SOFTWARE ENGINEERING

UNIT 1

What is software engineering?

- **Software engineering** is an engineering discipline whose aim is the production of quality software, software that is delivered on time, within budget and that satisfies its requirements.
- **Software engineers** should
  - adopt a systematic and organized approach to their work
  - use appropriate tools and techniques depending on
    - the problem to be solved,
    - the development constraints and
  - use the resources available

Software Paradigm

- Software paradigm refers to methods and steps, which are taken while designing the software.
- Programming paradigm is a subset of software design paradigm which is future for other a subset of software development paradigm.
- Software is considered to be a collection of executable programming code, associated libraries, and documentation.
- Software development paradigm is also known as software engineering, all the engineering concepts pertaining to developments software applied. It consists of the following parts as Requirement Gathering, Software design, Programming, etc.
- The software design paradigm is a part of software development. It includes design, maintenance, and programming.

Software Paradigm

❖ **Advantages of using a software paradigm:**

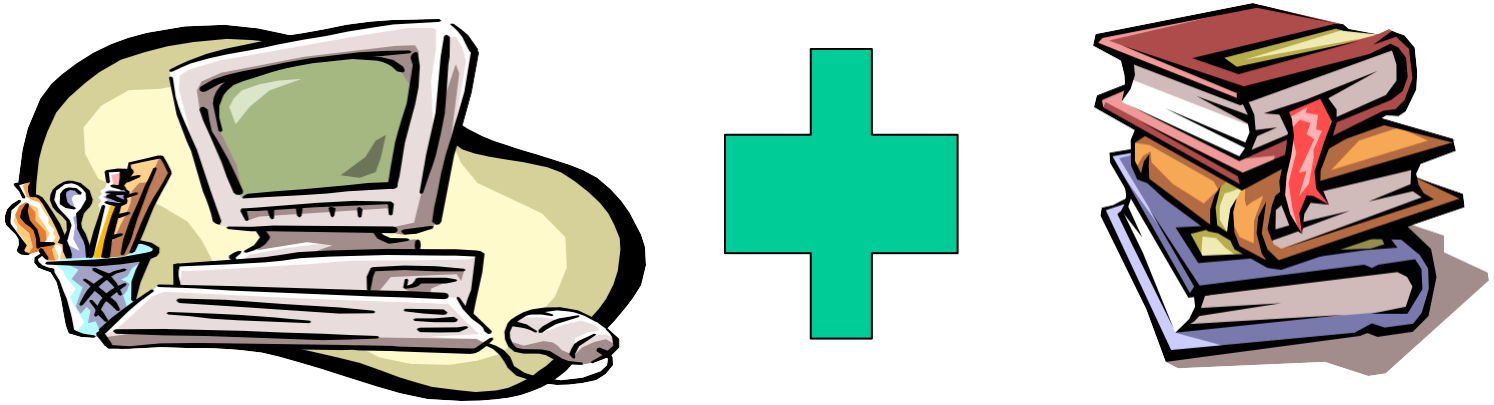
• Provide a consistent structure for developing software systems.
•Help developers understand the problem they are trying to solve.
•Help developers design and implement solutions more effectively.
•Help developers organize and reuse code more efficiently.
•Help developers create more reliable and maintainable software.

❖ **Disadvantages of using a software paradigm:**

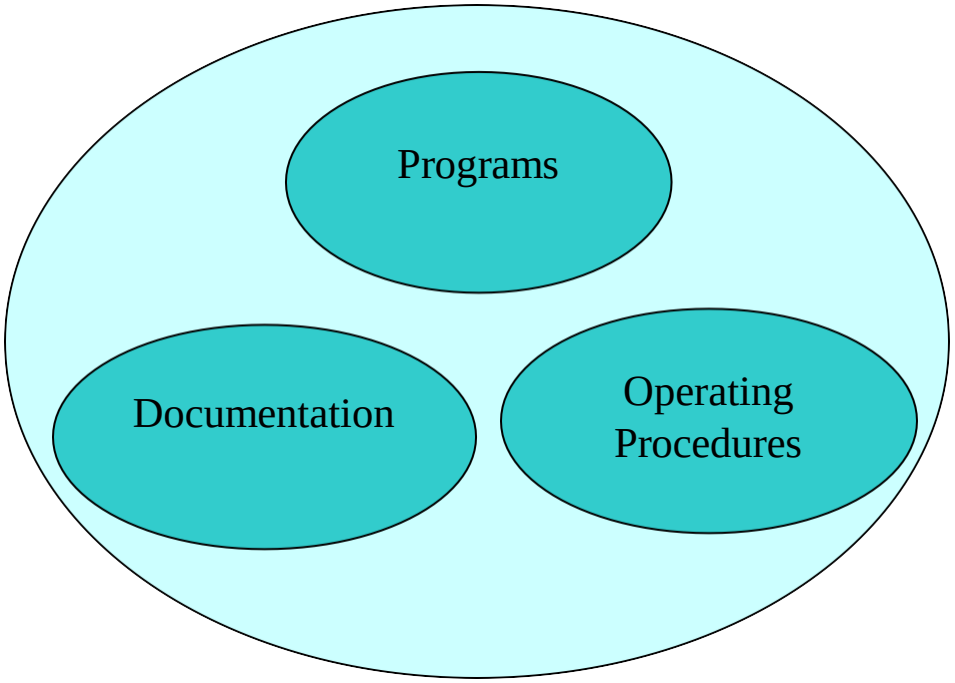
•Can be difficult to learn and understand for new developers.
•Can be limiting if a problem does not fit well into a specific paradigm.
•Can make it difficult to integrate systems developed using different paradigms.
•Can make it difficult to adapt to new technologies or changing requirements.

What is software?

- **Computer programs** and **associated documentation**



What is software?



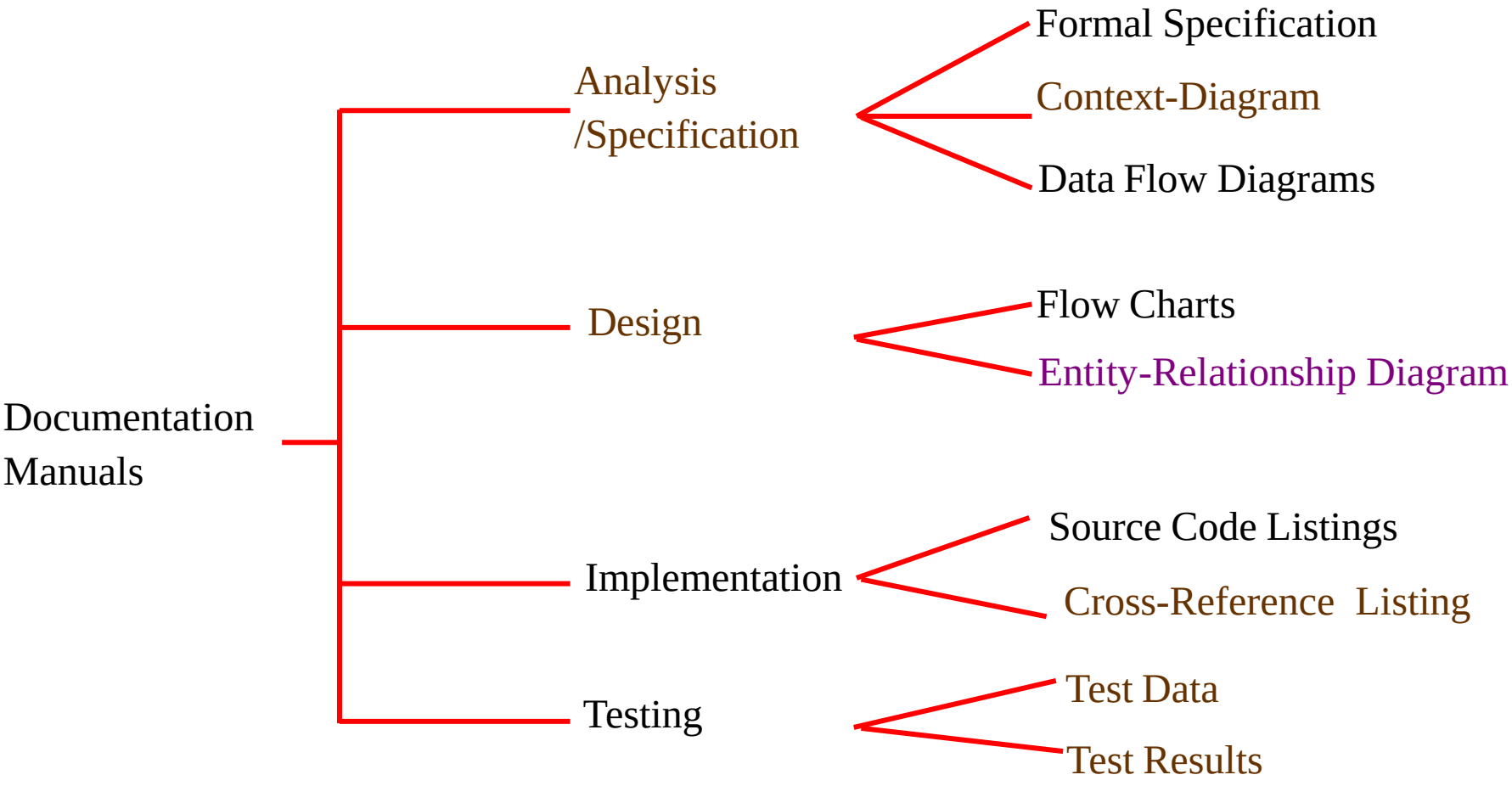
Software=Program+Documentation+Operating Procedures

Components of software

System Documentation

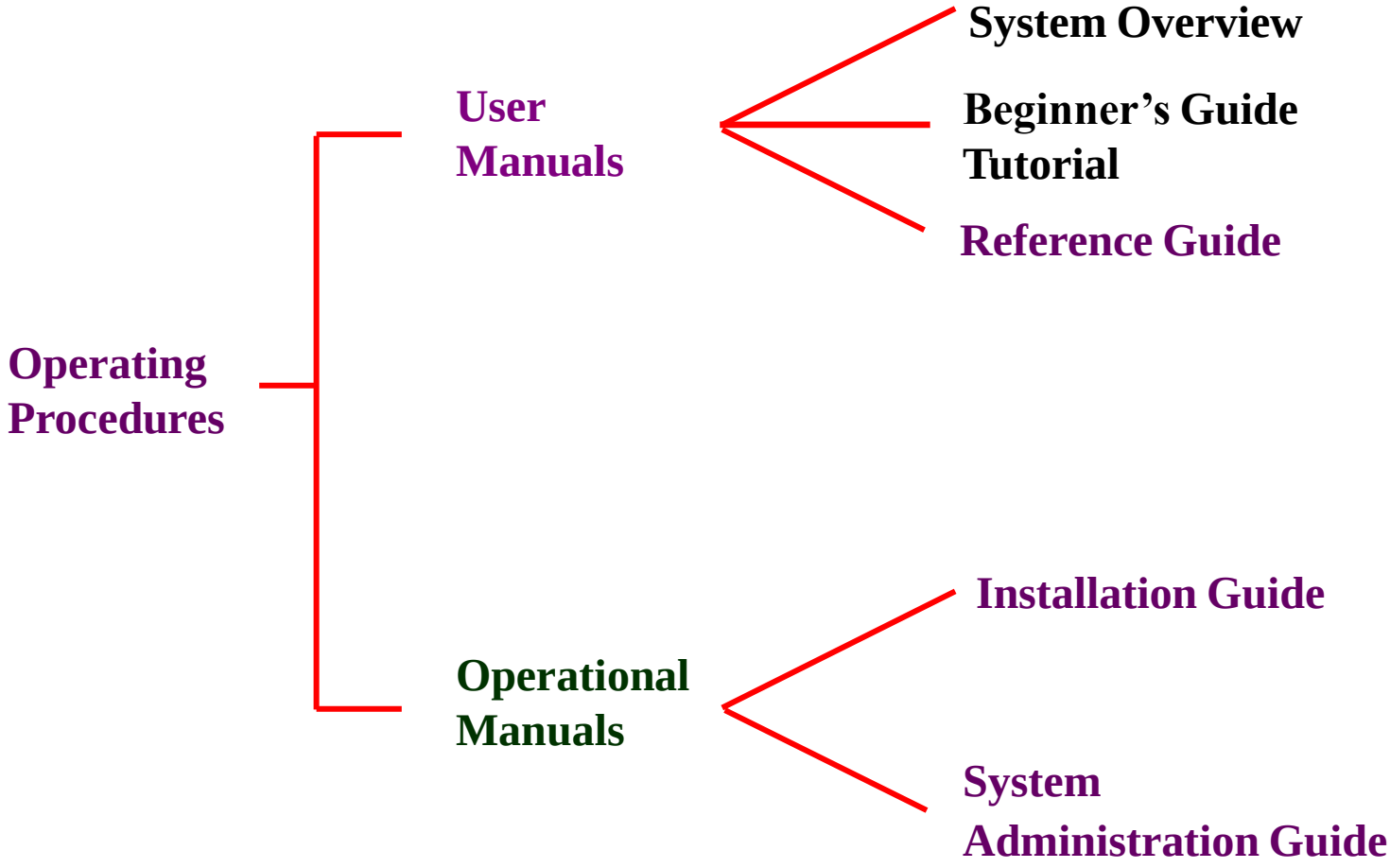
- SD consists all descriptions, programs, graphics, instructions pertaining to the design, coding, testing, and preparation of software.

Documentation consists of different types of manuals are



List of documentation manuals

Documentation consists of different types of manuals are



List of operating procedure manuals.

Why Software Engineering ?

- Change in nature & complexity of software
- We all want improvement

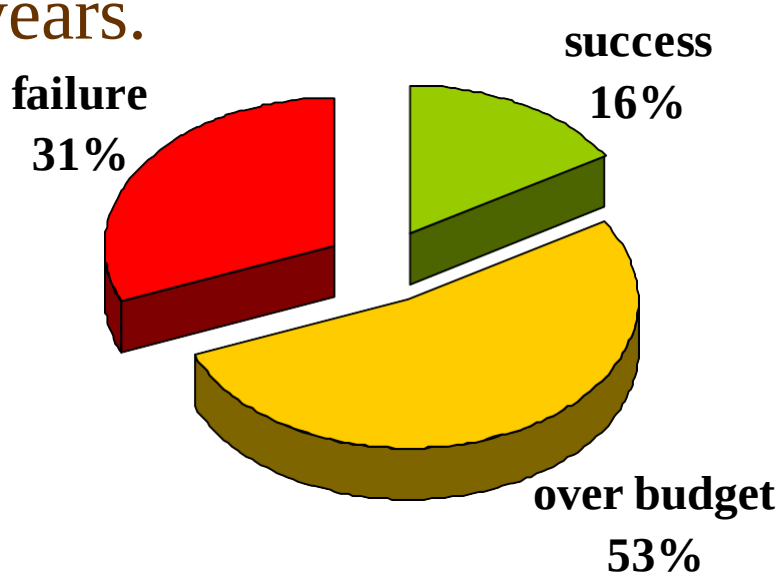
Ready for change

Need of Software Engineering ?

- Large software - It is easier to build a wall than to a house or building, likewise, as the size of software become large engineering has to step to give it a scientific process.
- Scalability- If the software process were not based on scientific and engineering concepts, it would be easier to re-create new software than to scale an existing one.
- Cost- As hardware industry has shown its skills and huge manufacturing has lower down the price of computer and electronic hardware. But the cost of software remains high if proper process is not adapted.
- Dynamic Nature- The always growing and adapting nature of software hugely depends upon the environment in which the user works. If the nature of software is always changing, new enhancements need to be done in the existing one. This is where software engineering plays a good role.
- Quality Management- Better process of software development provides better and quality software product.

The Evolving Role of Software

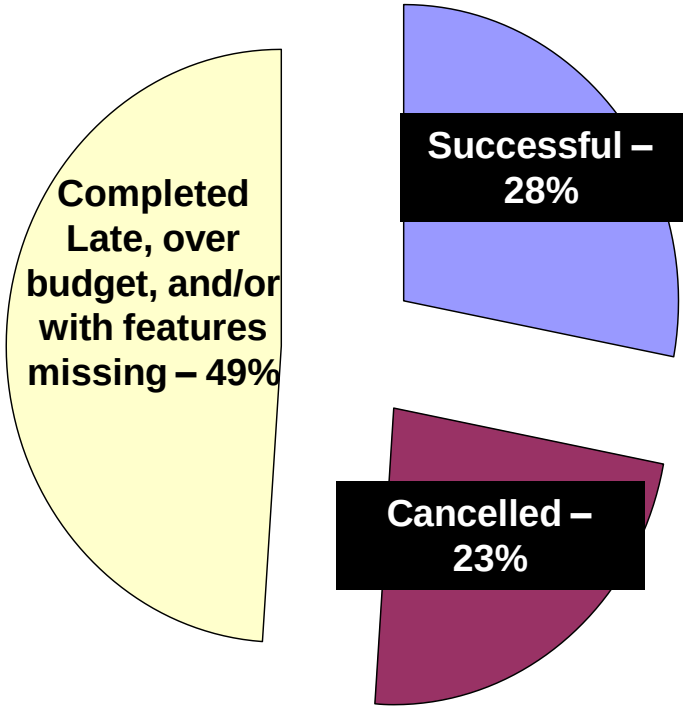
- Software industry is in Crisis!
- Software crisis is the expanse that organizations all around the world are incurring on software purchases compared to those on hardware purchases have been showing a worrying trend over the years.



Source: The Standish Group International, Inc. (CHAOS research)

The Evolving Role of Software

This is the SORRY state of Software Engineering Today!

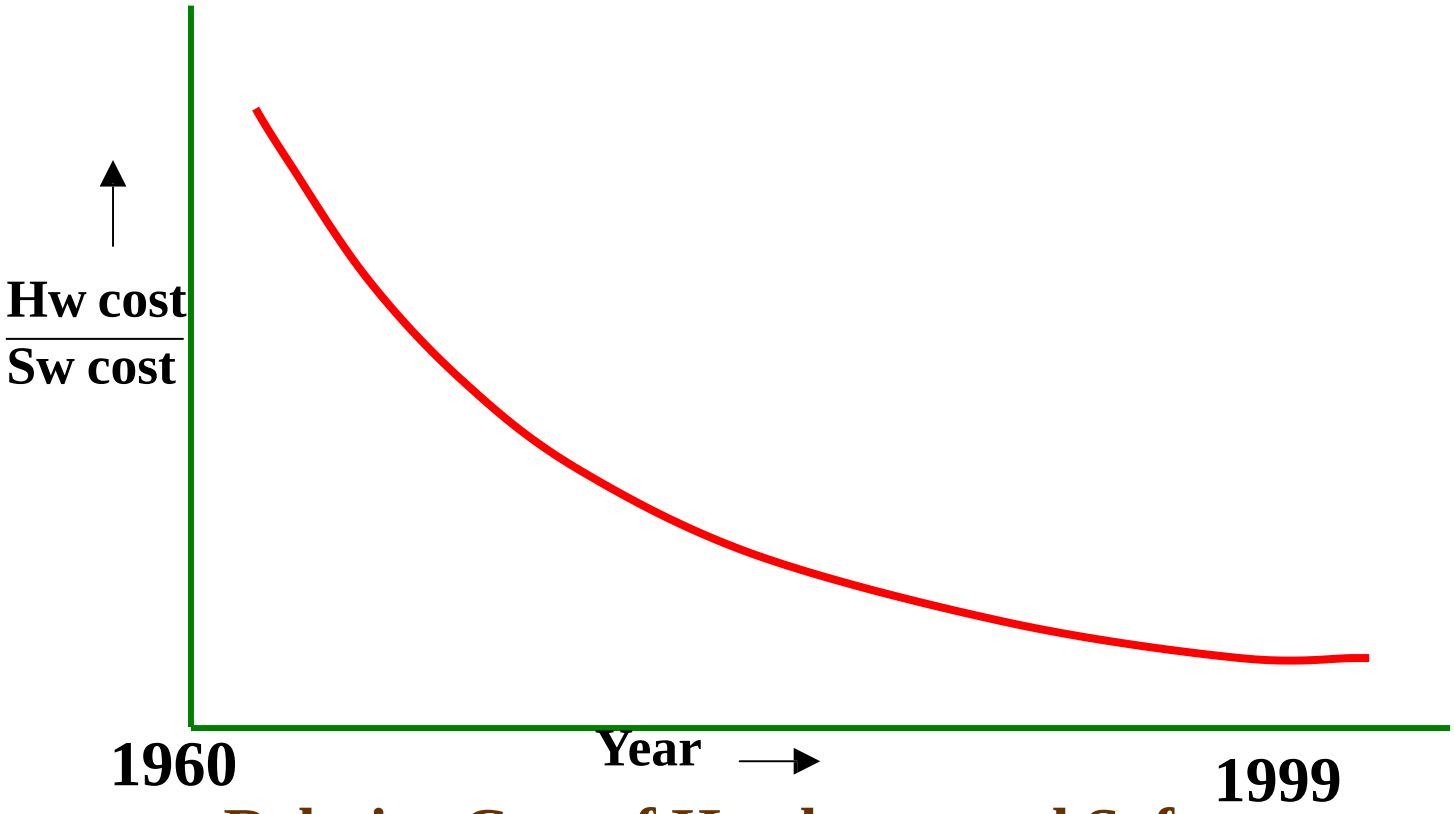


- **Data on 28,000 projects completed in 2000**

The Evolving Role of Software

As per the IBM report, “31%of the project get cancelled before they are completed, 53% over-run their cost estimates by an average of 189% and for every 100 projects, there are 94 restarts”.

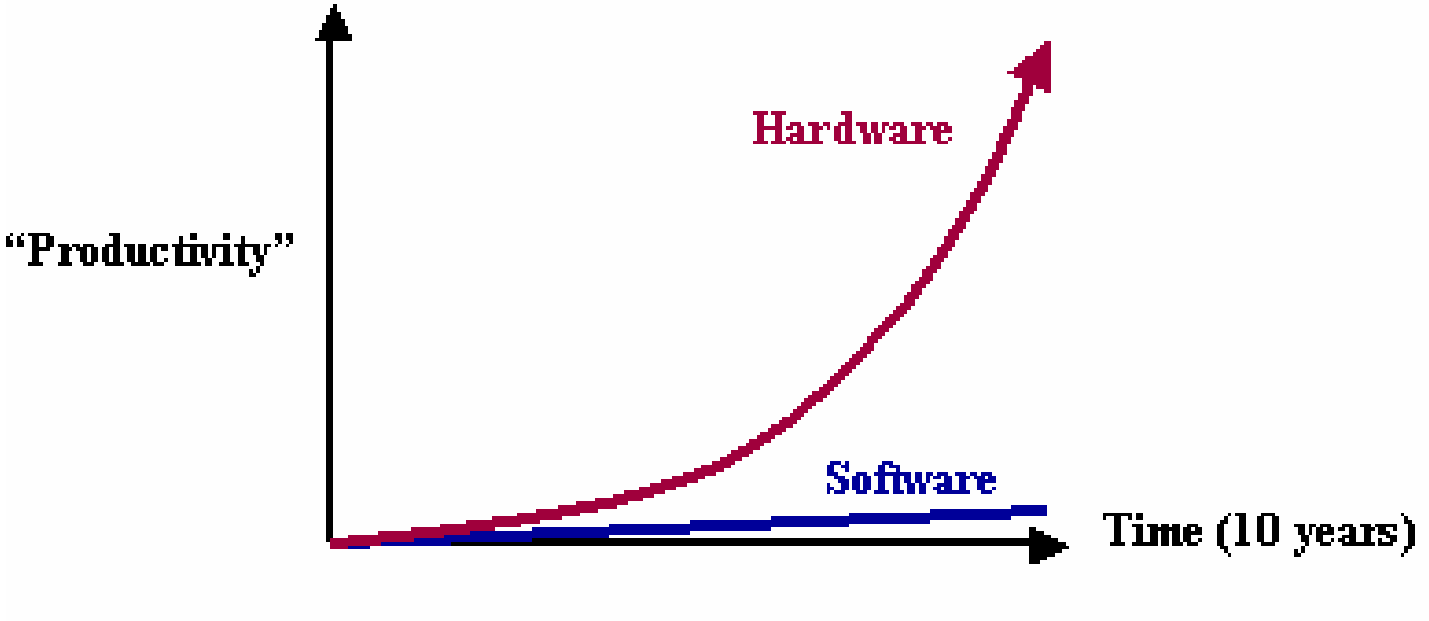
The Evolving Role of Software



Relative Cost of Hardware and Software

The Evolving Role of Software

- Unlike Hardware
  - Moore's law: processor speed/memory capacity doubles every two years



The Evolving Role of Software

***Managers* and *Technical Persons* are asked:**

- ✓ 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?

Factors Contributing to the Software Crisis

- Larger problems,
- Lack of adequate training in software engineering,
- Increasing skill shortage,
- Low productivity improvements.
- Lack of communication between software developers and users

Results of Software Crisis

- Poor qualities software was produced,
- Development team exceeded the budget,
- Late delivery of software,
- User requirements not completely supported by the software.
- Unreliable software,
- Software is inefficient

Some Software failures

Ariane 5

It took the European Space Agency **10 years and $7 billion** to produce Ariane 5, a giant rocket capable of hurling a pair of three-ton satellites into orbit with each launch and intended to give Europe overwhelming supremacy in the commercial space business.

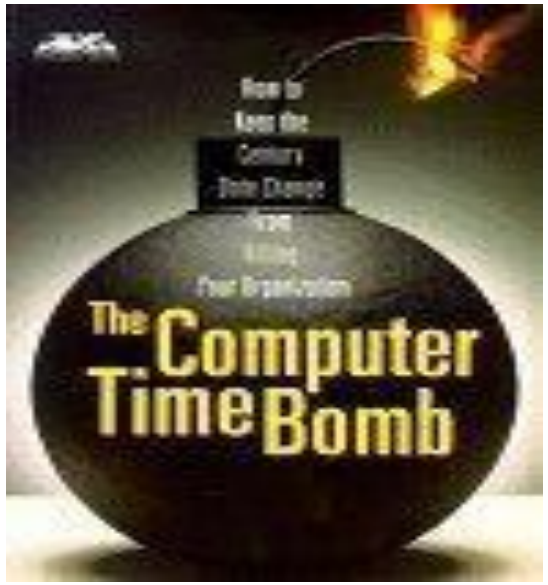
The rocket was destroyed after 39 seconds of its launch, at an altitude of two and a half miles along with its payload of four expensive and uninsured scientific satellites.

Some Software failures

Y2K problem:

It was simply the ignorance about the adequacy or otherwise of using only last two digits of the year.

The 4-digit date format, like 1964, was shortened to 2-digit format, like 64.



Some Software failures

The Patriot Missile

- First time used in Gulf war
- Used as a defense from Iraqi Scud missiles
- Failed several times including one that killed 28 US soldiers in Dhahran, Saudi Arabia

Reasons:

A small timing error in the system's clock accumulated to the point that after 14 hours, the tracking system was no longer accurate. In the Dhahran attack, the system had been operating for more than 100 hours.

Some Software failures

The Space Shuttle

Part of an abort scenario for the Shuttle requires fuel dumps to lighten the spacecraft. It was during the second of these dumps that a (software) crash occurred.

...the fuel management module, which had performed one dump and successfully exited, restarted when recalled for the second fuel dump...

Some Software failures

A simple fix took care of the problem…but the programmers decided to see if they could come up with a systematic way to eliminate these generic sorts of bugs in the future. A random group of programmers applied this system to the fuel dump module and other modules.

Seventeen additional, previously unknown problems surfaced!

Some Software failures

Financial Software

Many companies have experienced failures in their accounting system due to faults in the software itself. The failures range from producing the wrong information to the whole system crashing.

Some Software failures

Windows XP

- Microsoft released Windows XP on October 25, 2001.
- On the same day company posted 18 MB of compatibility patches on the website for bug fixes, compatibility updates, and enhancements.
- Two patches fixed important security holes.

This is **Software Engineering.**

Software Product

- **Software products** may be developed for a particular customer or may be developed for a general market

- **Software products** may be
  - –**Generic** - developed to be sold to a range of different customers
  - –Customized (**Bespoke)** - developed for a single customer according to their specification

Software Product

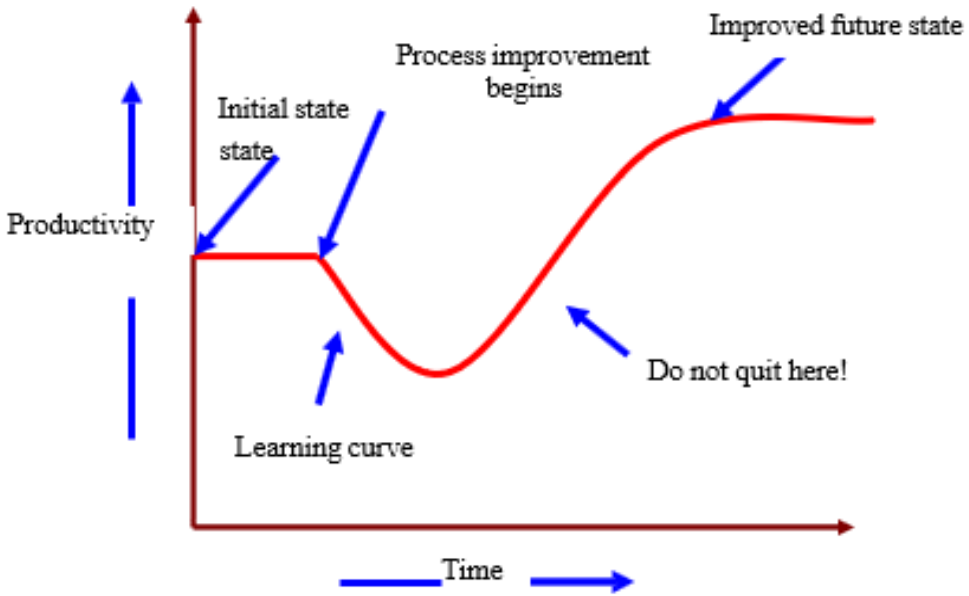
Software product is a product designated for delivery to the user

Software Process

The software process is the way in which we produce software.

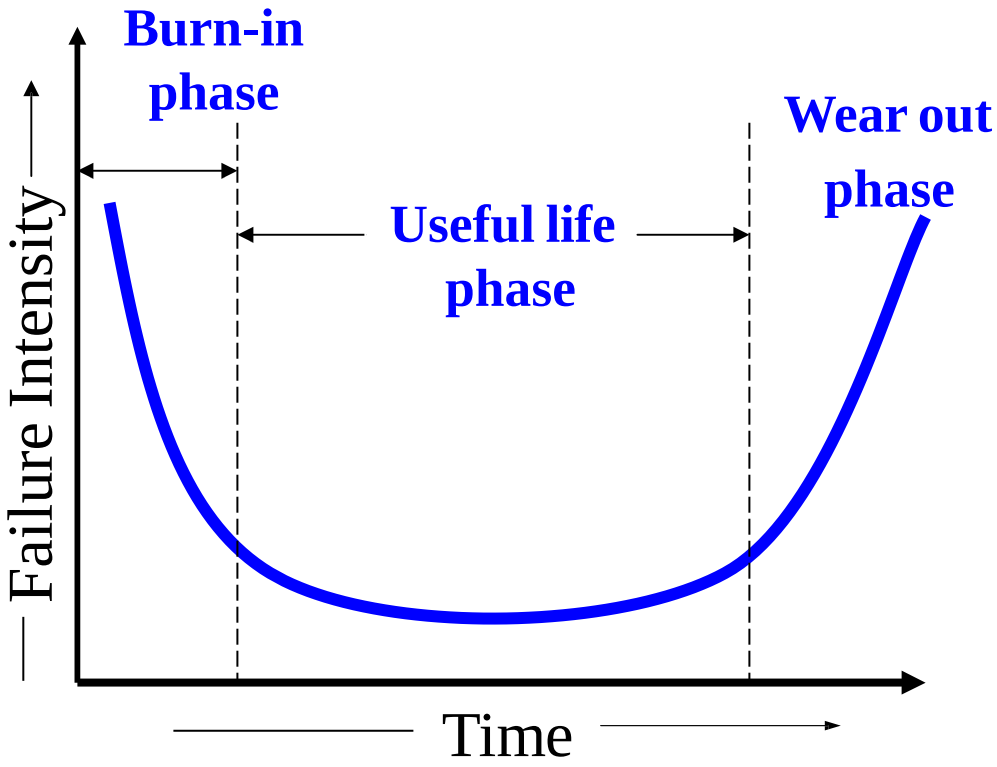
Why is it difficult to improve software process ?

- Not enough time
- Lack of knowledge
- Wrong motivations
- Insufficient commitment



Software Characteristics:

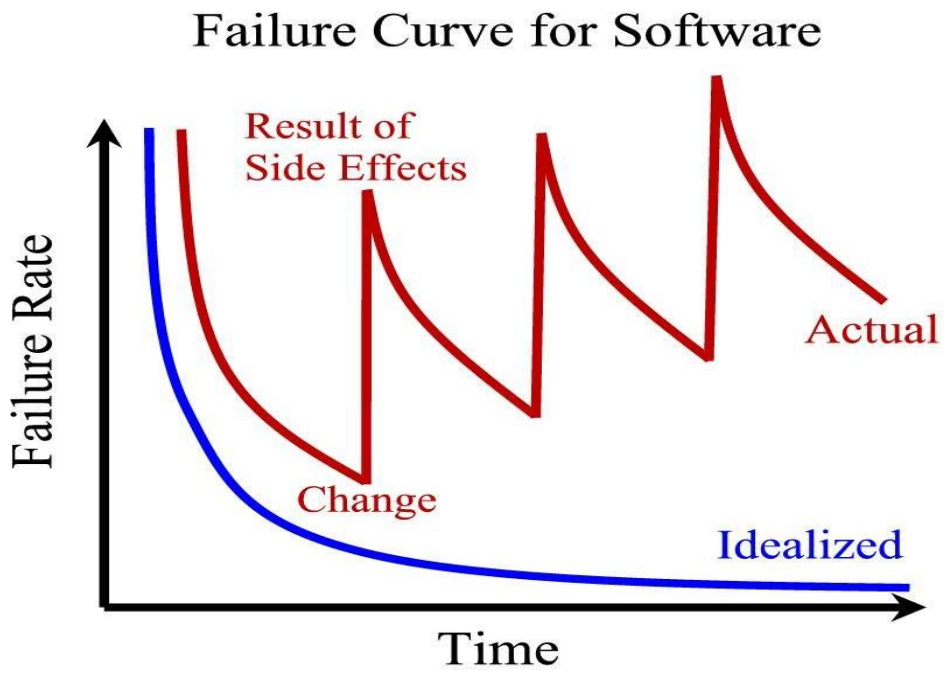
✓Software does not wear out.



- **Burn in phase**: just after the development of the product, high failure intensity
- **Useful life phase**: while testing of product and fixing faults before delivery
- **Wear-out phase**: when the component starts wearing out

Software Characteristics:

- ✓ S/w is developed or engineered, it is not manufactured
- ✓ S/w is one-time development effort and continuously maintains its efforts to keep it operational.
- ✓ Reusability of components
- ✓ Software is flexible
- ✓ Software is intangible

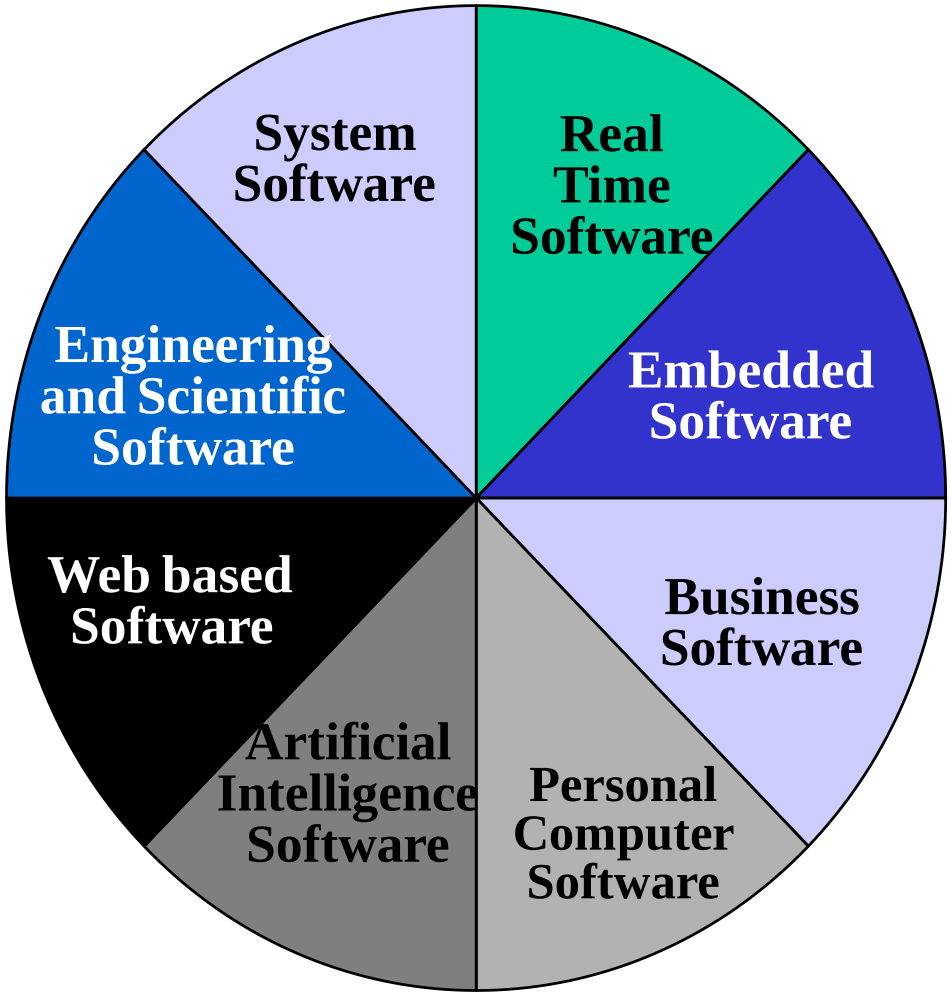


Software Characteristics:

Comparison of constructing a bridge vis-à-vis writing a program.

Sr. No	Constructing a bridge	Writing a program
1.	The problem is well understood	Only some parts of the problem are understood, others are not
2.	There are many existing bridges	Every program is different and designed for special applications.
3.	The requirement for a bridge typically do not change much during construction	Requirements typically change during all phases of development.
4.	The strength and stability of a bridge can be calculated with reasonable precision	Not possible to calculate correctness of a program with existing methods.
5.	When a bridge collapses, there is a detailed investigation and report	When a program fails, the reasons are often unavailable or even deliberately concealed.
6.	Engineers have been constructing bridges for thousands of years	Developers have been writing programs for 50 years or so.
7.	Materials (wood, stone,iron, steel) and techniques (making joints in wood, carving stone, casting iron) change slowly.	Hardware and software changes rapidly.

The Changing Nature of Software



The Changing Nature of Software

Trend has emerged to provide source code to the customer and organizations.

Software where source codes are available are known as open source software.

Examples

Open source software: LINUX, MySQL, PHP, Open office, Apache webserver etc.

Software Myths (Management Perspectives)

Management may be confident about good standards and clear procedures of the company.

But the taste of any food item is in the eating; not in the Recipe !

Software Myths (Management Perspectives)

Company has latest computers and state-of-the-art software tools, so we shouldn't worry about the quality of the product.

The infrastructure is only one of the several factors that determine the quality of the product!

Software Myths (Management Perspectives)

Addition of more software specialists, those with higher skills and longer experience may bring the schedule back on the track!

Unfortunately,
that may further delay the schedule!

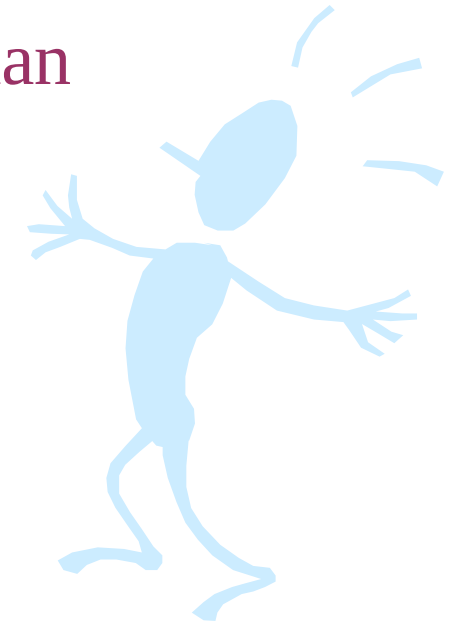
Software Myths (Management Perspectives)

Software is easy to change

The reality is totally different.

Software Myths (Management Perspectives)

Computers provide greater reliability than the devices they replace

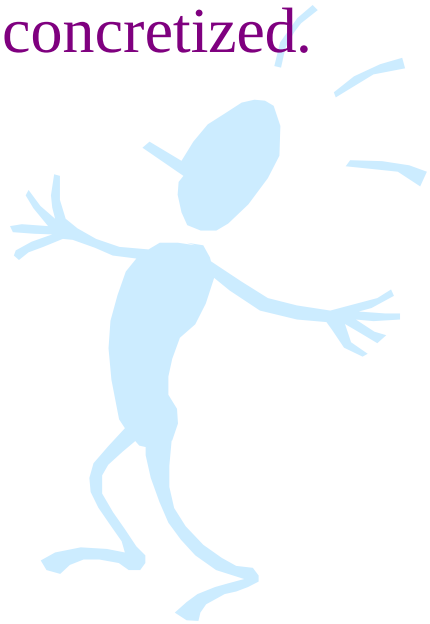


This is not always true.

Software Myths (Customer Perspectives)

A general statement of objectives is sufficient to get started with the development of software. Missing/vague requirements can easily be incorporated/detailed out as they get concretized.

If we do so, we are heading towards a disaster.



Software Myths (Customer Perspectives)

Software with more features is better software

Software can work right the first time

Both are only myths!

Software Myths (Developer Perspectives)

Once the software is demonstrated, the job is done.

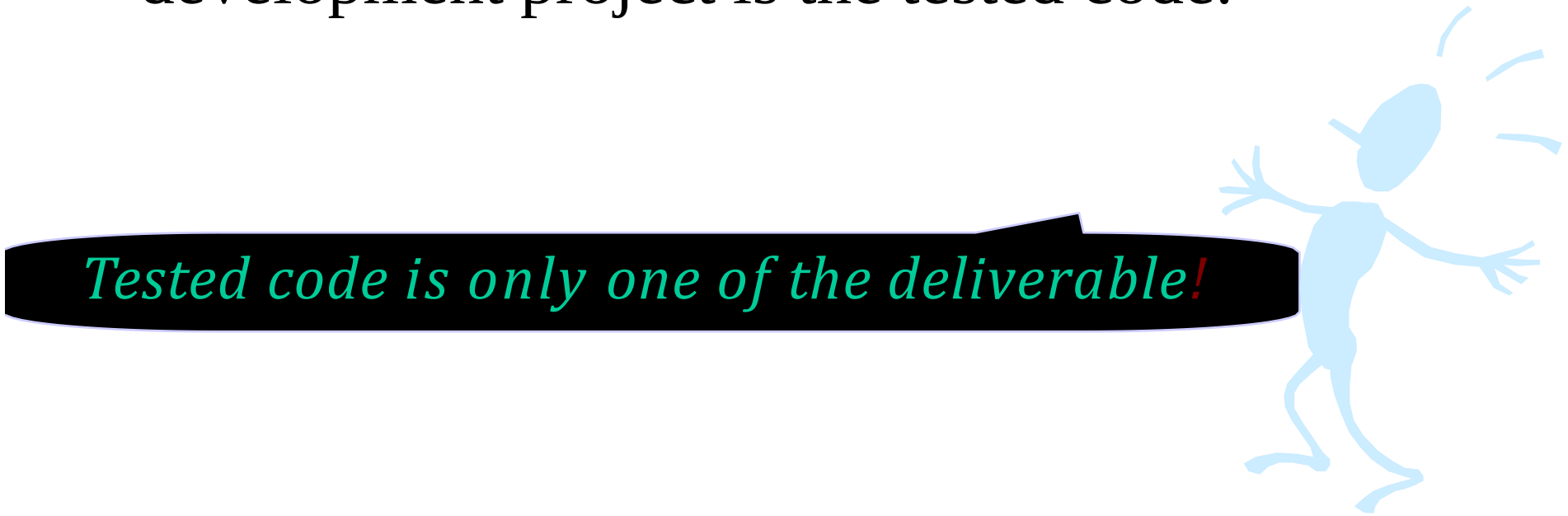
Usually, the problems just begin!

Software Myths (Developer Perspectives)

Software quality can not be assessed before testing.

Software Myths (Developer Perspectives)

The only deliverable for a software development project is the tested code.



Software Myths (Developer Perspectives)

Aim is to develop working programs

Those days are over. Now objective is to develop good quality maintainable programs!

Some Terminologies

- Deliverables and Milestones

Different deliverables are generated during software development. The examples are source code, user manuals, operating procedure manuals etc.

The milestones are the events that are used to ascertain the status of the project. Finalization of specification is a milestone. Completion of design documentation is another milestone. The milestones are essential for project planning and management.

Some Terminologies

- Product and Process

Product: What is delivered to the customer, is called a product. It may include source code, specification document, manuals, documentation etc. Basically, it is nothing but a set of deliverables only.

Process: Process is the way in which we produce software. It is the collection of activities that leads to (a part of) a product. An efficient process is required to produce good quality products.

If the process is weak, the end product will undoubtedly suffer, but an obsessive over reliance on process is also dangerous.

Some Terminologies

➢ Measures, Metrics and Measurement

A measure provides a quantitative indication of the extent, dimension, size, capacity, efficiency, productivity or reliability of some attributes of a product or process.

Measurement is the act of evaluating a measure.

A metric is a quantitative measure of the degree to which a system, component or process possesses a given attribute.

Some Terminologies

➢ Software Process and Product Metrics

Process metrics quantify the attributes of software development process and environment;

whereas product metrics are measures for the software product.

Examples

Process metrics: Productivity, Quality, Efficiency etc.

Product metrics: Size, Reliability, Complexity etc.

Some Terminologies

➢ Productivity and Effort

Productivity is defined as the rate of output, or production per unit of effort, i.e. the output achieved with regard to the time taken but irrespective of the cost incurred.

Hence most appropriate unit of effort is Person Months (PMs), meaning thereby number of persons involved for specified months. So, productivity may be measured as LOC/PM (lines of code produced/person month)

Some Terminologies

- Module and Software Components

There are many definitions of the term module. They range from “a module is a FORTRAN subroutine” to “a module is an Ada Package”, to “Procedures and functions of PASCAL and C”, to “C++ Java classes” to “Java packages” to “a module is a work assignment for an individual developer”. All these definitions are correct. The term subprogram is also used sometimes in place of module.

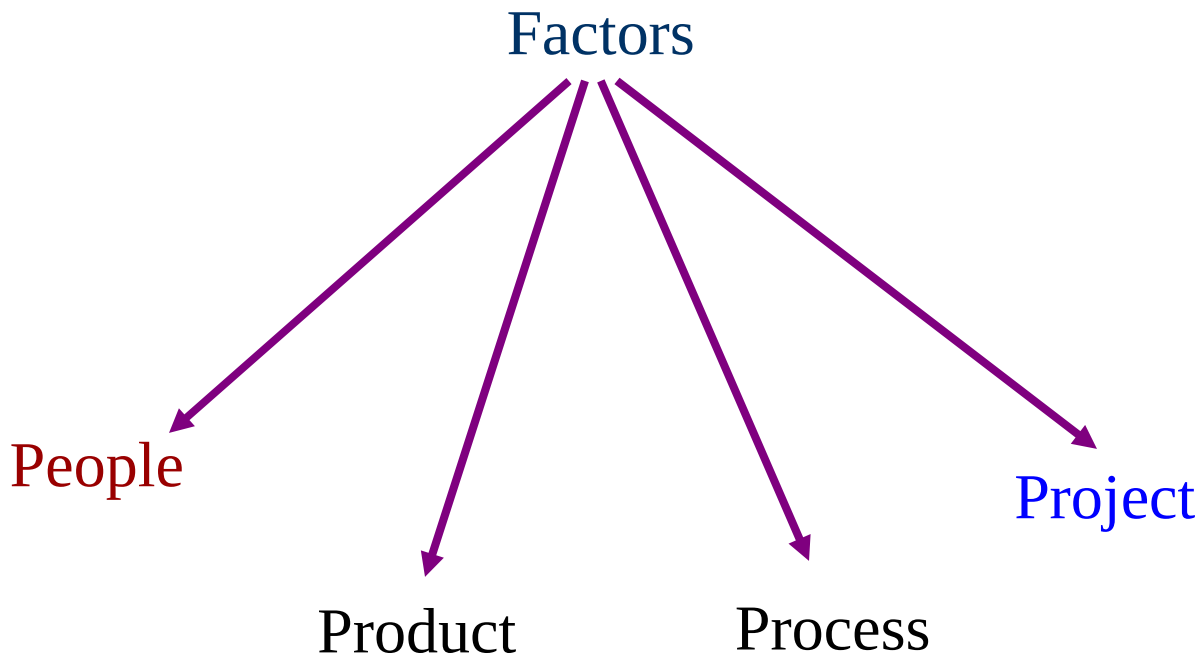
Some Terminologies

- Generic and Customized Software Products

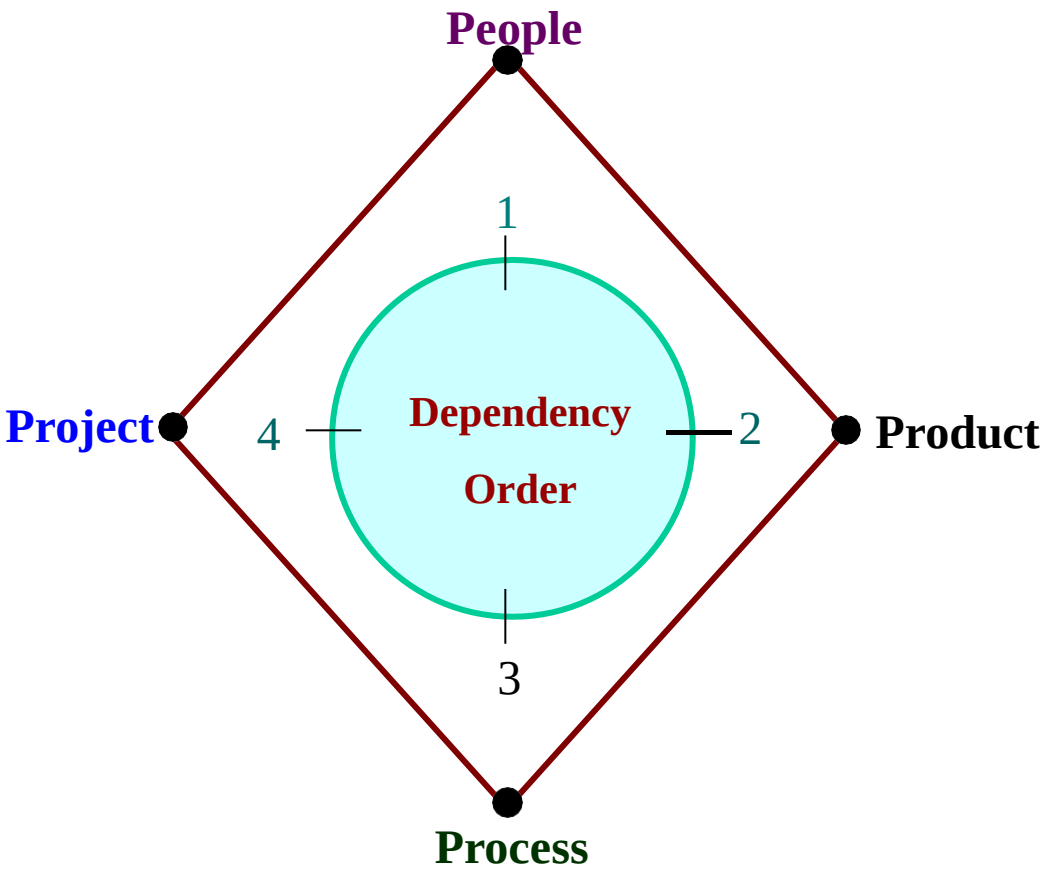
Generic products are developed for anonymous customers. The target is generally the entire world and many copies are expected to be sold. Infrastructure software like operating system, compilers, analyzers, word processors, CASE tools etc. are covered in this category.

The customized products are developed for particular customers. The specific product is designed and developed as per customer requirements. Most of the development projects (say about 80%)come under this category.

Role of Management in Software Development



Role of Management in Software Development



Software Life Cycle Models

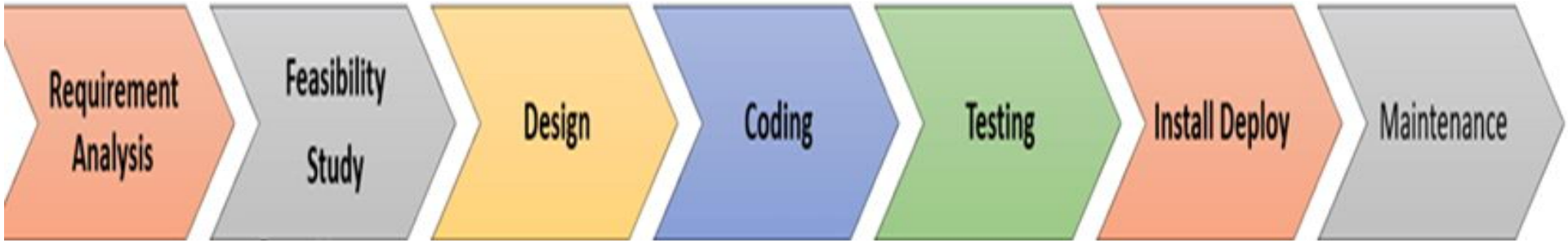
Software Life Cycle Models

The goal of Software Engineering is to provide models and processes that lead to the production of well-documented maintainable software in a manner that is predictable.

Software Life Cycle Models

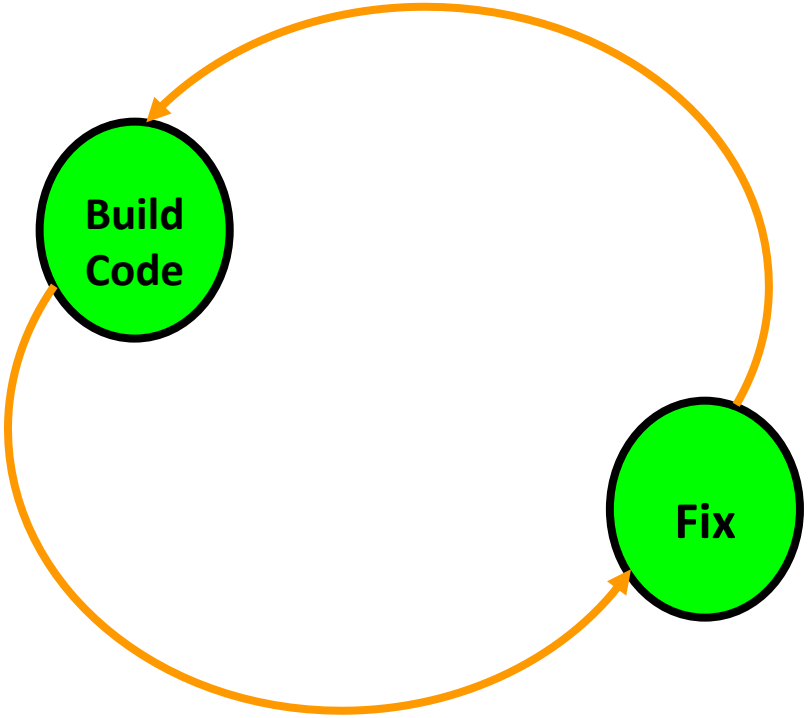
- “The period of time that starts when a software product is conceived and ends when the product is no longer available for use. The software life cycle typically includes a requirement phase, design phase, implementation phase, test phase, installation and check out phase, operation and maintenance phase, and sometimes retirement phase”.
- Life cycle model represents all the activities to make a software product. It also captures the order in which these activities are to be undertaken.
- Software development is not a “one-man show” that the person decides on his own which work to do and when. It is teamwork.

Common SDLC Phases



Build & Fix Model

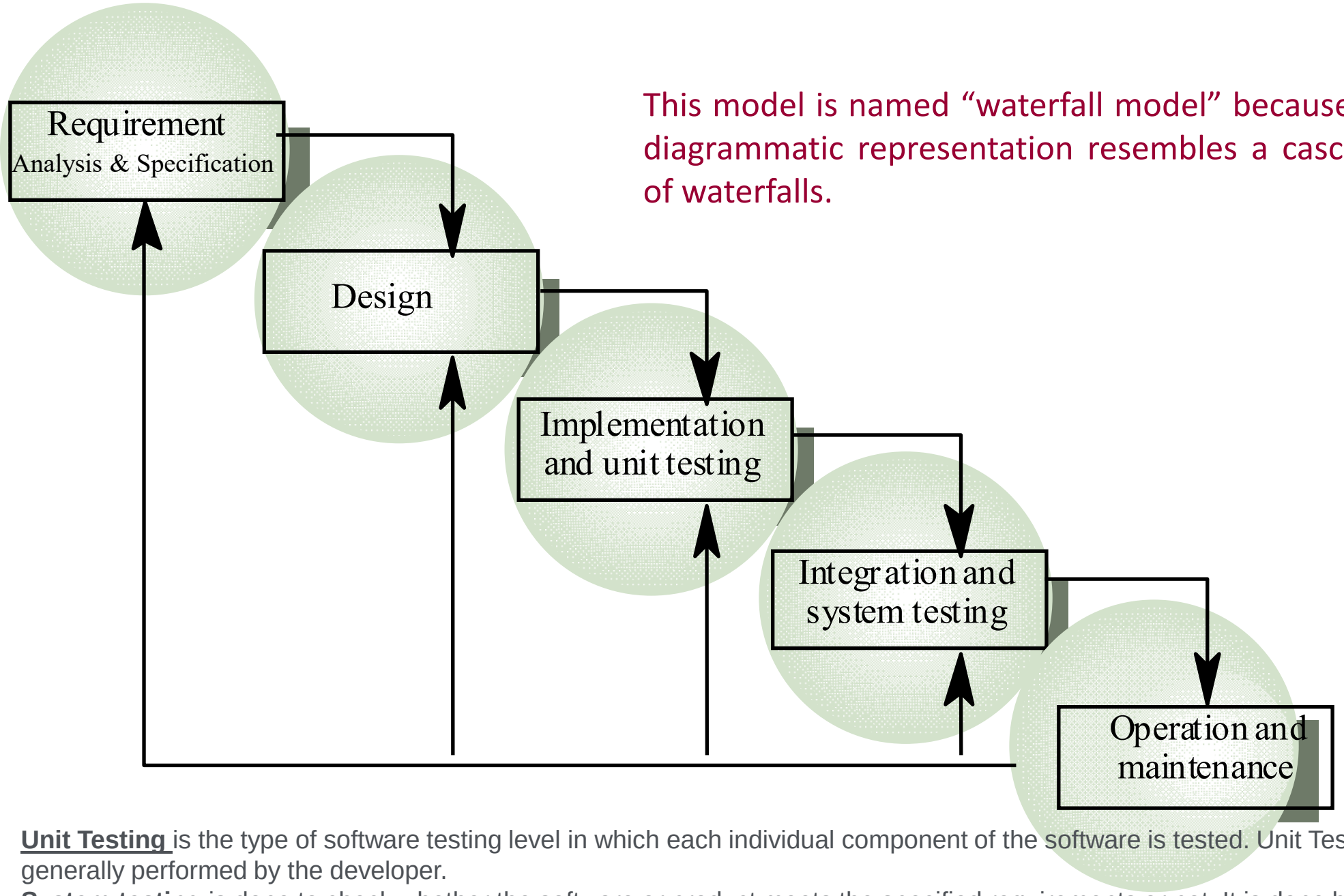
- Product is constructed without specifications or any attempt at design
- Adhoc approach and not well defined
- Simple two phase model



Build & Fix Model

- Suitable for small programming exercises of 100 or 200 lines
- Unsatisfactory for software for any reasonable size
- Code soon becomes unfixable & unenhanceable
- No room for structured design
- Maintenance is practically not possible

Waterfall Model



This model is named “waterfall model” because its diagrammatic representation resembles a cascade of waterfalls.

Unit Testing is the type of software testing level in which each individual component of the software is tested. Unit Testing is generally performed by the developer.

System testing is done to check whether the software or product meets the specified requirements or not. It is done by both testers and developers.

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.

Waterfall Model

Advantages of waterfall model

i. Simplicity

ii. Clearly divided phases: each phase deals with a separate logical concern.

iii. Easy to understand by a non-technical person.

iv. Each stage has well-defined deliverables/outputs.

v. Defects are easy to find.

Waterfall Model

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.

vi. Users have little interaction with the project team and their feedback is not taken during development.

Software projects that were developed using Waterfall model

1. Microsoft Office: The original version of Microsoft Office was developed using the Waterfall model.
2. SAP: The original version of SAP, an enterprise resource planning (ERP) software suite, was developed using the Waterfall model.
3. NASA Space Shuttle software: The software used to control the Space Shuttle was developed using a variation of the Waterfall model, with an emphasis on rigorous testing and quality control.
4. Medical devices: Many medical devices, such as pacemakers and insulin pumps, are developed using the Waterfall model due to the need for strict quality control and regulatory compliance
5. Google Map, Google Search and Google drive were initially developed using waterfall model but later they adopted agile approach for development.

Incremental Process Models

- Incremental Model is a process of software development where requirements are divided into multiple standalone modules of the software development cycle.
- They are effective in situations where requirements are defined precisely and there is no confusion about the functionality of the final product.
- After every cycle a usable product is given to the customer.
- Popular particularly when we have to quickly deliver a limited functionality system.

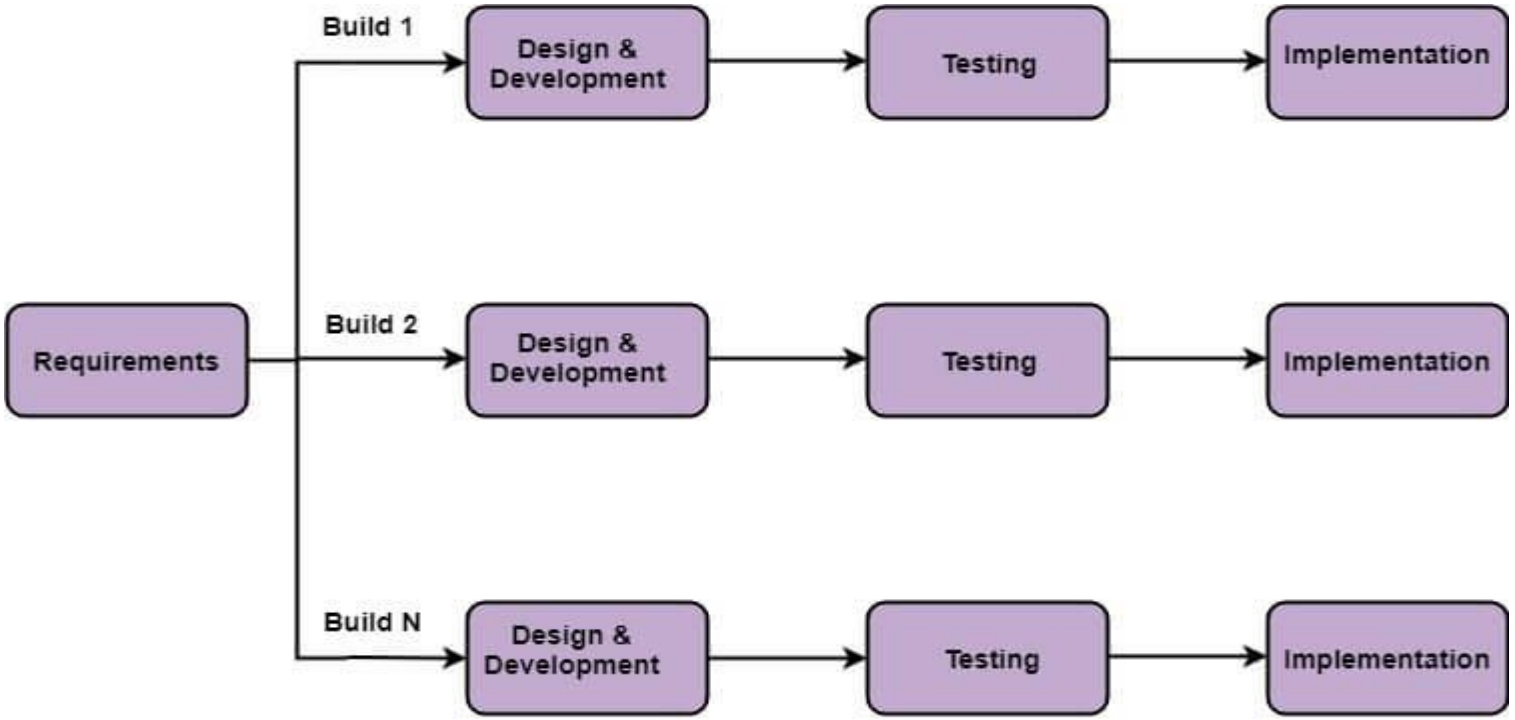


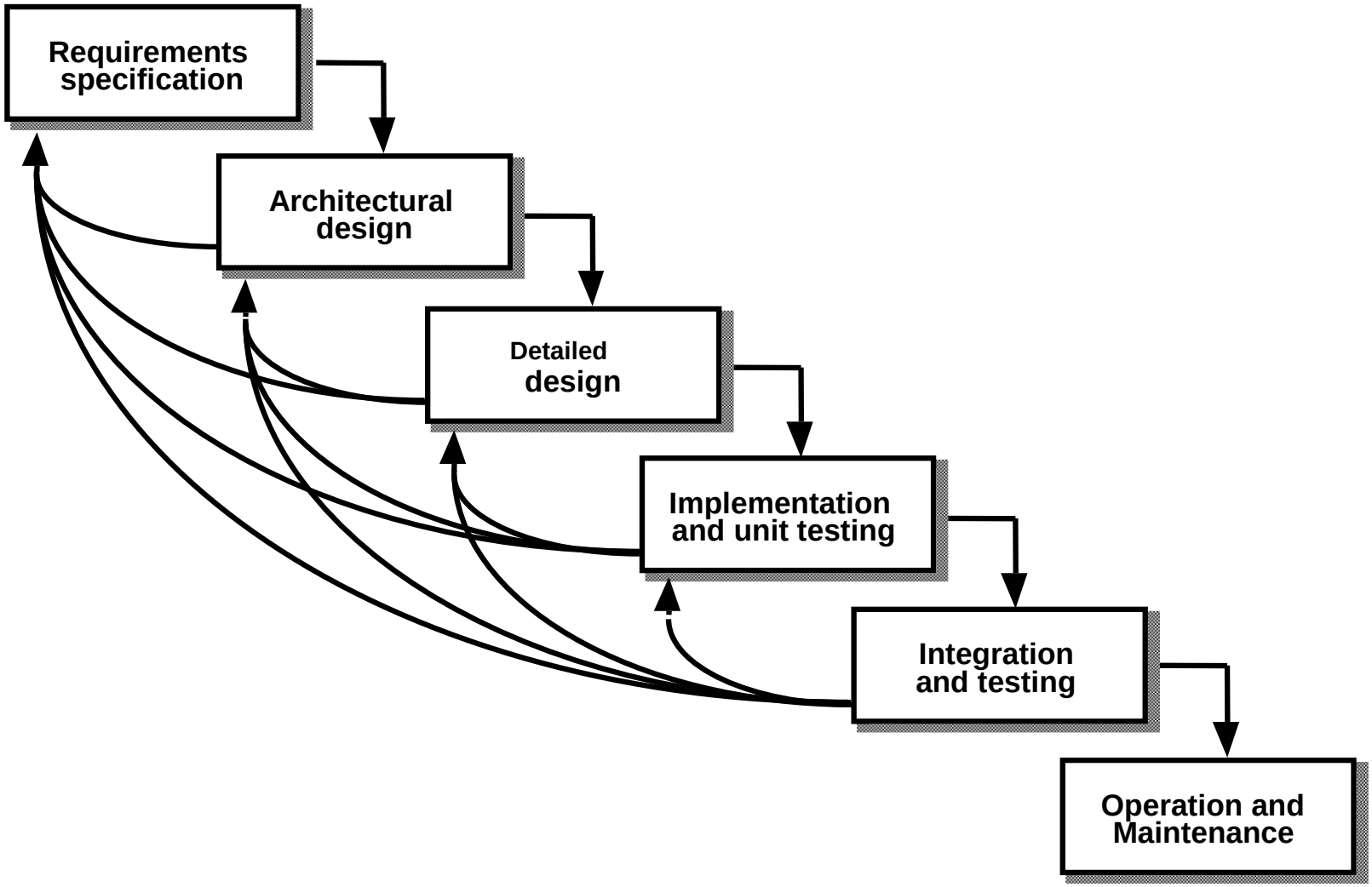
Fig: Incremental Model

Iterative Enhancement Model

This model has the same phases as the waterfall model, but with fewer restrictions. Generally, the phases occur in the same order as in the waterfall model, but they may be conducted in several cycles. A usable product is released at the end of each cycle, with each release providing additional functionality.

- ✓ Customers and developers specify as many requirements as possible and prepare a SRS document.
- ✓ Developers and customers then prioritize these requirements
- ✓ Developers implement the specified requirements in one or more cycles of design, implementation and test based on the defined priorities.

Iterative Enhancement Model



Characteristics of Iterative Enhancement Model

- The iterative model of software development allows for flexibility and adaptability, as the software can be developed and refined over time based on feedback from the customer or stakeholders.
- It is particularly useful when the customer's requirements are not fully known at the outset of the project, or when there is a high level of complexity or uncertainty in the development process.
- By breaking the development process into smaller iterations, the iterative model reduces the risk of project failure and allows the development team to deliver working software more quickly.
- Iterative Enhancement model is used where it is difficult to understand the detailed requirements of whole software in one go instead the major functionalities of the software are identified and customer is asked to prioritize the requirements.
- As per the prioritization of the requirements by the customer, the detailed requirements of one part are understood, analyzed, developed and deliver to the customer and then the next set of requirements as per the priority are understood, specified , analysed, developed, tested and delivered to customer.
- This way the whole software is developed iteratively in various cycles till all requirements are met.
- The prerequisite to use this model is that we should have senior analyst which can understand the relation between different functionalities properly so that integration of all cycles doesn’t cause any problem.

Software projects that developed using Iterative Enhancement model:

The iterative model of software development is commonly used for developing complex and large-scale software applications that require ongoing improvement and refinement. Some examples of software that have been developed using the iterative model include:

- Microsoft Windows operating system: The development of Windows operating system involves a series of iterations or releases, each adding new features and improvements based on customer feedback and changing market trends.
- Google Chrome web browser: Chrome is developed using an iterative model, with regular releases that add new features, improve performance and security, and fix bugs.
- Agile project management tools: Many agile project management tools, such as Jira and Trello, are developed using iterative development to continuously improve their functionality and user experience.
- Mobile applications: Mobile applications, such as Instagram and Snapchat, are often developed using iterative development to quickly respond to changing user needs and improve user engagement.
- Enterprise-level software: Large enterprise software applications, such as SAP and Salesforce, are typically developed using iterative development to accommodate the changing needs of their users and to adapt to new technologies and business trends.

Spotify : developed using Iterative Enhancement model

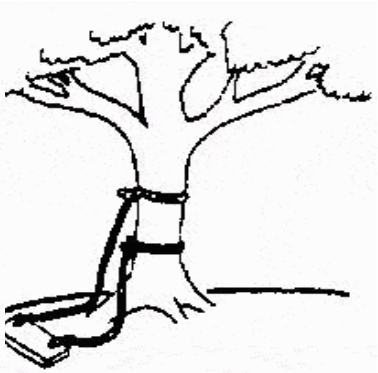
Spotify: Spotify was developed using an iterative model to ensure that it provided a personalized and engaging music streaming experience for its users.
Spotify has undergone several iterations in its development process. Here are some notable ones:
1.Spotify Desktop (2006): The first iteration of Spotify was a desktop application that allowed users to stream music and create playlists.
2.Spotify Mobile (2009): In 2009, Spotify launched its first mobile app for the iPhone. This allowed users to access their music on the go and was the first step in the company's move towards a mobile-first approach.
3.Spotify Discover (2012): The Discover feature was introduced in 2012, which used an algorithm to recommend new music to users based on their listening history.
4.Spotify Running (2015): In 2015, Spotify introduced a new feature called Running, which used sensors in users' smartphones to detect their pace and play music that matched their workout tempo.
5.Spotify for Artists (2018): Spotify for Artists was launched in 2018, which allowed musicians to manage their profiles on the platform, track their streams and demographics, and promote their music.
6.Spotify Podcasts (2019): Spotify made a big push into the podcasting market in 2019, by acquiring several podcast production companies and launching exclusive podcasts on the platform.
Throughout its development, Spotify has continued to use an iterative approach, constantly testing and iterating on new features based on user feedback and data analysis.

The Rapid Application Development (RAD) Model

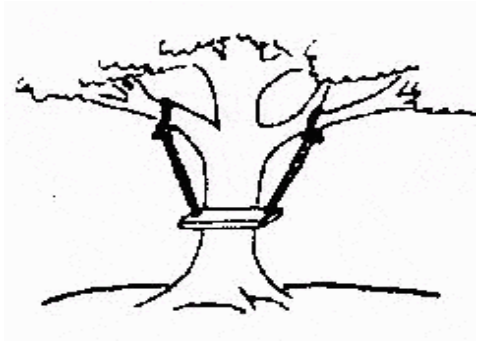
- Developed by IBM in 1980
- User participation is essential

The requirements specification was defined like this

The developers understood it in that way

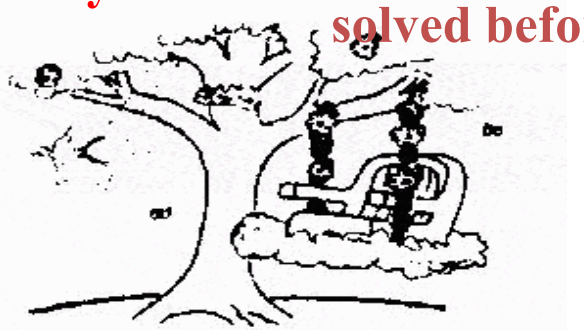


This is how the problem was solved before.

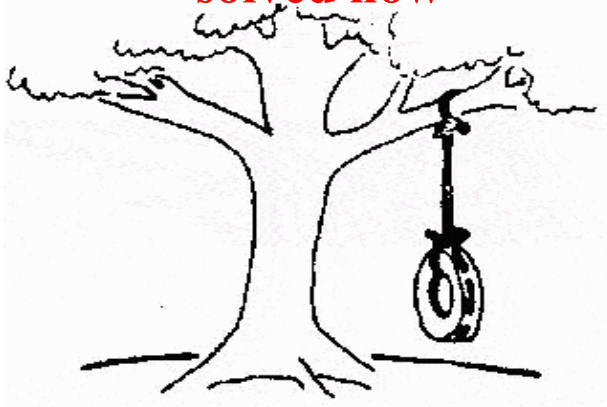


This is how the problem is solved now

That is the program after debugging

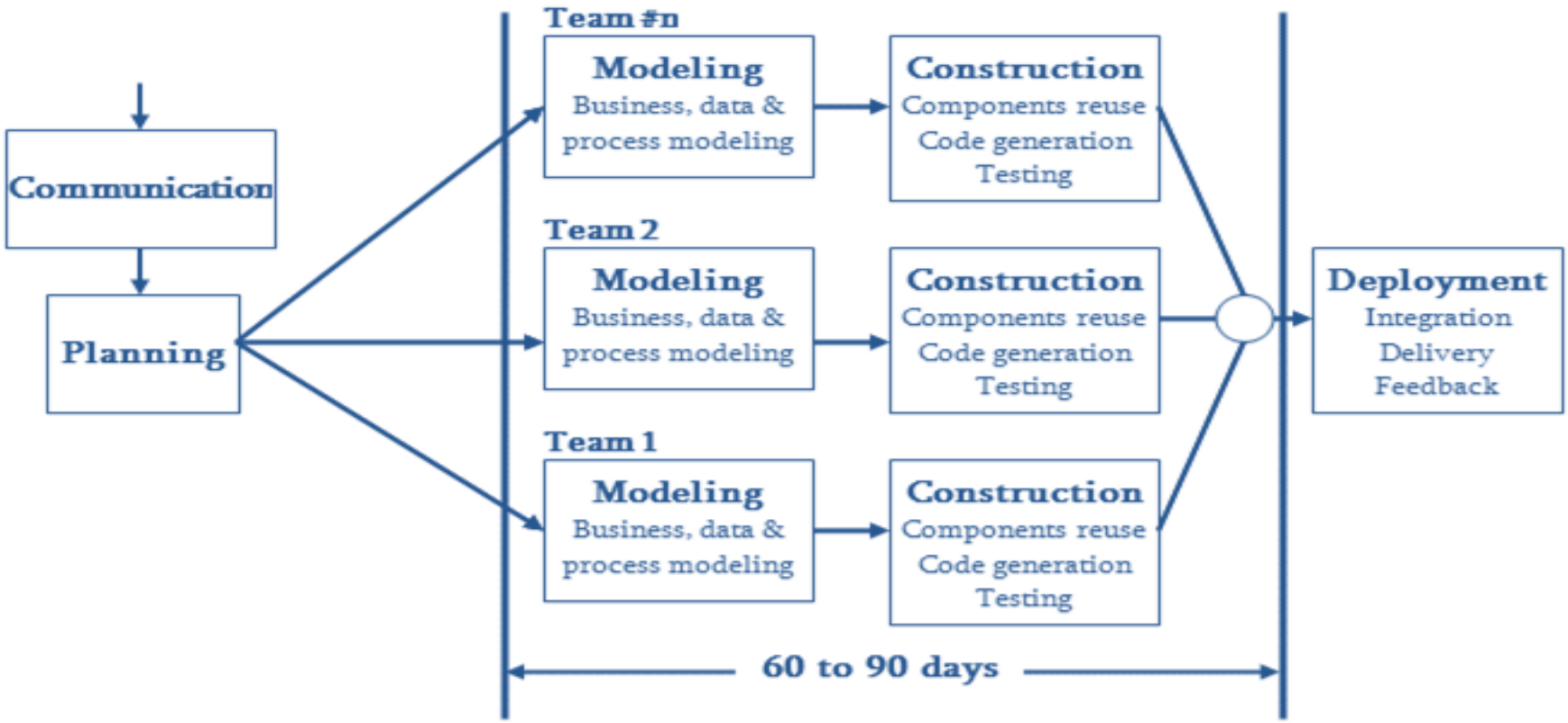


This is how the program is described by marketing department



This, in fact, is what the customer wanted …

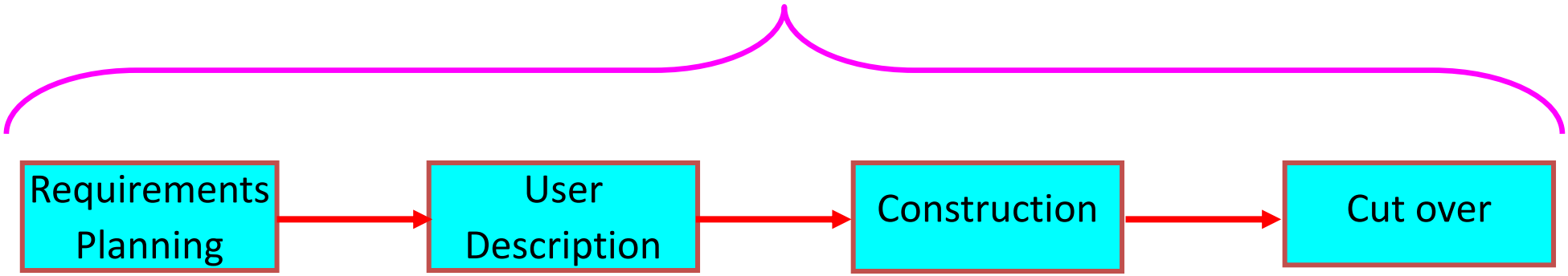
RAD MODEL


Communication
Planning
Team #n
Modeling
Business, data & process modeling
Construction
Components reuse
Code generation
Testing
Team 2
Modeling
Business, data & process modeling
Construction
Components reuse
Code generation
Testing
Team 1
Modeling
Business, data & process modeling
Construction
Components reuse
Code generation
Testing
Deployment
Integration
Delivery
Feedback
60 to 90 days

The Rapid Application Development (RAD) Model

- Build a rapid prototype
- Give it to user for evaluation & obtain feedback
- Prototype is refined

With active participation of users



The Rapid Application Development (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.

The Rapid Application Development (RAD) Model

Advantages of RAD Model

i. This model is flexible for change.

ii. In this model, changes are adoptable.

iii. Each phase in RAD brings highest priority functionality to the customer.

iv. It reduced development time.

v. It increases the reusability of features.

Disadvantage of RAD Model

i. It required highly skilled designers.

ii. All application is not compatible with RAD.

iii. For smaller projects, we cannot use the RAD model.

iv. On the high technical risk, it's not suitable.

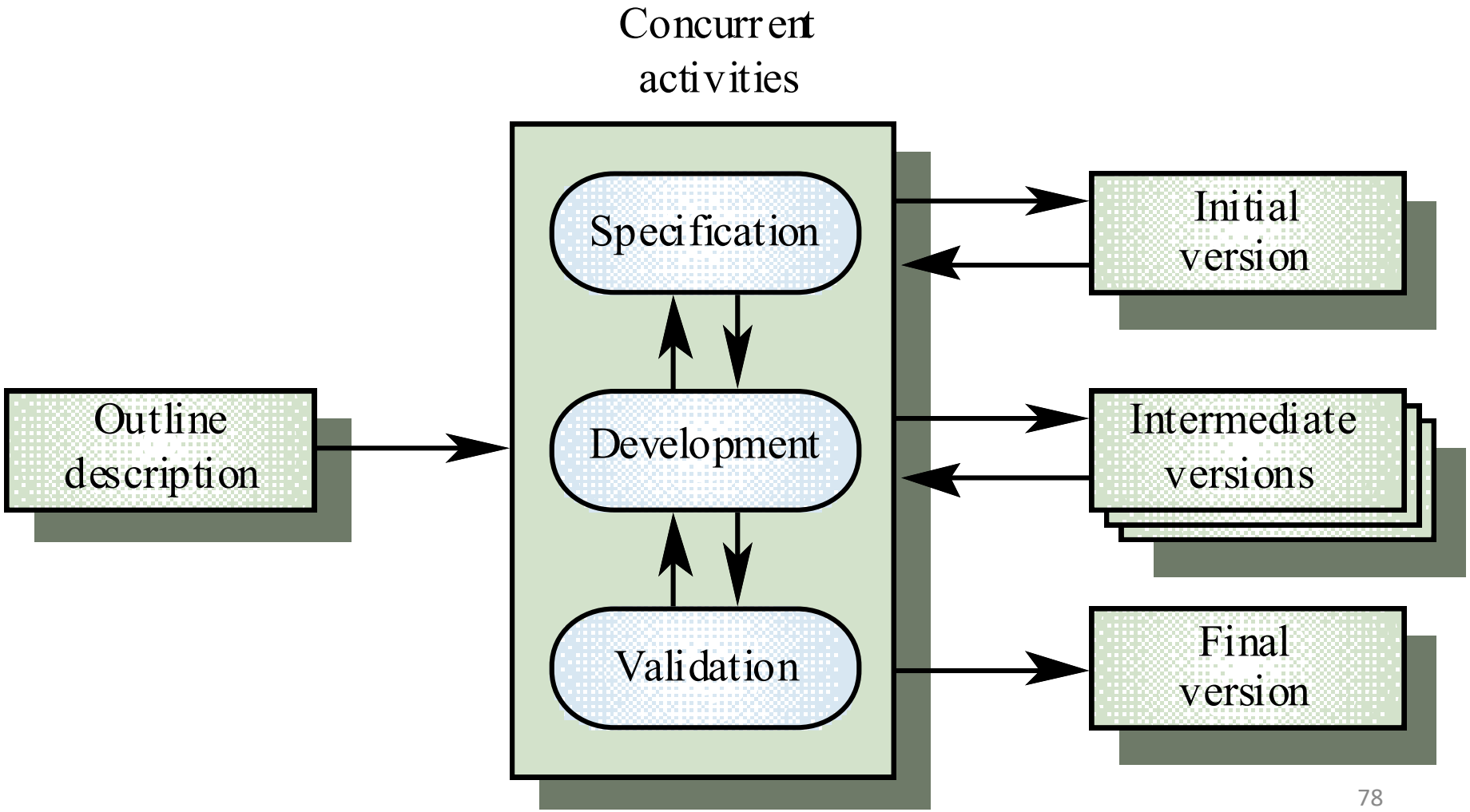
v. Required user involvement.

Evolutionary Process Models

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.

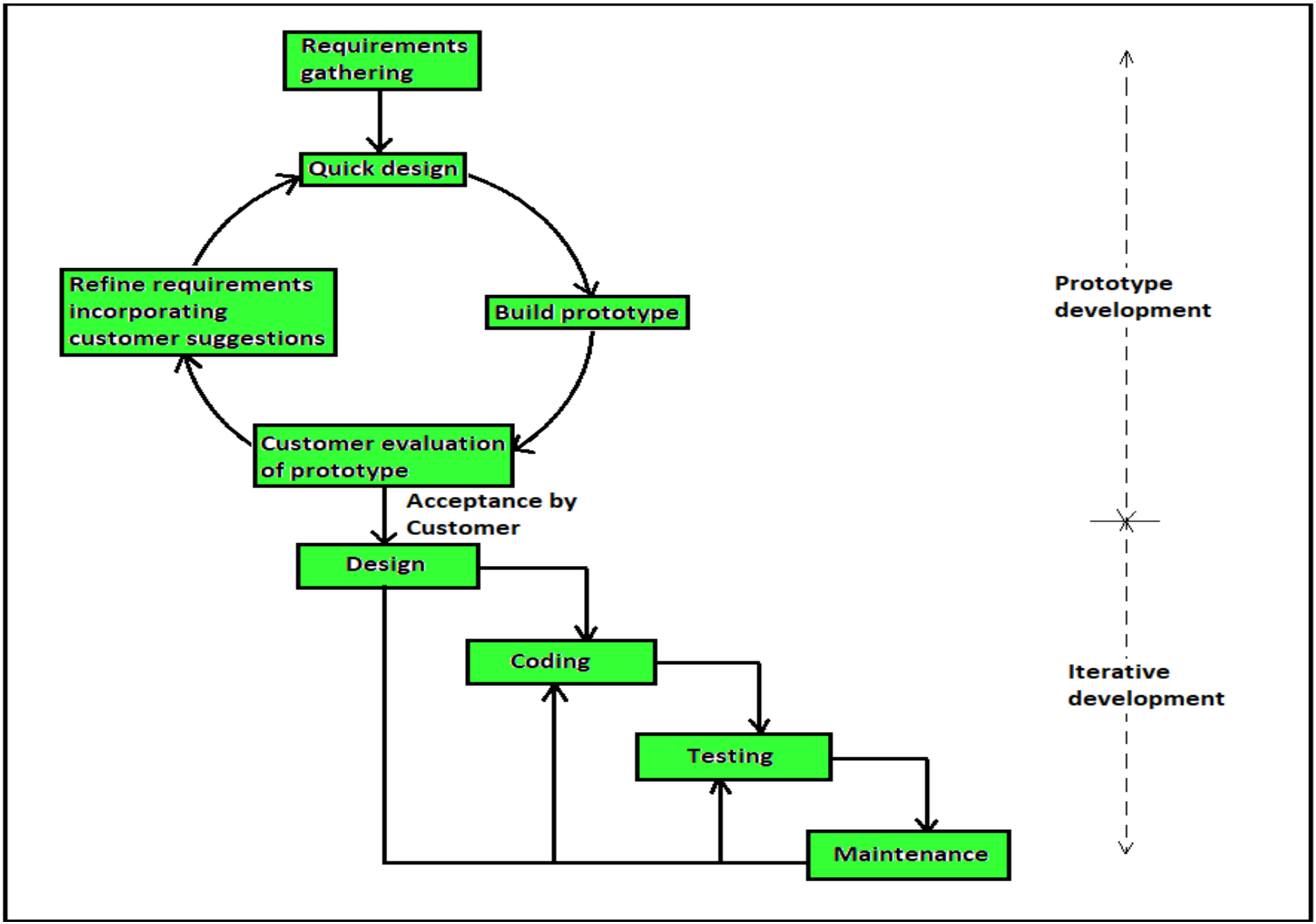
Evolutionary Process Model



Prototyping Model

- 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.

Prototyping Model



Spiral Model

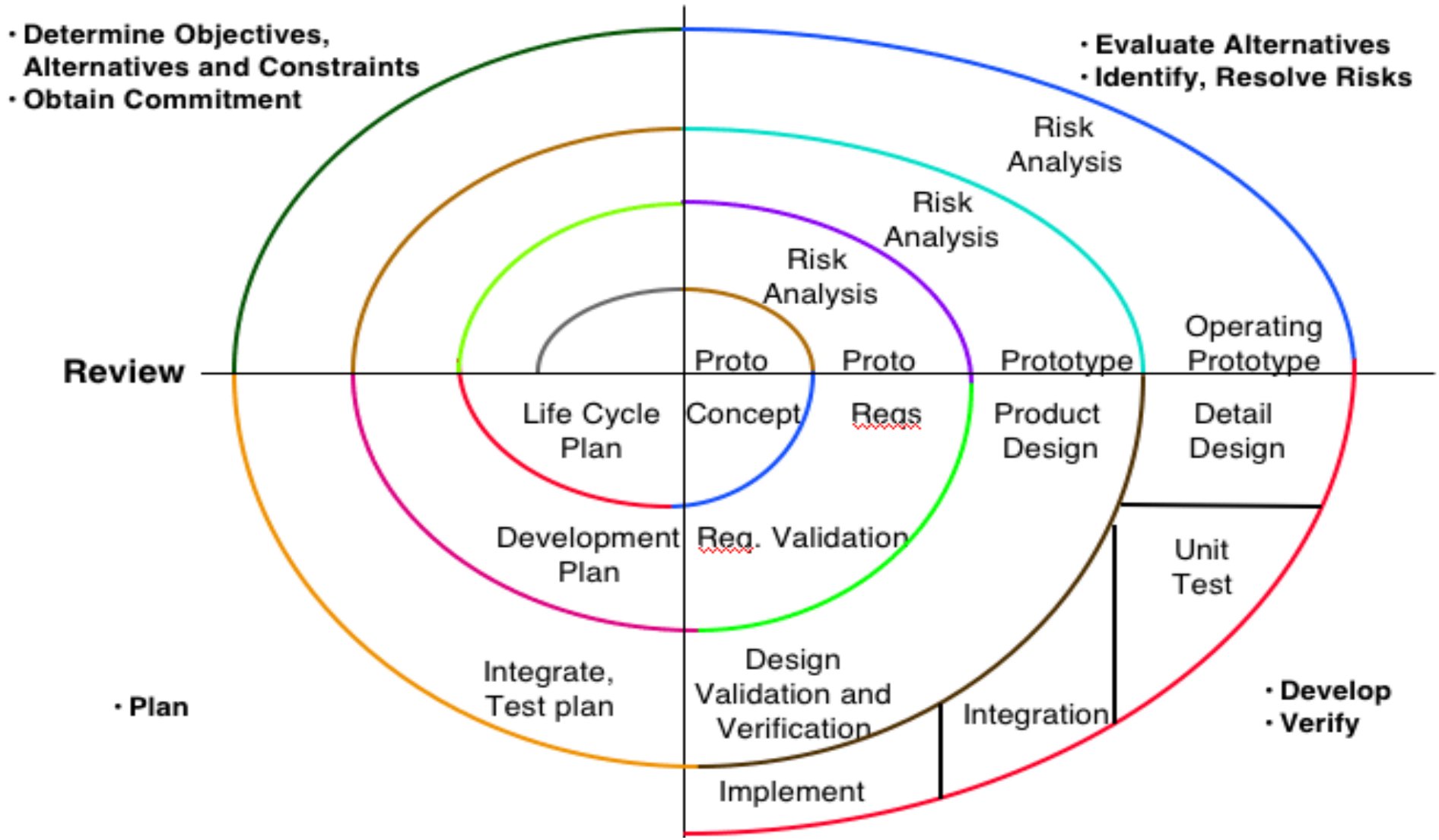
Models do not deal with uncertainly which is inherent to software projects.

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.

Spiral Model



Spiral Model

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 360^{o} 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

Spiral Model

- 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.

Which type of software are developed using spiral model

The spiral model of software development is suitable for projects that involve a high level of risk or uncertainty and require a systematic approach to risk management.

It is particularly well-suited for large and complex software projects that have changing requirements, and where there is a need to continually evaluate and mitigate risks.

Examples of software projects that may be developed using the spiral model include:

Defense systems: Defense systems, such as missile defense systems, require a high level of reliability and safety. The spiral model is suitable for such projects as it allows for a systematic approach to risk management and continuous evaluation and testing.

Medical devices: Medical devices, such as implantable devices or diagnostic systems, require a high level of safety and reliability. The spiral model is well-suited for such projects as it allows for continuous risk assessment and testing throughout the development process.

Large-scale enterprise systems: Large-scale enterprise systems, such as ERP systems, involve multiple stakeholders and complex requirements. The spiral model is suitable for such projects as it allows for flexibility and adaptation to changing requirements, as well as systematic risk management.

Complex software products: Complex software products, such as CAD or simulation software, involve multiple features and functionalities. The spiral model is well-suited for such projects as it allows for iterative development and testing of new features.

Which software have been developed using spiral model

Several software products have been developed using the spiral model of software development. Here are a few examples:

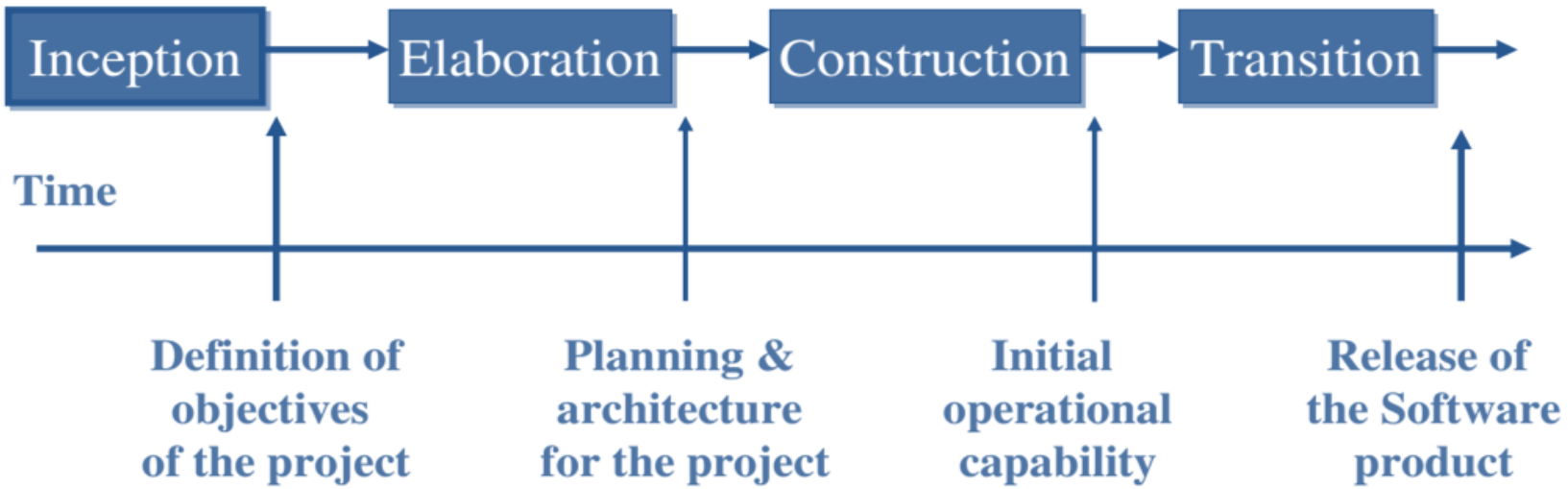
Microsoft Office Suite: Microsoft Office Suite is a widely used productivity software that includes Word, Excel, PowerPoint, and other applications. The software was developed using the spiral model, which allowed for continuous evaluation and testing of new features.

NASA Mission Control Software: NASA Mission Control software was developed using the spiral model to ensure reliability and safety in the control of space missions. The software went through several iterations and evaluations to ensure that it met the required safety standards.

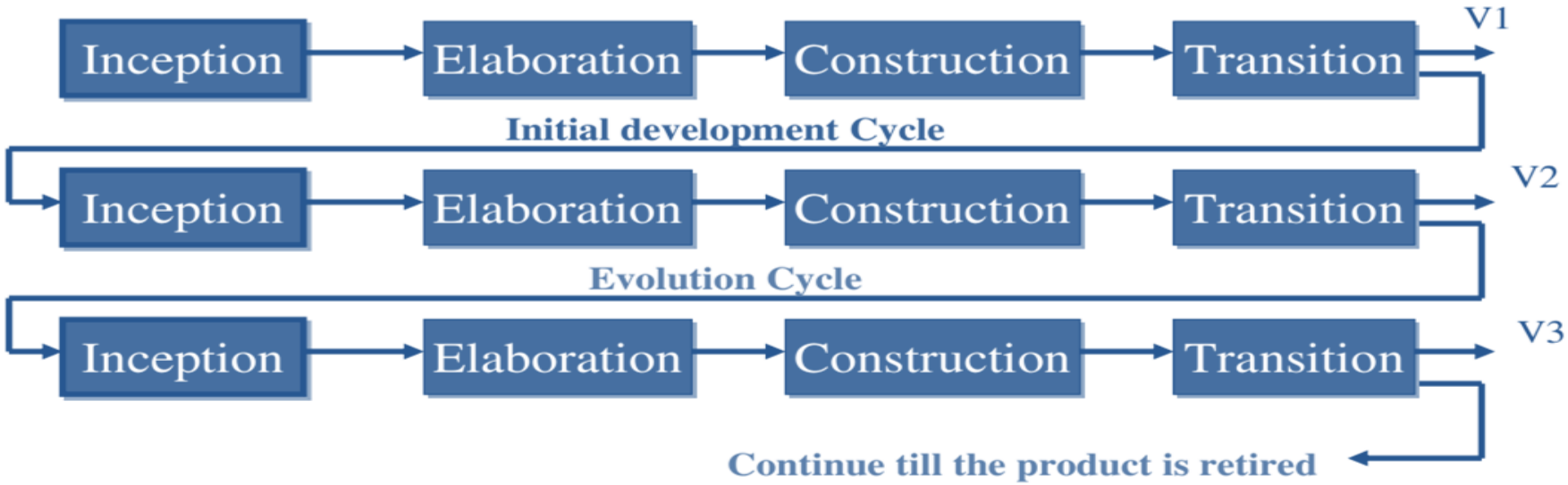
Internet Explorer: Internet Explorer was developed using the spiral model, which allowed for iterative development and testing of new features. The model also allowed for flexibility in adapting to changing market conditions and user requirements.

HP LaserJet Printers: HP LaserJet Printers were developed using the spiral model to ensure reliability and quality in their printing capabilities. The model allowed for continuous testing and evaluation of the printers' hardware and software components.

Rational Unified Process

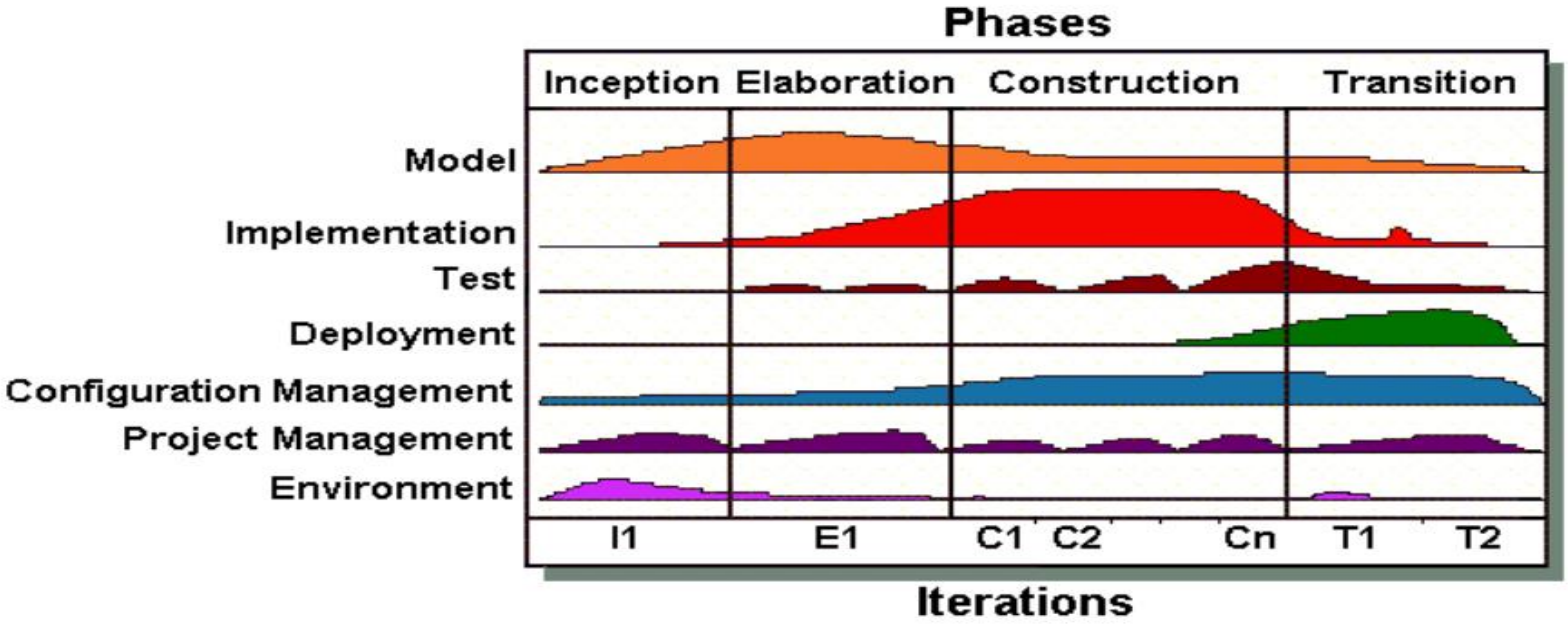


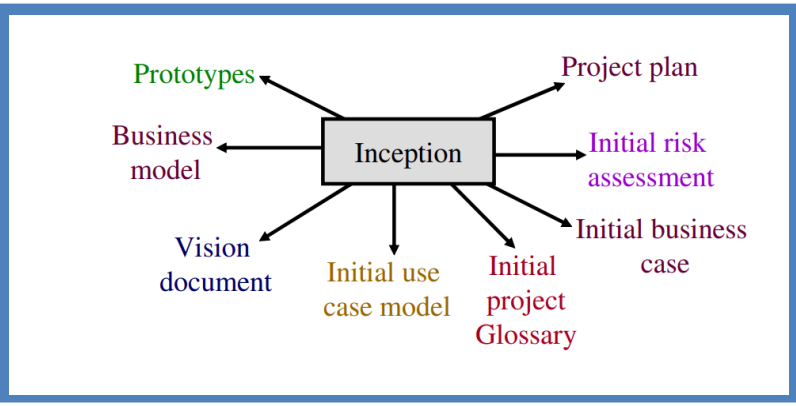
Initial Development and Evolution Cycles

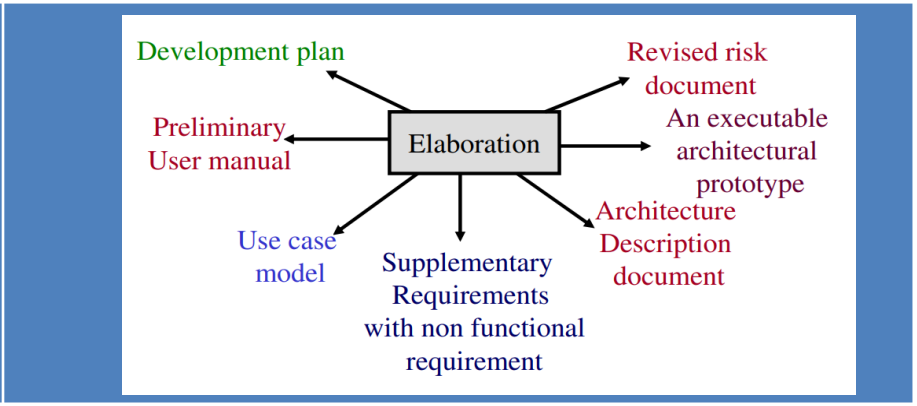


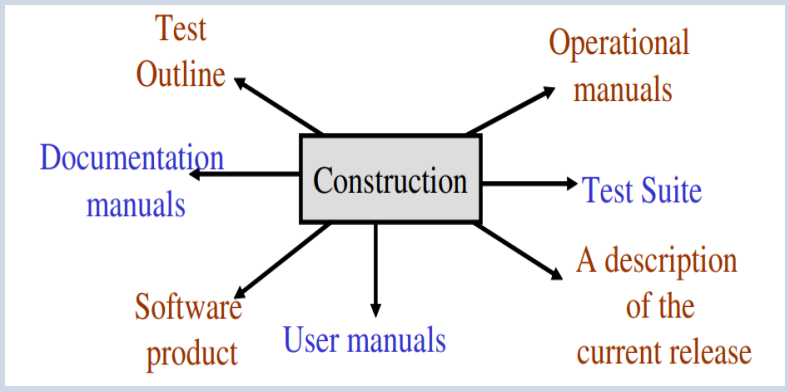
V1=version1, V2 =version2, V3=version3

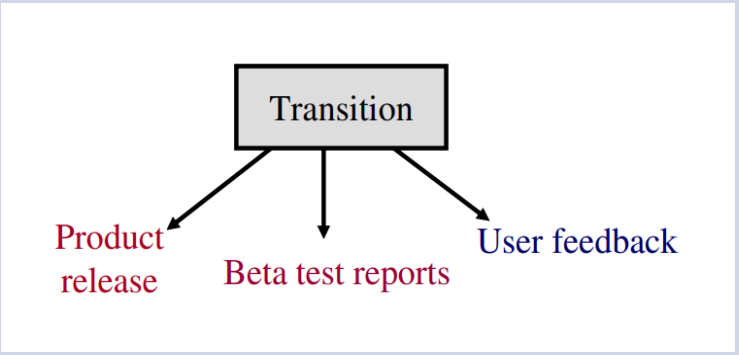
Iteration and Workflow of Unified Process




Prototypes
Project plan
Business model
Inception
Initial risk assessment
Initial business case
Vision document
Initial use case model
Initial project Glossary


Development plan
Revised risk document
Preliminary User manual
Elaboration
An executable architectural prototype
Architecture Description document
Use case model
Supplementary Requirements with non functional requirement


Test Outline
Operational manuals
Documentation manuals
Construction
Test Suite
A description of the current release
Software product
User manuals


Transition
Product release
Beta test reports
User feedback

Selection of a Life Cycle Model

Selection of a model is based on:

a) Requirements

b) Development team

c) Users

d) Project type and associated risk

Based On Characteristics Of Requirements

Requirements	Waterfall	Prototype	Iterative enhancement	Evolutionary development	Spiral	RAD
Are requirements easily understandable and defined?	Yes	No	No	No	No	Yes
Do we change requirements quite often?	No	Yes	No	No	Yes	No
Can we define requirements early in the cycle?	Yes	No	Yes	Yes	No	Yes
Requirements are indicating a complex system to be built	No	Yes	Yes	Yes	Yes	No

Based On Status Of Development Team

Development team	Waterfall	Prototype	Iterative enhancement	Evolutionary development	Spiral	RAD
Less experience on similar projects?	No	Yes	No	No	Yes	No
Less domain knowledge (new to the technology)	Yes	No	Yes	Yes	Yes	No
Less experience on tools to be used	Yes	No	No	No	Yes	No
Availability of training if required	No	No	Yes	Yes	No	Yes

Based On User's Participation

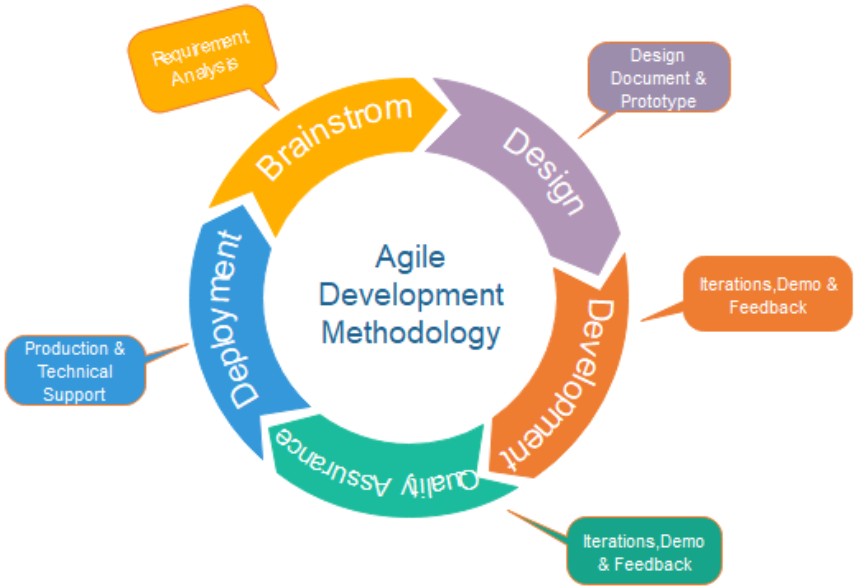
Involvement of Users	Waterfall	Prototype	Iterative enhancement	Evolutionary development	Spiral	RAD
User involvement in all phases	No	Yes	No	No	No	Yes
Limited user participation	Yes	No	Yes	Yes	Yes	No
User have no previous experience of participation in similar projects	No	Yes	Yes	Yes	Yes	No
Users are experts of problem domain	No	Yes	Yes	Yes	No	Yes

Based On Type Of Project With Associated Risk

Project type and risk	Waterfall	Prototype	Iterative enhancement	Evolutionary development	Spiral	RAD
Project is the enhancement of the existing system	No	No	Yes	Yes	No	Yes
Funding is stable for the project	Yes	Yes	No	No	No	Yes
High reliability requirements	No	No	Yes	Yes	Yes	No
Tight project schedule	No	Yes	Yes	Yes	Yes	Yes
Use of reusable components	No	Yes	No	No	Yes	Yes
Are resources (time, money, people etc.) scare?	No	Yes	No	No	Yes	No

Agile Model

- The Agile model is an iterative and incremental approach to software development that focuses on delivering working software quickly and efficiently.
- The Agile model involves continuous collaboration and feedback between the development team and the customer or end-users throughout the development process.



Agile Model- Steps

The Agile model consists of the following phases:

- Requirements gathering: In the Agile model, requirements are gathered in the form of user stories, which are short descriptions of a feature or functionality from the perspective of the end-user. These user stories are captured in a product backlog, which is a prioritized list of all the features or functionalities that need to be developed.
- Planning: In the planning phase, the development team and the customer prioritize the user stories based on business value, complexity, and other factors. The team estimates the time and effort required to complete each user story, and they plan the development iterations or sprints accordingly.
- Design: In the design phase, the development team creates a high-level design for the software application based on the requirements and user stories. The design may evolve over time as the team receives feedback and new requirements from the customer.
- Implementation: In the implementation phase, the development team works on completing the user stories that are planned for the current iteration or sprint. The team holds daily stand-up meetings to review progress, identify any issues or obstacles, and plan the work for the next day.
- Testing: In the Agile model, testing is integrated throughout the development process, rather than being a separate phase. The development team writes automated tests for each user story, and they conduct manual testing to ensure that the software meets the specified requirements.
- Review and feedback: At the end of each iteration or sprint, the development team holds a review meeting with the customer to demonstrate the working software and receive feedback. The customer provides feedback on the features and functionalities that have been completed, and they may request changes or additional features.
- Deployment: Once the software has been developed and tested, it is deployed to the production environment for end-users to use. The development team continues to provide maintenance and support for the software over its lifetime.

Characteristics of Agile Model

- The Agile model is based on the principles of flexibility, collaboration, and continuous improvement.
- It allows for changing requirements and priorities, and it emphasizes working software over documentation.
- The Agile model is particularly useful for complex projects where requirements may evolve over time, and where there is a need for regular feedback and collaboration between the development team and the customer.

Examples of software projects that may be developed using the Agile model include:

- Mobile applications: Mobile applications are typically developed in a fast-paced environment with rapidly evolving user requirements. The Agile model is well-suited for mobile app development as it allows for continuous feedback and regular updates to the app.
- E-commerce platforms: E-commerce platforms require frequent updates to accommodate changes in customer behavior and new technologies. The Agile model is ideal for e-commerce platform development as it allows for flexibility in accommodating changing customer needs.
- Web-based software applications: Web-based software applications are typically developed in a fast-paced environment with rapidly evolving user requirements. The Agile model is well-suited for web-based application development as it allows for continuous feedback and regular updates to the application.
- Gaming software: Gaming software requires frequent updates to accommodate changing user preferences and advancements in technology. The Agile model is ideal for gaming software development as it allows for flexibility in accommodating changing user needs.
- Cloud-based software: Cloud-based software requires frequent updates to accommodate changing user needs and new technologies. The Agile model is well-suited for cloud-based software development as it allows for continuous feedback and regular updates to the software.

Which software have been developed using Agile model

Spotify: The popular music streaming platform, Spotify, is known for using Agile methodologies in their software development process. They use the Agile model to develop their mobile and web applications, as well as their backend systems.

Microsoft Office: Microsoft uses the Agile model in their software development process for many of their products, including Microsoft Office. They use the Agile model to quickly deliver new features and improvements to their users.

Salesforce: Salesforce, the popular customer relationship management (CRM) software, uses Agile methodologies to develop their software products. They use the Agile model to deliver new features and enhancements to their users on a regular basis.

Airbnb: Airbnb, the online marketplace for short-term accommodations, uses Agile methodologies to develop their software products. They use the Agile model to quickly deliver new features and improvements to their users.

Amazon Web Services (AWS): Amazon uses Agile methodologies to develop their cloud-based software products, including Amazon Web Services (AWS). They use the Agile model to deliver new features and enhancements to their users on a regular basis.

What are different agile software methodologies?

There are several different Agile methodologies that can be used for software development. Some of the most popular Agile methodologies include:

- Scrum: Scrum is a popular Agile methodology that emphasizes teamwork, collaboration, and iterative development. In Scrum, development work is divided into small iterations called sprints, which typically last 2-4 weeks. During each sprint, the development team works to deliver a small, working piece of software that can be tested and evaluated by the customer.
- Kanban: Kanban is another popular Agile methodology that emphasizes continuous delivery and visual management. In Kanban, work is organized on a visual board, which allows the team to see the status of each task and identify any bottlenecks or issues that need to be addressed. The goal of Kanban is to minimize work in progress and optimize the flow of work through the development process.
- Extreme Programming (XP): Extreme Programming is an Agile methodology that emphasizes the importance of quality software development. In XP, development work is organized into small, iterative cycles, and the development team works closely with the customer to ensure that the software meets their needs. XP also emphasizes the use of automated testing and continuous integration to ensure that the software is of high quality.
- Lean Development: Lean Development is an Agile methodology that emphasizes the importance of reducing waste and optimizing the development process. In Lean Development, work is organized into small batches, and the development team works to continuously improve the development process and eliminate any waste or inefficiencies.
- Crystal: Crystal is an Agile methodology that emphasizes the importance of communication, collaboration, and simplicity. Crystal is designed to be flexible and adaptable, and it can be customized to fit the needs of different software development projects.

Scrum Model

- Scrum is an Agile methodology for software development that emphasizes teamwork, collaboration, and iterative development.
- The Scrum model is based on the idea of breaking development work into small iterations called sprints, which typically last 2-4 weeks.
- During each sprint, the development team works to deliver a small, working piece of software that can be tested and evaluated by the customer.
- The scrum team tracks the progress of the project in 15-minute time-boxed meetings called daily Scrum.
- After sprint completion, the team holds a review meeting to demonstrate the work done and a retrospective to continually improve.

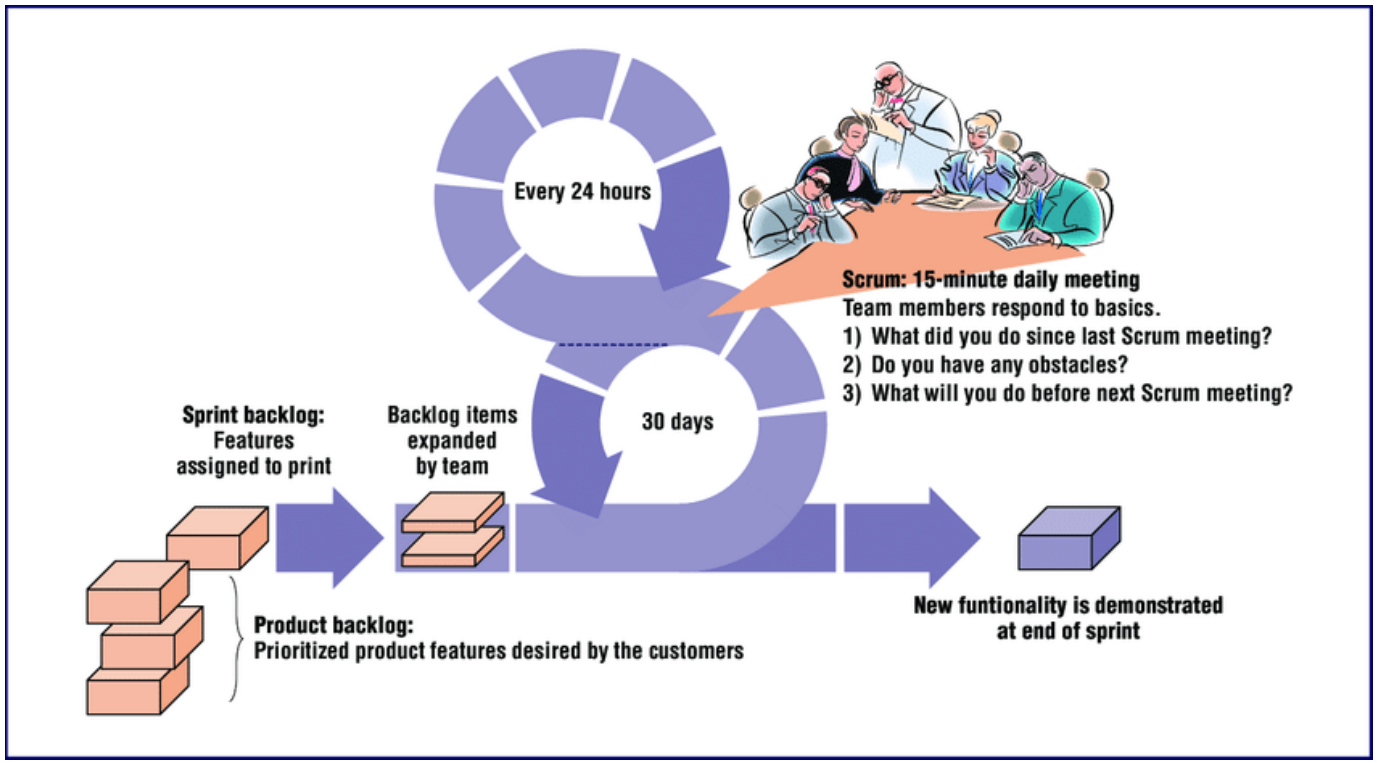
Scrum Roles

The core roles involved in the Scrum process are Product Owner, Scrum Master, and development team.

•**Product owner** - The product owner in the Scrum team is primarily responsible for working with the user group to determine the features to be included in the product release.

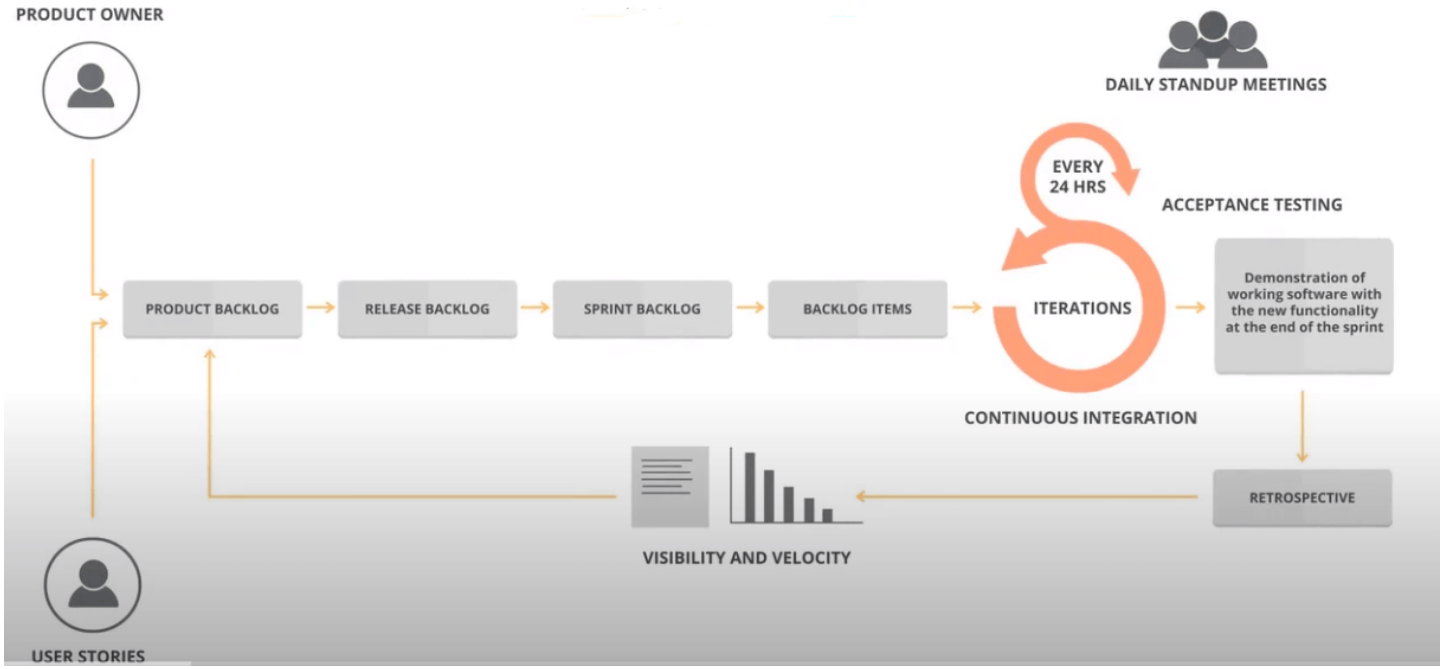
•**Scrum Master** - The Scrum Master in the team is responsible for promoting and supporting Scrum. They guide the team, product owner, and business on Scrum and look for ways to fine-tune their practice. An effective scrum master helps the team in optimizing their transparency and delivery flow.

•**Development Team** - The development teams are cross-functional and have the skills to deliver product increments. This team includes developers, testers, designers, etc., to have less dependency on the third party. All the members of the scrum team are self-organizing and ensure successful sprint completion by turning Product Backlog into increments of potentially releasable functionality.


Every 24 hours
Scrum: 15-minute daily meeting
Team members respond to basics.
1) What did you do since last Scrum meeting?
2) Do you have any obstacles?
3) What will you do before next Scrum meeting?
30 days
Sprint backlog:
Features
assigned to print
Backlog items
expanded
by team
Product backlog:
Prioritized product features desired by the customers
New funtionality is demonstrated
at end of sprint

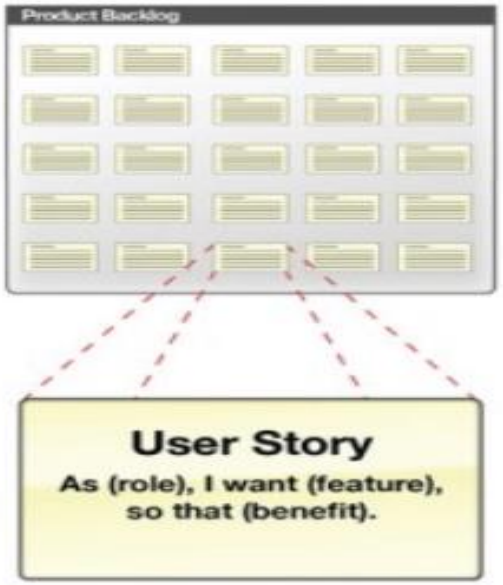
Scrum Workflow Steps

- In Scrum workflow, the process is regularly customized based on the continuous feedback loops.
- The below diagram depicts the step-by-step process involved in scrum workflow:



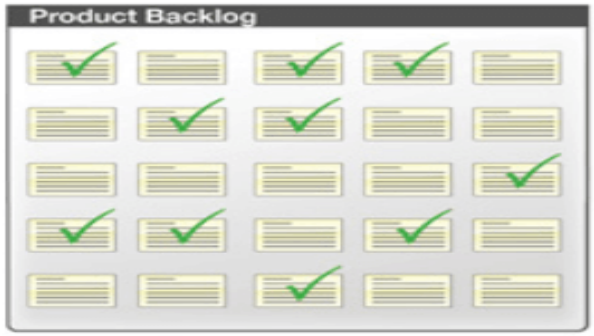
1. Product Backlog Creation

- The first phase of the Scrum workflow process begins with the visioning phase, where the Stakeholders meet to decide the list of features that should be implemented and develop a product roadmap.
- A product backlog is a prioritized list of all the product requirements or user stories that a scrum team maintains for a product.
- In Scrum, features are known as user stories and are written from the end-user perspective. A Product owner decides which user stories or items make it into the product backlog.



2. Release Backlog

- Based on the product roadmap developed, in collaboration with the product owner, the team decides how to group user stories into releases. The objective of the release is to deliver a subset of the product backlog known as the release backlog.
- After determining which user stories will go into a particular release, the development team estimates the time duration needed to complete each item. Once the release planning has been completed, the user stories are then selected for a sprint.



3. Sprint Backlog Creation

- A Sprint is a predefined timeframe within which the team performs a set of tasks from the Backlog.
- The duration of each Sprint lasts 2-4 weeks.
- Each Sprint takes a manageable chunk of the release backlog and gets it a ship-steady state.
- A set of product backlog items that must be delivered within a single sprint iteration is called a Sprint backlog.
- Once the sprint backlog is determined, the team then divides each user story into a task. And then in each Sprint, the product is developed

4. Working on Sprint and Scrum Meetings

- After the user stories for the current phase are selected, the development process begins. For tracking the current working process, a task board is commonly used, which represents particular user stories with a description of tasks needed for implementation.

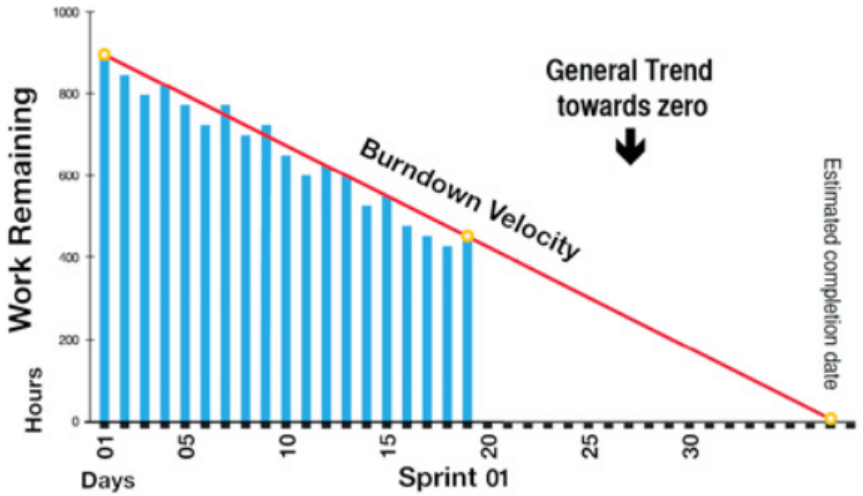
- Daily scrums or daily stand-ups are conducted by development teams to monitor the progress made towards the Sprint Goal and progress performed in the Sprint Backlog, to adjust the plan for the rest of the Sprint.

- Each of the meetings is held timeboxed in about 15 minutes. The main goal of these meetings is to get accurate information about the current project status.

5. **Burndown Charts**

The progress of the team is tracked through a burndown chart. It provides a day-by-day measure of the work that remains in a given sprint or release.

The slope of the graph (burndown velocity) is calculated by comparing the number of hours worked to the original project estimation and displays the average rate of productivity for each day.



6. **Testing and Product Demonstration**

- If all the user stories are completed then the sprint backlog is also completed, which means a sprint completion.
- A sprint review is held after sprint completion, where the working software is demonstrated and presented for acceptance to the customers.

7. **Retrospective and Next Sprint Planning**

Finally, the team then conducts a sprint retrospective to ask themselves questions about what could be done better to improve themselves. The team primarily looks at three things during a retrospective:

- What went well?
- What did not go well?
- What should be done differently?

The time duration of retrospectives lasts max of 90 minutes. They help us to incorporate continuous changes into sprint cadence and team.

At last, the team velocity is updated and it acts as information radiators to display the status and progress of the project, which again make their way back to user stories, and then the whole cycle repeats until the project is completed.

Example of User Story in Scrum

Employee Management System

As a HR manager, I want to be able to use an employee management system to:

1.Create new employee profiles with their personal information, job title, and start date.
2.View and update employee information such as contact details, job details, salary details, and other relevant information.
3.Assign tasks and projects to employees and track their progress.
4.Generate reports on employee performance, attendance, and other relevant metrics.
5.Manage employee leaves, including vacation time, sick leave, and other types of leaves.
6.Store and access employee documents such as contracts, resumes, and performance reviews.
7.Receive notifications and alerts for employee birthdays, work anniversaries, and other important events.
8.Allow employees to access and update their own personal information, request leaves, and view their tasks and projects.
9.Ensure the system is secure and complies with data privacy regulations.

By having an employee management system in place, I can more efficiently manage our company's human resources and make informed decisions based on real-time data.

Scrum Model- Example

Example of how the Scrum model might be used in software development:

Sprint planning: At the beginning of each sprint, the development team and the product owner meet to plan the work for the sprint. They review the product backlog, which is a prioritized list of features or user stories that the customer wants in the software. The development team then selects the highest priority items from the product backlog that they think they can complete during the sprint.

Daily stand-up meetings: Throughout the sprint, the development team holds daily stand-up meetings to check in on their progress and identify any issues or roadblocks that need to be addressed. The meetings are typically brief and focused, and each team member answers three questions: What did I accomplish yesterday? What will I work on today? Do I have any blockers or issues that need to be addressed?

Sprint review: At the end of each sprint, the development team presents their work to the product owner and any other stakeholders who are interested. The team demonstrates the working software that they completed during the sprint, and the product owner provides feedback on the features or user stories that were completed.

Sprint retrospective: After the sprint review, the development team holds a retrospective meeting to reflect on the sprint and identify areas for improvement. They discuss what went well during the sprint, what didn't go well, and any ideas for how to improve their processes or communication.

Repeat: The development team then starts the next sprint, using the feedback and insights gained from the retrospective meeting to improve their work.

By using the Scrum model, the development team can quickly deliver working software that meets the customer's needs. The model emphasizes collaboration, flexibility, and continuous improvement, which can lead to faster and more effective software development.

Advantage (Pros) of Agile Method

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

Disadvantages (Cons) of Agile Model

1. Due to the lack of formal documents, it creates confusion and vital decisions taken throughout various phases can be misunderstood at any time by different team members.

2. Due to the lack of proper documentation, once the project completes and the developers allotted to another project, maintenance of the finished project can become a difficulty.