# Software,

Software=Program+Documentation+Operating Procedures

# Software engineering :

The application of a systematic,disciplined,quantifiable approach to the development,operation and maintenance of software; that is, the application of engineering to software.

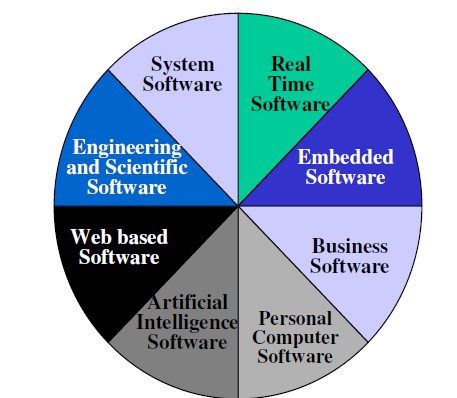
Let us first understand what software engineering stands for. The term is made of two words, software and engineering.

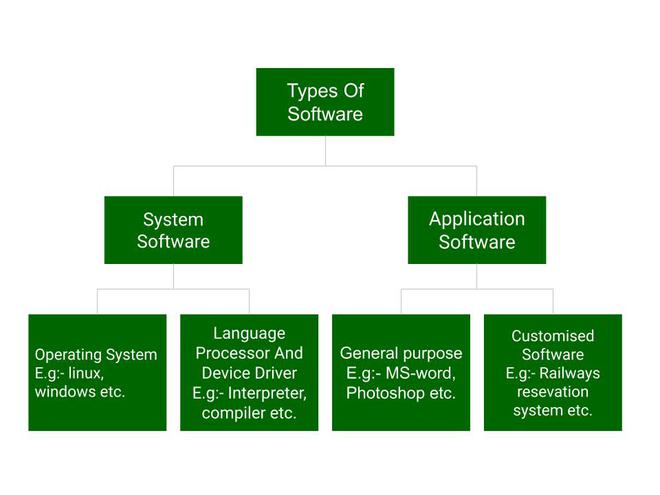
**Software**is more than just a program code. A program is an executable code, which serves some computational purpose. Software is considered to be collection of executable programming code, associated libraries and documentations. Software, when made for a specific requirement is called **software product.**

**Engineering** on the other hand, is all about developing products, using well-defined, scientific principles and methods.

**Software engineering** is an engineering branch associated with development of software product using well-defined scientific principles, methods and procedures. The outcome of software engineering is an efficient and reliable software product.

# types of software





Above is the diagram of types of software. Now we will briefly describe each type and its subtypes:

1. **System Software**
   * Operating System
   * Language Processor
   * Device Driver
2. **Application Software**
   * General Purpose Software
   * Customize Software
   * Utility Software

**System Software**

[System software](https://www.geeksforgeeks.org/system-software/)is software that directly operates the [computer hardware](https://www.geeksforgeeks.org/computer-hardware/) and provides the basic functionality to the users as well as to the other software to operate smoothly. Or in other words, system software basically controls a computer’s internal functioning and also controls hardware devices such as monitors, printers, and storage devices, etc. It is like an interface between hardware and user applications, it helps them to communicate with each other because hardware understands machine language(i.e. 1 or 0) whereas user applications are work in human-readable languages like English, Hindi, German, etc. so system software converts the human-readable language into machine language and vice versa.

**Types of System Software**

It has two subtypes which are:

1. **Operating System:** It is the main program of a computer system. When the computer system ON it is the first software that loads into the computer’s memory. Basically, it manages all the resources such as [computer memory](https://www.geeksforgeeks.org/computer-memory/), [CPU](https://www.geeksforgeeks.org/central-processing-unit-cpu/), [printer](https://www.geeksforgeeks.org/what-is-a-printer/), hard disk, etc., and provides an interface to the user, which helps the user to interact with the computer system. It also provides various services to other computer software. Examples of operating systems are [Linux](https://www.geeksforgeeks.org/introduction-to-linux-operating-system/), Apple macOS, [Microsoft Windows](https://www.geeksforgeeks.org/interesting-facts-about-windows/), etc.
2. **Language Processor:**As we know that system software converts the human-readable language into a machine language and vice versa. So, the conversion is done by the language processor. It converts programs written in high-level [programming languages](https://www.geeksforgeeks.org/introduction-to-programming-languages/) like[Java](https://www.geeksforgeeks.org/introduction-to-java/), [C](https://www.geeksforgeeks.org/c-plus-plus/),[C++](https://www.geeksforgeeks.org/c-plus-plus/), [Python](https://www.geeksforgeeks.org/history-of-python/), etc(known as source code), into sets of instructions that are easily readable by machines(known as object code or machine code).
3. **Device Driver:**A [device driver](https://www.geeksforgeeks.org/device-driver-and-its-purpose/)is a program or software that controls a device and helps that device to perform its functions. Every device like a printer, mouse, [modem](https://www.geeksforgeeks.org/how-to-install-a-modem/), etc. needs a driver to connect with the computer system eternally. So, when you connect a new device with your computer system, first you need to install the driver of that device so that your operating system knows how to control or manage that device.

**Features of System Software**

Let us discuss some of the features of System Software:

* System Software is closer to the computer system.
* System Software is written in a low-level language in general.
* System software is difficult to design and understand.
* System software is fast in speed(working speed).
* System software is less interactive for the users in comparison to application software.

**Application Software**

Software that performs special functions or provides functions that are much more than the basic operation of the computer is known as [application software](https://www.geeksforgeeks.org/what-is-application-software/). Or in other words, application software is designed to perform a specific task for end-users. It is a product or a program that is designed only to fulfill end-users’ requirements. It includes word processors, [spreadsheets](https://www.geeksforgeeks.org/introduction-to-excel-spreadsheet/), database management, inventory, payroll programs, etc.

**Types of Application Software**

There are different types of application software and those are:

1. **General Purpose Software:**This type of application software is used for a variety of tasks and it is not limited to performing a specific task only. For example, MS-Word, MS-Excel, PowerPoint, etc.
2. **Customized Software:**This type of application software is used or designed to perform specific tasks or functions or designed for specific organizations. For example, [railway reservation system](https://www.geeksforgeeks.org/railway-reservation-system-in-c/), airline reservation system, invoice management system, etc.
3. **Utility Software:**This type of application software is used to support the computer infrastructure. It is designed to analyze, configure, optimize and maintains the system, and take care of its requirements as well. For example, [antivirus](https://www.geeksforgeeks.org/how-an-antivirus-works/), disk fragmenter, memory tester, disk repair, disk cleaners, registry cleaners, disk space analyzer, etc.

**Features of Application Software**

Let us discuss some of the features of Application Software:

* An important feature of application software is it performs more specialized tasks like word processing, spreadsheets, [email](https://www.geeksforgeeks.org/what-is-an-email/), etc.
* Mostly, the size of the software is big, so it requires more storage space.
* Application software is more interactive for the users, so it is easy to use and design.
* The application software is easy to design and understand.
* Application software is written in a high-level language in general.

**Difference Between System Software and Application Software**

Now, let us discuss some difference between system software and application software:

| **System Software** | **Application Software** |
| --- | --- |
| It is designed to manage the resources of the computer system, like memory and process management, etc. | It is designed to fulfill the requirements of the user for performing specific tasks. |
| Written in a low-level language. | Written in a high-level language. |
| Less interactive for the users. | More interactive for the users. |
| System software plays vital role for the effective functioning of a system. | Application software is not so important for the functioning of the system, as it is task specific. |
| It is independent of the application software to run. | It needs system software to run. |

For more Information you can refer to this article on – [Difference between System Software and Application Software](https://www.geeksforgeeks.org/difference-between-system-software-and-application-software/).

# types of software engineering

# evolving role of software,

# challenges in software development.

Why does it take so long to get the program finished?

Why are costs so high?

Why can not we find all errors before release?

Why do we have difficulty in measuring progress of software development?

• Larger problems,

• Lack of adequate training in software engineering,

• Increasing skill shortage,

• Low productivity improvements.

**1. Health Problems Because of Longer Working Periods:**

Long periods of time spent in front of a computer by software engineers can cause a number of health concerns, including back discomfort, eye strain, and disorders associated with a sedentary lifestyle. It is crucial that software developers put their health first, make time for regular exercise, and follow ergonomic guidelines.

**2. Project timelines that are difficult to meet:**

Software engineering projects frequently have rushed completion dates and stressful work conditions. It can be challenging to meet project deadlines and deliver on schedule, which can result in stress and burnout. These difficulties can be lessened with good time management, thorough preparation, and reasonable expectations.

**3. Periodical updates in Technology:**

Technology is always changing, and the software engineering industry is no exception. It may be difficult to stay current with the newest programming languages, frameworks, and tools. To stay competitive, software engineers must regularly upgrade their knowledge and follow market trends.

**4. Security Concerns:**

As our reliance on software systems grows, security flaws and online dangers have taken on greater importance. Implementing strong security measures and having a thorough grasp of potential threats is essential for creating secure software. Inadequate security measures can lead to data breaches and jeopardize user privacy.

**5. Price issues:**

Software system development and upkeep can be costly. Hardware, software licenses, development tools, and ongoing maintenance are frequently expensive components of software engineering projects. Budget restraints and cost overruns can be problematic for businesses and have an impact on the success of projects.

**6. Restricted Monitoring:**

The project scope, decision-making process, and project management elements of software projects may be subject to restricted software engineer control. They could have to operate within the limitations imposed by stakeholders or project managers, which may limit their creativity and liberty.

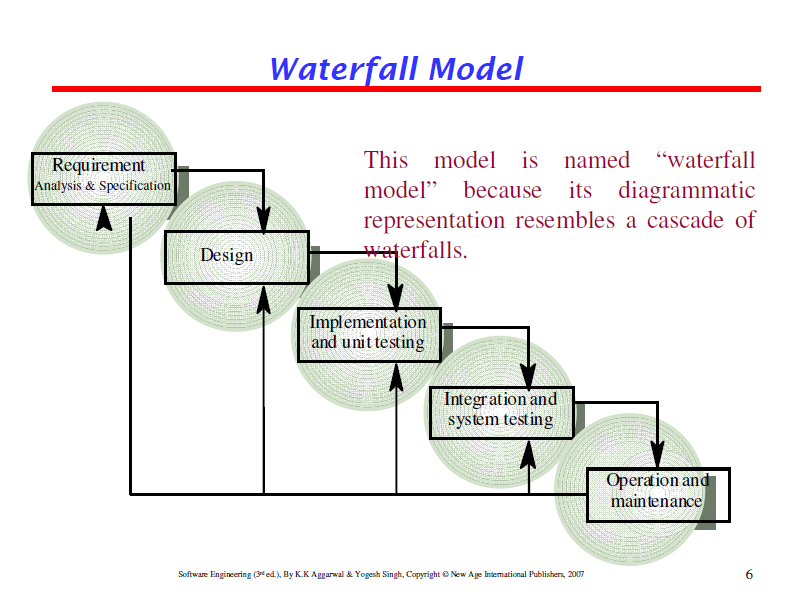
**7. Not enough Assistance:**

Software engineers may experience an insufficient amount of mentoring, resources, or support in some organizations. This may impede their capacity to advance professionally and efficiently handle difficult problems. To get over these constraints, software developers must actively seek out learning and growth opportunities.

These are the different disadvantages of software engineering, some of them feel very easy to overcome, and some are difficult to overcome, but everything needs a start.

**software Development Process Model**

Waterfall Model,



This model is easy to understand and reinforces

the notion of “define before design” and “design

before code”.

The model expects complete & accurate

requirements early in the process, which is

unrealistic

**Problems of waterfall model**

i. It is difficult to define all requirements at the beginning of a

project

ii. This model is not suitable for accommodating any change

iii. A working version of the system is not seen until late in

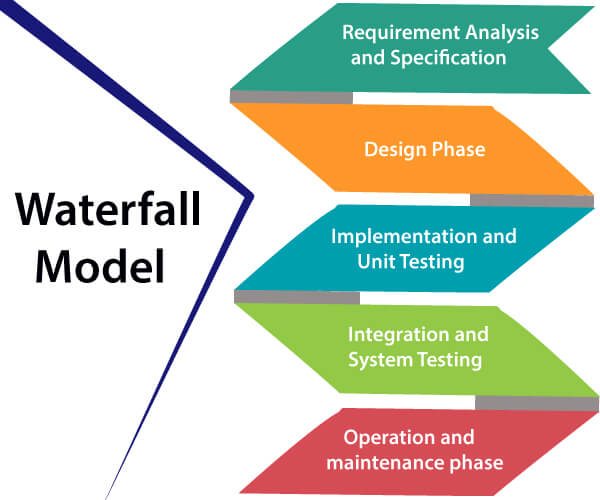
the project’s life

iv. It does not scale up well to large projects.

v. Real projects are rarely sequential.

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.



**2. 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).

**3. 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.

**4. 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.

**5. 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 SDLC 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.

# Evolutionary Process Model:

Evolutionary process model resembles iterative enhancement

model. The same phases as defined for the waterfall model occur

here in a cyclical fashion. This model differs from iterative

enhancement model in the sense that this does not require a

useable product at the end of each cycle. In evolutionary

development, requirements are implemented by category rather

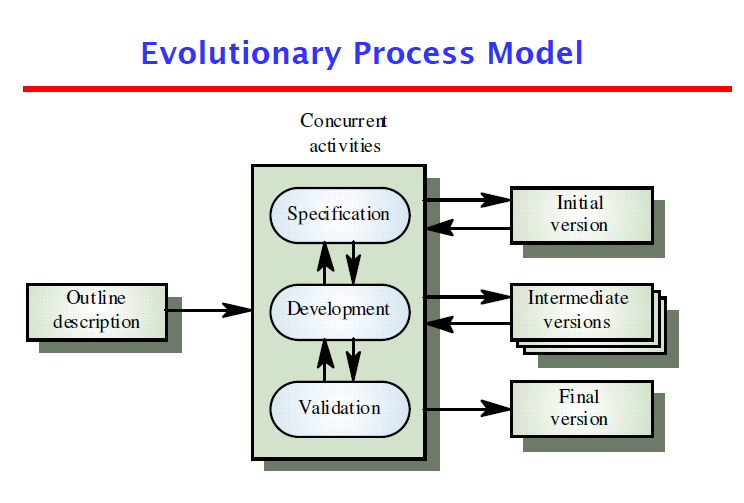
than by priority.

This model is useful for projects using new technology that is not

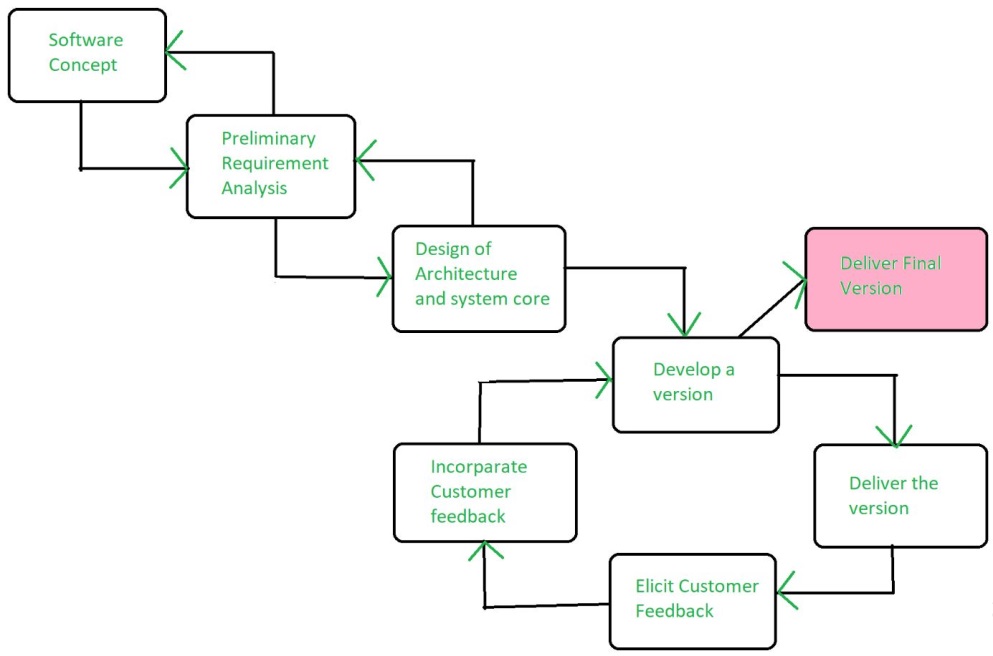
well understood. This is also used for complex projects where all

functionality must be delivered at one time, but the requirements

are unstable or not well understood at the beginning.



**Evolutionary model** is a combination of [Iterative](https://www.geeksforgeeks.org/software-engineering-iterative-waterfall-model/)and [Incremental model](https://www.geeksforgeeks.org/software-engineering-incremental-process-model/) of software development life cycle. Delivering your system in a big bang release, delivering it in incremental process over time is the action done in this model. Some initial requirements and architecture envisioning need to be done. It is better for software products that have their feature sets redefined during development because of user feedback and other factors. The Evolutionary development model divides the development cycle into smaller, incremental waterfall models in which users are able to get access to the product at the end of each cycle. Feedback is provided by the users on the product for the planning stage of the next cycle and the development team responds, often by changing the product, plan or process. Therefore, the software product evolves with time. All the models have the disadvantage that the duration of time from start of the project to the delivery time of a solution is very high. Evolutionary model solves this problem in a different approach.



Evolutionary model suggests breaking down of work into smaller chunks, prioritizing them and then delivering those chunks to the customer one by one. The number of chunks is huge and is the number of deliveries made to the customer. The main advantage is that the customer’s confidence increases as he constantly gets quantifiable goods or services from the beginning of the project to verify and validate his requirements. The model allows for changing requirements as well as all work is broken down into maintainable work chunks.

**Application of Evolutionary Model:**

1. It is used in large projects where you can easily find modules for incremental implementation. Evolutionary model is commonly used when the customer wants to start using the core features instead of waiting for the full software.
2. Evolutionary model is also used in object oriented software development because the system can be easily portioned into units in terms of objects.

**Necessary conditions for implementing this model:-**

* Customer needs are clear and been explained in deep to the developer team.
* There might be small changes required in separate parts but not a major change.
* As it requires time, so there must be some time left for the market constraints.
* Risk is high and continuous targets to achieve and report to customer repeatedly.
* It is used when working on a technology is new and requires time to learn.

**Advantages:**

* In evolutionary model, a user gets a chance to experiment partially developed system.
* It reduces the error because the core modules get tested thoroughly.

**Disadvantages:**

* Sometimes it is hard to divide the problem into several versions that would be acceptable to the customer which can be incrementally implemented and delivered.

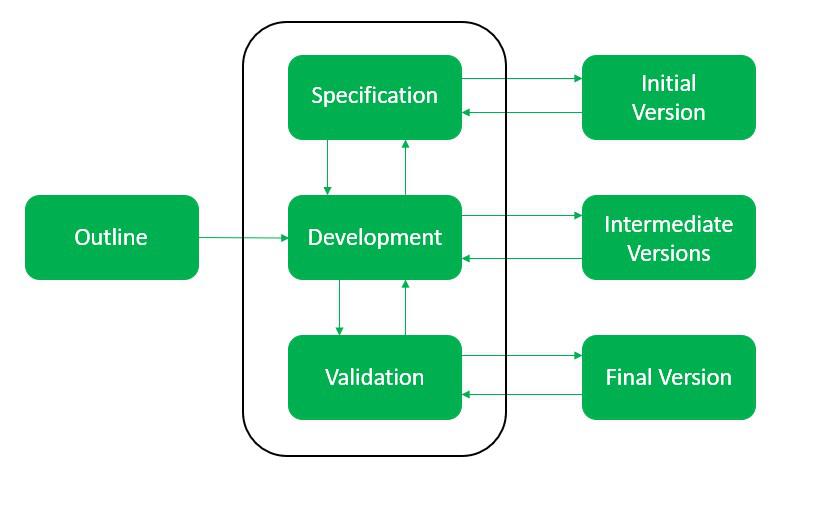
Evolutionary model is a combination of Iterative and Incremental model of [software development life cycle](https://www.geeksforgeeks.org/software-development-life-cycle-sdlc/). In this article, we are going to understand different types of evolutionary process models with the help of examples.

**Software Process Model**

A software process model is a structured representation of the activities of the software development process. During the development of software, various steps that are important for the successful development of the project are taken and if we structured them according to the proper order in a model then it is called a [software process model](https://www.geeksforgeeks.org/software-processes-in-software-engineering/). The software process model includes various activities such as steps like planning, designing, implementation, defining tasks, setting up milestones, roles, and responsibilities, etc.

**Evolutionary Process Model**

The evolutionary model is based on the concept of making an initial product and then evolving the software product over time with iterative and incremental approaches with proper feedback. In this type of model, the product will go through several iterations and come up when the final product is built through multiple iterations. The development is carried out simultaneously with the feedback during the development. This model has a number of advantages such as customer involvement, taking feedback from the customer during development, and building the exact product that the user wants. Because of the multiple iterations, the chances of errors get reduced and the reliability and efficiency will increase.



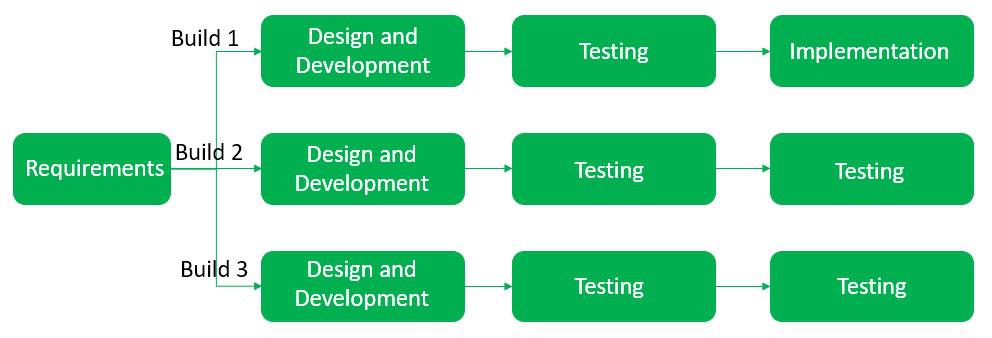
*Evolutionary Model*

Types of Evolutionary Process Models

1. Iterative Model
2. Incremental Model
3. Spiral Model

**Iterative Model**

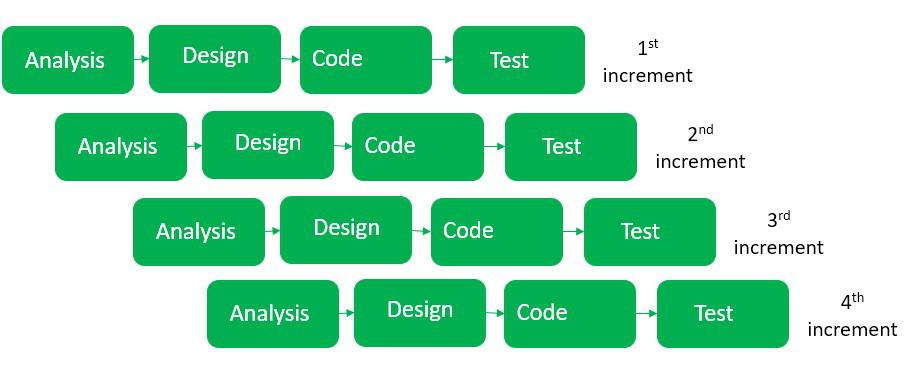
In the iterative model first, we take the initial requirements then we enhance the product over multiple iterations until the final product gets ready. In every iteration, some design modifications were made and some changes in functional requirements is added. The main idea behind this approach is to build the final product through multiple iterations that result in the final product being almost the same as the user wants with fewer errors and the performance, and quality would be high.



*Iterative model*

**Incremental Model**

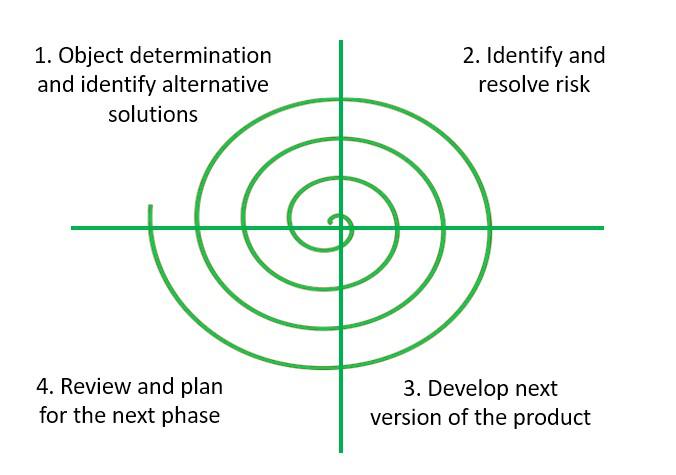
In the incremental model, we first build the project with basic features and then evolve the project in every iteration, it is mainly used for large projects. The first step is to gather the requirements and then perform analysis, design, code, and test and this process goes the same over and over again until our final project is ready.



*Incremental Model*

**Spiral Model**

The spiral model is a combination of waterfall and iterative models and in this, we focused on risk handling along with developing the project with the incremental and iterative approach, producing the output quickly as well as it is good for big projects. The software is created through multiple iterations using a spiral approach. Later on, after successive development the final product will develop, and the customer interaction is there so the chances of error get reduced.



*Spiral Model*

**Advantages of the Evolutionary Process Model**

1. During the development phase, the customer gives feedback regularly because the customer’s requirement gets clearly specified.
2. After every iteration risk gets analyzed.
3. Suitable for big complex projects.
4. The first build gets delivered quickly as it used an iterative and incremental approach.

**Disadvantages of the Evolutionary Process Model**

1. It is not suitable for small projects.
2. The complexity of the spiral model can be more than the other sequential models.
3. The cost of developing a product through a spiral model is high.

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

Use of EVO brings a significant reduction in risk for software projects.

EVO can reduce costs by providing a structured, disciplined avenue for experimentation.

EVO allows the marketing department access to early deliveries, facilitating the development of documentation and demonstration.

Better fit the product to user needs and market requirements.

Manage project risk with the definition of early cycle content.

Uncover key issues early and focus attention appropriately.

Increase the opportunity to hit market windows.

Accelerate sales cycles with early customer exposure.

Increase management visibility of project progress.

Increase product team productivity and motivations.

# Prototype model and

The prototype may be a usable program but is not suitable as

the final software product.

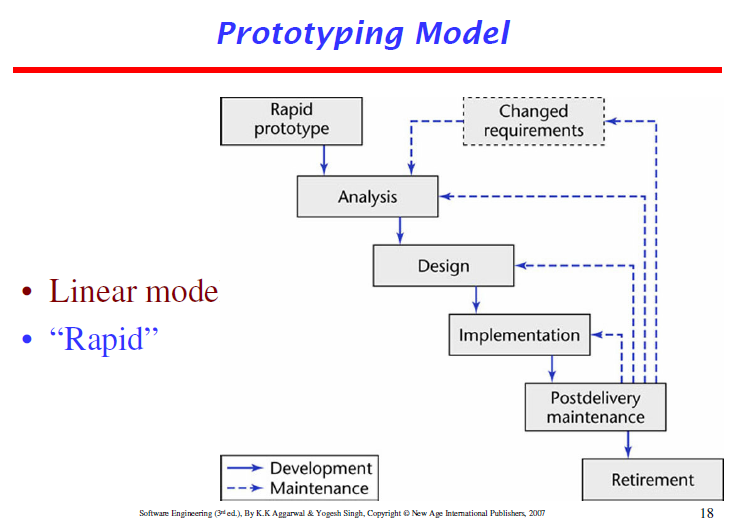
The code for the prototype is thrown away. However

experience gathered helps in developing the actual system.

The development of a prototype might involve extra cost, but

overall cost might turnout to be lower than that of an

equivalent system developed using the waterfall model.



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.

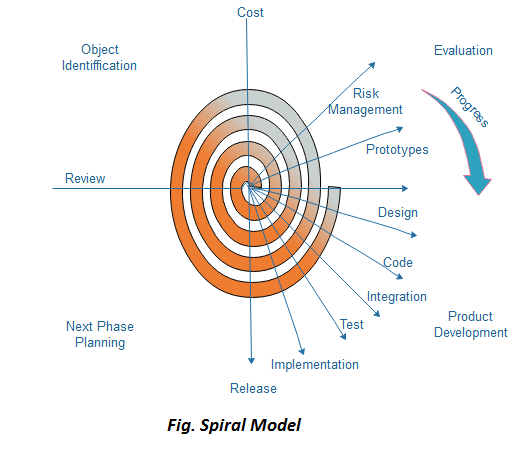
Disadvantage of Prototype Model

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.

# Spiral Model,

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.

Models do not deal with uncertainly which is inherent to software

projects.

Important software projects have failed because project risks were

neglected & nobody was prepared when something unforeseen

happened.

Barry Boehm recognized this and tired to incorporate the “project

risk” factor into a life cycle model.

The result is the spiral model, which was presented in 1986.



The radial dimension of the model represents the cumulative costs.

Each path around the spiral is indicative of increased costs. The

angular dimension represents the progress made in completing each

cycle. Each loop of the spiral from X-axis clockwise through 360o

represents one phase. One phase is split roughly into four sectors of

major activities.

Planning: Determination of objectives, alternatives &

constraints.

Risk Analysis: Analyze alternatives and attempts to identify

and resolve the risks involved.

Development: Product development and testing product.

Assessment: Customer evaluation

An important feature of the spiral model is that each phase is

completed with a review by the people concerned with the

project (designers and programmers)

The advantage of this model is the wide range of options to

accommodate the good features of other life cycle models.

It becomes equivalent to another life cycle model in

appropriate situations.

The spiral model has some difficulties that need to be resolved

before it can be a universally applied life cycle model. These

difficulties include lack of explicit process guidance in determining

objectives, constraints, alternatives; relying on risk assessment

expertise; and provides more flexibility than required for many

applications.

# Spiral vs waterfall

# Incremental Process model:

They are effective in the situations where requirements are

defined precisely and there is no confusion about the

functionality of the final product.

After every cycle a useable product is given to the customer.

Popular particularly when we have to quickly deliver a limited

functionality system.

# Iterative approach,

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.

**2. 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.

**3. 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.

**4. 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.

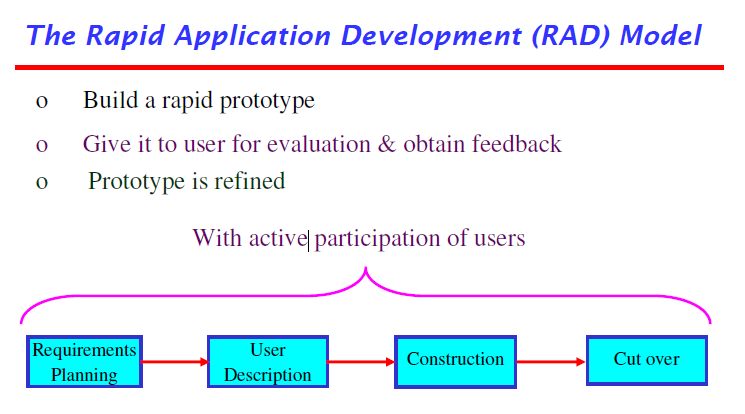
Advantage of Incremental Model

* 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.

# RAD model,



Not an appropriate model in the absence of user

participation.

Reusable components are required to reduce development

time.

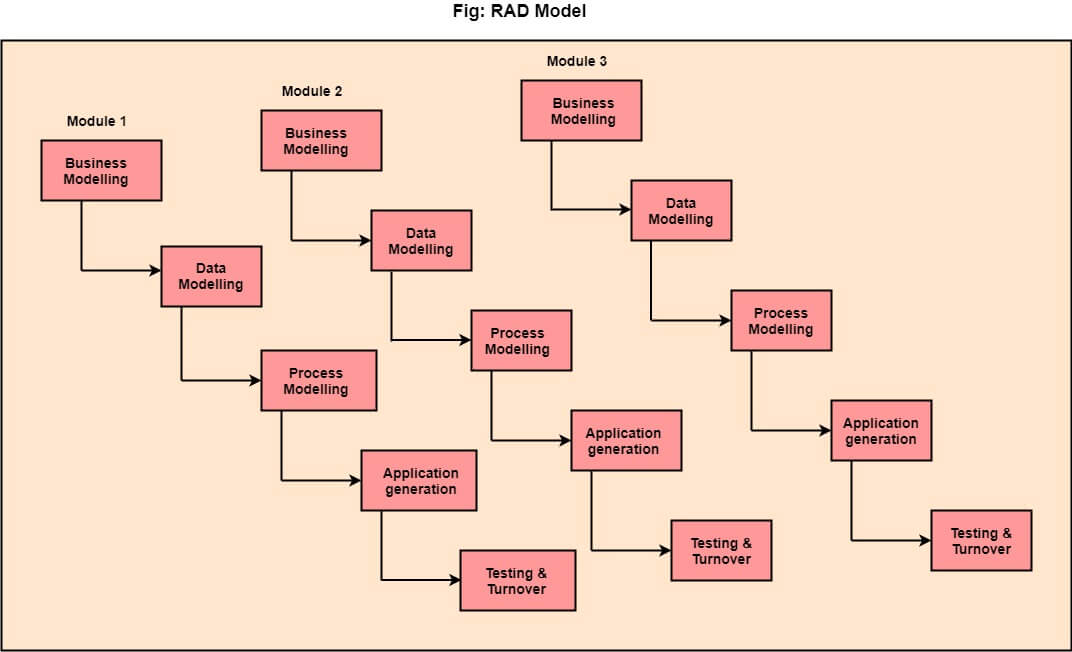
Highly specialized & skilled developers are required and

such developers are not easily available.

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.

# What is Agility?

Agility means characteristics of being dynamic, content specific, aggressively change embracing and growth oriented.

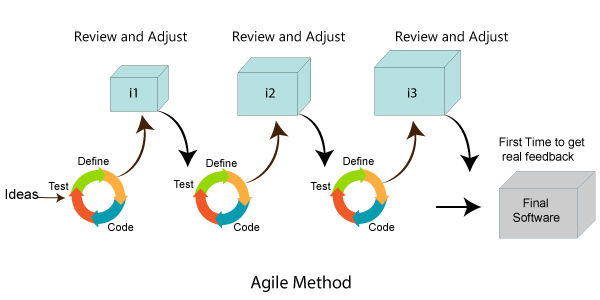
# 12 Agile principles

The Twelve Principle of Agile Manifesto

1. **Customer Satisfaction:** Manifesto provides high priority to satisfy the costumer's requirements. This is done through early and continuous delivery of valuable software.
2. **Welcome Change:** Making changes during software development is common and inevitable. Every changingrequirement should be welcome, evenin the late development phase. Agile process works to increase the customers' competitive advantage.
3. **Deliver the Working Software:** Deliver the working software frequently, ranging from a few weeks to a few months with considering the shortest timeperiod.
4. **Collaboration:** Business people (Scrum Master and Project Owner) and developers must work together during the entire life of a project development phase.
5. **Motivation:** Projects should be build around motivated team members. Provide such environment that supportsindividual team members and trust them. It makes them feel responsible for gettingthe job donethoroughly.
6. **Face-to-face Conversation:** Face-to-face conversation betweenScrum Master anddevelopment team and between the Scrum Master and customers for the most efficient and effective method of conveying information to and within a development team.
7. **Measure the Progress as per the Working Software:** The working software is the key and primary measure of the progress.
8. **Maintain Constant Pace:** The aim of agile development is sustainable development. All the businesses and users should be able to maintain a constant pace with the project.
9. **Monitoring:** Pay regular attention to technical excellence and good design to maximize agility.
10. **Simplicity:** Keep things simple and use simple terms to measure the work that is not completed.
11. **Self-organized Teams:** The Agile team should be self-organized. They should not be depending heavily on other teams because the best architectures, requirements, and designs emerge from self-organized teams.
12. **Review the Work Regularly:** The work should be reviewed at regular intervals, so that the team canreflect on how to become more productive and adjust its behavior accordingly.

Advantages of Agile Methodology

1. Customer satisfaction is rapid, continuous development and delivery of useful software.
2. Customer, Developer, and Product Owner interact regularly to emphasize rather than processes and tools.
3. Product is developed fast and frequently delivered (weeks rather than months.)
4. A face-to-face conversation is the best form of communication.
5. It continuously gave attention to technical excellence and good design.
6. Daily and close cooperation between business people and developers.
7. Regular adaptation to changing circumstances.
8. Even late changes in requirements are welcomed.

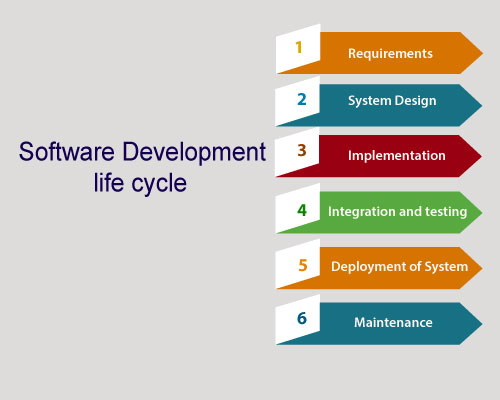


Disadvantages of Agile methodology:

1. It is not useful for small development projects.
2. There is a lack of intensity on necessary designing and documentation.
3. It requires an expert project member to take crucial decisions in the meeting.
4. Cost of Agile development methodology is slightly more as compared to other development methodology.
5. The project can quickly go out off track if the project manager is not clear about requirements and what outcome he/she wants.

Advantages of the Waterfall Model:

1. It is one of the easiest and traditional model to manage. Because of its traditional development nature, each phase has specific deliverables and a review process.
2. It works well in smaller size projects where requirements are easily understandable.
3. It has a faster product delivery model.
4. There are well-documented process and results.
5. Easily adaptable method for shifting teams
6. This project management methodology is beneficial to manage dependencies.



Disadvantages of Waterfall Model:

1. It is not an ideal model to develop a large scale project size.
2. It requires a clear-cut requirement at the beginning time; otherwise, it may lead to a less effective method.
3. It is difficult to move back to make changes in the previous phase.
4. The testing process starts once development is completed. Hence, it has high chances of bugs to be found later in project development. Due to this, it is costly to fix.

Compression between the Agile methodology and Waterfall model:

|  |  |
| --- | --- |
| **Agile methodology** | **Waterfall model** |
| It follows the incremental approach. | It is a sequential design process. |
| It divides the project development lifecycle into a sprint. | The software development process is divided into distinct phases. |
| Agile methodology is a flexible methodology. | The Waterfall is a structured software development methodology. |
| Agile is the collection of many different projects. | It is completed as one single project. |
| The test plan is reviewed after each sprint | Test plan is reviewed after complete development. |
| Testing team can take part in the requirements change phase without problems. | It is difficult for the test to initiate any change in needs. |

# Agile Definition of Done

Agile Definition of **done** is defined into three different stages called User Story (Requirement), Iteration, and product Release. These are given below:

User Story (requirement)

A user story is a requirement which is formulated into few sentences. The user requirement is the everyday language of user. This user story should be completed within iteration. The user story is done when

* All the related code and documentation have been checked-in.
* The product passed all the processes of unit test.
* All the processes of the acceptance test case have been moved.
* The product owner must have accepted the story.
* The help text (documentation) is written.

Iteration

An iteration is a time-based collection of a user story. It works on the defected product and accepted within the release of a product. Iteration is defined at the time of the iteration planning meeting and completed within the iteration demo and review meeting. The iteration is also known as a sprint. The repetition is required when:

* Performance of the product has been tested.
* Product backup is complete.
* User requirement has been accepted or moved for the next iteration.
* Defected product has been fixed or postponed for the next iteration.

Release

The product release is a major occasion that represents an internal and external delivery of work. It also tests the version of the product or system. The product release is done when:

* The system is stress tested.
* Performance is high.
* Contain the security validation in the product.
* Disaster recovery plan is tested.

# Agility

# and the cost of change,

# what is an agile process?,

# Extreme programming,

Extreme programming (XP) is one of the most important software development frameworks of Agile models. It is used to improve software quality and responsiveness to customer requirements. The extreme programming model recommends taking the best practices that have worked well in the past in program development projects to extreme levels.

**Good practices need to be practiced in extreme programming:** Some of the good practices that have been recognized in the extreme programming model and suggested to maximize their use are given below:

* **Code Review:** Code review detects and corrects errors efficiently. It suggests pair programming as coding and reviewing of written code carried out by a pair of programmers who switch their works between them every hour.
* **Testing:** Testing code helps to remove errors and improves its reliability. XP suggests test-driven development (TDD) to continually write and execute test cases. In the TDD approach, test cases are written even before any code is written.
* **Incremental development:**Incremental development is very good because customer feedback is gained and based on this development team comes up with new increments every few days after each iteration.
* **Simplicity:**Simplicity makes it easier to develop good quality code as well as to test and debug it.
* **Design:**Good quality design is important to develop good quality software. So, everybody should design daily.
* **Integration testing:** It helps to identify bugs at the interfaces of different functionalities. Extreme programming suggests that the developers should achieve continuous integration by building and performing integration testing several times a day.

**Basic principles of Extreme programming:** XP is based on the frequent iteration through which the developers implement User Stories. User stories are simple and informal statements of the customer about the functionalities needed. A User Story is a conventional description by the user of a feature of the required system. It does not mention finer details such as the different scenarios that can occur. Based on User stories, the project team proposes Metaphors. Metaphors are a common vision of how the system would work. The development team may decide to build a Spike for some features. A Spike is a very simple program that is constructed to explore the suitability of a solution being proposed. It can be considered similar to a prototype. Some of the basic activities that are followed during software development by using the XP model are given below:

* **Coding:** The concept of coding which is used in the XP model is slightly different from traditional coding. Here, the coding activity includes drawing diagrams (modeling) that will be transformed into code, scripting a web-based system, and choosing among several alternative solutions.
* **Testing:** XP model gives high importance to testing and considers it to be the primary factor to develop fault-free software.
* **Listening:** The developers need to carefully listen to the customers if they have to develop good quality software. Sometimes programmers may not have the depth knowledge of the system to be developed. So, the programmers should understand properly the functionality of the system and they have to listen to the customers.
* **Designing:** Without a proper design, a system implementation becomes too complex and very difficult to understand the solution, thus making maintenance expensive. A good design results elimination of complex dependencies within a system. So, effective use of suitable design is emphasized.
* **Feedback:** One of the most important aspects of the XP model is to gain feedback to understand the exact customer needs. Frequent contact with the customer makes the development effective.
* **Simplicity:** The main principle of the XP model is to develop a simple system that will work efficiently in the present time, rather than trying to build something that would take time and may never be used. It focuses on some specific features that are immediately needed, rather than engaging time and effort on speculations of future requirements.
* **Pair Programming:**XP encourages pair programming where two developers work together at the same workstation. This approach helps in knowledge sharing, reduces errors and improves code quality.
* **Continuous Integration:**In XP, developers integrate their code into a shared repository several times a day. This helps to detect and resolve integration issues early on in the development process.
* **Refactoring:**XP encourages refactoring, which is the process of restructuring existing code to make it more efficient and maintainable. Refactoring helps to keep the codebase clean, organized and easy to understand.
* **Collective Code Ownership:** In XP, there is no individual ownership of code. Instead, the entire team is responsible for the codebase. This approach ensures that all team members have a sense of ownership and responsibility towards the code.
* **Planning Game:** XP follows a planning game, where the customer and the development team collaborate to prioritize and plan development tasks. This approach helps to ensure that the team is working on the most important features and delivers value to the customer.
* **On-site Customer:** XP requires an on-site customer who works closely with the development team throughout the project. This approach helps to ensure that the customer’s needs are understood and met, and also facilitates communication and feedback.

**Applications of Extreme Programming (XP):** Some of the projects that are suitable to develop using the XP model are given below:

* **Small projects:** XP model is very useful in small projects consisting of small teams as face-to-face meeting is easier to achieve.
* **Projects involving new technology or Research projects:** This type of project face changing requirements rapidly and technical problems. So XP model is used to complete this type of project.
* **Web development projects:** XP model is well-suited for web development projects as the development process is iterative and requires frequent testing to ensure the system meets the requirements.
* **Collaborative projects:** XP model is useful for collaborative projects that require close collaboration between the development team and the customer.
* **Projects with tight deadlines:**XP model can be used in projects that have a tight deadline, as it emphasizes simplicity and iterative development.
* **Projects with rapidly changing requirements:**XP model is designed to handle rapidly changing requirements, making it suitable for projects where requirements may change frequently.
* **Projects where quality is a high priority:**XP model places a strong emphasis on testing and quality assurance, making it a suitable approach for projects where quality is a high priority.

Extreme Programming (XP) is an Agile software development methodology that focuses on delivering high-quality software through frequent and continuous feedback, collaboration, and adaptation. XP emphasizes a close working relationship between the development team, the customer, and stakeholders, with an emphasis on rapid, iterative development and deployment.

Agile development approaches evolved in the 1990s as a reaction to documentation and bureaucracy based processes, particularly the waterfall approach. Agile approaches are based on some common principles, some of which are:

1. Working software is the key measure of progress in a project.
2. For progress in a project, therefore software should be developed and delivered rapidly in small increments.
3. Even late changes in the requirements should be entertained.
4. Face-to-face communication is preferred over documentation.
5. Continuous feedback and involvement of customer is necessary for developing good-quality software.
6. Simple design which envolves and improves with time is a better approach than doing an elaborate design up front for handling all possible scenarios.
7. The delivery dates are decided by empowered teams of talented individuals.

Extreme programming is one of the most popular and well-known approaches in the family of agile methods. an XP project starts with user stories which are short description of what scenarios the customers and users would like the system to support. Each story is written on a separate card, so they can be flexibly grouped.

XP, and other agile methods, are suitable for situations where the volume and space of requirements change is high and where requirement reisks are considerable.

XP includes the following practices:

1. Continuous Integration: Code is integrated and tested frequently, with all changes reviewed by the development team.
2. Test-Driven Development: Tests are written before code is written, and the code is developed to pass those tests.
3. Pair Programming: Developers work together in pairs to write code and review each other’s work.
4. Continuous Feedback: Feedback is obtained from customers and stakeholders through frequent demonstrations of working software.
5. Simplicity: XP prioritizes simplicity in design and implementation, with the goal of reducing complexity and improving maintainability.
6. Collective Ownership: All team members are responsible for the code, and anyone can make changes to any part of the codebase.
7. Coding Standards: Coding standards are established and followed to ensure consistency and maintainability of the code.
8. Sustainable Pace: The pace of work is maintained at a sustainable level, with regular breaks and opportunities for rest and rejuvenation.
9. XP is well-suited to projects with rapidly changing requirements, as it emphasizes flexibility and adaptability. It is also well-suited to projects with tight timelines, as it emphasizes rapid development and deployment.
10. Refactoring: Code is regularly refactored to improve its design and maintainability, without changing its functionality.
11. Small Releases: Software is released in small increments, allowing for frequent feedback and adjustments based on that feedback.
12. Customer Involvement: Customers are actively involved in the development process, providing feedback and clarifying requirements.
13. On-Site Customer: A representative from the customer’s organization is present with the development team to provide continuous feedback and answer questions.
14. Short Iterations: Work is broken down into short iterations, usually one to two weeks in length, to allow for rapid development and frequent feedback.
15. Planning Game: The team and customer work together to plan and prioritize the work for each iteration, with the goal of delivering the most valuable features first.
16. Metaphor: A shared metaphor is used to guide the design and implementation of the system.
17. Coding Standards: Coding standards are established and followed to ensure consistency and maintainability of the code.

**Advantages of Extreme Programming (XP):**

* **Slipped schedules −**Timely delivery is ensured through slipping timetables and doable development cycles.
* **Misunderstanding the business and/or domain −** Constant contact and explanations are ensured by including the client on the team.
* **Canceled projects −** Focusing on ongoing customer engagement guarantees open communication with the consumer and prompt problem-solving.
* **Staff turnover −**Teamwork that is focused on cooperation provides excitement and goodwill. Team spirit is fostered by multidisciplinary cohesion.
* **Costs incurred in changes −** Extensive and continuing testing ensures that the modifications do not impair the functioning of the system. A functioning system always guarantees that there is enough time to accommodate changes without impairing ongoing operations.
* **Business changes −** Changes are accepted at any moment since they are seen to be inevitable.
* **Production and post-delivery defects: Emphasis is on −** the unit tests to find and repair bugs as soon as possible.

# Scrum,

# Scrum Life cycle

# and Agile Scrum Framework,

**Self Learning Topics:-**

Kanban,

Kaizen **05**

**03 Module:Agile Process Models**

# Adaptive Software Development (ASD),

Adaptive Software Development has evolved from RAD practices. The team aspects also were added to these practices. Companies from New Zealand to Canada, for a wide range of project and product types, have used adaptive Software Development.

Jim Highsmith published Adaptive Software Development in 2000.

Adaptive Software Development practices provide ability to accommodate change and are adaptable in turbulent environments with products evolving with little planning and learning.

Phases of ASD Life Cycle

Adaptive Software Development is cyclical like the Evolutionary model, with the phase names reflecting the unpredictability in the complex systems. The phases in the Adaptive development life cycle are −

* Speculate
* Collaborate
* Learn

These three phases reflect the dynamic nature of Adaptive Software Development. The Adaptive Development explicitly replaces Determinism with Emergence. It goes beyond a mere change in lifecycle to a deeper change in management style. Adaptive Software Development has a dynamic Speculate-Collaborate-Learn Lifecycle.

The Adaptive Software Development Lifecycle focuses on results, not tasks, and the results are identified as application features.



Speculate

The term plan is too deterministic and indicates a reasonably high degree of certainty about the desired result. The implicit and explicit goal of conformance to plan, restricts the manager's ability to steer the project in innovative directions.

In Adaptive Software Development, the term plan is replaced by the term speculate. While speculating, the team does not abandon planning, but it acknowledges the reality of uncertainty in complex problems. Speculate encourages exploration and experimentation. Iterations with short cycles are encouraged.

Collaborate

Complex applications are not built, they evolve. Complex applications require that a large volume of information be collected, analyzed, and applied to the problem. Turbulent environments have high rates of information flow. Hence, complex applications require that a large volume of information be collected, analyzed, and applied to the problem. This results in diverse Knowledge requirements that can only be handled by team collaboration.

Collaborate would require the ability to work jointly to produce results, share knowledge or make decisions.

In the context of project management, Collaboration portrays a balance between managing with traditional management techniques and creating and maintaining the collaborative environment needed for emergence.

Learn

The Learn part of the Lifecycle is vital for the success of the project. Team has to enhance their knowledge constantly, using practices such as −

* Technical Reviews
* Project Retrospectives
* Customer Focus Groups

Reviews should be done after each iteration. Both, the developers and customers examine their assumptions and use the results of each development cycle to learn the direction of the next. The team learns −

* About product changes
* More fundamental changes in underlying assumptions about how the products are being developed

The iterations need to be short, so that the team can learn from small rather than large mistakes.

Speculate - Collaborate - Learn Cycle as a Whole

As you observe from the Speculate-Collaborate-Learn cycle, given above, it is obvious that the three phases are nonlinear and overlap.

We observe the following from an Adaptive framework.

* It is difficult to Collaborate without Learning or to Learn without Collaborating.
* It is difficult to Speculate without Learning or to Learn without Speculating.
* It is difficult to Speculate without Collaborating or to Collaborate without Speculating.

# Scrum, ,

What is Scrum?

Scrum is a management framework that teams use to self-organize and work towards a common goal. It describes a set of meetings, tools, and roles for efficient project delivery. Much like a sports team practicing for a big match, Scrum practices allow teams to self-manage, learn from experience, and adapt to change. Software teams use Scrum to solve complex problems cost effectively and sustainably.

What is Scrum methodology?

Certain principles and values characterize Scrum methodology:

Scrum principles for project success

Transparency

Teams work in an environment where everyone is aware of the challenges that others might be experiencing. Regular face-to-face conversations between cross-functional team members and project owners prevent miscommunication and information bottlenecks.

Reflection

Frequent reflection points are built into the framework to allow team members to review their progress. Project managers use insights from these review meetings for estimation and future planning. As a result, projects can run more efficiently, within budget, and on schedule.

Adaptation

Team members can reprioritize tasks based on changing customer requirements. They decide which tasks to complete first and which to revisit in the future.

Scrum values for project teams

Scrum Teams follow five core values.

Commitment

Scrum Team members are committed to time-based tasks and goals and are dedicated to continuous improvement to find the best solution.

Courage

Scrum Teams show courage by asking open, challenging questions. They have honest and transparent discussions to arrive at the best solution.

Focus

During any given period, team members will work from a Product Backlog of tasks. They will focus on the selected tasks to provide deliverables within a limited time frame.

Openness

Scrum Team members are open to new ideas and opportunities that support individual learning and overall project quality.

Respect

Team members respect the project managers, each other, and the Scrum process. This culture of respect creates a spirit of mutual collaboration and cooperation within the team.

How does Scrum work?

Scrum is a framework that is easy to learn but difficult to become an expert in. The co-creators of scrum, Jeff Sutherland and Ken Schwaber, have explained the underlying concepts in The Scrum Guide. The guide gives a detailed overview of scrum processes and how to implement them effectively.  
  
The essence of Scrum is a self-organizing team delivering customer value in a time-boxed period called a Sprint. Scrum defines artifacts, roles, and events associated with each Sprint. Let’s look at each of these in detail.

What are Scrum artifacts?

Scrum Teams use tools called Scrum artifacts to solve problems and manage projects. Scrum artifacts provide critical planning and task information to team members and stakeholders. There are three primary artifacts:

Product Backlog

The Product Backlog is a dynamic list of features, requirements, enhancements, and fixes that must be completed for project success. It is essentially the team’s to-do list, which is constantly revisited and reprioritized to adapt to market changes. The product owner maintains and updates the list, removing irrelevant items or adding new requests from customers.

Sprint Backlog

The Sprint Backlog is the list of items to be completed by the development team in the current Sprint cycle. Before each Sprint, the team chooses which items it will work on from the Product Backlog. A Sprint Backlog is flexible and can evolve during a Sprint.

Increment

The Increment is a step towards a goal or vision. It is the usable end product from a Sprint. Teams can adopt different methods to define and demonstrate their Sprint Goals. Despite the flexibility, the fundamental Sprint Goal—what the team wants to achieve from the current Sprint—can’t be compromised.  
  
For example, some teams choose to release something to their customers at the end of the Sprint, so their Sprint Goal would be completed once the software change is released. Other teams might work on completing a set of features that will be released together. In this case, the Sprint Goal would be completed when a feature is tested successfully.

What are Scrum roles?

A Scrum Team needs three specific roles: a Product Owner, Scrum leader, and development team.

Product Owner

The Product Owner focuses on ensuring the development team delivers the most value to the business. They understand and prioritize the changing needs of end users and customers. Effective product owners do the following:

* Give the team clear guidance on which features to deliver next.
* Bridge the gap between what the business wants and what the team understands.
* Decide when and how frequently releases should happen.

Scrum leader

Scrum leaders are the champions for Scrum within their teams. They are accountable for the Scrum Team’s effectiveness. They coach teams, Product Owners, and the business to improve its Scrum processes and optimize delivery. Scrum leaders are also responsible for doing the following:

* Schedule the resources needed for each Sprint.
* Facilitate other Sprint events and team meetings.
* Lead digital transformation within the team.
* Facilitate any team training when adopting new technologies.
* Communicate with external groups to solve any challenges the team might be facing as a whole.

Scrum development team

The Scrum Team consists of testers, designers, UX specialists, Ops engineers, and developers. Team members have different skill sets and cross-train each other, so no one person becomes a bottleneck in delivering work.   
  
Jeff Bezos, the founder of Amazon, recommends the two-pizza rule when deciding team size:A team should be small enough to share two pizzas.

Scrum development teams do the following:

* Work collaboratively to ensure a successful Sprint completion.
* Champion sustainable development practices.
* Self-organize and approach their projects with an evident we attitude.
* Drive the planning and estimating for how much work they can complete for each Sprint.

What are Scrum events?

Scrum events or Scrum ceremonies are a set of sequential meetings that Scrum Teams perform regularly. Some Scrum events include the following:

Sprint Planning

In this event, the team estimates the work to be completed in the next Sprint. Members define Sprint Goals that are specific, measurable, and attainable. At the end of the planning meeting, every Scrum member knows how each Increment can be delivered in the Sprint.

Sprint

A Sprint is the actual time period when the Scrum Team works together to finish an Increment. Two weeks is the typical length for a Sprint but can vary depending on the needs of the project and the team. The more complex the work and the more unknowns, the shorter the Sprint should be.

Daily Scrum or stand-up

A Daily Scrum is a short meeting in which team members check in and plan for the day. They report on work completed and voice any challenges in meeting Sprint Goals. It is called a stand-up because it aims to keep the meeting as short as practical—like when everybody is standing.

Sprint Review

At the end of the Sprint, the team gets together for an informal session to review the work completed and showcase it to stakeholders. The Product Owner might also rework the Product Backlog based on the current Sprint.

Sprint Retrospective

The team comes together to document and discuss what worked and what didn’t work during the Sprint. Ideas generated are used to improve future Sprints.

Why is Scrum important in software development?

All kinds of teams, such as HR, marketing, and design, use Scrum effectively. However, Scrum is more prevalent in software development and engineering teams. It allows teams to respond faster to changing requirements without letting costs and budgets spin out of control. It is important for the following reasons:

Ability to maintain quality in challenging situations

Quality assurance checks are built into the Scrum framework. Teams define requirements at the start of each Sprint. Teams also comprehensively assess the software or product life cycle while establishing a team vision of done. This means requirements remain relevant and achievable within a short time frame. Regular Product Owner feedback and Sprint reviews allow continuous team improvement throughout the project.

Increased return on investment

Scrum Teams prioritize requirements based on customer value and risk analysis. The focus is on developing a primary working product that can be released to market to gather early customer feedback. Scrum development is characterized by fewer costly defects, team efficiency, and a fail-fast approach that saves money in the long run.

Happier and more productive teams

Self-managed and self-organized team structures allow members to be more creative and innovative. Members have the flexibility to organize their work according to their work styles, personalities, and personal life goals. Working cross-functionally allows members to learn new skills and mentor each other. As a result, Scrum creates an environment of support and trust, increasing people’s overall motivation and morale.

Relevant metrics that improve estimation

Scrum Teams choose their own metrics to measure project performance. They estimate timelines, budgets, and quality metrics based on their experience and capabilities. The Product Owner has control because estimates are relative. Teams get more support at the beginning of the project and naturally speed up over time. Project stakeholders review working products and provide regular feedback to ensure the project stays on track.

Scrum vs. agile—what is the difference?

Agile refers to a mindset or way of thinking in software development. It is a philosophy adopted at an organizational level to get every team member to focus on continuous improvement and value delivery to customers. Scrum is a framework for getting work done within agile. Scrum uses all the core principles of agile to define methods to facilitate a project.However, it is important to note that agile does not always mean Scrum. Many different methodologies take an agile approach to project management.

# Dynamic Systems Development Method (DSDM)

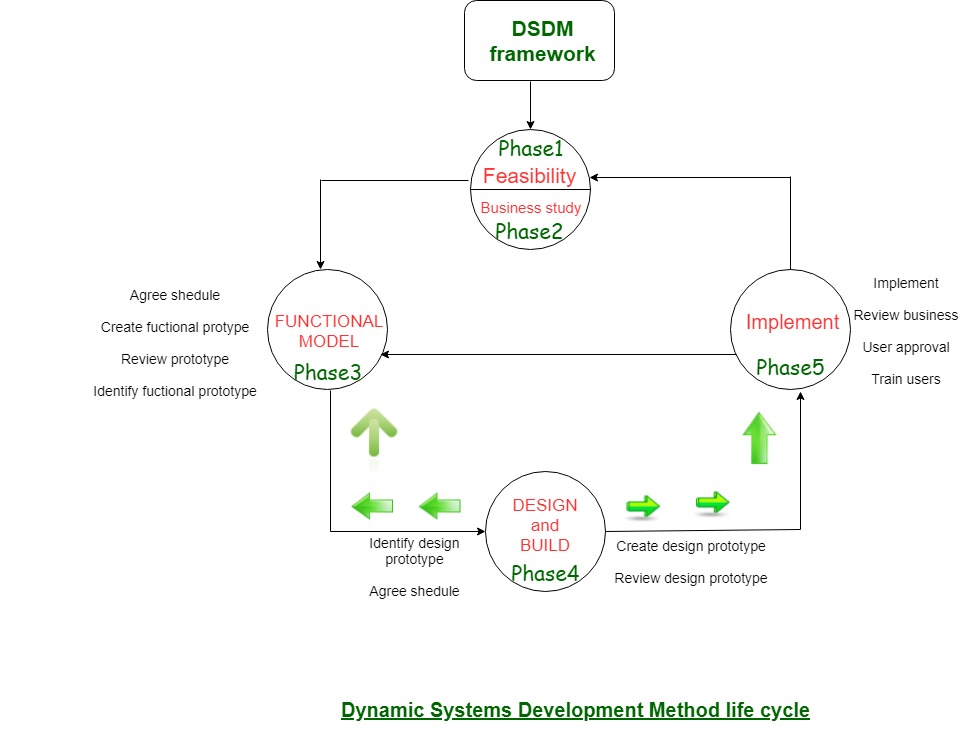
The **Dynamic Systems Development technique (DSDM)** is an associate degree agile code development approach that provides a framework for building and maintaining systems. The DSDM philosophy is borrowed from a modified version of the sociologist principle—80 % of An application is often delivered in twenty percent of the time it’d desire deliver the entire (100 percent) application.

DSDM is An iterative code method within which every iteration follows the 80% rule that simply enough work is needed for every increment to facilitate movement to the following increment. The remaining detail is often completed later once a lot of business necessities are noted or changes are requested and accommodated.

The DSDM tool (www.dsdm.org) could be a worldwide cluster of member companies that put together tackle the role of “keeper” of the strategy. The pool has outlined AN [Agile Development Model](https://www.geeksforgeeks.org/software-engineering-agile-development-models/), known as the DSDM life cycle that defines 3 different unvarying cycles, preceded by 2 further life cycle activities:

1. **Feasibility Study:**  
   It establishes the essential business necessities and constraints related to the applying to be designed then assesses whether or not the application could be a viable candidate for the DSDM method.
2. **Business Study:**  
   It establishes the use and knowledge necessities that may permit the applying to supply business value; additionally, it is the essential application design and identifies the maintainability necessities for the applying.
3. **Functional Model Iteration:**  
   It produces a collection of progressive prototypes that demonstrate practicality for the client.  
   (Note: All DSDM prototypes are supposed to evolve into the deliverable application.) The intent throughout this unvarying cycle is to collect further necessities by eliciting feedback from users as they exercise the paradigm.
4. **Design and Build Iteration:**  
   It revisits prototypes designed throughout useful model iteration to make sure that everyone has been designed during a manner that may alter it to supply operational business price for finish users. In some cases, useful model iteration and style and build iteration occur at the same time.
5. **Implementation:**  
   It places the newest code increment (an “operationalized” prototype) into the operational surroundings. It ought to be noted that:
   * **(a)** the increment might not 100% complete or,
   * **(b)** changes are also requested because the increment is placed into place. In either case, DSDM development work continues by returning to the useful model iteration activity.

Below diagram describe the DSDM life cycle:



DSDM is often combined with XP to supply a mixed approach that defines a solid method model (the DSDM life cycle) with the barmy and bolt practices (XP) that are needed to create code increments. additionally, the ASD ideas of collaboration and self-organizing groups are often tailored to a combined method model.

# Crystal,

**Crystal methods in Agile Development/Framework:**The crystal method is an agile framework that is considered a lightweight or agile methodology that focuses on individuals and their interactions. The methods are color-coded to significant risk to human life. It is mainly for short-term projects by a team of developers working out of a single workspace. Among a few Agile [Software Development Life Cycle (SDLC)](https://www.geeksforgeeks.org/software-development-life-cycle-sdlc/) models crystal is considered as one of the Agile SDLC models.  
Two core beliefs of the Crystal method:

* Find your own way and methods to optimize workflow.
* Make use of unique methods to make the project unique and dynamic.

**Let’s know about the history of the Crystal Method**: The crystal method was developed by an American scientist named Alistair Cockburn who worked at IBM. He decided not to focus on step-by-step developmental strategies, but to develop team collaboration and communication. Some of the traits of Cockburn’s Crystal method were:

* Human-powered i.e. the project should be flexible and people involved in preferred work.
* Adaptive i.e. approaches don’t have any fixed tools but can be changed anytime to meet the team’s specific needs.
* Ultra-light i.e. this methodology doesn’t require much documentation.

**Properties of Crystal Agile Framework:**

1. **Frequent Delivery-** It allows you regularly deliver the products and test code to real users. Without this, you might build a product that nobody needs.
2. **Reflective Improvement-**No matter how good you have done or how bad you have done. Since there are always areas where the product can be improved, so the teams can implement to improve their future practices.
3. **Osmotic Communication-**Alistair stated that having the teams in the same physical phase is very much important as it allows information to flow in between members of a team as in osmosis.
4. **Personal Safety-**There are no bad suggestions in a crystal team, team members should feel safe to discuss ideas openly without any fear.
5. **Focus-**Each member of the team knows exactly what to do, which enables them to focus their attention. This boosts team interaction and works towards the same goal.
6. **Easy access to expert users-**It enhances team communication with users and gets regular feedback from real users.
7. **Technical tooling-** It contains very specific technical tools which to be used by the software development team during testing, management, and configuration. These tools make it enable the team to identify any error within less time.
8. **Continuous learning –** The framework emphasizes on continuous learning, enabling team members to acquire new skills and knowledge, and apply them in their work.
9. **Teamwork –** The framework stresses on the importance of teamwork, promoting collaboration, and mutual support among team members.
10. **Timeboxing –**The framework adopts timeboxing to manage project deadlines, ensuring that the team delivers within set timelines.
11. **Incremental development –**The framework promotes incremental development, enabling the team to deliver working software frequently, and adapt to changes as they arise.
12. **Automated testing –**The framework emphasizes on automated testing, enabling the team to detect and fix bugs early, reducing the cost of fixing errors at later stages.
13. **Customer involvement –** The framework emphasizes on involving customers in the development process, promoting customer satisfaction, and delivering products that meet their needs.
14. **Leadership –** The framework encourages leadership, enabling team members to take ownership of their work and make decisions that contribute to the success of the project.

**How does Crystal function?**

Till now, we got to know that crystal is a family of various developmental approaches, and it is not a group of prescribed developmental tools and methods. In the beginning, the approach is set by considering the business requirements and the needs of the project. Various methodologies in the Crystal family also known as weights of the Crystal approach are represented by different colors of the spectrum.  
Crystal family consists of many variants like Crystal Clear, Crystal Yellow, Crystal Red, Crystal Sapphire, Crystal Red, Crystal Orange Web, and Crystal Diamond.

1. **Crystal Clear-**The team consists of only 1-6 members that is suitable for short-term projects where members work out in a single workspace.
2. **Crystal Yellow-** It has a small team size of 7-20 members, where feedback is taken from Real Users. This variant involves automated testing which resolves bugs faster and reduces the use of too much documentation.
3. **Crystal Orange-** It has a team size of 21-40 members, where the team is split according to their functional skills. Here the project generally lasts for 1-2 years and the release is required every 3 to 4 months.
4. **Crystal Orange Web-**It has also a team size of 21-40 members were the projects that have a continually evolving code base that is being used by the public. It is also similar to Crystal Orange but here they do not deal with a single project but a series of initiatives that required programming.
5. **Crystal Red-** The software development is led by 40-80 members where the teams can be formed and divided according to requirements.
6. **Crystal Maroon-** It involves large-sized projects where the team size is 80-200 members and where methods are different and as per the requirement of the software.
7. **Crystal Diamond & Sapphire-** This variant is used in large projects where there is a potential risk to human life.

The below figure illustrates about crystal team



*CRYSTAL FAMILY (TEAM MEMBERS)*

**Benefits of using the Crystal Agile Framework :**

* Facilitate and enhance team communication and accountability.
* The adaptive approach lets the team respond well to the demanding requirements.
* Allows team to work with the one they see as the most effective.
* Teams talk directly with each other, which reduces management overhead.
* **Faster delivery –** The framework enables the team to deliver working software faster, which can help gain a competitive advantage in the market.
* **Higher quality –**The framework emphasizes on quality, enabling the team to detect and fix defects early in the development process, resulting in a higher quality product.
* **Improved customer satisfaction –** The framework promotes customer involvement, enabling the team to deliver products that meet customer needs, resulting in higher customer satisfaction.
* **Increased productivity –**The framework enables the team to focus on delivering the highest value features, which can increase productivity and reduce waste.
* **Flexibility –**The framework is highly adaptable, enabling the team to adjust to changing requirements, and make decisions based on real-time feedback.
* **Empowerment –** The framework promotes empowerment, enabling team members to take ownership of their work, and make decisions that contribute to the success of the project.
* **Reduced risk –** The framework promotes risk management, enabling the team to identify and mitigate potential risks early in the development process, reducing the likelihood of project failure.

**Drawbacks of using the Crystal Agile Framework :**

* A lack of pre-defined plans may lead to confusion and loss of focus.
* Lack of structure may slow down inexperienced teams.
* Not clear on how a remote team can share knowledge informally.
* **Lack of predictability –** The framework’s emphasis on adaptability and flexibility may result in a lack of predictability, making it difficult to plan and estimate project timelines and budgets.
* **Lack of documentation –** The framework’s emphasis on communication and collaboration may result in a lack of documentation, making it difficult to track progress and maintain a record of decisions.
* **Limited scalability –** The framework may not be suitable for large or complex projects, as the lack of structure and predefined plans may make it difficult to manage teams at scale.
* **Dependence on team expertise –**The framework relies heavily on the expertise and skills of the development team, which may not be suitable for teams with limited experience or knowledge.
* **Lack of clarity on roles and responsibilities –**The framework’s emphasis on self-organizing teams may result in a lack of clarity on roles and responsibilities, leading to confusion and a loss of focus.
* **Inability to handle regulatory requirements –** The framework may not be suitable for projects with strict regulatory requirements, as the lack of documentation and structure may not meet compliance standards.
* **Potential for informal knowledge sharing –** The framework’s emphasis on osmotic communication may result in informal knowledge sharing, which may be difficult to track and monitor for accuracy and completeness.

The Crystal Method is expandable. It may be used by small teams or large teams to work on simple or complex objects. It places importance on developmental skills and interactions which in turn encourage the exchange of ideas. It is also beneficial for the clients as it delivers the most important components of the product first. But on the other hand, the Crystal Method does not plan based on the requirements of the projects.

# Feature Driven Development (FDD)

An Agile methodology for developing software, Feature-Driven Development (FDD) is customer-centric, iterative, and incremental, with the goal of delivering tangible software results often and efficiently. FDD in Agile encourages status reporting at all levels, which helps to track progress and results.

FDD allows teams to update the project regularly and identify errors quickly. Plus, clients can be provided with information and substantial results at any time. FDD is a favorite method among development teams because it helps reduce two known morale-killers in the development world: Confusion and rework./p>

First applied in 1997 during [a project for a Singapore bank](https://ipfs.io/ipfs/QmXoypizjW3WknFiJnKLwHCnL72vedxjQkDDP1mXWo6uco/wiki/Feature-driven_development.html), FDD was developed and refined by Jeff De Luca, Peter Coad and others. The original project took 15 months with 50 people, and it worked; it was followed by a second, 18-month long, 250-person project./p>

Since then, it’s become a pragmatic approach ideal for long-term, complex projects looking for a simple but comprehensive methodology. While Scrum and new variations of Agile are [more widely recognized methods](https://dzone.com/articles/fdd-processes-amp-comparison-to-other-agile-method) (especially outside of software development), FDD can be a good option for software development teams looking for a structured, focused Agile methodology that can be scaled across the product organization and will deliver clear outcomes.

How is FDD Different from Scrum?

FDD is related to Scrum, but as its name implies, it’s a feature-focused method (as opposed to a delivery-focused method). Features are a foundational piece of FDD; they’re to FDD what user stories are to Scrum: Small functions that are, most importantly, client-valued.

*“During FDD, a feature should be delivered every 2-10 days – which differs from Scrum, in which sprints typically last two, but sometimes four, weeks.”*

FDD values documentation more than other methods (Scrum and XP included), which also creates differences in the roles of meetings. In Scrum, teams [typically meet on a daily basis](https://www.planview.com/resources/guide/what-is-scrum/); in FDD, teams rely on documentation to communicate important information, and thus don’t usually meet as frequently.

Another key difference is the end user; in FDD, the actual user is viewed as the end user, whereas in Scrum it’s typically the Product Owner who is seen as the end user.

How Does FDD Work?

Typically used in large-scale development projects, five basic activities exist during FDD:

* Develop overall model
* Build feature list
* Plan by feature
* Design by feature
* Build by feature

An overall model shape is formed during the first two steps, while the final three are repeated for each feature. The majority (roughly 75%) of effort during FDD will be spent on the fourth and fifth steps – Design by Feature and Build by Feature.

*Teams using all Agile methodologies operate with the primary goal of quickly and effectively satisfying the needs of their customers; FDD is no exception.*

However, the difference is that once a goal has been identified, teams following FDD organize their activities by features, rather than by project milestones or other indicators of progress.

How is a Feature Defined?

In FDD, each feature is useful and important to the client and results in something tangible to showcase. And because businesses appreciate quick results, the methodology depends on its two-week cycle.

Stages of Feature-Driven Development

Stage 0: Gather Data

As with all Agile methodologies, the first step in FDD is to gain an accurate understanding of content and context of the project, and to develop a clear, shared understanding of the target audience and their needs. During this time, teams should aim to learn everything they can about the why, the what, and the for whom about the project they’re about to begin (the next few steps will help clarify the how). This data-gathering can be thought of as stage 0, but one that cannot be skipped. To compare product development with writing a research paper, this is the research and thesis development step.

Once teams have a clear understanding of their goals, the targeted audience and their current (and potentially, future) needs, the first named stage in FDD can begin: Developing an Overall Model.

Develop an overall model

Continuing the research paper metaphor, this stage is when the outline is drafted. Using the “thesis” (aka primary goal) as a guide, the team will develop detailed domain models, which will then be merged into one overall model that acts as a rough outline of the system. As it develops and as the team learns, details will be added.

Build a features list

Use the information assembled in the first step to create a list of the required features. Remember, a feature is a client-valued output. Make a list of features (that can be completed in two weeks’ time), and keep in mind that these features should be purposes or smaller goals, rather than tasks.

Plan by Feature

Enter: Tasks. Analyze the complexity of each feature and plan tasks that are related for team members to accomplish. During the planning stage, all members of the team should take part in the evaluation of features with the perspective of each development stage in mind. Then, use the assessment of complexity to determine the order in which each feature will implemented, as well as the team members that will be assigned to each feature set.

This stage should also identify class owners, individual developers who are assigned to classes. Because every class of the developing feature belongs to a specific developer, someone is responsible for the conceptual principles of that class, and should changes be required to multiple classes, then collaboration is necessary between the owners of each to implement them.

And while class owners are important to FDD, so are feature teams. In feature teams, specific roles are defined, and a variety of viewpoints are encouraged. This ensures that design decisions consider multiple thoughts and perspectives.

*“Kanban provides a shared, single source of truth that enables teams to not only manage support tickets more effectively, but also add infinitely more value to the product organization.”*

Design by Feature

A chief programmer will determine the feature that will be designed and build. He or she will also determine the class owners and feature teams involved, while defining the feature priorities. Part of the group might be working on technical design, while others work on framework. By the end of the design stage, a design review is completed by the whole team before moving forward.

Build by Feature

This step implements all the necessary items that will support the design. Here, user interfaces are built, as are components detailed in the technical design, and a feature prototype is created. The unit is tested, inspected and approved, then the completed feature can be promoted to the main build. Any feature that requires longer time than two weeks to design and build is further broken into features until it meets the two-week rule.

Conclusion

Feature-Driven Development is a practical Agile approach suited for long-term, complex projects. It is a suitable choice for development teams seeking a simple but structured Agile method that is scalable and delivers predictable results.

# Lean Software Development (LSD)

**Lean Software Development (LSD)** is an [agile framework](https://www.geeksforgeeks.org/software-engineering-agile-software-development/) that is used to streamline & optimize the software development process. It may also be referred to as Minimum Viable Product (MVP) strategy as these ways of thinking are very much alike since both intend to speed up development by focusing on new deliverables.

Toyota has been credited to inspire the lean development approach which is meant for optimizing production and minimize waste. Seeing Toyota’s lean approach many other manufacturing teams started to follow the same strategy. And it was first adopted in software development in 2003.

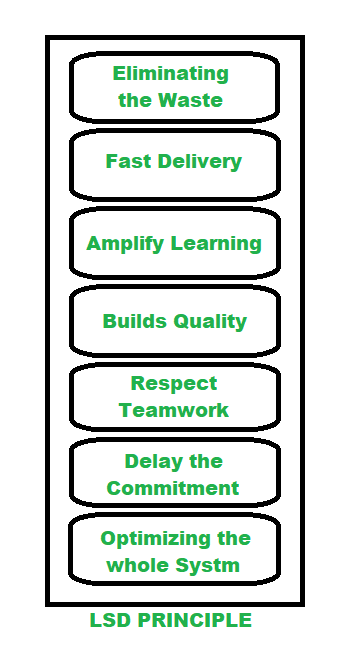
**Advantages of LSD :**  
LSD has proved to improve software development in the following ways :

1. LSD removes the unnecessary process stages when designing software so that it acts as a time saver as simplifies the development process.
2. With a focus on MVP, Lean Software Development prioritizes essential functions so this removes the risk of spending time on valueless builds.
3. It increases the involvement power of your team as more and more members participate due to which the overall workflow becomes optimized and losses get reduced.
4. LSD simplifies the development process and saves time by removing unnecessary stages.
5. It prioritizes essential functions and removes the risk of spending time on valueless builds.
6. It increases the involvement of team members, optimizing the workflow and reducing losses.
7. LSD encourages collaboration and communication among team members and stakeholders.
8. It fosters innovation through experimentation and creativity.
9. It reduces waste by minimizing unnecessary features.
10. LSD increases customer satisfaction through customer involvement and feedback.
11. It improves flexibility by allowing for adaptability to changing requirements and circumstances.

**Key Principles of Lean Software Development :**  
There are 7 established lean principles that come with a set of tactics, practices, and processes that builds more efficient software products :

1. Eliminating the waste
2. Fast Delivery
3. Amplify Learning
4. Builds Quality
5. Respect Teamwork
6. Delay the commitment
7. Optimizing the whole system

The below figure illustrates the principles of LSD :



*LEAN SOFTWARE DEVELOPMENT (LSD) PRINCIPLE*

**Eliminating the Waste:** To identify and eliminate wastes e.g. unnecessary code, delay in processes, inefficient communication, the issue with quality, data duplication, more tasks in the log than completed, etc. regular meetings are held by Project Managers. Which allows team members to point out faults and suggest changes in the next turn.

**Fast Delivery:**Previously long time planning used to be the key success in business, but in the passage of time it is found that engineers spend too much time on building complex systems with unwanted features. So they came up with an MVP strategy which resulted in the building products quickly that included a little functionality and launch the product to market and see the reaction. Such an approach allows them to enhance the product on the basis of customer feedback.

**Amplify Learning:**Learning is improved through ample code reviewing, meeting that is cross-team applicable. It is also ensured that particular knowledge isn’t accumulated by one engineer who’s writing a particular piece of code so paired programming is used.

**Builds Quality:** LSD is all about prevent waste, keeping an eye on not sacrificing quality. Developers often apply test-driven programming to examine the code before it is written. The quality can also be gained to get constant feedback from team members and project managers.

**Respect Teamwork**: LSD focuses on empowering team members, rather than controlling them. Setting up a collaborative atmosphere, keep perfect balance when there are short deadlines and immense workload. This method becomes very much important when new members join a well-established team.

**Delay the Commitment:** In traditional project management it often happens when you make your application and it turns out to be completely unfit for the market. LSD method recognizes this threat and makes room for improvement by postponing irreversible decisions until all experiment is done. This methodology always constructs software as flexible, so the new knowledge is available and engineers can make improvements.

**Optimizing the whole system:** lean’s principle allows managers to break an issue into small constituent parts to optimize the team’s workflow, create unity among members, and inspire a sense of shared responsibility which results in enhancing the team performance.

**Key feature of LSD:**

* Focus on continuous improvement and waste elimination
* Iterative and collaborative approach
* Development of Minimum Viable Product (MVP)
* Customer involvement and feedback
* Flexible and adaptive approach
* Prioritization of essential functions and minimization of waste

**Weakness in LSD :**

* Make it scalable as other frameworks since it strongly depends on the team involved.
* It is hard to keep pace so it is not easy for developers to work with team members as conflict may occur between them.
* It leads to a difficult decision-making process as it is mandatory for customers to clearly set their requirements for the development not to be interrupted.
* Lack of documentation
* Inflexibility for structured projects or strict deadlines
* Lack of predictability
* Dependence on customer involvement

Lean Software Development is one of the proactive approaches that drives your body through productivity and cleanliness. It closely connects to Agile methodology, knowledge-sharing experience, fast product delivery. All processes and stages of development are accurately built to deliver the end product at minimum cost in a timely manner.

# Agile Modeling (AM)

# Agile Unified Process (AUP).

Agile Unified Process (AUP) is a software development methodology that combines the principles and practices of agile software development with the principles and practices of the [Rational Unified Process (RUP)](https://www.agilelonestar.com/knowledge-base/rational-unified-process). AUP is based on the following principles:

**Emphasize collaboration and communication**: AUP emphasizes collaboration and communication among the team members and the stakeholders, in order to ensure that the software development process is transparent and visible, and that the team members and the stakeholders are involved and engaged in the software development process. This emphasis may involve using tools and techniques, such as agile project management and agile modeling, to support and facilitate the collaboration and the communication among the team members and the stakeholders, and to ensure that the team members and the stakeholders are working together towards the common goals and objectives of the software development project.

**Emphasize flexibility and adaptability**: AUP emphasizes flexibility and adaptability in the software development process, in order to allow the team members and the stakeholders to respond and adapt to the changing requirements and the emerging technologies, and to ensure that the software development process is agile and responsive to the changing needs and the priorities of the team members and the stakeholders. This emphasis may involve using tools and techniques, such as agile project management and agile modeling, to support and facilitate the flexibility and the adaptability of the software development process, and to allow the team members and the stakeholders to adapt and respond to the changing requirements and the emerging technologies, and to ensure that the software development process is agile and responsive.

**Emphasize iterative and incremental development**: AUP emphasizes iterative and incremental development in the software development process, in order to allow the team members and the stakeholders to iterate and increment the software development process, and to ensure that the software development process is incremental and evolutionary, and that the software is developed and delivered in small and frequent iterations. This emphasis may involve using tools and techniques, such as agile project management and agile modeling, to support and facilitate the iterative and incremental development of the software, and to allow the team members and the stakeholders to iterate and increment the software development process, and to ensure that the software development process is incremental and evolutionary.

**Emphasize quality and testing**: AUP emphasizes quality and testing in the software development process, in order to ensure that the software is developed and delivered with high quality, and to reduce the risks and the costs associated with the software quality. This emphasis may involve using tools and techniques, such as agile project management and agile modeling, to support and facilitate the quality and the testing of the software, and to ensure that the software is developed and delivered with high quality, and to reduce the risks and the costs associated with the software quality.

**Emphasize customer satisfaction and value**: AUP emphasizes customer satisfaction and value in the software development process, in order to ensure that the software development process is aligned with the needs and the expectations of the customers, and to ensure that the software is developed and delivered with value, and that the customers are satisfied with the software. This emphasis may involve using tools and techniques, such as agile project management and agile modeling, to support and facilitate the customer satisfaction and the value of the software development process, and to ensure that the software development process is aligned with the needs and the expectations of the customers, and that the customers are satisfied with the software.

The core principles of AUP include:

* The use of an iterative and incremental approach to software development
* The use of a common set of artifacts and deliverables to guide the development process
* The emphasis on collaboration, communication, and customer-centricity in the development process
* The use of a flexible and customizable framework that can be adapted to the specific needs of the project
* The incorporation of key principles and practices from agile methodologies, such as collaboration, flexibility, and adaptability

By following these principles, AUP can help organizations and teams to develop and deliver software with high quality and value, and to reduce the risks and the costs associated with the software development, and to improve the satisfaction and the engagement of the customers.

When To Choose AUP

* When you want to use a lightweight and iterative approach to software development
* When you want to quickly deliver working software and respond to changes and feedback from stakeholders
* When you want to emphasize collaboration, communication, and customer-centricity in the development process
* When you want to use a flexible and adaptable framework that can be tailored to the specific needs of the project
* When you want to incorporate key principles and practices from agile methodologies in the development process

AUP pros and cons

Pros of AUP:

* AUP provides a lightweight and iterative approach to software development
* AUP allows teams to quickly deliver working software and respond to changes and feedback from stakeholders
* AUP emphasizes collaboration, communication, and customer-centricity in the development process
* AUP provides a flexible and adaptable framework that can be tailored to the specific needs of the project
* AUP incorporates key principles and practices from agile methodologies in the development process

Cons of AUP:

* AUP may require additional time and resources to implement
* AUP may not be suitable for projects with a fixed deadline or budget
* AUP may not be suitable for projects with a large number of complex features
* AUP may not be suitable for projects with well-defined requirements that do not require significant changes