to adopt these new practices. DevOps had officially caught on as the next big thing since Agile for the IT industry.

**What Is DevOps?**

**Definition**

DevOps is a software development approach that emphasizes collaboration, communication, and integration between development teams and IT operations teams to automate and streamline the software delivery process. The goal of DevOps is to improve the speed and efficiency of software development and deployment while maintaining high quality and reliability

**How does DevOps work?**

A DevOps team includes developers and IT operations working collaboratively throughout the product lifecycle, in order to increase the speed and quality of software deployment. It’s a new way of working, a cultural shift , that has significant implications for teams and the organizations they work for.

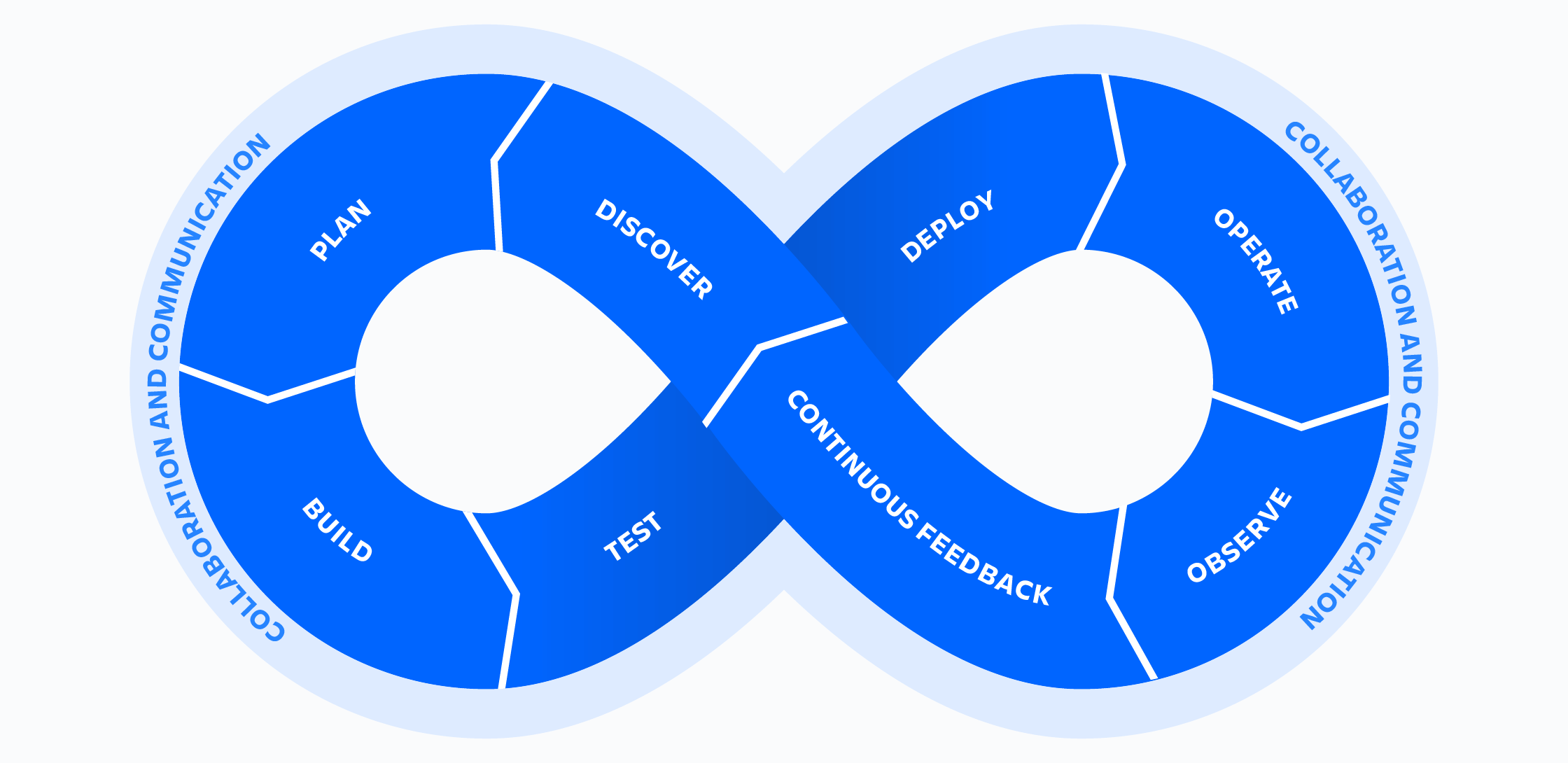
Under a DevOps model, development and operations teams are no longer “siloed.” Sometimes, these two teams merge into a single team where the engineers work across the entire application lifecycle — from development and test to deployment and operations — and have a range of multidisciplinary skills.

DevOps teams use tools to automate and accelerate processes, which helps to increase reliability. A DevOps toolchain helps teams tackle important DevOps fundamentals including continuous integration, continuous delivery, automation, and collaboration.

DevOps values are sometimes applied to teams other than development. When security teams adopt a DevOps approach, security is an active and integrated part of the development process. This is called [DevSecOps](https://www.atlassian.com/continuous-delivery/principles/devsecops).

**The DevOps lifecycle**

Because of the continuous nature of DevOps, practitioners use the infinity loop to show how the phases of the DevOps lifecycle relate to each other. Despite appearing to flow sequentially, the loop symbolizes the need for constant collaboration and iterative improvement throughout the entire lifecycle.



The DevOps lifecycle consists of eight phases representing the processes, capabilities, and tools needed for development (on the left side of the loop) and operations (on the right side of the loop). Throughout each phase, teams collaborate and communicate to maintain alignment, velocity, and quality.

**1. Plan:** Defining the objectives, requirements, and scope of the software project.Tools used: Trello, Asana, Jira

**2. Code:** Writing and reviewing code for the software. It includes version control, code review, and collaboration among developers*.*Tools used: Bitbucket, GitLab, Git, GitHub

**3. Build:** Compiling the code and generating executable files Tools used: Gradle, Maven

**4. Test:** Testing for bugs and errors. Tools used: Selenium, JUni

**5. Release:** Preparing the software for release and production. Tools used: Jenkins

**6. Deploy:** Deploying the software to production environments. Tools used: Kubernetes, Docker, Ansible, Chef, Puppet

**7. Operate:** Smooth production assurance. Tools used: Nagios, New Relic

**8. Monitor:** Monitoring for issues, bugs, and errors, and taking corrective action as needed. Tools used: Nagios

**Stages of DevOps**

**1. Version Control:** Collaborative coding, change/ update code. (GitHub)

**2. Continuous Integration (CI):** Integrating and testing code changes frequently to ensure that code is working as expected and there are no conflicts. (Jenkins)

**3. Continuous Delivery (CD):** Deploying code changes to a testing environment, and automating the delivery process to ensure that new features and updates are released quickly and reliably. (Maven integrated with Jenkins)

**4. Continuous Deployment (CD):** Deploying code changes directly to production without any manual intervention, ensuring that code is delivered quickly and consistently. (Ansible, Chef, Puppet, Kubernetes, Docker)

**5. Continuous Monitoring:** Monitoring the software in production and identifying issues or bugs as they arise, so that they can be quickly resolved.

**What is DevOps culture?**

DevOps culture involves closer collaboration and a shared responsibility between development and operations for the products they create and maintain. This helps companies align their people, processes, and tools toward a more unified customer focus.

It involves cultivating multidisciplinary teams who take accountability for the entire lifecycle of a product. DevOps teams work autonomously and embrace a software engineering culture, workflow, and toolset that elevates operational requirements to the same level of importance as architecture, design and development. The understanding that developers who build it, also run it, brings developers closer to the user, with a greater understanding of user requirements and needs. With operations teams more involved in the development process, they can add maintenance requirements and customer needs for a better product.

At the heart of DevOps culture is **increased transparency, communication, and** [collaboration between teams](https://www.atlassian.com/work-management/project-collaboration/collaborative-culture/build-collaborative-communication) that traditionally worked in siloes. But there are important cultural shifts that need to happen to bring these teams closer together. DevOps is an organizational culture shift that emphasizes continuous learning and continuous improvement, especially through team autonomy, fast feedback, high empathy and trust, and cross-team collaboration.

DevOps entails **shared responsibilities**. Development and operations staff should both be responsible for the success or failure of a product. Developers are expected to do more than just build and hand off to operations — they are expected to share the responsibility of overseeing a product through the entire course of its lifetime, adopting a "[you build it, you run it](https://www.atlassian.com/incident-management/devops/you-built-it-you-run-it)" mentality. They test and operate software and collaborate more with QA (Quality Assurance) and ITOps (IT Operations) . When they understand the challenges faced by operations, they are more likely to simplify deployment and maintenance. Likewise, when operations understand the system’s business goals, they can work with developers to help define the operational needs of a system and adopt automation tools.

**Autonomous teams** are another important aspect of DevOps. For the development and operations teams to collaborate effectively, they need to make decisions and implement changes without a cumbersome and lengthy approval process. This involves handing over trust to teams and establishing an environment where there is no fear of failure. These teams should have the right processes and tools in place to make decisions faster and easier, for each level of risk to the customer.

For example, a typical development workflow might require engagement from several contributors on different teams to deploy code changes. The developer makes a code change and pushes it to a source control repository, engineer builds and deploys the code to a test environment, a product owner updates the status of the work in an issue tracking tool, etc. An autonomous team will take advantage of tools that automate these processes, so that pushing new code triggers the building and deployment of a new feature into a test environment and the issue tracking tool is updated automatically.

A DevOps team culture values fast **feedback** that can help with continuous improvement of a unified development and operations team. In an environment where the development and operations teams are in isolated silos, feedback about the performance and stability of application software in production is often slow to make it back to the development team, if it makes it at all. DevOps ensures that developers get the fast feedback they need to rapidly iterate and improve on application code by requiring collaboration between operations folks in designing and implementing application monitoring and reporting strategies. For example, any sufficiently capable continuous integration tool will enable automated build and testing of new code pushes and provide the developer with immediate feedback on the quality of their code.

**Automation** is essential to DevOps culture, since it allows great collaboration and frees up resources. Automating and integrating the processes between software development and IT teams helps them to build, test, and release software faster and more reliably.

**What are the benefits of DevOps culture?**

The most obvious and impactful benefit of embracing a DevOps culture is streamlined, frequent, and high-quality software releases. This not only increases company performance, but also employee satisfaction.

A DevOps culture fosters high levels of trust and collaboration, results in higher quality decision making, and even higher levels of job satisfaction, according to the book “Accelerate: Building and Scaling High Performing Technology Organizations.”

Embracing a DevOps culture is key to building a high-performing engineering organization without sacrificing employee contentment. It’s a win-win. For an engineer there is nothing like the feeling of frequently and easily deploying and running stable, high-performing software that makes its users happy, and for executives the improved business outcomes are a hit.

**Software Development Life Cycle (SDLC)**

**1. Definition**

The Software Development Life Cycle (SDLC) is a structured process that enables the production of high-quality, low-cost software, in the shortest possible production time. The goal of the SDLC is to produce superior software that meets and exceeds all customer expectations and demands. The SDLC defines and outlines a detailed plan with stages, or phases, that each encompass their own process and deliverables. Adherence to the SDLC enhances development speed and minimizes project risks and costs associated with alternative methods of production

**2. Definition**

Software Development life cycle (SDLC) is a spiritual model used in project management that defines the stages include in an information system development project, from an initial feasibility study to the maintenance of the completed application.



**Importance of SDLC**

* It provides a standardized framework that defines activities and deliverables
* It aids in project planning, estimating, and scheduling
* It makes project tracking and control easier
* It increases visibility on all aspects of the life cycle to all stakeholders involved in the development process
* It increases the speed of development
* It improves client relations
* It decreases project risks
* It decreases project management expenses and the overall cost of production

**Working of SDLC**

**Planning phase**

The planning phase encompasses all aspects of project and product management. This typically includes resource allocation, capacity planning, project scheduling, cost estimation, and provisioning.

During the planning phase, the development team collects input from stakeholders involved in the project; customers, sales, internal and external experts, and developers. This input is synthesized into a detailed definition of the requirements for creating the desired software. The team also determines what resources are required to satisfy the project requirements, and then infers the associated cost.

Expectations are clearly defined during this stage as well, the team determines not only what is desired in the software, but also what is NOT. The tangible deliverables produced from this phase include project plans, estimated costs, projected schedules, and procurement needs.

**Coding phase**

The coding phase includes system design in an integrated development environment. It also includes static code analysis and code review for multiple types of devices.

**Building Phase**

The building phase takes the code requirements determined earlier and uses those to begin actually building the software.

**Testing Phase**

The phase entails the evaluation of the created software. The testing team evaluates the developed products in order to assess whether they meet the requirements specified in the ‘planning’ phase.

**Assessments entail the performance of functional testing**

unit testing, code quality testing, integration testing, system testing, security testing, performance testing and acceptance testing, as well as nonfunctional testing. If a defect is identified, developers are notified. Validated (actual) defects are resolved, and a new version of the software is produced.

The best method for ensuring that all tests are run regularly and reliably , is to implement automated testing. Continuous integration tools assist with this need.

**Release Phase**

The release phase involves the team packaging, managing and deploying releases across different environments.

**Deploy Phase**

In the deployment phase, the software is officially released into the production environment.

**Operate Phase**

The operate phase entails the use of the software in the production environment.

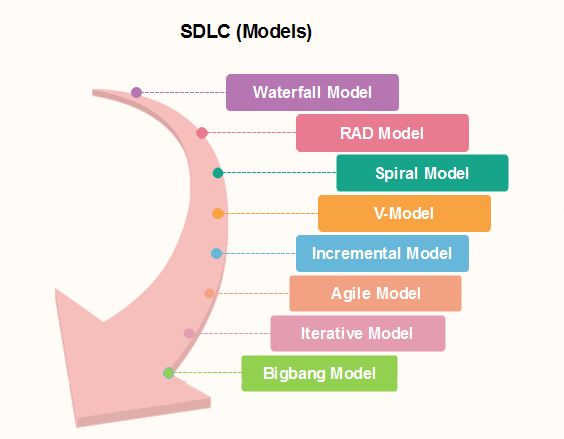
**Monitor Phase**

In the monitor phase, various elements of the software are monitored. These could include the overall system performance, user experience, new security vulnerabilities, an analysis of bugs or errors in the system.

**SDLC Models / Methodologies**

There are different software development life cycle models specify and design, which are followed during the software development phase. These models are also called "**Software Development Process Models**." Each process model follows a series of phase unique to its type to ensure success in the step of software development.

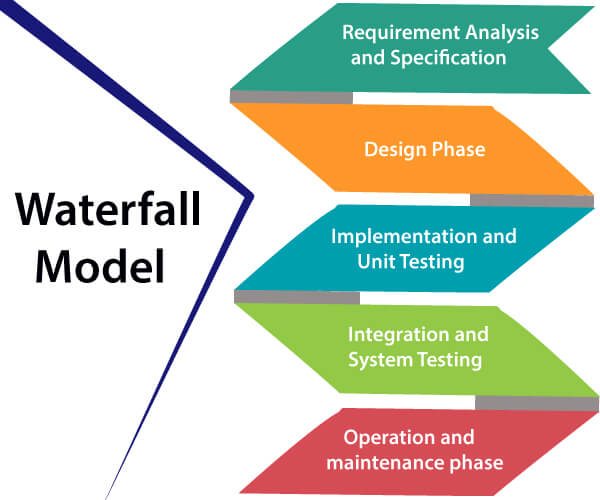
**Here, are some important phases of SDLC life cycle:**



**Waterfall Model**

[Waterfall](https://en.wikipedia.org/wiki/Waterfall_model) represents the oldest, simplest, and most structured methodology. Each phase depends on the outcome of the previous phase, and all phases run sequentially. This model provides discipline and gives a tangible output at the end of each phase. However, this model doesn’t work well when flexibility is a requirement. There is little room for change once a phase is deemed complete, as changes can affect the cost, delivery time, and quality of the software.The biggest drawback of this model is that small details left incomplete can hold up the entire process.

Winston Royce introduced the Waterfall Model in 1970.This model has five phases: Requirements analysis and specification, design, implementation, and unit testing, integration and system testing, and operation and maintenance. The steps always follow in this order and do not overlap. The developer must complete every phase before the next phase begins. This model is named "**Waterfall Model**", because its diagrammatic representation resembles a cascade of waterfalls.



1. **Requirements analysis and specification phase**

The aim of this phase is to understand the exact requirements of the customer and to document them properly. Both the customer and the software developer work together so as to document all the functions, performance, and interfacing requirement of the software. It describes the "what" of the system to be produced and not "how". In this phase, a large document called **Software Requirement Specification (SRS)** document is created which contained a detailed description of what the system will do in the common language.

1. **Design Phase**

This phase aims to transform the requirements gathered in the SRS into a suitable form which permits further coding in a programming language. It defines the overall software architecture together with high level and detailed design. All this work is documented as a Software Design Document (SDD).

1. **Implementation and unit testing**

During this phase, design is implemented. If the SDD is complete, the implementation or coding phase proceeds smoothly, because all the information needed by software developers is contained in the SDD.

During testing, the code is thoroughly examined and modified. Small modules are tested in isolation initially. After that these modules are tested by writing some overhead code to check the interaction between these modules and the flow of intermediate output.

1. **Integration and System Testing**

This phase is highly crucial as the quality of the end product is determined by the effectiveness of the testing carried out. The better output will lead to satisfied customers, lower maintenance costs, and accurate results. Unit testing determines the efficiency of individual modules. However, in this phase, the modules are tested for their interactions with each other and with the system.

1. **Operation and maintenance phase**

Maintenance is the task performed by every user once the software has been delivered to the customer, installed, and operational.

**When to use Waterfall Model?**

Some Circumstances where the use of the Waterfall model is most suited are:

* When the requirements are constant and not changed regularly.
* A project is short
* The situation is calm
* Where the tools and technology used is consistent and is not changing
* When resources are well prepared and are available to use.

**Advantages of Waterfall model**

* This model is simple to implement also the number of resources that are required for it is minimal.
* The requirements are simple and explicitly declared; they remain unchanged during the entire project development.
* The start and end points for each phase is fixed, which makes it easy to cover progress.
* The release date for the complete product, as well as its final cost, can be determined before development.
* It gives easy to control and clarity for the customer due to a strict reporting system.

**Disadvantages of Waterfall model**

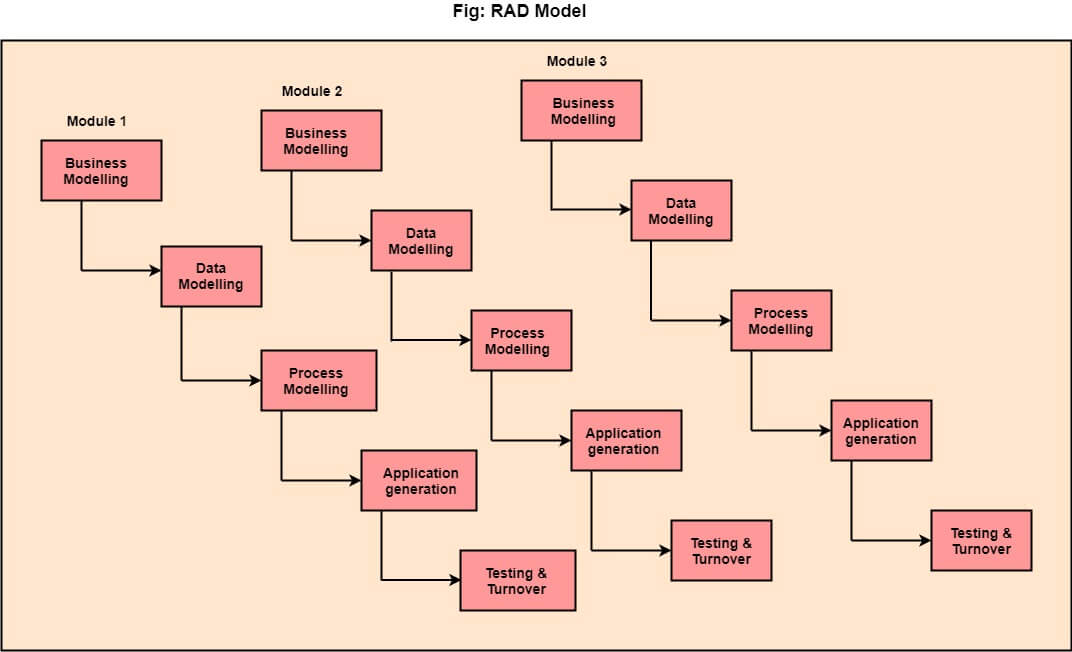
* In this model, the risk factor is higher, so this model is not suitable for more significant and complex projects.
* This model cannot accept the changes in requirements during development.
* It becomes tough to go back to the phase. For example, if the application has now shifted to the coding phase, and there is a change in requirement, it becomes tough to go back and change it.
* Since the testing done at a later stage, it does not allow identifying the challenges and risks in the earlier phase, so the risk reduction strategy is difficult to prepare.

### [RAD Model](https://www.javatpoint.com/software-engineering-rapid-application-development-model) (Rapid Application Development)

RAD is a linear sequential software development process model that emphasizes a concise development cycle using an element based construction approach. If the requirements are well understood and described, and the project scope is a constraint, the RAD process enables a development team to create a fully functional system within a concise time period.

RAD (Rapid Application Development) is a concept that products can be developed faster and of higher quality through:

* Gathering requirements using workshops or focus groups
* Prototyping and early, reiterative user testing of designs
* The re-use of software components
* A rigidly paced schedule that refers design improvements to the next product version
* Less formality in reviews and other team communication



**The various phases of RAD are as follows:**

**1. Business Modelling**

The information flow among business functions is defined by answering questions like what data drives the business process, what data is generated, who generates it, where does the information go, who process it and so on.

**2. Data Modelling**

The data collected from business modeling is refined into a set of data objects (entities) that are needed to support the business. The attributes (character of each entity) are identified, and the relation between these data objects (entities) is defined.

**3. Process Modelling**

The information object defined in the data modeling phase are transformed to achieve the data flow necessary to implement a business function. Processing descriptions are created for adding, modifying, deleting, or retrieving a data object.

**4. Application Generation**

Automated tools are used to facilitate construction of the software; even they use the 4th GL techniques.

**5. Testing & Turnover:**

Many of the programming components have already been tested since RAD emphasis reuse. This reduces the overall testing time. But the new part must be tested, and all interfaces must be fully exercised.

**When to use RAD Model?**

* When the system should need to create the project that modularizes in a short span time (2-3 months).
* When the requirements are well-known.
* When the technical risk is limited.
* When there's a necessity to make a system, which modularized in 2-3 months of period.
* It should be used only if the budget allows the use of automatic code generating tools.

**Advantage of RAD Model**

* This model is flexible for change.
* In this model, changes are adoptable.
* Each phase in RAD brings highest priority functionality to the customer.
* It reduced development time.
* It increases the reusability of features.

**Disadvantage of RAD Model**

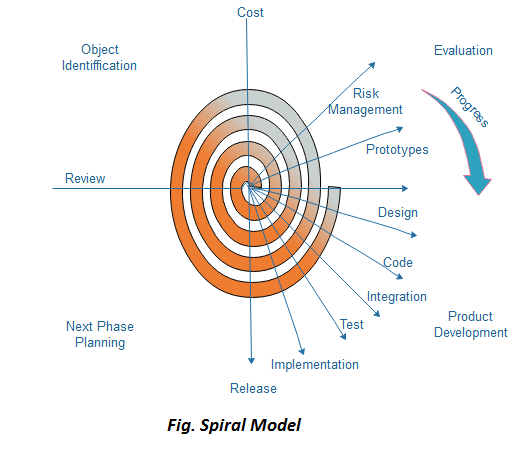
* It required highly skilled designers.
* All application is not compatible with RAD.
* For smaller projects, we cannot use the RAD model.
* On the high technical risk, it's not suitable.
* Required user involvement.

### Spiral Model

The Spiral Model is a software development process model that combines elements of both waterfall and iterative development models. It is a flexible and adaptable approach to software development that emphasizes risk management and continuous improvement.The Spiral Model consists of four main phases that are repeated over and over again, with each iteration called a "spiral.

The spiral model, initially proposed by Boehm, is an evolutionary software process model that couples the iterative feature of prototyping with the controlled and systematic aspects of the linear sequential model. It implements the potential for rapid development of new versions of the software. Using the spiral model, the software is developed in a series of incremental releases. During the early iterations, the additional release may be a paper model or prototype. During later iterations, more and more complete versions of the engineered system are produced.

**The Spiral Model is shown in fig:**



**Each cycle in the spiral is divided into four parts:**

**Objective setting**

Each cycle in the spiral starts with the identification of purpose for that cycle, the various alternatives that are possible for achieving the targets, and the constraints that exists.

**Risk Assessment and reduction**

The next phase in the cycle is to calculate these various alternatives based on the goals and constraints. The focus of evaluation in this stage is located on the risk perception for the project.

**Development and validation**

The next phase is to develop strategies that resolve uncertainties and risks. This process may include activities such as benchmarking, simulation, and prototyping.

**Planning**

Finally, the next step is planned. The project is reviewed, and a choice made whether to continue with a further period of the spiral. If it is determined to keep, plans are drawn up for the next step of the project.

The development phase depends on the remaining risks. For example, if performance or user-interface risks are treated more essential than the program development risks, the next phase may be an evolutionary development that includes developing a more detailed prototype for solving the risks.

The **risk-driven** feature of the spiral model allows it to accommodate any mixture of a specification-oriented, prototype-oriented, simulation-oriented, or another type of approach. An essential element of the model is that each period of the spiral is completed by a review that includes all the products developed during that cycle, including plans for the next cycle. The spiral model works for development as well as enhancement projects.

**When to use Spiral Model?**

* When deliverance is required to be frequent.
* When the project is large
* When requirements are unclear and complex
* When changes may require at any time
* Large and high budget projects
* **Advantages**
* High amount of risk analysis
* Useful for large and mission-critical projects.

**Disadvantages**

* Can be a costly model to use.
* Risk analysis needed highly particular expertise
* Doesn't work well for smaller projects.

The Spiral Model is an effective approach to software development because it allows for a more flexible and adaptable process than traditional waterfall models. It enables developers to identify and address potential problems early in the development process, which can save time and resources in the long run. The model is also useful for projects where requirements are not well-defined or where there is a high degree of uncertainty or risk.

**V-Shaped Model**

V-Model also referred to as the Verification and Validation Model. In this, each phase of SDLC must complete before the next phase starts. It follows a sequential design process same as the waterfall model. Testing of the device is planned in parallel with a corresponding stage of development.



**Verification:** It involves a static analysis method (review) done without executing code. It is the process of evaluation of the product development process to find whether specified requirements meet.

**Validation:** It involves dynamic analysis method (functional, non-functional), testing is done by executing code. Validation is the process to classify the software after the completion of the development process to determine whether the software meets the customer expectations and requirements.

So V-Model contains Verification phases on one side and Validation phases on the other side. Verification and Validation process is joined by coding phase in V-shape. Thus it is known as V-Model.

**There are the various phases of Verification Phase of V-model:**

1. **Business requirement analysis:** This is the first step where product requirements understood from the customer's side. This phase contains detailed communication to understand customer's expectations and exact requirements.
2. **System Design:** In this stage system engineers analyze and interpret the business of the proposed system by studying the user requirements document.
3. **Architecture Design:** The baseline in selecting the architecture is that it should understand all which typically consists of the list of modules, brief functionality of each module, their interface relationships, dependencies, database tables, architecture diagrams, technology detail, etc. The integration testing model is carried out in a particular phase.
4. **Module Design:** In the module design phase, the system breaks down into small modules. The detailed design of the modules is specified, which is known as Low-Level Design
5. **Coding Phase:** After designing, the coding phase is started. Based on the requirements, a suitable programming language is decided. There are some guidelines and standards for coding. Before checking in the repository, the final build is optimized for better performance, and the code goes through many code reviews to check the performance.

**There are the various phases of Validation Phase of V-model:**

1. **Unit Testing:** In the V-Model, Unit Test Plans (UTPs) are developed during the module design phase. These UTPs are executed to eliminate errors at code level or unit level. A unit is the smallest entity which can independently exist, e.g., a program module. Unit testing verifies that the smallest entity can function correctly when isolated from the rest of the codes/ units.
2. **Integration Testing:** Integration Test Plans are developed during the Architectural Design Phase. These tests verify that groups created and tested independently can coexist and communicate among themselves.
3. **System Testing:** System Tests Plans are developed during System Design Phase. Unlike Unit and Integration Test Plans, System Tests Plans are composed by the client?s business team. System Test ensures that expectations from an application developer are met.
4. **Acceptance Testing:** Acceptance testing is related to the business requirement analysis part. It includes testing the software product in user atmosphere. Acceptance tests reveal the compatibility problems with the different systems, which is available within the user atmosphere. It conjointly discovers the non-functional problems like load and performance defects within the real user atmosphere.

**When to use V-Model?**

* When the requirement is well defined and not ambiguous.
* The V-shaped model should be used for small to medium-sized projects where requirements are clearly defined and fixed.
* The V-shaped model should be chosen when sample technical resources are available with essential technical expertise.

**Advantage (Pros) of V-Model:**

1. Easy to Understand.
2. Testing Methods like planning, test designing happens well before coding.
3. This saves a lot of time. Hence a higher chance of success over the waterfall model.
4. Avoids the downward flow of the defects.
5. Works well for small plans where requirements are easily understood.

**Disadvantage (Cons) of V-Model:**

1. Very rigid and least flexible.
2. Not a good for a complex project.
3. Software is developed during the implementation stage, so no early prototypes of the software are produced.
4. If any changes happen in the midway, then the test documents along with the required documents, has to be updated.

The V-Shaped Model is an effective approach to software development because it emphasizes testing throughout the development process. This can help to identify defects early in the development process, reducing the time and cost of fixing them later. It also ensures that the software meets the specified requirements and is suitable for its intended use.

### [Incremental Model](https://www.javatpoint.com/software-engineering-incremental-model)

An incremental model is a software development model where a software product is developed through a series of iterations, each building on the previous one. In other words, an incremental model breaks down the development process into smaller, more manageable chunks that can be completed and delivered one at a time.

Incremental Model is a process of software development where requirements divided into multiple standalone modules of the software development cycle. In this model, each module goes through the requirements, design, implementation and testing phases. Every subsequent release of the module adds function to the previous release. The process continues until the complete system achieved.



**The various phases of incremental model are as follows:**

* 1. **Requirement analysis**

In the first phase of the incremental model, the product analysis expertise identifies the requirements. And the system functional requirements are understood by the requirement analysis team. To develop the software under the incremental model, this phase performs a crucial role.

* 1. **Design & Development:**

In this phase of the Incremental model of SDLC, the design of the system functionality and the development method are finished with success. When software develops new practicality, the incremental model uses style and development phase.

* 1. **Testing:**

In the incremental model, the testing phase checks the performance of each existing function as well as additional functionality. In the testing phase, the various methods are used to test the behavior of each task.

* 1. **Implementation:**

Implementation phase enables the coding phase of the development system. It involves the final coding that design in the designing and development phase and tests the functionality in the testing phase. After completion of this phase, the number of the product working is enhanced and upgraded up to the final system product

**When we use the Incremental Model?**

* When the requirements are superior.
* A project has a lengthy development schedule.
* When Software team are not very well skilled or trained.
* When the customer demands a quick release of the product.
* You can develop prioritized requirements first.

**Advantage of Incremental Model**

* Errors are easy to be recognized.
* Easier to test and debug
* More flexible.
* Simple to manage risk because it handled during its iteration.
* The Client gets important functionality early.

**Disadvantage of Incremental Model**

* Need for good planning
* Total Cost is high.
* Well defined module interfaces are needed.

Overall, the incremental model can be an effective approach for developing complex software projects, particularly when requirements are likely to change over time or when a high degree of flexibility is required. However, it may not be the best choice for every project, and developers should carefully evaluate the needs of the project before selecting a development model

### [Agile Model](https://www.javatpoint.com/software-engineering-agile-model)

The Agile model is a software development methodology that emphasizes flexibility, adaptability, and collaboration among team members. It involves breaking down the development process into small iterations or sprints, each of which typically lasts between one to four weeks

The meaning of Agile is swift or versatile."**Agile process model**" refers to a software development approach based on iterative development. Agile methods break tasks into smaller iterations, or parts do not directly involve long term planning. The project scope and requirements are laid down at the beginning of the development process. Plans regarding the number of iterations, the duration and the scope of each iteration are clearly defined in advance.

Each iteration is considered as a short time "frame" in the Agile process model, which typically lasts from one to four weeks. The division of the entire project into smaller parts helps to minimize the project risk and to reduce the overall project delivery time requirements. Each iteration involves a team working through a full software development life cycle including planning, requirements analysis, design, coding, and testing before a working product is demonstrated to the client.



## Phases of Agile Model:

Following are the phases in the agile model are as follows:

1. Requirements gathering
2. Design the requirements
3. Construction/ iteration
4. Testing/ Quality assurance
5. Deployment
6. Feedback

**1. Requirements gathering:** In this phase, you must define the requirements. You should explain business opportunities and plan the time and effort needed to build the project. Based on this information, you can evaluate technical and economic feasibility.

**2. Design the requirements:** When you have identified the project, work with stakeholders to define requirements. You can use the user flow diagram or the high-level UML diagram to show the work of new features and show how it will apply to your existing system.

**3. Construction/ iteration:** When the team defines the requirements, the work begins. Designers and developers start working on their project, which aims to deploy a working product. The product will undergo various stages of improvement, so it includes simple, minimal functionality.

**4. Testing:** In this phase, the Quality Assurance team examines the product's performance and looks for the bug.

**5. Deployment:** In this phase, the team issues a product for the user's work environment.

**6. Feedback:** After releasing the product, the last step is feedback. In this, the team receives feedback about the product and works through the feedback.

**Agile Testing Methods:**

* Scrum
* Crystal
* Dynamic Software Development Method(DSDM)
* Feature Driven Development(FDD)
* Lean Software Development
* Extreme Programming(XP)

## Scrum

SCRUM is an agile development process focused primarily on ways to manage tasks in team-based development conditions.

There are three roles in it, and their responsibilities are:

* **Scrum Master:** The scrum can set up the master team, arrange the meeting and remove obstacles for the process
* **Product owner:** The product owner makes the product backlog, prioritizes the delay and is responsible for the distribution of functionality on each repetition.
* **Scrum Team:** The team manages its work and organizes the work to complete the sprint or cycle.

### Extreme Programming (XP)

This type of methodology is used when customers are constantly changing demands or requirements, or when they are not sure about the system's performance.

### Crystal:

There are three concepts of this method-

1. Chartering: Multi activities are involved in this phase such as making a development team, performing feasibility analysis, developing plans, etc.
2. Cyclic delivery: under this, two more cycles consist, these are:
   * Team updates the release plan.
   * Integrated product delivers to the users.
3. Wrap up: According to the user environment, this phase performs deployment, post-deployment.

### Dynamic Software Development Method (DSDM):

DSDM is a rapid application development strategy for software development and gives an agile project distribution structure. The essential features of DSDM are that users must be actively connected, and teams have been given the right to make decisions. The techniques used in DSDM are:

1. Time Boxing
2. Moscow Rules
3. Prototyping

**The DSDM project contains seven stages:**

1. Pre-project
2. Feasibility Study
3. Business Study
4. Functional Model Iteration
5. Design and build Iteration
6. Implementation
7. Post-project

### Feature Driven Development (FDD)

This method focuses on "Designing and Building" features. In contrast to other smart methods, FDD describes the small steps of the work that should be obtained separately per function.

### Lean Software Development

Lean software development methodology follows the principle "just in time production." The lean method indicates the increasing speed of software development and reducing costs. Lean development can be summarized in seven phases.

1. Eliminating Waste
2. Amplifying learning
3. Defer commitment (deciding as late as possible)
4. Early delivery
5. Empowering the team
6. Building Integrity
7. Optimize the whole

## When to use the Agile Model?

* When frequent changes are required.
* When a highly qualified and experienced team is available.
* When a customer is ready to have a meeting with a software team all the time.
* When project size is small.

## Advantage (Pros) of Agile Method:

1. Frequent Delivery
2. Face-to-Face Communication with clients.
3. Efficient design and fulfils the business requirement.
4. Anytime changes are acceptable.
5. It reduces total development time.

## Disadvantages (Cons) of Agile Model:

1. Due to the shortage of formal documents, it creates confusion and crucial decisions taken throughout various phases can be misinterpreted at any time by different team members.
2. Due to the lack of proper documentation, once the project completes and the developers allotted to another project, maintenance of the finished project can become a difficulty.

However, the agile model may not be suitable for every project, and it can be challenging to implement in organizations that are used to more traditional software development methodologies. Additionally, some stakeholders may find it difficult to adjust to the iterative and constantly evolving nature of the agile model.

**Iterative Model**

The Iterative Model is a software development methodology that involves developing software through a repeated cycle of planning, designing, building, and testing. It is a type of incremental development approach where software is developed in small portions, known as iterations, and each iteration is built upon the previous one.

In this Model, you can start with some of the software specifications and develop the first version of the software. After the first version if there is a need to change the software, then a new version of the software is created with a new iteration. Every release of the Iterative Model finishes in an exact and fixed period that is called iteration.

The Iterative Model allows the accessing earlier phases, in which the variations made respectively. The final output of the project renewed at the end of the Software Development Life Cycle (SDLC) process.



### The various phases of Iterative model are as follows:

**1. Requirement gathering & analysis:** In this phase, requirements are gathered from customers and check by an analyst whether requirements will fulfill or not. Analyst checks that need will achieve within budget or not. After all of this, the software team skips to the next phase.

**2. Design:** In the design phase, team design the software by the different diagrams like Data Flow diagram, activity diagram, class diagram, state transition diagram, etc.

**3. Implementation:** In the implementation, requirements are written in the coding language and transformed into computer programmes which are called Software.

**4. Testing:** After completing the coding phase, software testing starts using different test methods. There are many test methods, but the most common are white box, black box, and grey box test methods.

**5. Deployment:** After completing all the phases, software is deployed to its work environment.

**6. Review:** In this phase, after the product deployment, review phase is performed to check the behavior and validity of the developed product. And if there are any error found then the process starts again from the requirement gathering.

**7. Maintenance:** In the maintenance phase, after deployment of the software in the working environment there may be some bugs, some errors or new updates are required. Maintenance involves debugging and new addition options.

## When to use the Iterative Model?

1. When requirements are defined clearly and easy to understand.
2. When the software application is large.
3. When there is a requirement of changes in future.

## Advantage (Pros) of Iterative Model:

1. Testing and debugging during smaller iteration is easy.
2. A Parallel development can plan.
3. It is easily acceptable to ever-changing needs of the project.
4. Risks are identified and resolved during iteration.
5. Limited time spent on documentation and extra time on designing.

## Disadvantage (Cons) of Iterative Model:

1. It is not suitable for smaller projects.
2. More Resources may be required.
3. Design can be changed again and again because of imperfect requirements.
4. Requirement changes can cause over budget.
5. Project completion date not confirmed because of changing requirements.

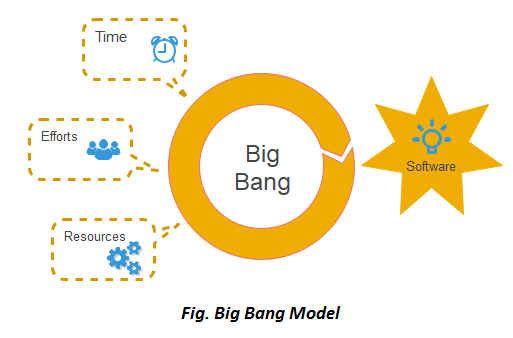
Overall, the Iterative Model can be an effective approach for developing complex software projects, particularly when requirements are likely to change over time or when a high degree of flexibility is required.

### [Big bang model](https://www.javatpoint.com/software-engineering-big-bang-model)

The Big Bang model is a software development model that involves developing and delivering the software in a single, all-encompassing effort, without any prior planning or design. In other words, it is a non-incremental and non-iterative model, where the entire software is developed in a single step.

In this model, developers do not follow any specific process. Development begins with the necessary funds and efforts in the form of inputs. And the result may or may not be as per the customer's requirement, because in this model, even the customer requirements are not defined.

This model is ideal for small projects like academic projects or practical projects. One or two developers can work together on this model.



**When to use Big Bang Model?**

As we discussed above, this model is required when this project is small like an academic project or a practical project. This method is also used when the size of the developer team is small and when requirements are not defined, and the release date is not confirmed or given by the customer.

**Advantage (Pros) of Big Bang Model:**

1. There is no planning required.
2. Simple Model.
3. Few resources required.
4. Easy to manage.
5. Flexible for developers.

**Disadvantage (Cons) of Big Bang Model:**

1. There are high risk and uncertainty.
2. Not acceptable for a large project.
3. If requirements are not clear that can cause very expensive.

Overall, the Big Bang model is generally considered to be a high-risk approach to software development and is not recommended for most software projects. Other software development models, such as the incremental or iterative models, are generally more suitable for complex or large-scale projects.

### [Prototype Model](https://www.javatpoint.com/software-engineering-prototype-model)

The Prototype Model is a software development methodology that involves building a working model of the software product to be developed. The prototype model is an iterative and incremental approach where a working version of the software is developed quickly to provide a visual representation of the final product.

The prototype model requires that before carrying out the development of actual software, a working prototype of the system should be built. A prototype is a toy implementation of the system. A prototype usually turns out to be a very crude version of the actual system, possible exhibiting limited functional capabilities, low reliability, and inefficient performance as compared to actual software. In many instances, the client only has a general view of what is expected from the software product. In such a scenario where there is an absence of detailed information regarding the input to the system, the processing needs, and the output requirement, the prototyping model may be employed.



**Steps of Prototype Model**

1. Requirement Gathering and Analyst
2. Quick Decision
3. Build a Prototype
4. Assessment or User Evaluation
5. Prototype Refinement
6. Engineer Product

**Advantage of Prototype Model**

1. Reduce the risk of incorrect user requirement
2. Good where requirement are changing/uncommitted
3. Regular visible process aids management
4. Support early product marketing
5. Reduce Maintenance cost.
6. Errors can be detected much earlier as the system is made side by side.

**Disadvantage of Prototype Model**

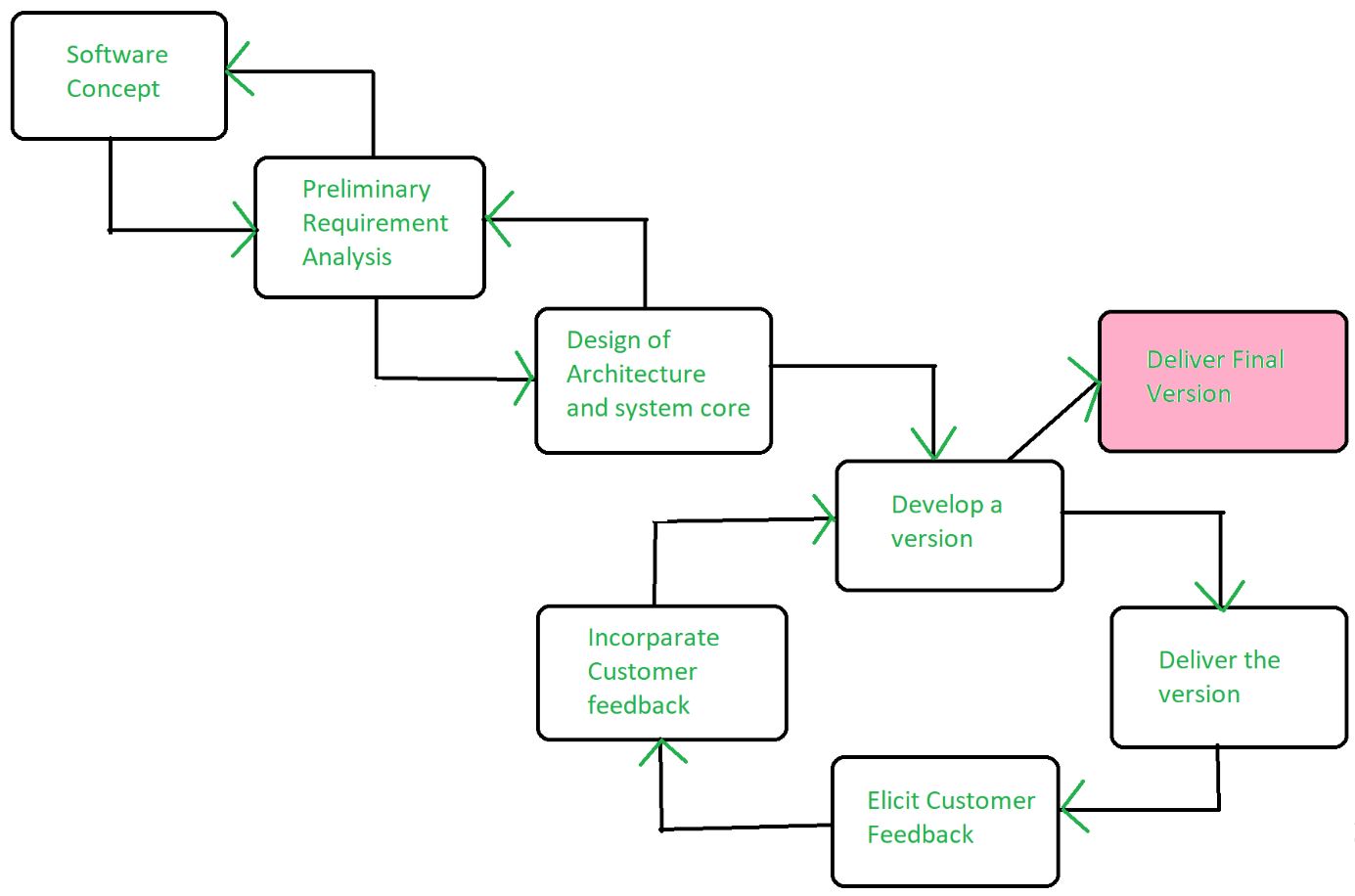
1. An unstable/badly implemented prototype often becomes the final product.
2. Require extensive customer collaboration
   * Costs customer money
   * Needs committed customer
   * Difficult to finish if customer withdraw
   * May be too customer specific, no broad market
3. Difficult to know how long the project will last.
4. Easy to fall back into the code and fix without proper requirement analysis, design, customer evaluation, and feedback.
5. Prototyping tools are expensive.
6. Special tools & techniques are required to build a prototype.
7. It is a time-consuming process.

Overall, the Prototype Model can be an effective approach for developing software products that require a high degree of flexibility and adaptability. However, it should be used in conjunction with other development methodologies to ensure that the final product meets all requirements and is delivered on time and within budget.

**Evolutionary Process Model**

Evolutionary process model resembles the iterative enhancement model. The same phases are defined for the waterfall model occurs here in a cyclical fashion. This model differs from the iterative enhancement model in the sense that this does not require a useful product at the end of each cycle. In evolutionary development, requirements are implemented by category rather than by priority.

For example, in a simple database application, one cycle might implement the graphical user Interface (GUI), another file manipulation, another queries and another updates. All four cycles must complete before there is a working product available. GUI allows the users to interact with the system, file manipulation allow the data to be saved and retrieved, queries allow user to get out of the system, and updates allows users to put data into the system.



**Benefits of Evolutionary Process Model**

1. Use of EVO brings a significant reduction in risk for software projects.
2. EVO can reduce costs by providing a structured, disciplined avenue for experimentation.
3. EVO allows the marketing department access to early deliveries, facilitating the development of documentation and demonstration.
4. Better fit the product to user needs and market requirements.
5. Manage project risk with the definition of early cycle content.
6. Uncover key issues early and focus attention appropriately.
7. Increase the opportunity to hit market windows.
8. Accelerate sales cycles with early customer exposure.
9. Increase management visibility of project progress.
10. Increase product team productivity and motivations.

**Difference between Waterfall and Agile model**

## Agile vs. Waterfall: At a Glance

### Agile Methodology

* **Approach:** Frequent stakeholder interaction
* **Flexibility:** High
* **Requires:** Team initiative and short-term deadlines

[Agile methodology](https://www.forbes.com/advisor/business/what-is-agile-project-management/) was developed as a response to Waterfall’s more rigid structure. As a result, it’s a much more fluid form of project management. A software development project can take years to complete, and technology can change significantly during that time. Agile was developed as a flexible method that welcomes incorporating changes of direction even late in the process, as well as accounting for stakeholders’ feedback throughout the process.

In Agile, the team will work on phases of the project concurrently, often with short-term deadlines. Additionally, the team, rather than a project manager, drives the project’s direction. This can empower the team to be motivated and more productive, but also requires a more self-directed team.

| **Pros** | **Cons** |
| --- | --- |
| Short-term deadlines encourage productivity and efficiency | Because team members are working on multiple phases at a time, there is potential for overlap or unnecessary effort spent on later stages if an early phase needs to be modified |
| There is a lot of flexibility to change project direction and experiment with new directions | Deliverables are not a requirement to progress to the following phase. It can be harder to ensure the entire team is on the same page—especially if it’s a large team with different departments. It also means that work can get lost or miscommunicated between team members, especially when people leave and join the team in the middle of the projects. |
| The methodology is client-facing, which means that the team shares progress and incorporates feedback into the process | The project timeline is more difficult to determine from the start, and it is also more susceptible to change |

### Waterfall Methodology

* **Approach:** Hands-off; goals and outcome established from the beginning
* **Flexibility:** Low
* **Requires:** Completing deliverables to progress to the next phase

[Waterfall methodology](https://www.forbes.com/advisor/business/what-is-waterfall-methodology/) is a linear form of project management ideal for projects where the end result is clearly established from the beginning of the project. The expectations for the project and the deliverables of each stage are clear and are required in order to progress to the next phase.

| **Pros** | **Cons** |
| --- | --- |
| Provides a concrete plan of the project from start to finish | Because each project phase needs to be completed before progressing to the next stage, the process can take longer |
| The team establishes project requirements early on, which can save time | You might not realize an issue with a phase until you have already progressed to the next one. This would mean going back through each phase and checking where the mistake or error occurred, which can be a timely process. |
| Each phase of the project requires a deliverable to progress to the next phase, making the workflow more structured | The Waterfall methodology requires that you outline the project from start to finish before you begin. This doesn't allow for a lot of flexibility or change. Additionally, it can become problematic if the stakeholders disagree on the project’s vision and don’t find out until it is executed or in a later phase. |
|  |  |

## Agile and Waterfall Comparison Chart

|  | **Waterfall** | **Agile** |
| --- | --- | --- |
| Timeline | Waterfall has a fixed timeline. The idea is that the start and finish of the project are already mapped out from the beginning. | Agile is a lot more flexible and accounts for experimenting with different directions. Rather than a fixed timeline, the schedule adapts as the project progresses. The [Agile Manifesto](http://agilemanifesto.org/principles.html), an online document released in 2001 by a group of software developers, says team members are expected to, “​​Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.” |
| Client Involvement | Once the end goal is established, Waterfall does not involve the client or project owner during the process, apart from specific check-ins or for deliverables. The course of the project is outlined from the start, so incorporating client feedback is not an ongoing part of the process. | A fundamental part of Agile is including clients in the project development at every step. The Agile Manifesto states, “Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.” Therefore, business owners are expected to be involved and give feedback to the software development team as they progress through the different phases of the project. |
| Flexibility | Waterfall is not as flexible as Agile because each phase needs to be fully completed before moving on to the next phase. The project is also planned out ahead of time, making this management system ideal for teams with a clear vision of where they are headed from start to finish. | Flexibility is built into the Agile method. Agile values short bursts of work, which are called sprints. The method welcomes adapting to different directions, incorporating new information even at a later stage of the project. |
| Budget | Fixed.  The budget for projects using the Waterfall methodology is generally fixed. Because the project is determined from start to finish, there is less room to change the budget mid-project. | Flexible.  Agile is open to adaptation, encourages experimentation and welcomes changes of direction, even in later phases of the project. Because of this, the budget tends to be more flexible. |

Waterfall is a linear project progression, so it’s best suited for projects with a defined end goal. If a project owner has a clear and specific vision of an app, for example, and is confident it will not change throughout the project development, Waterfall methodologies could be a good system to follow.

Meanwhile, Agile leaves a lot of room to adapt and change course as the project develops. It’s better suited for projects where the outcome may be dependent on more research or testing.

The budget for projects using Waterfall methodologies tends to be less flexible because the project is mapped out from the beginning. With Agile, there is more room to change direction as the project develops, so the budget is also subject to change. Similarly, the timeline with Waterfall is set from the start, while it’s more flexible with Agile and dependent on how the project develops.

## Waterfall vs. Agile: How to Choose the Right Methodology for Your Project

There are several factors to consider when you are choosing between Waterfall and Agile. Here are a few questions to consider:

Does your project require strict regulations or requirements? Waterfall is better suited for projects with regulations or requirements because each phase’s deliverables and strict procedures ensure that they are met. For instance, the Department of Defense and the aerospace industry are a couple of industries that would more likely use Waterfall over Agile, since the requirements are a safety factor. [Dr. Chris Mattmann](https://www.mattmann.ai/) , Chief Technology and Innovation Officer (CTIO) at NASA Jet Propulsion Laboratory, told Forbes Advisor that “agile methodology is used more for IT companies, [companies] that fail fast and move fast, types of places where you can proceed in parallel in different phases.”

When choosing between Agile and Waterfall, consider how involved the project owners or stakeholders will be in the project. Agile is better suited for projects where stakeholders are closely involved every step of the way. Waterfall is a more structured project management method and does not lend itself to the same type of flexibility.

### Bottom Line

To summarize, Agile and Waterfall are two different management methodologies best suited for different types of projects. If you clearly understand the project outcomes from the beginning, Waterfall may be the best fit. Waterfall is a better method when a project must meet strict regulations as it requires deliverables for each phase before proceeding to the next one.

Alternatively, Agile is better suited for teams that plan on moving fast, experimenting with direction and don’t know how the final project will look before they start. Agile is flexible and requires a collaborative and self-motivated team, plus frequent check-ins with business owners and stakeholders about the progress.

Finally, in March of 2011, Cameron Haight of Gartner presented his predictions for the trajectory of DevOps over the next few years. His positive outlook on its impact on the industry lead to more attention for the DevOps movement, and it wasn’t long before enterprises of all sizes were beginning to adopt these new practices. DevOps had officially caught on as the next big thing since Agile for the IT industry.