



# SOFTWARE ENGINEERING

## MODULE I

Dr.Preethi  
Assistant Professor  
School of CS and IT

# Module 1

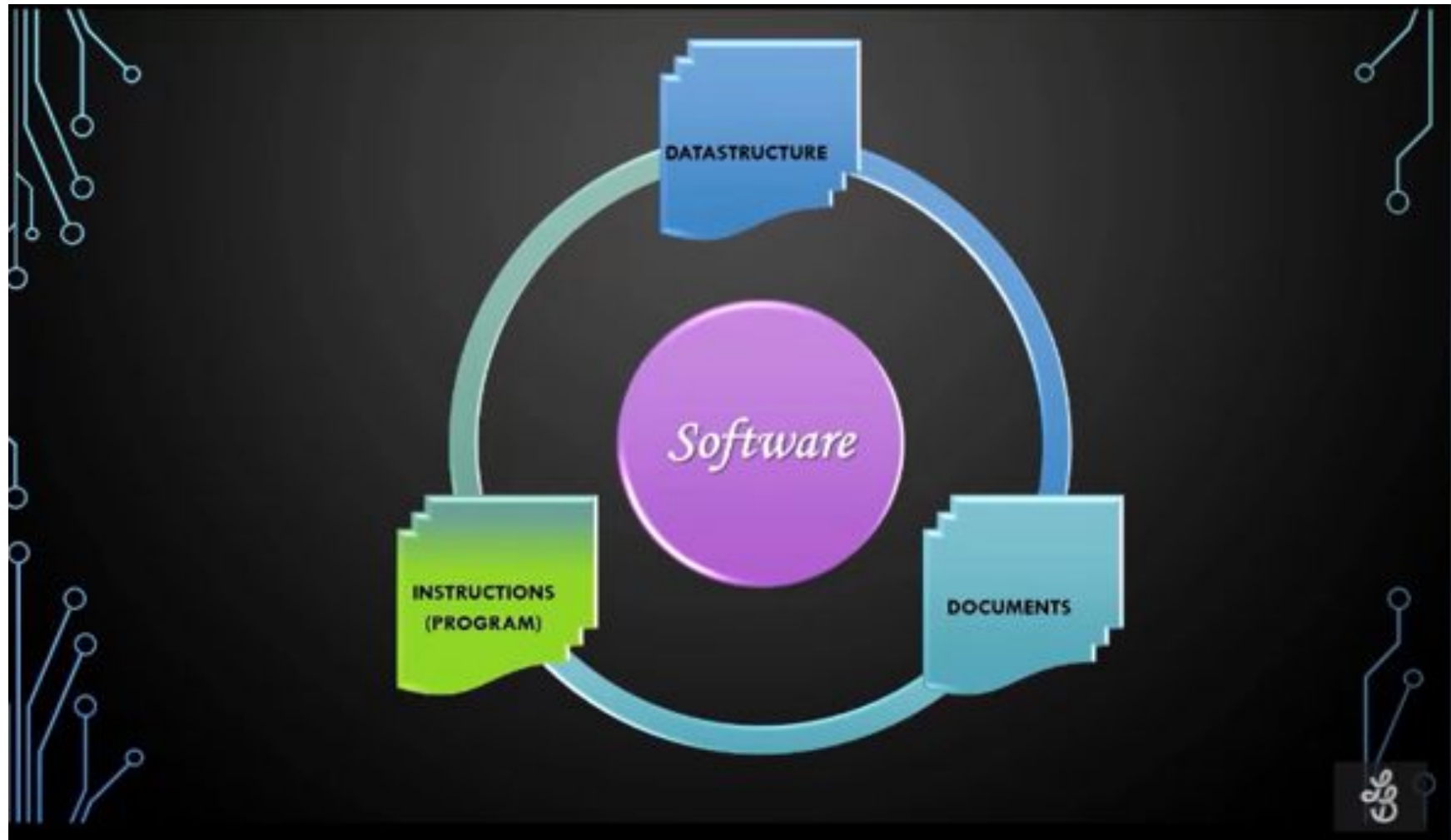
Software Product and Process: Defining Software, Categories of Software, Legacy Software, Changing Nature of Software, Defining Software Engineering, Software Process Framework, SDLC, Prescriptive Process Models: Waterfall, V-Model, Incremental Model, Prototyping, Spiral Model; Agile Development: Agile Process, Extreme Programming, Scrum; Characteristics of a Software Engineer, Software Team

## OVERVIEW

- *WHAT IS SOFTWARE?*
- *COMPONENTS OF SOFTWARE*
- *TYPES OF SOFTWARE*
- *CHARACTERISTICS OF SOFTWARE*
- *ROLE OF SOFTWARE*
- *WHY WE NEED SOFTWARE?*
- *WHAT IS SOFTWARE ENGINEERING?*

# *WHAT IS SOFTWARE?*





## ESSENTIAL COMPONENTS OF SOFTWARE

- *INSTRUCTIONS:*

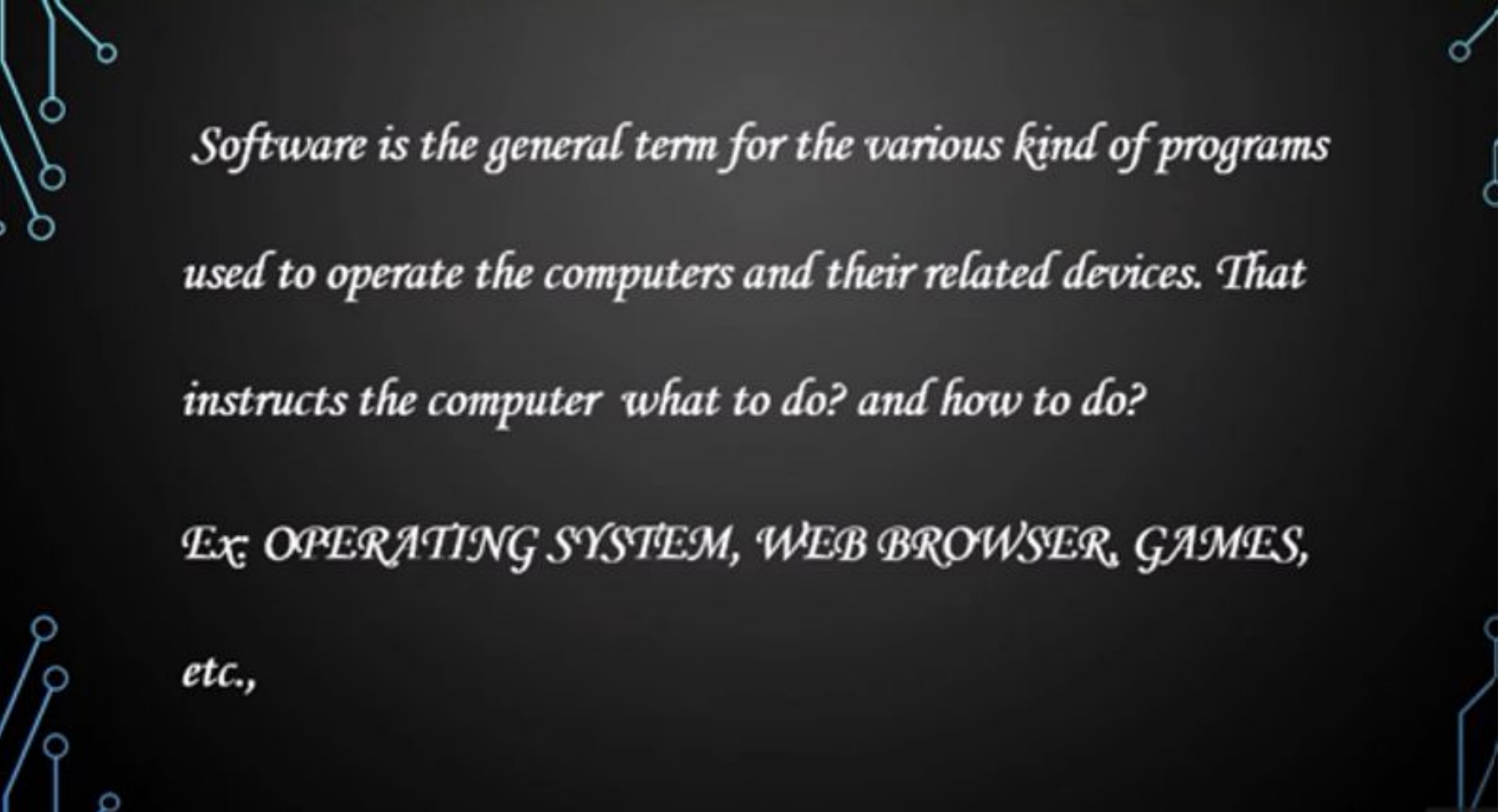
- *functionality*
- *performance*

- *DATA STRUCTURES:*

- *Essential components*
- *Maintains the data*
- *Program logic*
- *Design*

- *DOCUMENTS:*

- *user manual*
- *design manual*

The slide features a dark grey background with decorative light blue circuit-like lines in the corners. These lines consist of small circles connected by thin lines, resembling a stylized electronic circuit.

*Software is the general term for the various kind of programs used to operate the computers and their related devices. That instructs the computer what to do? and how to do?*

*Ex: OPERATING SYSTEM, WEB BROWSER, GAMES,  
etc.,*

# *TYPES OF SOFTWARE*



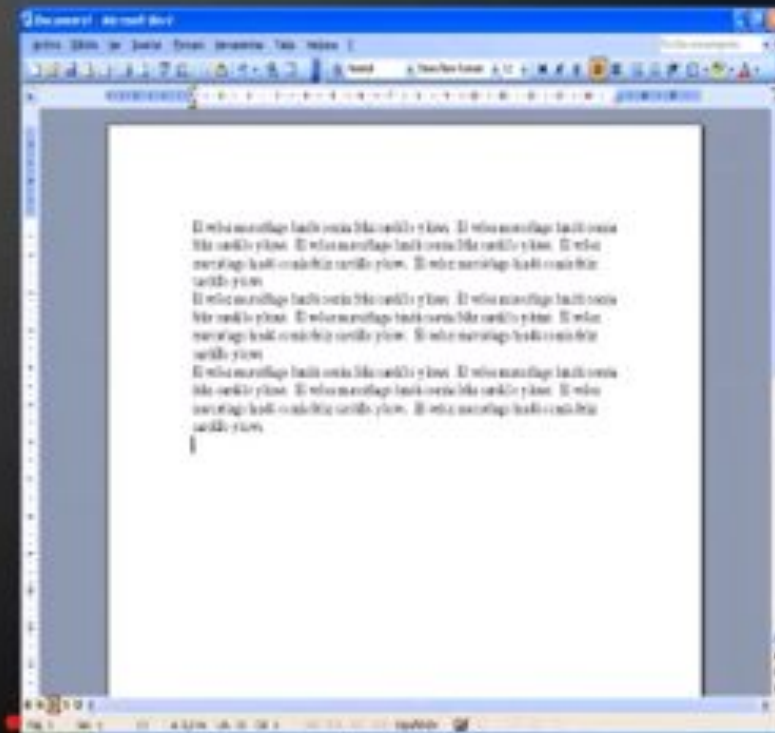


## *TYPES OF SOFTWARE*

- *Based on functional Domain,*
  - (i) *System software*
  - (ii) *Real-time software*
  - (iii) *Business software*
  - (iv) *Scientific and Engineering software*
  - (v) *Embedded software*
  - (vi) *Personal computer software*
  - (viii) *Web based software*

# SYSTEM SOFTWARE

- *Provide interface to the other applications*
- *Hide the complexity from higher level application*



# REAL-TIME SOFTWARE

- *Input data*
- *Analyze the data*
- *Take appropriate action*

*Ex: Air traffic control system, Flood control*

*System, network management system etc.,*



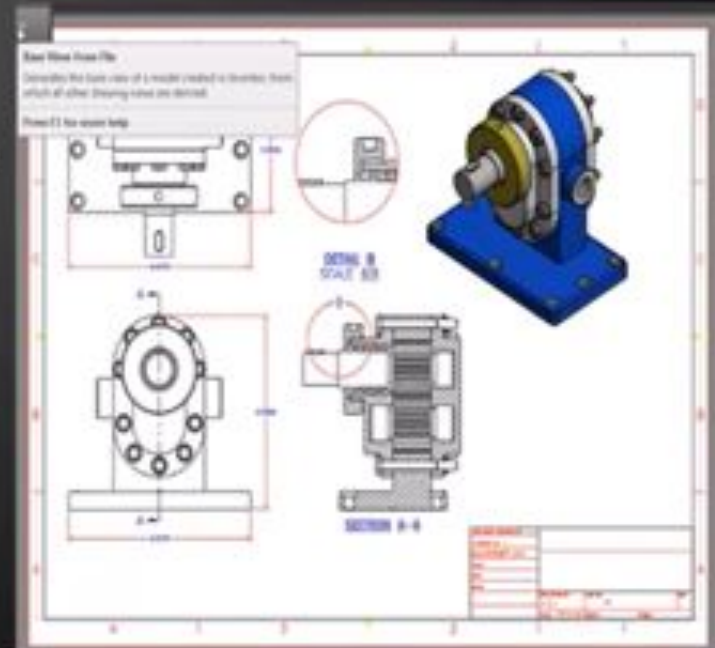
## BUSINESS SOFTWARE

- *Software use is most prevalent in this application.*
- *Ex: payroll system, sales analyze system etc.,*



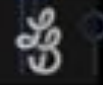
# SCIENTIFIC AND ENGINEERING SOFTWARE

- *Scientific nature of the computation they perform.*
- *Ex: AutoCAD, MATLAB, etc.,*



## EMBEDDED SOFTWARE

- *Resides in Read only memory*
- *It gives products an intelligent look*
- *Ex: ATMs, cell phones, printers, thermostats, calculators, etc.,*





## PERSONAL COMPUTER SOFTWARE

- *It is prolific application domain*
- *It is a multi-purpose computer whose size, capabilities, and price make it feasible for individual use.*
- *PCs are intended to be operated directly by an end user, rather than by a computer expert or technician.*
- *Ex: Spread sheet, multimedia software, etc.,*



## WEB BASED SOFTWARE

- *It is software we use over the internet with a web browser.*
- *We don't have to install anything, download any software, or worry about upgrades.*
- *If we use an online bank or web-based email program like Gmail, Hotmail, or Yahoo Mail, we are already using web-based software.*



# CHARACTERISTICS OF SOFTWARE

## *CHARACTERISTICS OF SOFTWARE*

- *Maintainability*
- *Correctness*
- *Reusability*
- *Reliability*
- *Portability*
- *Efficiency*

## MAINTAINABILITY



- *Allows organizations to identify improvement areas as well as determine the value supplied by current applications or during development changes.*

## CORRECTNESS



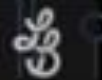
- *The degree with which software adheres to its specified requirements*



## REUSABILITY



- *The ease with which software can be reused in developing other software*
- *The use of existing assets in some form within the software product development process.*
- *Assets are products and by-products of the software development life cycle and include code, software components, test suites, designs and documentation.*





# RELIABILITY



- *The frequency and criticality of software failure, where failure is an unacceptable effect or behavior occurring under permissible operating conditions.*
- *The probability of failure-free software operation for a specified period of time in a specified environment.*

## PORTABILITY



- *Measure of how easily an application can be transferred from one computer environment to another.*
- *A computer software application is considered portable to a new environment if the effort required to adapt it to the new environment is within reasonable limits.*
- *The meaning of the abstract term 'reasonable' depends upon the nature of the application and is often difficult to express in quantifiable units.*

## EFFICIENCY



- *Ability to avoid wasting materials, energy, efforts, money, and time in doing something or in producing a desired result.*
- *In a more general sense, it is the ability to do things well, successfully, and without waste.*



## ROLE OF SOFTWARE

- *Integration technology*
- *Captures application functionality*
- *Defines lot of system behavior*
- *Determines how much of potential system performance is achieved*
- *Acts as director*



## WHY WE NEED SOFTWARE

- *Each device needs at least one corresponding device driver; because a computer typically has at minimum at least one input device and at least one output device, a computer typically needs more than one device driver.*

# Changing Nature of Software

# SOFTWARE ENGINEERING

- *Software engineering is the application of principles used in the field of engineering, which usually deals with physical systems, to the design, development, testing, deployment and management of software systems.*

- The field of software engineering applies the *disciplined, structured approach to programming* that is used in engineering to software development with the stated *goal* of improving the quality, time and budget efficiency, along with the assurance of structured testing and engineer certification.
- Software engineering is typically used for large and intricate software systems rather than single *applications* or *programs*.

# What is 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.
- **Software engineering** is the establishment and use of sound engineering principles in order to obtain economically software that is reliable and works efficiently on real machines.
- **Software engineering** is also:-
  - A modeling activity
  - A problem-solving activity
  - A knowledge acquisition activity
  - A rationale-driven activity

□ **Software engineers** should adopt

- Systematic and organized approach to their work
- Use appropriate tools and techniques depending on the problem to be solved
- The development constraints and the resources available

□ **Software Engineering** is the science and art of building (designing and writing programs) a software

systems that are:

- On time
- On budget
- With acceptable performance
- With correct operation

# A Layered Technology





# A Layered Technology

- ❑ **Software Engineering** is a layered technology.

## **Quality focus:-**

- ❑ Main principle of Software Engineering is Quality focus.
- ❑ An engineering approach must have a focus on quality.
- ❑ Total Quality Management(TQM), Six Sigma, ISO 9001, ISO 9000-3 , Capability Maturity Model(CMM) , CMMI & similar approaches encourages a continuous process improvement culture.

## **Process :-**

- ❑ The foundation for software engineering is the process layer.
- ❑ Software engineering process is the glue that holds the technology layers together.
- ❑ Process defines a framework activities for effective delivery of software engineering technology.

# Methods

:-

- Software engineering methods provide the technical how-to's for building software.
- Methods encompass a broad array of tasks that include requirements analysis, design, program construction, testing, and support.
- Software engineering methods rely on a set of basic principles that govern each area of the technology and include modeling activities and other descriptive techniques.

## Tools :-

- Software engineering tools provide automated or semi-automated support for the process and the methods.
- When tools are integrated so that information created by one tool can be used by another, a system for the support of software development, called computer-aided software engineering, is established.
- CASE combines software, hardware, and a software engineering database (a repository containing important information about analysis, design, program construction, and testing) to create a software engineering environment analogous to CAD/CAE (computer-aided design/engineering) for hardware.

# Categories of Software

- **System software**—a collection of programs written to service other programs. Some system software (e.g., compilers, editors, and file management utilities) processes complex, but determinate, information structures. Other systems applications (e.g., operating system components, drivers, networking software, telecommunications processors) process largely indeterminate data.
- **Application software**—stand-alone programs that solve a specific business need. Applications in this area process business or technical data in a way that facilitates business operations or management/technical decision making.
- **Engineering/scientific software**—has been characterized by “number crunching” algorithms. Applications range from astronomy to volcanology, from automotive stress analysis to space shuttle orbital dynamics, and from molecular biology to automated manufacturing.
- **Embedded software**—resides within a product or system and is used to implement and control features and functions for the end user and for the system itself.
- **Product-line software**—designed to provide a specific capability for use by many different customers. Product-line software can focus on a limited and esoteric marketplace or address mass consumer markets.
- **Web applications**—called “WebApps,” this network-centric software category spans a wide array of applications. In their simplest form, WebApps can be little more than a set of linked hypertext files that present information using text and limited graphics.
- **Artificial intelligence software**—makes use of nonnumerical algorithms to solve complex problems that are not amenable to computation or straightforward analysis. Applications within this area include robotics, expert systems, pattern recognition (image and voice), artificial neural networks, theorem proving, and game playing.

# Legacy Software

These older programs—often referred to as legacy software. Legacy software systems . were developed decades ago and have been continually modified to meet changes in business requirements and computing platforms. Legacy software is characterized by longevity and business criticality.

Unfortunately, there is sometimes one additional characteristic that is present in legacy software—poor quality. Legacy systems sometimes have inextensible designs, convoluted code, poor or nonexistent documentation, test cases and results.

- Legacy systems often evolve for one or more of the following reasons:
  - The software must be adapted to meet the needs of new computing environments or technology.
  - The software must be enhanced to implement new business requirements.
  - The software must be extended to make it interoperable with other more modern systems or databases.
  - The software must be re-architected to make it viable within a network environment.

# Software process framework

## A software process:-

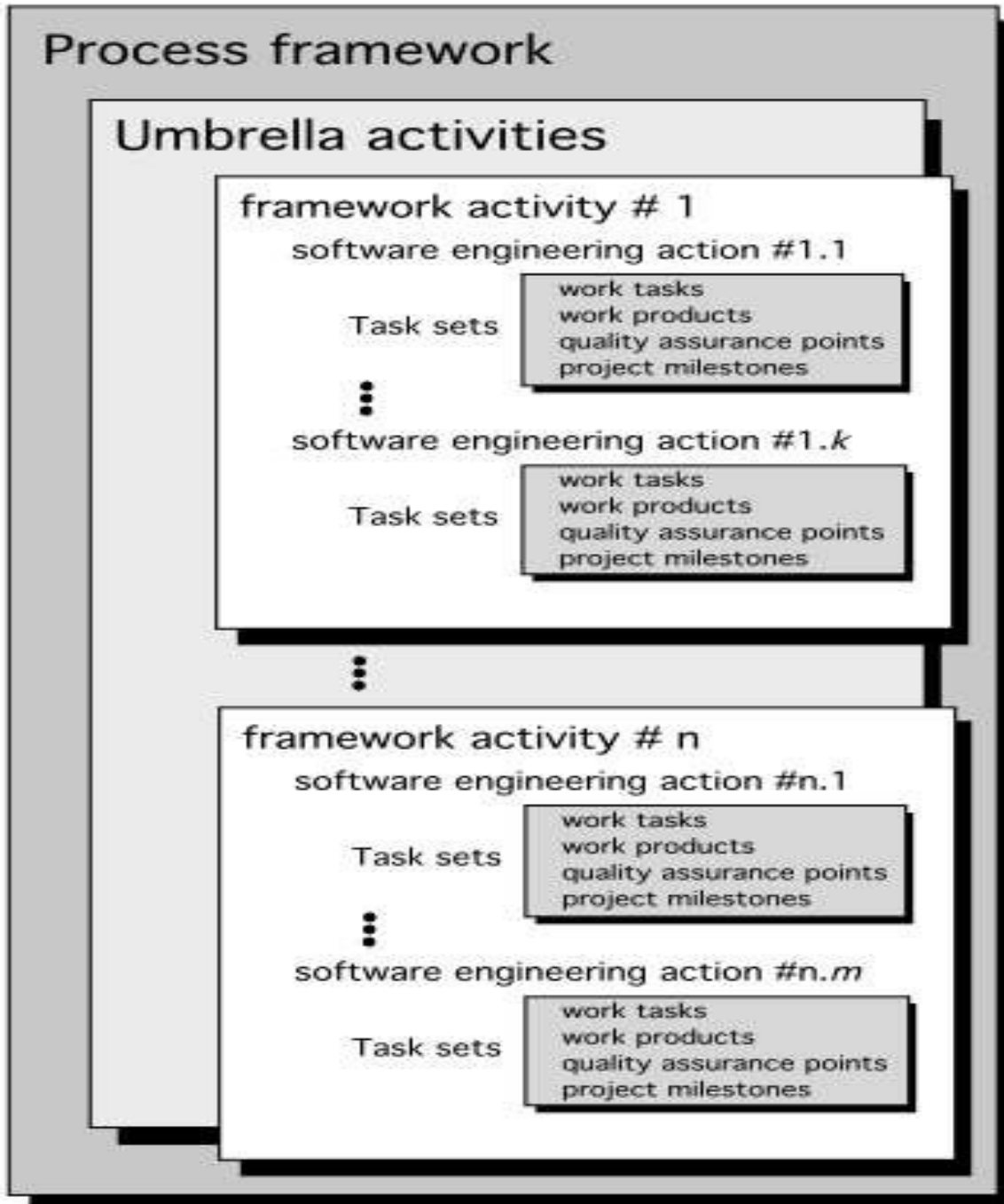
- is a roadmap to building high quality software products.
- provides a framework for managing activities.
- adapts to meet needs of software engineers and managers.

## What is a process?

- Sequence of steps required to develop or maintain software.

## Characteristics

- prescribes major activities
- constraints and controls apply to activities, resources, and products
- utilizes resources, subject to constraints such as schedule, to produce intermediate and final results
- constraints on activities: time, budget, *tools*



Process framework

Framework activities

work tasks

work products

milestones & deliverables

QA checkpoints

Umbrella Activities

# Common Process Framework Activities

## COMMUNICATION



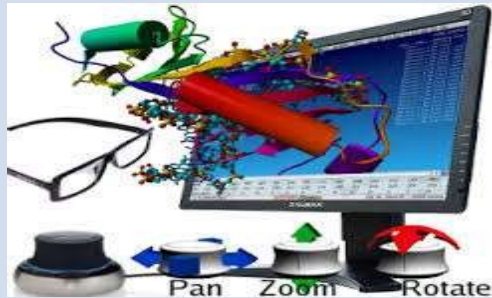
Communication with customers/ stakeholders to understand project requirements for defining software features.

## PLANNING



Software Project Plan which defines workflow that is to follow. It describes technical task, risks, resources, product to be produced & work schedule.

## MODELING



Creating models to understand requirements and shows design of software to achieve requirements.

## CONSTRUCTIONS



Code generation (manual or automated) & testing (to uncover errors in the code)

## DEPLOYMENT



Deliver Software to Customer collect feedback from customer based on evaluation Software Support

# Umbrella Activities

## **Software project tracking and control**

- ☐ Assessing progress against the project plan.
- ☐ Take adequate action to maintain schedule.

## **Formal technical reviews**

- ☐ Assessing software work products in an effort to uncover and remove errors before goes into next action or activity.

## **Software quality assurance**

- ☐ Define and conducts the activities required to ensure software quality.

## **Software configuration management**

- ☐ Manages the effects of change.

## **Document preparation and production**

- ☐ Help to create work products such as models, documents, logs, form and list.



## **Reusability management**

- ☐ Define criteria for work product reuse
- ☐ Mechanisms to achieve reusable components.

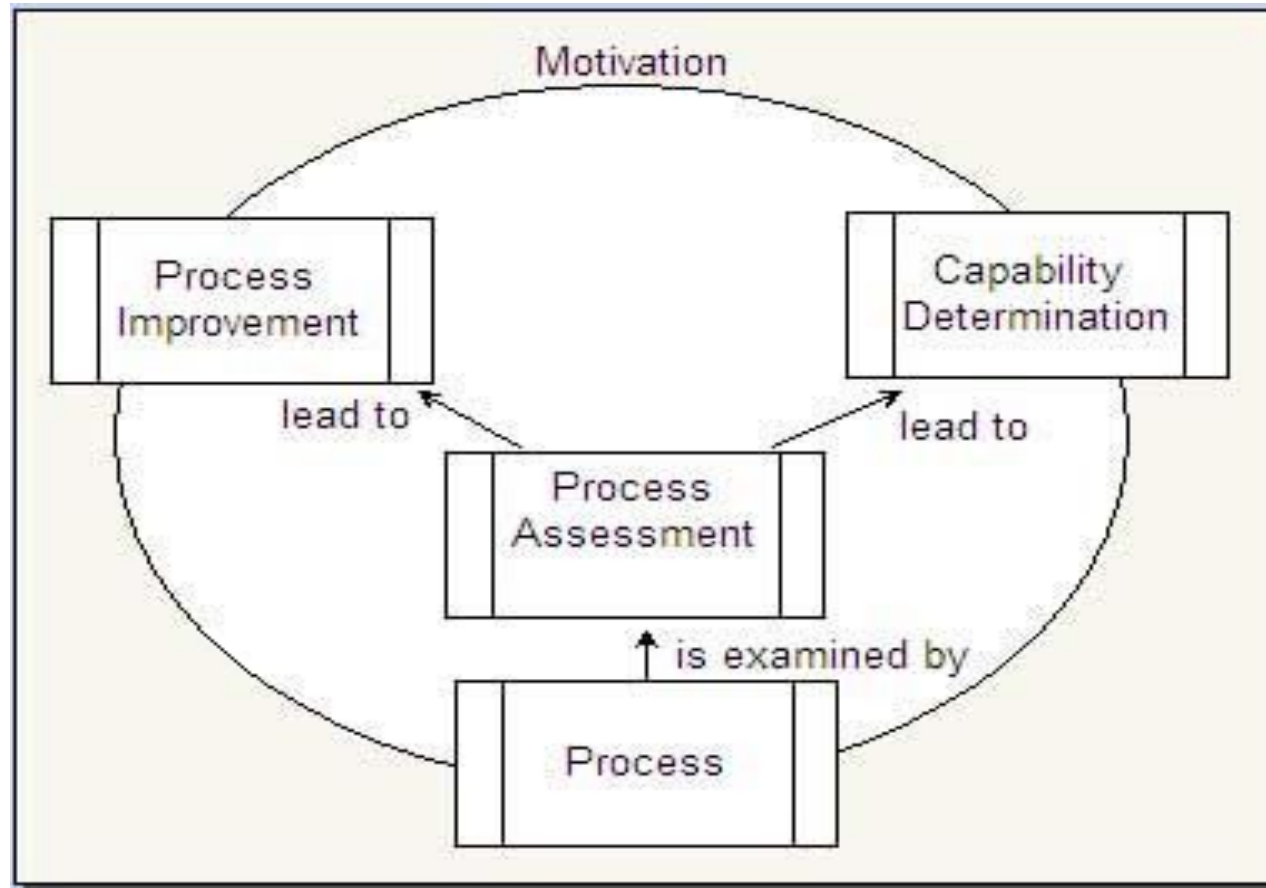
## **Measurement**

- ☐ Define and collects process, project, and product measures
- ☐ Assist the team in delivering software that meets customer's needs.

## **Risk management**

- ☐ Assesses risks that may effect that outcome of project or quality of product (i.e. software)

## 1. PROCESS ASSESSMENT AND IMPROVEMENT



## • Process Assessment and Improvement

1. **Standard CMMI Assessment Method for Process Improvement (SCAMPI)** — provides a five step process assessment model that incorporates five phases: initiating, diagnosing, establishing, acting and learning.
2. SCAMPI is used for process improvement by gaining insight into the process capability in the organization.
3. The major advantage of SCAMPI is that it supports process improvement




\*

- Process Assessment and Improvement


1. **CMM-Based Appraisal for Internal Process Improvement**

**(CBA IPI)**—CBA IPI is a diagnostics, enables, and encourages an organization's commitment to **process improvement**. It provides a diagnostic technique for assessing the relative maturity of a software organization; uses the SEI CMM as the basis for the assessment




- 
1. **SPICE**—The SPICE (ISO/IEC15504) **SPICE** (Software Process Improvement and Capability Determination) in **software engineering** standard defines a set of requirements for software process assessment. SPICE provides a framework for assessing the software process and is used by the organizations involved in
  2. planning,
  3. monitoring,
  4. developing,
  5. managing,
  6. and improving acquisitions.




- 
1. It is carried out in accordance with the **International Organization for Standardization (ISO)** and **International Electro-technical Committee (IEC)**, which are used together and known as **ISO/IEC 15504**.
  2. **ISO 9001:2000 for Software**—a generic standard that applies to any organization that wants to improve the overall quality of the products, systems, or services that it provides. Therefore, the standard is directly applicable to software organizations and companies



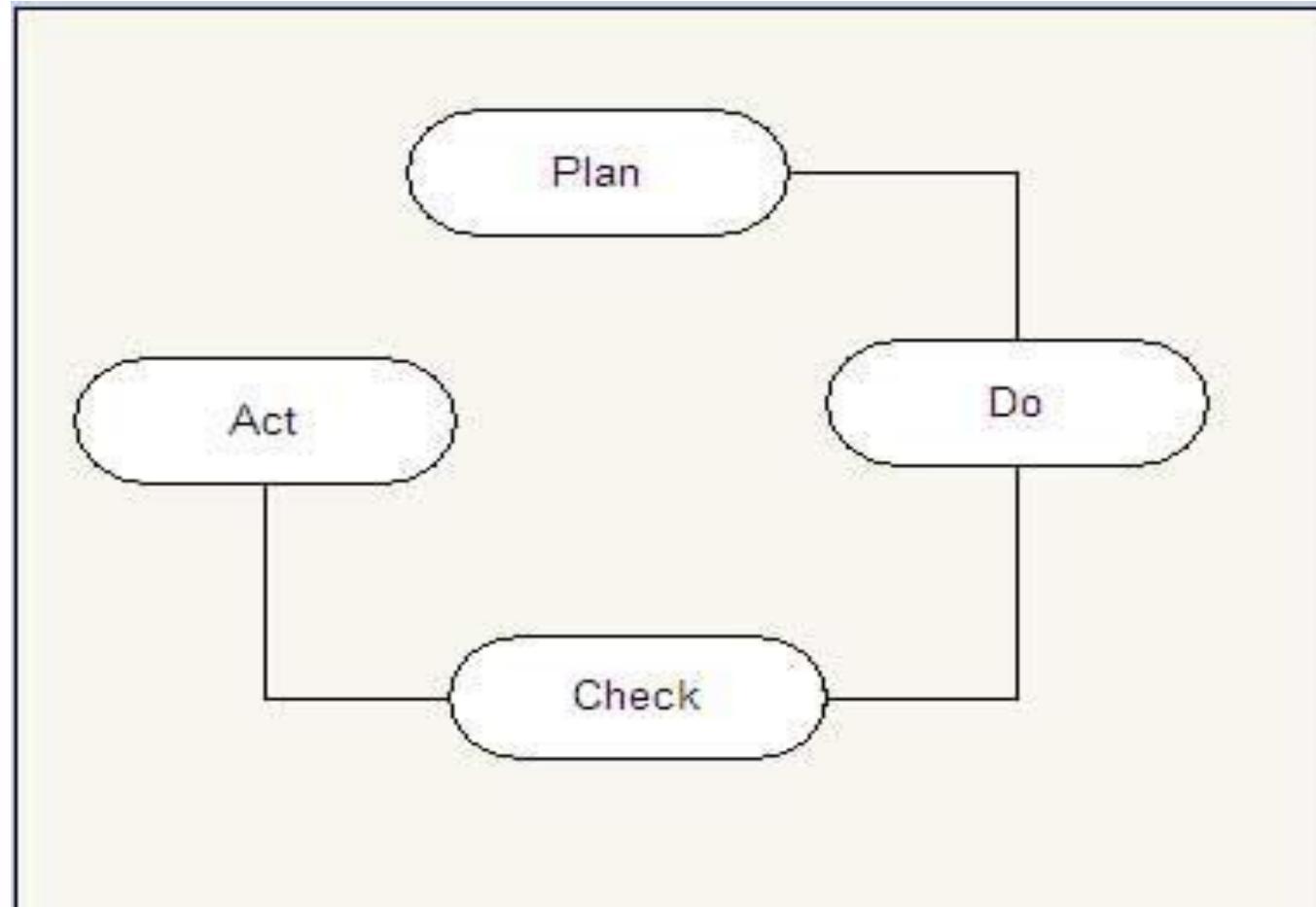
- 
1. ISO (International Organization for Standardization) established a standard known as ISO 9001:2000 to determine the requirements of quality management systems.
  2. A **quality management system** refers to the activities within an organization, which satisfies the quality related expectations of customers



- 
1. This standard follows a **plan-do-check-act (PDCA)** cycle, which includes a set of activities that are listed below.
  2. **Plan: Determines the processes and resources** which are required to **develop a quality product** according to the **user's satisfaction**.
  3. **Do: Performs activities** according to the plan to create the **desired product**.
  4. **Check: Measures** whether the **activities** for establishing **quality management** according to the **requirements** are accomplished.
  5. **Act: Initiates activities** which constantly **improve processes** in the organization.







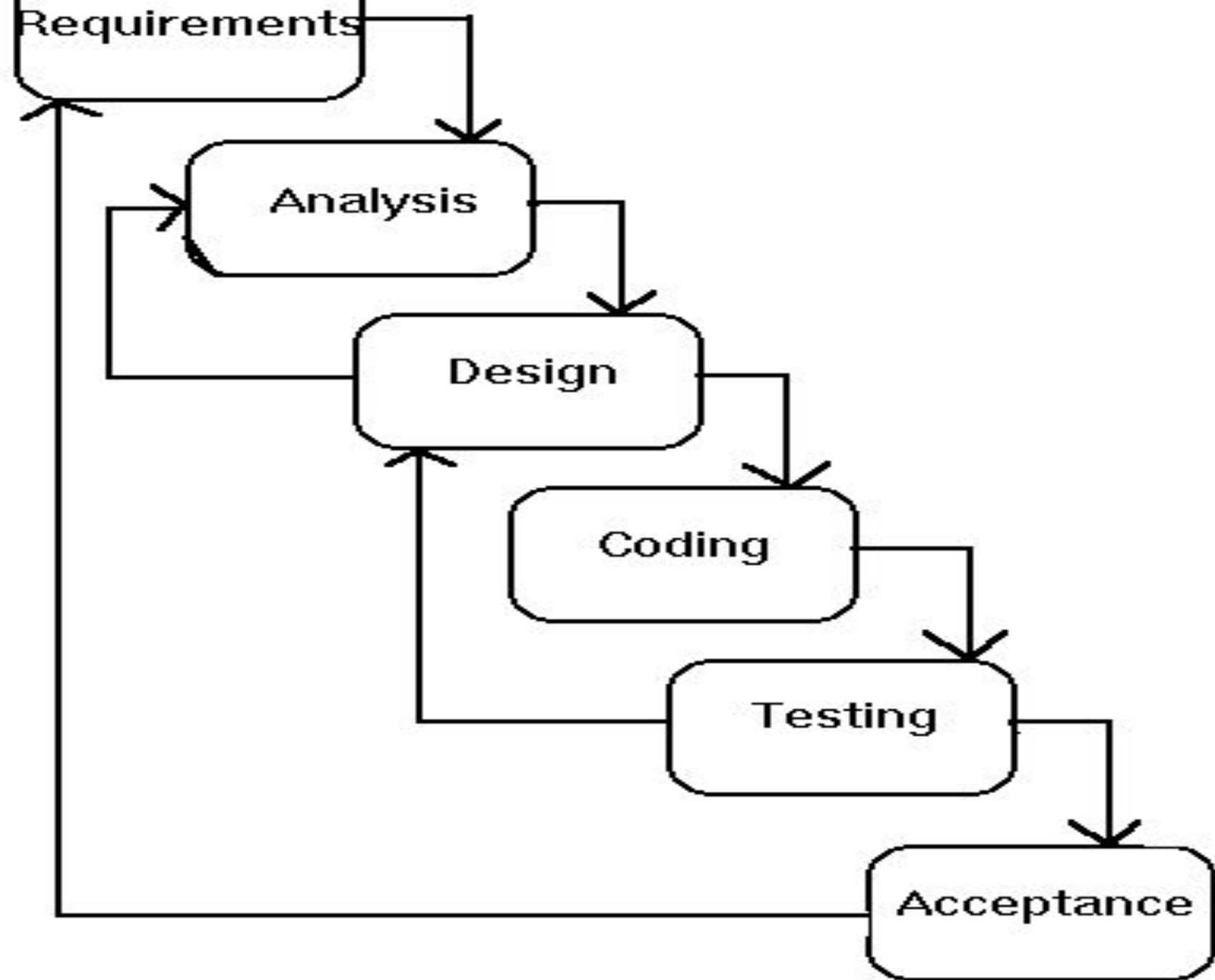
# SDLC

- The **Software Development Life Cycle (SDLC)**, or *System Development Life Cycle* in systems engineering, information systems and software engineering, is the entire process of formal, logical steps taken to develop a software product. The concept generally refers to computer or information systems.

## Phases of SDLC

The phases of SDLC can vary somewhat but generally include the following:

1. Problem Definition.
2. Program Design.
3. Coding.
4. Debugging.
5. Testing.
6. Documentation.
7. Maintenance.
8. Extension and Redesign



The waterfall model (Systems Development Life Cycle)

## **Problem Definition:**

Problem definition is the basic and primary step of software development life cycle. It includes the goal of system analysis and to determine where the problem is in an attempt to fix the system. This step involves "breaking down" the system in different pieces to analyze the situation. Requirements Gathering is also a step to be taken in this stage. Requirements Gathering sometimes requires individuals/teams from client as well as service provider sides to get detailed and accurate requirements.

## **Program Design:**

In systems, design functions and operations are described in detail, including screen layouts, business rules, process diagrams and other documentation. The output of this stage will describe the new system as a collection of modules or subsystems. The design stage takes as its initial input the requirements identified in the approved requirements document. Design elements describe the desired software features in detail, and generally include functional hierarchy diagrams, screen layout diagrams, tables of business rules, business process diagrams, pseudo code, and a complete entity-

## **Coding:**

Modular and subsystem programming code will be accomplished during this stage. Coding includes the application of various logic and internal work done by various specialists. This stage is intermingled with the next in that individual modules will need testing before integration to the main project



## **Debugging:**

Debugging is the process of removing the errors that occurs during the coding part. Debugging is essential for this stage establishes the platform for further stages of development.

## **Testing:**

The code is tested at various levels in software testing. Unit, system and user acceptance testing's are often performed. This is a grey area as many different opinions exist as to what the stages of testing are and how much if any iteration occurs.

## **Documentation:**

Documentation is the process of writing down every stages and each and every details of the process of life cycle development so that anyone who follows this process may be able to do it in the real sense. Documenting the internal design of software for the purpose of future maintenance and enhancement is done throughout development.

## **Maintenance:**

Maintaining the system is an important aspect of SDLC. As key personnel change positions in the organization, new changes will be implemented, which will require system updates. Maintenance is the process of keeping the software in its fully functional form and see to that nothing goes wrong. Maintaining and enhancing software to cope with newly discovered problems or new requirements can take far more time than the initial development of the software

## **Extension and Redesign:**

This is the last step of system design where there is always scope for extension and redesign whenever required. This stage allows for the extension of any part for the advancement of the software or so that the developed software does not become useless.

This stage allows further designing and following all the steps again.

## **Different types of SDLC models:**

Several models exist to streamline the development process. Each one has its pros and cons, and it's up to the development team to adopt the most appropriate one for the project. Sometimes a combination of the models may be more suitable.

1. Waterfall Model
2. Software Prototyping
3. Joint Applications Design (JAD)
4. Rapid Application Development (RAD)
5. Extreme Programming (XP); extension of earlier work in 6. Prototyping and RAD.
7. Open Source Development
8. End-user development

## **Strengths and weaknesses:**

Some people will argue that the SDLC no longer applies to models like Agile computing, but it is still a term widely in use in Technology circles. The SDLC practice has advantages in traditional models of software development that lends itself more to a structured environment. The disadvantages to using the SDLC methodology is when there is need for iterative development or (i.e. web development or e-commerce) where stakeholders need to review on a regular basis the software being designed..

## • Process Models

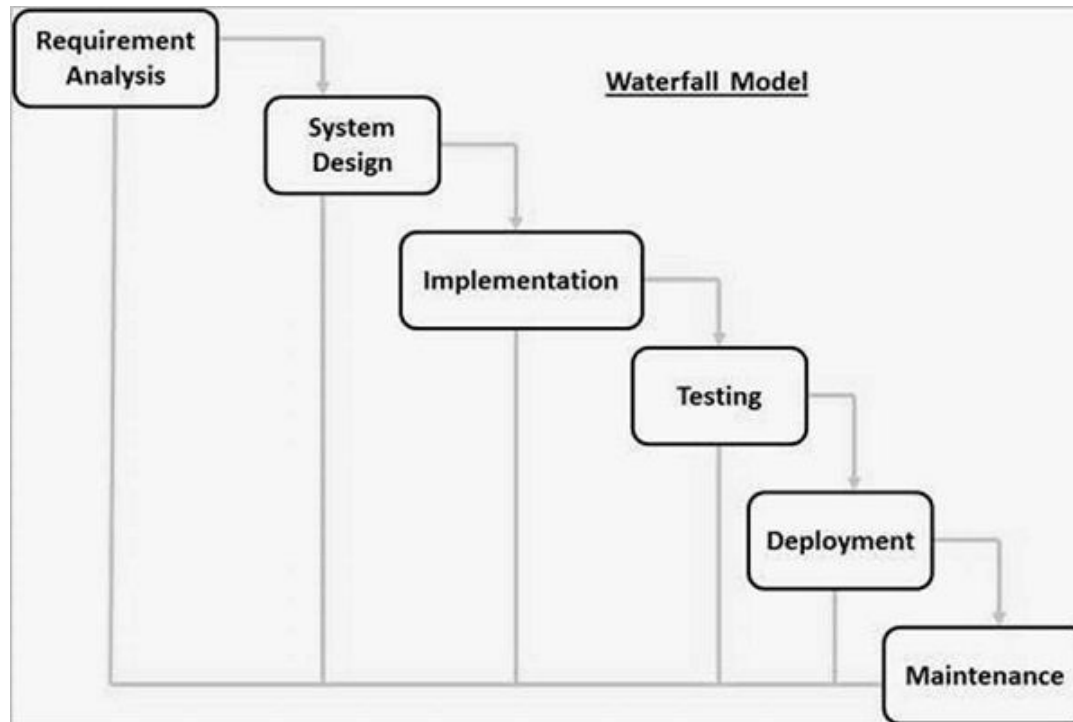
1. **A software process model is a simplified representation of a software process. Each model represents a process from a specific perspective.**
2. Following are the most important and popular SDLC models followed in the industry –
  - ❖ Waterfall Model
  - ❖ Iterative Model
  - ❖ Spiral Model
  - ❖ V-Model





# WATER FALL MODEL

- The Waterfall Model was the first Process Model to be introduced. It is also referred to as a **linear-sequential life cycle model**. It is very simple to understand and use



# WATER FALL MODEL

1. Winston Royce introduced the Waterfall Model in 1970
2. **WATERFALL MODEL** is a sequential model that divides software development into pre-defined phases.
3. Each phase must be completed before the next phase can begin with no overlap between the phases.
4. Each phase is designed for performing specific activity during the SDLC phase.





**Major advantages of the Waterfall Model** are as follows –

- Simple and easy to understand and use
- Easy to manage due to the rigidity of the model. Each phase has specific deliverables and a review process.
- Phases are processed and completed one at a time.
- Works well for smaller projects where requirements are very well understood.
- Clearly defined stages.
- Well understood milestones.
- Easy to arrange tasks.



- Process and results are well documented.

## • **DISADVANTAGES**

The major disadvantages of the Waterfall Model are as follows –

1. No working software is produced until late during the life cycle.
2. High amounts of risk and uncertainty.
3. Not a good model for complex and object-oriented projects.
4. Poor model for long and ongoing projects.
5. Not suitable for the projects where requirements are at a moderate to high risk of changing. So, risk and uncertainty is high with this process model.
6. It is difficult to measure progress within stages.
7. Cannot accommodate changing requirements.
8. Adjusting scope during the life cycle can end a project.

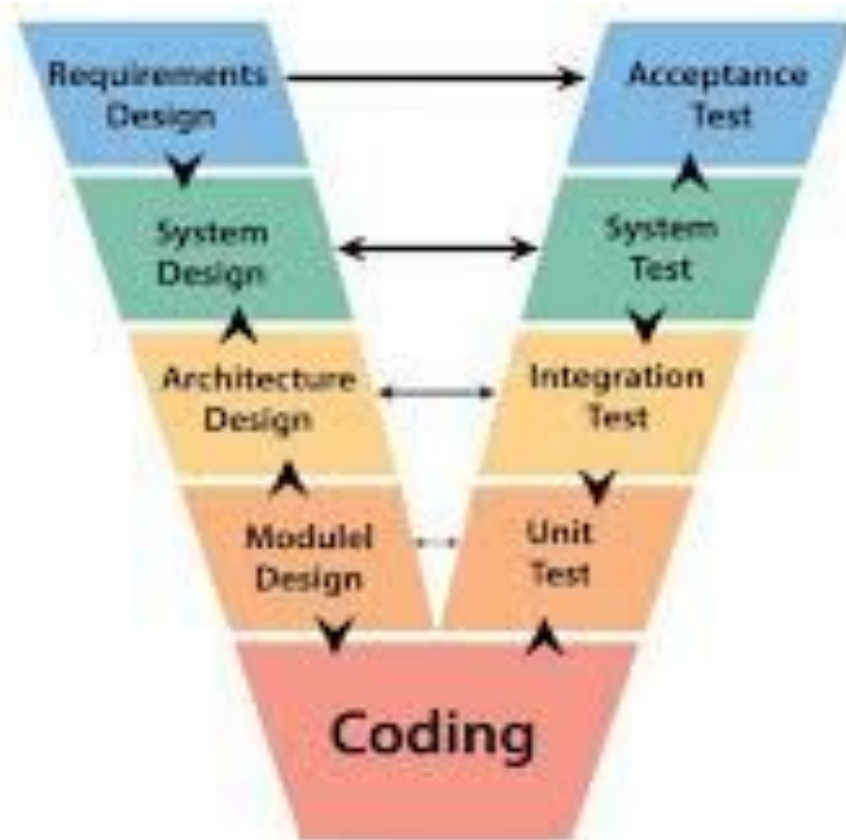


## • Application

1. Requirements are very well documented, clear and fixed.
2. Product definition is stable.
3. Technology is understood and is not dynamic.
4. There are no ambiguous requirements.
5. Ample resources with required expertise are available to support the product.
6. The project is short.
7. Example:customer relationship management(CRM),HRMS,supplychain management



- V-MODEL



# V-MODEL

1. The V-model is an SDLC model where execution of processes happens in a sequential manner in a V-shape.
2. The V model is an extension of the waterfall model in which testing is done on each stage parallel with development in a sequential way
3. . It is known as the Validation or Verification Model
4. The left side of the model is Software Development Life Cycle - **SDLC**
5. The right side of the model is Software Test Life Cycle - **STLC**
6. The entire figure looks like a V, hence the name **V - model**





# DESIGN PHASE:

1. **Requirement Analysis:** This phase contains detailed communication with the customer to understand their requirements and expectations. This stage is known as Requirement Gathering.
2. **System Design:** This phase contains the system design and the complete hardware and communication setup for developing product.
3. **Architectural Design:** System design is broken down further into modules taking up different functionalities. The data transfer and communication between the internal modules and with the outside world (other systems) is clearly understood.
4. **Module Design:** In this phase the system breaks down into small modules. The detailed design of modules is specified, also known as Low-Level Design (LLD).



# TESTING PHASES

1. **Unit Testing:** Unit Test Plans are developed during module design phase. These Unit Test Plans are executed to *eliminate bugs at code or unit level*.
2. **Integration testing:** In integration testing, the modules are integrated and the system is tested. This *test verifies the communication of modules among themselves*.
3. **System Testing:** System testing test the complete application with its functionality, inter dependency, and communication. It tests the *functional and non-functional requirements of the developed application*.
4. **User Acceptance Testing (UAT):** UAT is performed in a user environment that resembles the production environment. UAT verifies that the delivered system meets user's requirement and system is ready for use in real world.



## • Advantages of the V-Model

1. Advantages of the V-Model are as follows –
2. This is a highly-disciplined model and Phases are completed one at a time.
3. Support big project Works well for smaller projects and where requirements are very well understood.
4. Simple and easy to understand and use.
5. Easy to manage due to the rigidity of the model. Each phase has specific deliverables and a review process.



## • Disadvantages of the V-Model

1. Disadvantages of the V-Model method are as follows –
2. High risk and uncertainty.
3. Not a good model for complex and object-oriented projects.
4. Poor model for long and ongoing projects.
5. Not suitable for the projects where requirements are at a moderate to high risk of changing.
6. Once an application is in the testing stage, it is difficult to go back and change a functionality.
7. No working software is produced until late during the life cycle.



## • APPLICATION OF V - MODEL

Uses of the V-Model application.

1. Requirements are well defined, clearly documented and fixed.
2. Product definition is stable.
3. Technology is not dynamic and is well understood by the project team.
4. There are no ambiguous or undefined requirements.
5. The project is short.



## • Evolutionary Process Models

1. Evolutionary models are iterative type models.
2. They allow to develop more complete versions of the software.

### **3. Following are the evolutionary process models.**

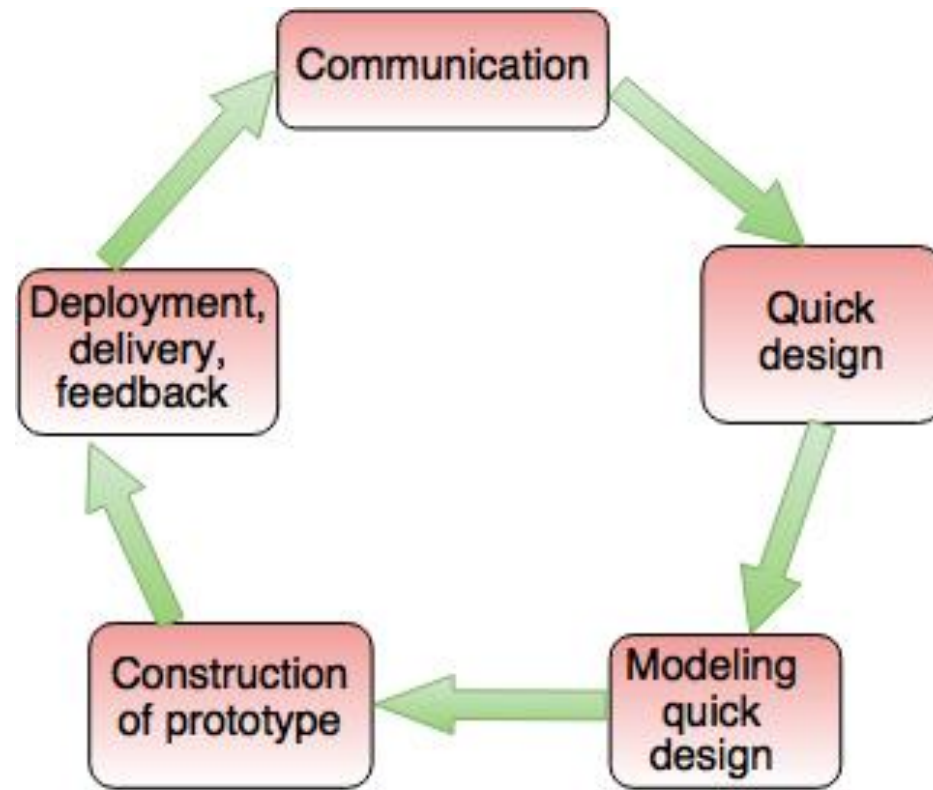
1. The prototyping model
2. The spiral model



## • What is Prototyping Model?

1. **Prototyping Model** is a software development model in which prototype is built, tested, and reworked until an acceptable prototype is achieved.
2. It also creates base to produce the final system or software.
3. It works best in scenarios where the project's requirements are not known in detail.
4. It is an iterative, trial and error method which takes place between developer and client.





**Fig. - The Prototyping Model**





# |

## 1. Advantages of Prototype model

- Users are actively involved in the development
- Since in this methodology a working model of the system is provided, the users get a better understanding of the system being developed.
- Errors can be detected much earlier.
- Quicker user feedback is available leading to better solutions.
- Missing functionality can be identified easily
- Confusing or difficult functions can be identified



## • DISADVANTAGE OF PROTOTYPE MODEL

1. **Time-consuming:** As the prototype is being modified time to time according to customer requirement which usually increases the time of completion of the product.
2. **Complexity:** Change in the requirement usually expand the scope of the product beyond its original plan and thus increase the complexity.
3. **Poor Documentation:** Continuous changing of requirement can lead to poor documentation.



## • DISADVANTAGE OF PROTOTYPE MODEL

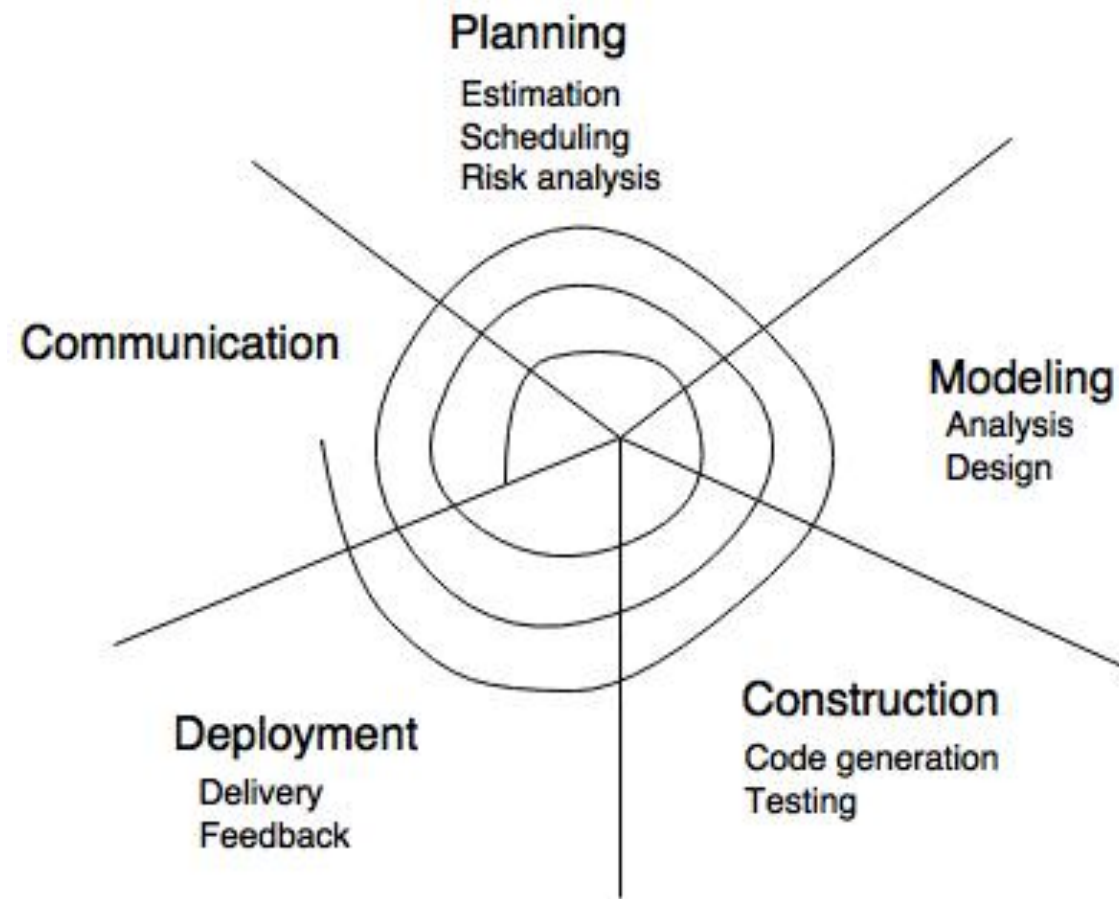
1. **Unpredictability of no of iteration:** It is difficult to determine the no of iteration required before the prototype is finally accepted by the customer.
2. **Confusion:** Customer can confuse between the actual product and prototype



# Spiral Model

1. The spiral model is a risk-driven where the process is represented as spiral rather than a sequence of activities.
2. It was designed to include the best features from the waterfall and prototyping models, and introduces a new component; risk-assessment.
3. Each loop (from *review* till *service* — see figure below) in the spiral represents a phase. Thus the first loop might be concerned with system feasibility, the next loop might be concerned with the requirements definition, the next loop with system design, and so on.
- 4.





**Fig. - The Spiral Model**



# ADVANTAGES

1. The advantages of the Spiral SDLC Model are as follows –
2. Changing requirements can be accommodated.
3. Allows extensive use of prototypes.
4. Requirements can be captured more accurately.
5. Users see the system early.
6. Development can be divided into smaller parts and the risky parts can be developed earlier which helps in better risk management.



## • Disadvantages

Disadvantages of the Spiral SDLC Model are as follows

1. Management is more complex.
2. End of the project may not be known early.
3. Not suitable for small or low risk projects and could be expensive for small projects.
4. Process is complex
5. Spiral may go on indefinitely.
6. Large number of intermediate stages requires excessive documentation



## Uses of a Spiral Model –

1. When there is a budget constraint and risk evaluation is important.
2. For medium to high-risk projects.
3. Long-term project commitment because of potential changes to economic priorities as the requirements change with time.
4. Customer is not sure of their requirements which is usually the case.
5. Requirements are complex and need evaluation to get clarity.
6. New product line which should be released in phases to get enough customer feedback.
7. Significant changes are expected in the product during the development cycle



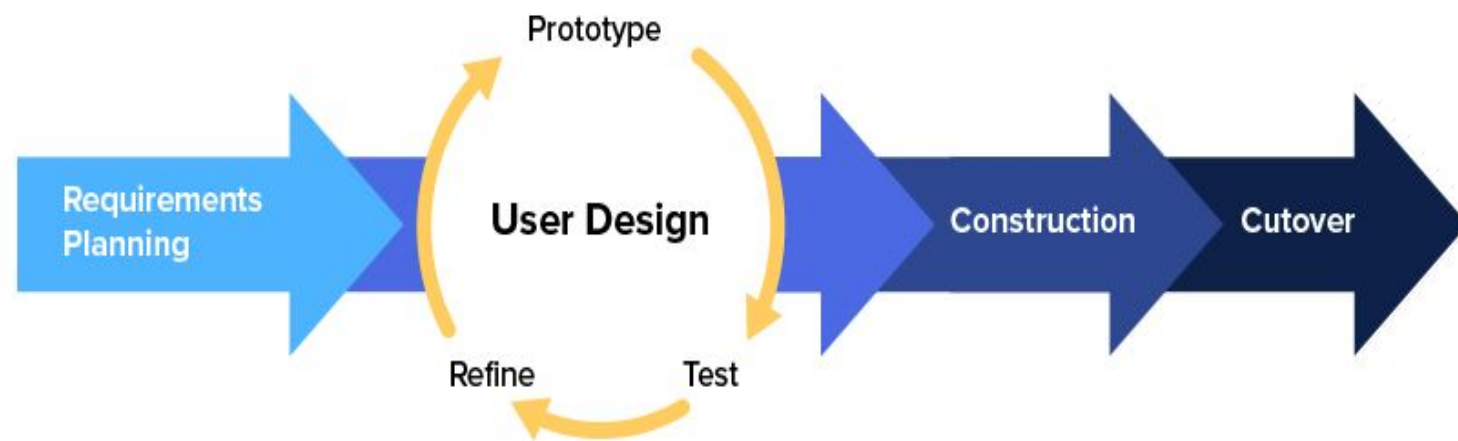


## • RAD MODEL

1. Rapid Application Development model is a software development process based on prototyping without any specific planning.
2. In RAD model, there is less attention paid to the planning and more priority is given to the development tasks.
3. It targets at developing software in a short span of time.
4. The Rapid Application Development Model was first proposed by IBM in 1980's.
5. The critical feature of this model is the use of powerful development tools and techniques.



## Rapid Application Development (RAD)



## USE OF RAD MODEL

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



## • Advantage of RAD Model

1. Advantage of RAD Model
2. This model is flexible for change.
3. In this model, changes are adoptable.
4. Each phase in RAD brings highest priority functionality to the customer.
5. It reduced development time.
6. It increases the reusability of features.



## Disadvantage of RAD Model

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



# INCREMENTAL MODEL

**Incremental Model** is a software development process where requirements are divided into several stand-alone software development modules.

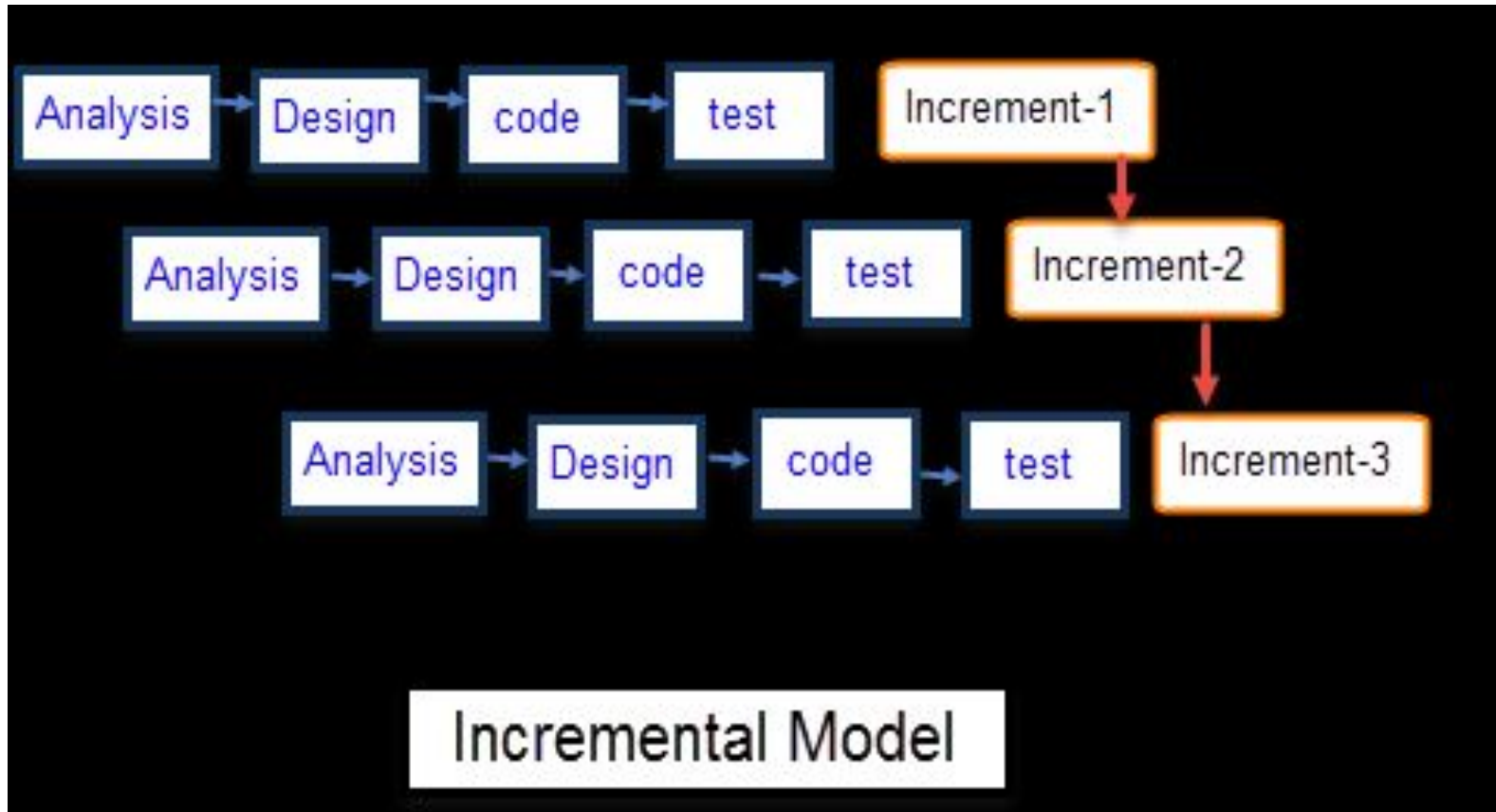
In this example, each module passes through the

- Requirement,
- Design,
- Development,
- Implementation,
- Testing Phases.

That subsequent release of the module adds a feature to the previous release.

The process will continue until the whole software is achieved.

# INCREMENTAL MODEL



## • Advantages of Incremental Model

1. Errors are easy to detect.
2. Easy to test and debug.
3. Flexible.
4. Easy to manage risk because it has been managed through iteration.
5. The client is provided with significant functionality at an early stage.





# DISADVANTAGES OF INCREMENTAL MODEL

1. It needs good planning.
2. The total cost is high.
3. It needs well-defined module interfaces.
4. Needs a clear and complete definition of the whole system before it can be broken down and built incrementally.



# USE THE INCREMENTAL MODEL

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



## Comparison of SDLC Models

Properties	Water-Fall	Incremental	Spiral	Rad
Objectives	High Assurance	Rapid Development	High Assurance	Rapid development
Planning in early stage	Yes	Yes	Yes	No
Returning to an earlier phase	No	Yes	Yes	Yes
Handle Large-Project	Not Appropriate	Not Appropriate	Appropriate	Not Appropriate
Time-Frame	Very Long	Long	Long	Short
Working software availability	At the end of the life-cycle	At the end of every iteration	At the end of every iteration	At the end of the life cycle
Risk Involvement	High	Low	Medium to high risk	Low
Software Team size	Large Software Team	Not Large Software Team	Large Software Team	Small Software Team
Customer control over administrator	Very Low	Yes	Yes	Yes
Maintenance	Least	Promotes Maintainability	Typical	Easily Maintained
Time Duration	Long	Very long	Long	Short
Re-usability	Least possible	To some extent	To some extent	Yes
Framework Type	Linear	Linear + Iterative	Linear + Iterative	Linear
When Testing?	After completion of development phase	After every iteration	At the end of the engineering phase	After completion of development
Maintenance	Least Maintainable	Maintainable	Yes	Easily Maintainable
Detailed Documentation	Necessary	Yes but not much	Yes	Limited
Overlapping Phases	No	Yes	No	Yes



# WHAT IS AGILITY

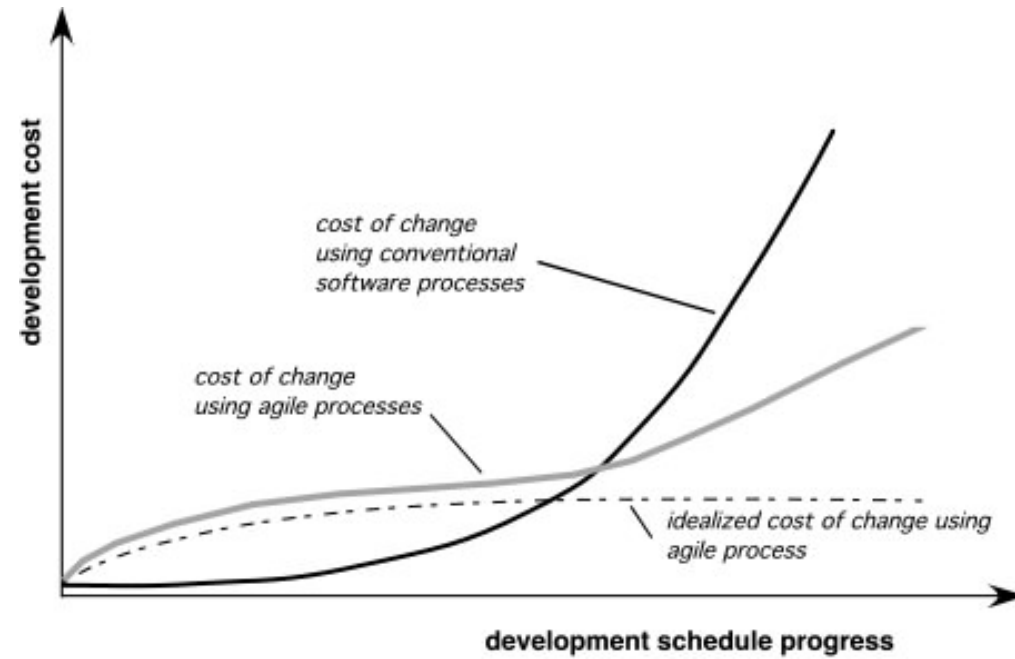
1. **Effective (rapid and adaptive) response to change**
2. Effective communication among all stakeholders
3. Drawing the customer onto the team
4. Organizing a team so that it is in control of the work performed

*Yielding ...*

1. Rapid, incremental delivery of software
2. ***The agile process forces the development team to focus on software itself rather than design and documentation.***



## •AGILITY AND COST OF CHANGE



# AN AGILE PROCESS

1. Is driven by customer descriptions of what is required (scenarios)
2. Recognizes that plans are short-lived
3. Develops software iteratively with a heavy emphasis on construction activities
4. Delivers multiple 'software increments'
5. Adapts as changes occur



# AN AGILE PROCESS



# AGILITY PRINCIPLES - I

1. **Our highest priority is to satisfy the customer** through early and continuous delivery of valuable software.
2. **Welcome changing requirements, even late in development.** Agile processes harness change for the customer's competitive advantage.
3. Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
4. Business people and developers must work together daily throughout the project.
5. Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
6. The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.





## AGILITY PRINCIPLES - II

7. Working software is the primary measure of progress.
8. Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
9. Continuous attention to technical excellence and good design enhances agility.
10. Simplicity – the art of maximizing the amount of work not done – is essential.
11. The best architectures, requirements, and designs emerge from self-organizing teams.
12. At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.



## • Human Factors

1. *the process molds to the needs of the people and team, not the other way around*
2. key traits must exist among the people on an agile team and the team itself:
  1. **Competence.**
  2. **Common focus.**
  3. **Collaboration.**
  4. **Decision-making ability.**
  5. **Fuzzy problem-solving ability.**
  6. **Mutual trust and respect.**
  7. **Self-organization.**





## • **Extreme Programming (XP)**

What is Extreme Programming?

1. XP is a lightweight, efficient, low-risk, flexible, predictable, scientific, and fun way to develop a software.
2. Extreme Programming is one of the Agile software development methodologies
3. It shares all Agile principles including strong customer involvement in the software development process, good communication inside of the teams, and iterative cycles of development.



- **Extreme Programming (XP)**

1. The most widely used agile process, originally proposed by Kent Beck

2. **XP Planning**

1. Begins with the creation of “user stories”
2. Agile team assesses each story and assigns a cost
3. Stories are grouped to for a deliverable increment
4. A commitment is made on delivery date
5. After the first increment “project velocity” is used to help define subsequent delivery dates for other increments



# • Extreme Programming (XP)

## 1. XP Design

1. Follows the KIS principle (KEEP IT SIMPLE)
2. Encourage the use of CRC(class responsibility collaborator) cards –Relevant to current s/w increment
3. Encourages “refactoring”—an iterative refinement of the internal program design
4. Spike solution (low risk and validate the story containing the design problem)

## 2. XP Coding

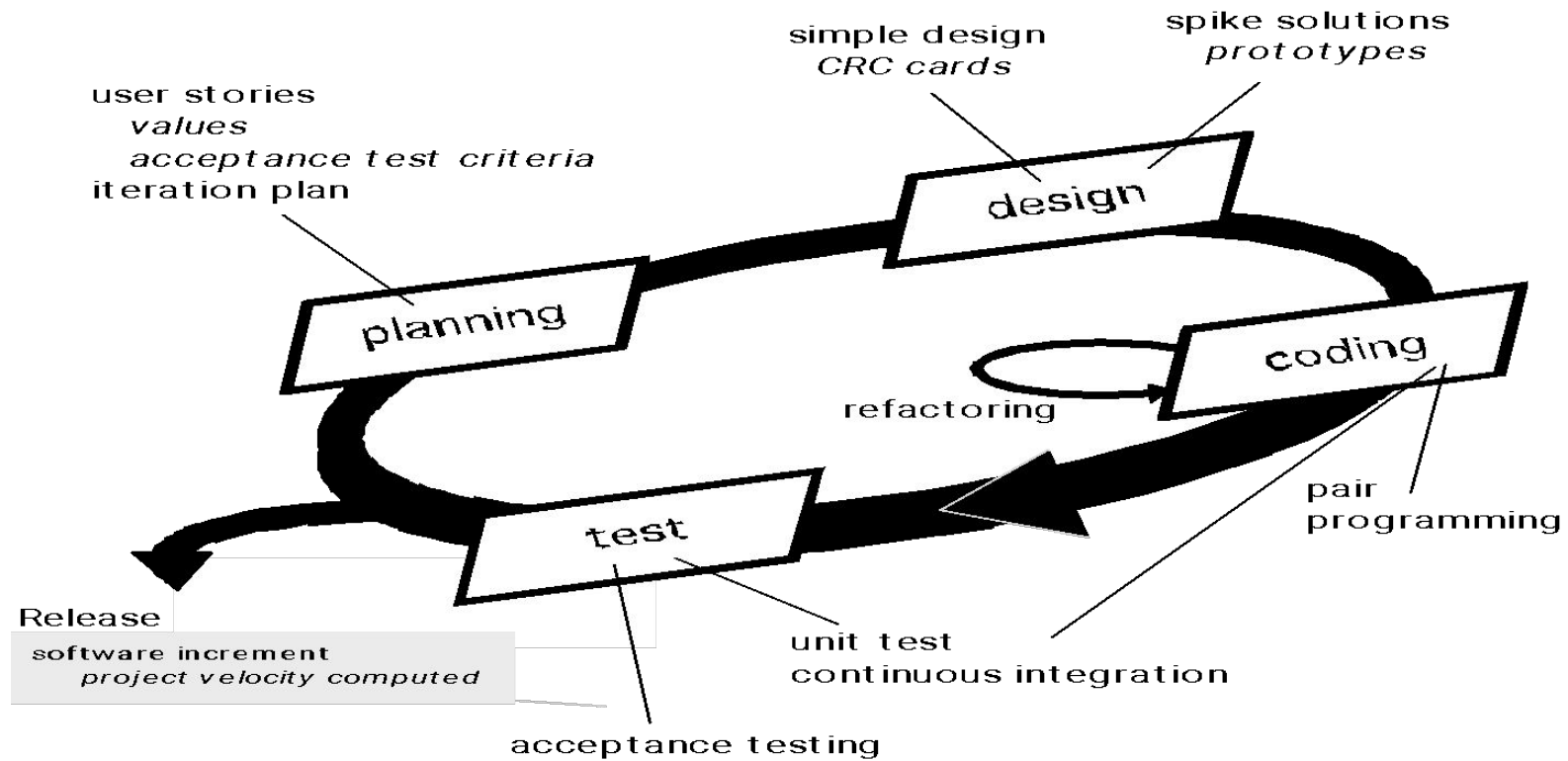
1. Recommends the construction of a unit test for a store *before* coding commences
2. Encourages “pair programming”

## 3. XP Testing

1. All unit tests are executed daily
2. “Acceptance tests” are defined by the customer and excuted to assess customer visible functionality



# EXTREME PROGRAMMING (XP)



# APPLICATIONS

1. **Applications of Extreme Programming (XP):** Some of the projects that are suitable to develop using XP model are given below:
2. **Small projects:** XP model is very useful in small projects consisting of small teams as face to face meeting is easier to achieve.
3. **Projects involving new technology or Research projects:** This type of projects face changing of requirements rapidly and technical problems. So XP model is used to complete this type of projects.





# ADVANTAGES OF EXTREME PROGRAMMING

1. save costs and time
2. reduces the risks
3. Simplicity
4. process is visible and accountable.
5. Constant feedback
6. working software faster



## • Disadvantages Extreme Programming

1. Some specialists say that Extreme Programming is focused on the code rather than on design.
2. This methodology does not measure code quality assurance.
3. It may cause defects in the initial code.
4. XP is not the best option if programmers are separated geographically.



## 1. Adaptive Software Development (ASD)



## Adaptive Software Development (ASD)

1. Adaptive software development (ASD) has been proposed by Jim Highsmith as a technique for building complex software and systems.
2. ASD focuses on human collaboration and self-organisation
3. It's focused on rapid creation and evolution of software systems.
4. It aims to enable teams to quickly and effectively adapt to changing requirements



## **Characteristics of ASD**

### **1. Mission Driven**

The activities in the each development cycle must be justified against the overall project mission.

### **2. Component Based**

Development activities should not be task oriented but rather focus on developing working software. It focuses on results.

### **3. Iterative**

Redoing of development instead of doing it right the first time

### **4. Time Boxed**

Setting fixed delivery times for projects



## Characteristics of ASD

### **5. Change Tolerant**

Able to incorporate change is viewed as a competitive advantage (not as a problem).

### **6. Risk Driven**

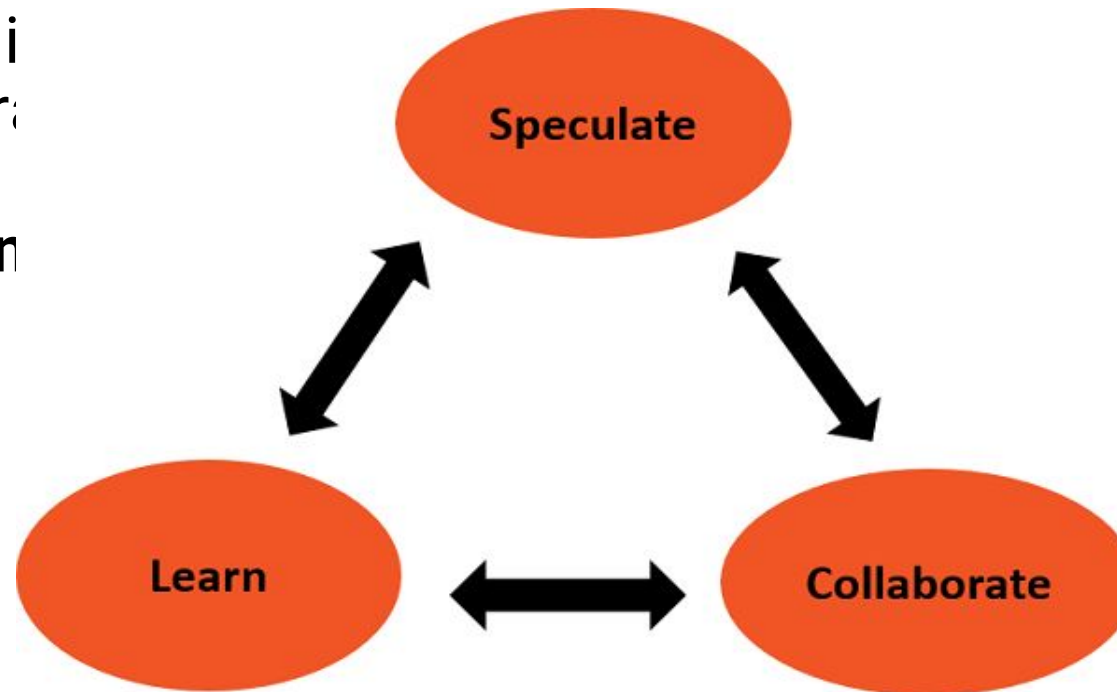
The development of high risk items should begin as early as possible.



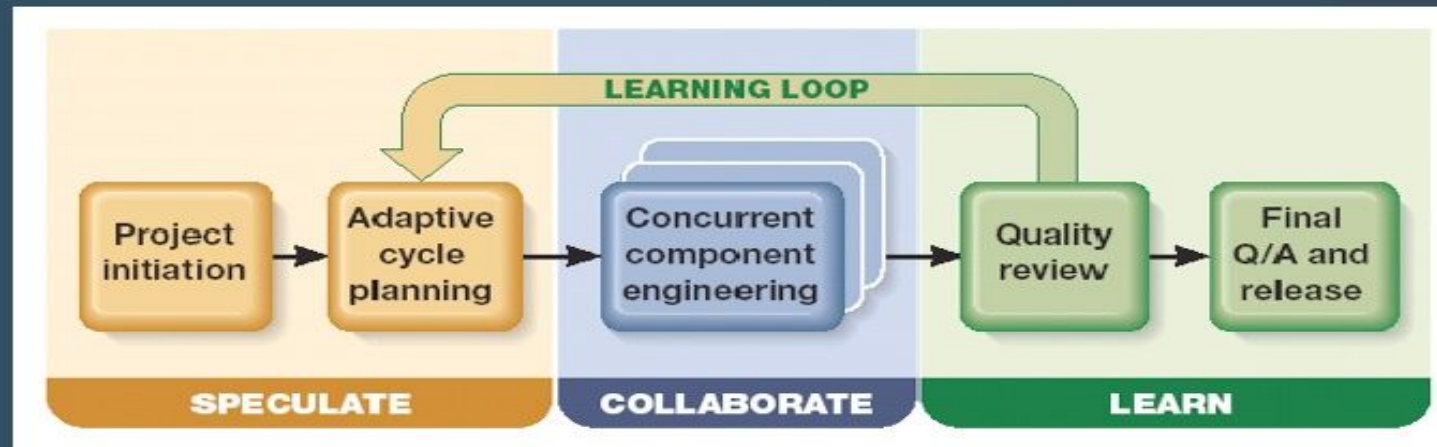
# 1. Adaptive Software Development (ASD)

2. ASD life cycle i  
diagram illustra

- **Speculation**
- **Collaboration**
- **Learning**



# Adaptive Software Development



- Short iterations
- Deliverable-centric instead of task-centric





## 1. Speculation:

During this phase project is initiated and planning is conducted. The project plan uses **project initiation** information like

- project requirements,
- user needs,
- customer mission statement etc, to define set of release cycles that the project wants.

## 2. Collaboration:

- Effective collaboration with customer is very important.
- Communication, teamwork, individual creativity is part of effective collaboration.
- Collaborate would require the ability to work jointly to produce results, share knowledge or make decisions.





**3.Learning** :-helps the workers to increase their level of understanding over the project. During the learning phase, the newest version of the software is released to users

Learning process is of 3 ways:

1. Focus groups
2. Technical reviews
3. Project postmortem



# ADVANTAGES OF ADAPTIVE SOFTWARE DEVELOPMENT :

## Advantages of Adaptive Software Development :

1. Useful for rapid and complex software product development.
2. Easy software incremental adjustment
3. Focus on end-users, meeting requirements, and fulfilling demands
4. Allows on-time delivery with maximum customer satisfaction
5. Provides high transparency between developers and clients



# DISADVANTAGES OF ADAPTIVE SOFTWARE DEVELOPMENT :

## Disadvantages of Adaptive Software Development :

1. Requires high user/client involvement
2. Requires testing into each iteration which increases the cost
3. Frequent changes undergo with less documentation
4. Requires strict time commitment between different teams involved in project



# SCRUM



# SCRUM!



# SCRUM

1. Scrum is a subset of Agile.
2. It is a lightweight process framework for agile development, and
3. The most widely-used one.
4. It stresses the importance of accountability, team collaboration,
5. product iterations aimed at achieving a high-quality product.
6. Scrum in Software Testing is a methodology for building complex software applications



# SCRUM PRINCIPLES



# SCRUM

- Scrum Principles

1. **Small working teams used to maximize communication and minimize overhead**
2. **Process must be adaptable** to both technical and business challenges to ensure best produced
3. **Process yields frequent increments** that can be inspected, adjusted, tested, documented & built on
  - **Development work** and people performing it are partitioned into clean, low coupling partitions
  - **Testing and documentation** is performed as the product is built
  - Ability to declare the product done whenever required





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

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



# SCRUM



# SCRUM PROCESS

## 1. **Step1: Create a Scrum project**

The list of tasks that needs to be done in the project is called a product backlog.

## 1. **Step 2: Create user stories or tasks in the backlog**

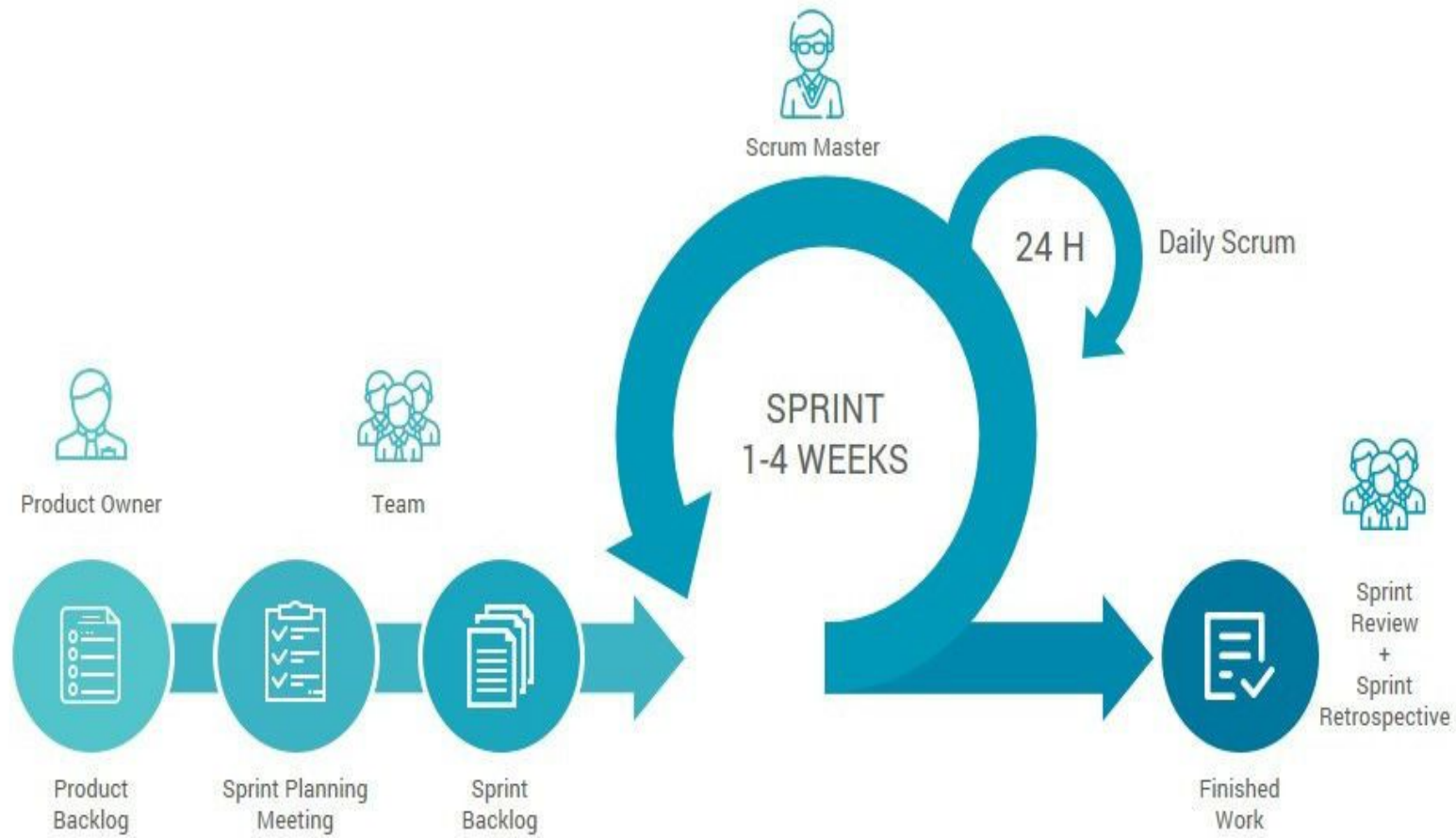
2. User stories describe the work items from the user's perspective in a non-technical language.
3. Once the user stories are created, we can prioritize them in the backlog.

## 4. **Step 3: Create a Sprint**

5. In Scrum, the team forecast to complete a set of user stories in a fixed time duration called a sprint. The optimal time mostly we set for a sprint is for two weeks.



# SCRUM



# SCRUM PROCESS

## **Step4: Hold the Sprint Planning Meeting**

1. The first meeting to start the sprint is called a Sprint planning meeting.
2. The product owner, scrum master, and the entire team members attend it to discuss the sprint goal and stories in the prioritized product backlog.

## **Step 5 : Hold the Daily Standup Meetings**

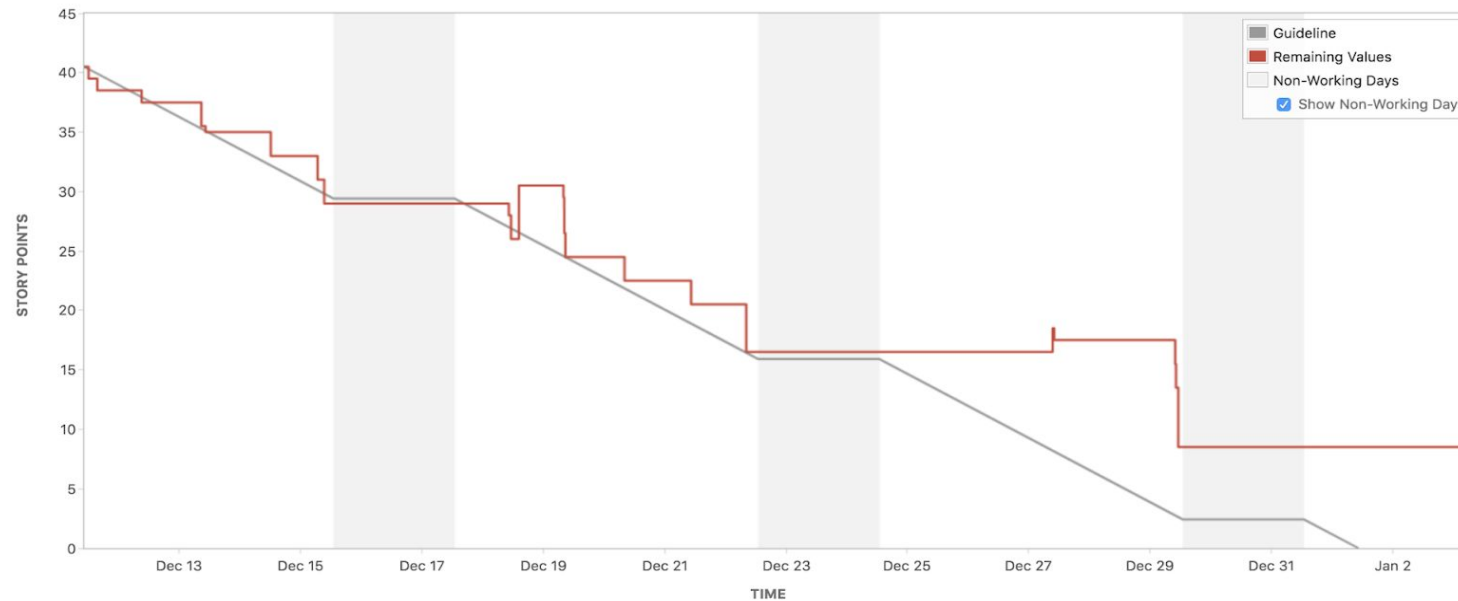
1. Daily standup meetings inspect the progress of the Sprint goal and how progress is trending towards work completion in the sprint backlog.



# SCRUM PROCESS

## Step 6 : View the Burndown Chart

1. A graphical representation of how quickly the team is working through customers' user stories is called the Burndown chart. It estimates the actual and estimated amount of work to be done in a sprint.



# SCRUM PROCESS

## Step 7: Hold the Sprint Review Meeting

1. Sprint reviews take place at the end of the sprint and are specially designed for gathering actionable feedback on what the team has completed.
2. Also, referred to as demos, which shows the overall work to the team and inspects the roadmap for the product.

## Step 8: Hold the Sprint Retrospective Meeting

1. Sprint retrospectives are used to inspect and improvements to be performed in the next sprint. Sprint retrospectives occur before the next sprint planning and after the Sprint review.



# SCRUM

- **Advantages of Scrum**

1. Quicker release
2. Increased project control
3. Greater ability to incorporate changes at any time
4. Improved progress visibility and control
5. Higher productivity
6. Lower costs
7. Higher customer satisfaction





# SCRUM

- **Disadvantages of Scrum**

1. The chances of project failure are high if individuals aren't very committed or cooperative
2. Daily meetings sometimes frustrate team members
3. The Scrum framework in large teams is challenging
4. Mostly suited for small teams that have great cohesion and understanding.
5. If a member of the team leaves the process during development, it has a negative impact on the project development.
6. With no deadline to deliver the product, it leads to scope creep

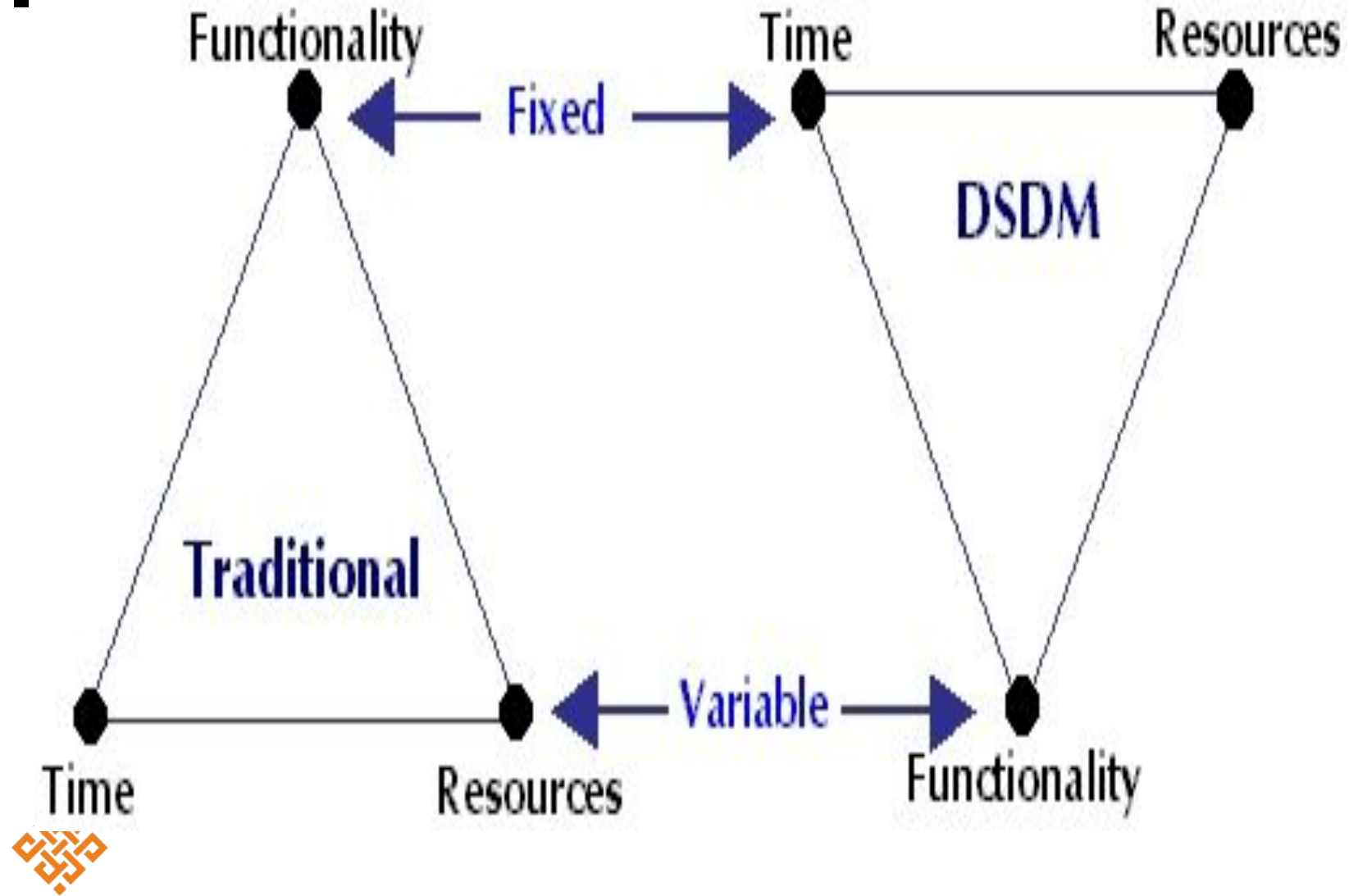


# DSDM

1. Dynamic Systems Development Method (DSDM) is an organized process focused on delivering business solutions quickly and efficiently.
2. It is similar in many ways to SCRUM and XP,
3. but it has its best uses where the time requirement is fixed.
4. DSDM is an iterative and incremental approach that emphasizes continuous user/customer involvement.
5. It provides the full roadmap for delivery on time and within limits of a budget.



# DSDM



## • **DSDM Advantages and Disadvantages**

DSDM's strengths include:

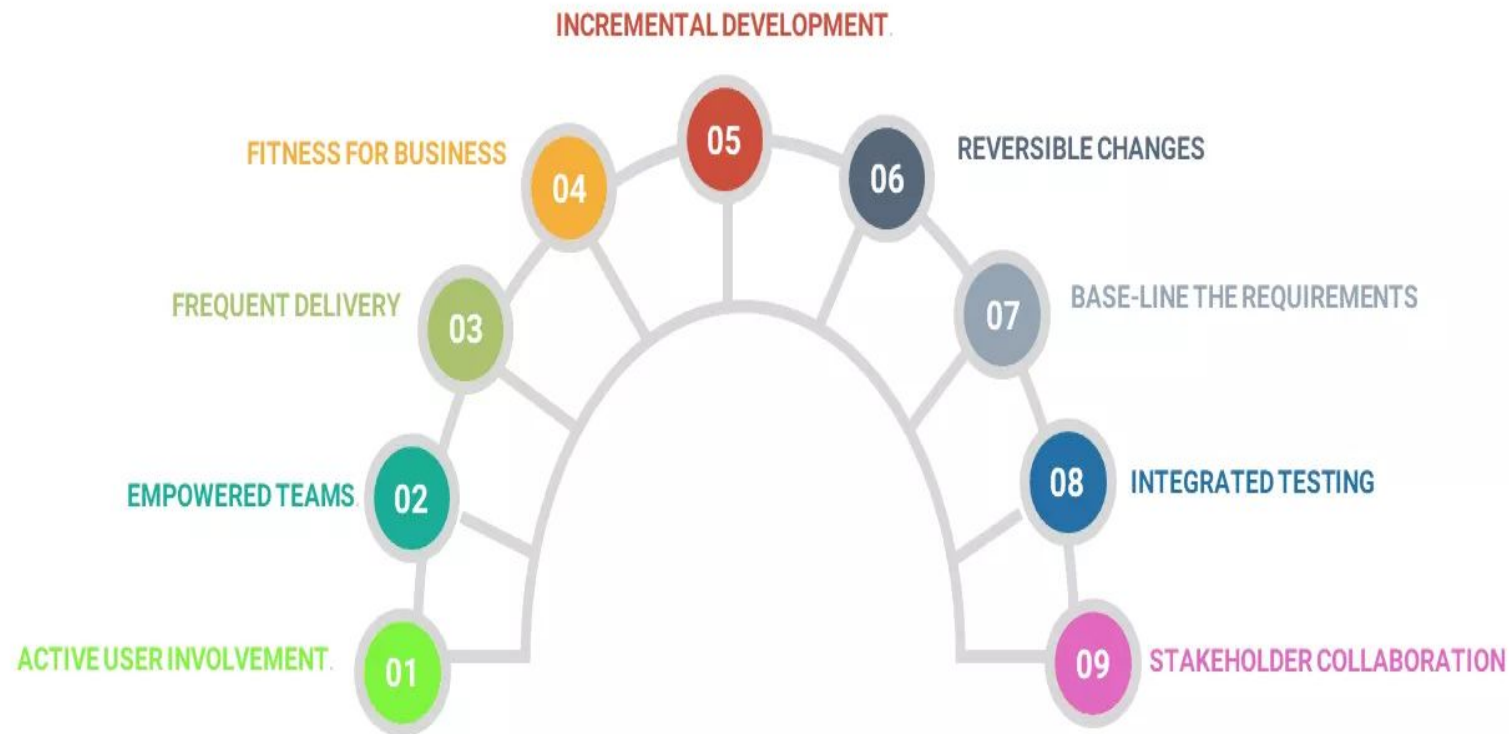
1. Developers have easy access to end-users
2. Projects are reliably completed on time
3. Basic product functionality can be delivered rapidly

DSDM's weaknesses include:

1. Costly to implement
2. Not ideal for small organizations



# 9 PRINCIPLES OF DSDM



# DSDM Nine guiding principles

- **Active user involvement** is imperative.
- **DSDM teams must be empowered** to make decisions.
- The focus is on **frequent delivery of products**.
- Fitness for business purpose is the essential criterion for **acceptance of deliverables**.
- Iterative and incremental development is **necessary to converge on an accurate business solution**.
- All changes during development are reversible.
- Requirements are baselined at a high level
- Testing is integrated throughout the life-cycle.

▪



# CRYSTAL

1. Crystal is an agile methodology for software development.
2. It places focus on people over processes, to empower teams to find their own solutions for each project
  - Face-to-face communication is emphasized
3. Crystal methods are said to be “lightweight methodologies
4. Crystal emphasizes people aspects of development: – Communication – Collaboration – Cooperation – Skills



# CRYSTAL

Crystal methods are focused on:

1. People
2. Interaction
3. Community
4. Skills
5. Talents
6. Communications







Crystal Clear uses seven methods / practices, three of which are mandatory:

1. Frequent delivery of the product
2. Improvements through reflection
3. Personal Communications
4. Sense of security
5. Focusing
6. Easy access to experts
7. Qualitative technical environment



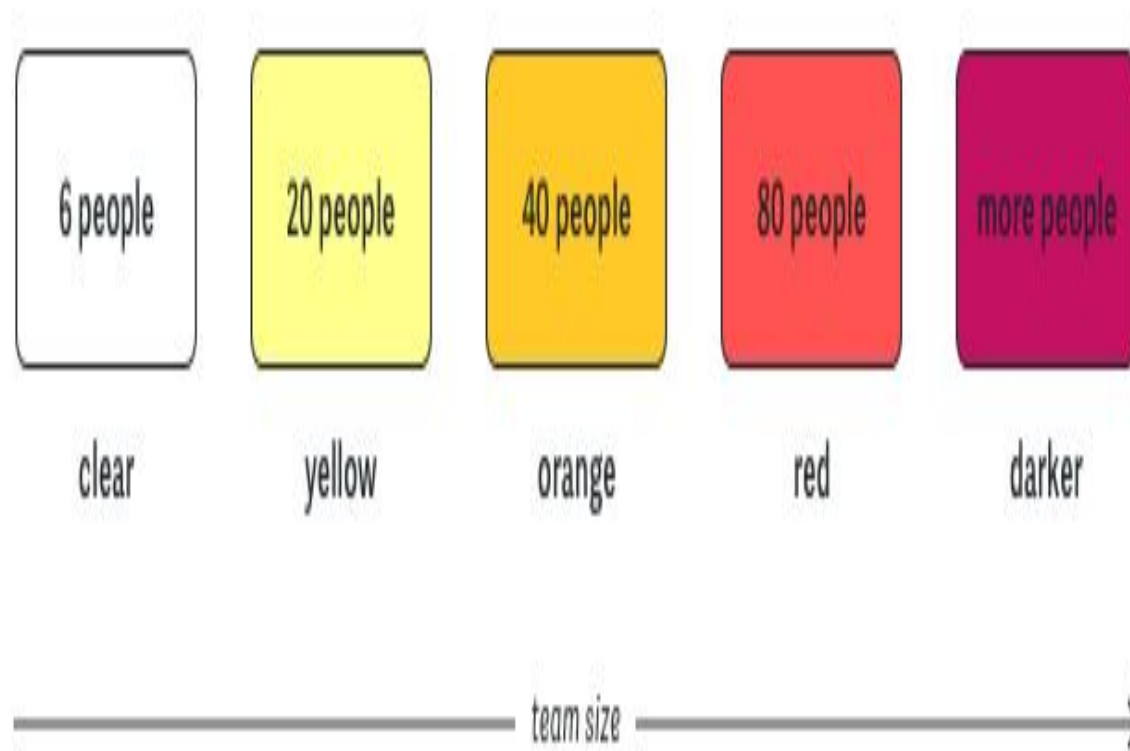
## • 7 Properties of Crystal Clear





1. Crystal Methodologies
2. Crystal Clear is designed for very small projects comprising up to six developers.
3. Crystal Clear is a light, tolerant, low ceremony, and barely sufficient methodology.
4. . **Clear** - for teams of 8 or fewer people
5. **Yellow** - for teams of 10-20 people
6. **Orange** - for teams of 20-50 people
7. **Red** - for teams of 50-100 people





# FEATURE-DRIVEN DEVELOPMENT.

1. **FDD** stands for **Feature-Driven Development**.
2. It is an agile **iterative and incremental model** that focuses on progressing the features of the developing software.
3. The main goal of FDD is to provide timely updated and working software to the client.
4. In FDD, reporting and progress tracking is necessary at all levels.



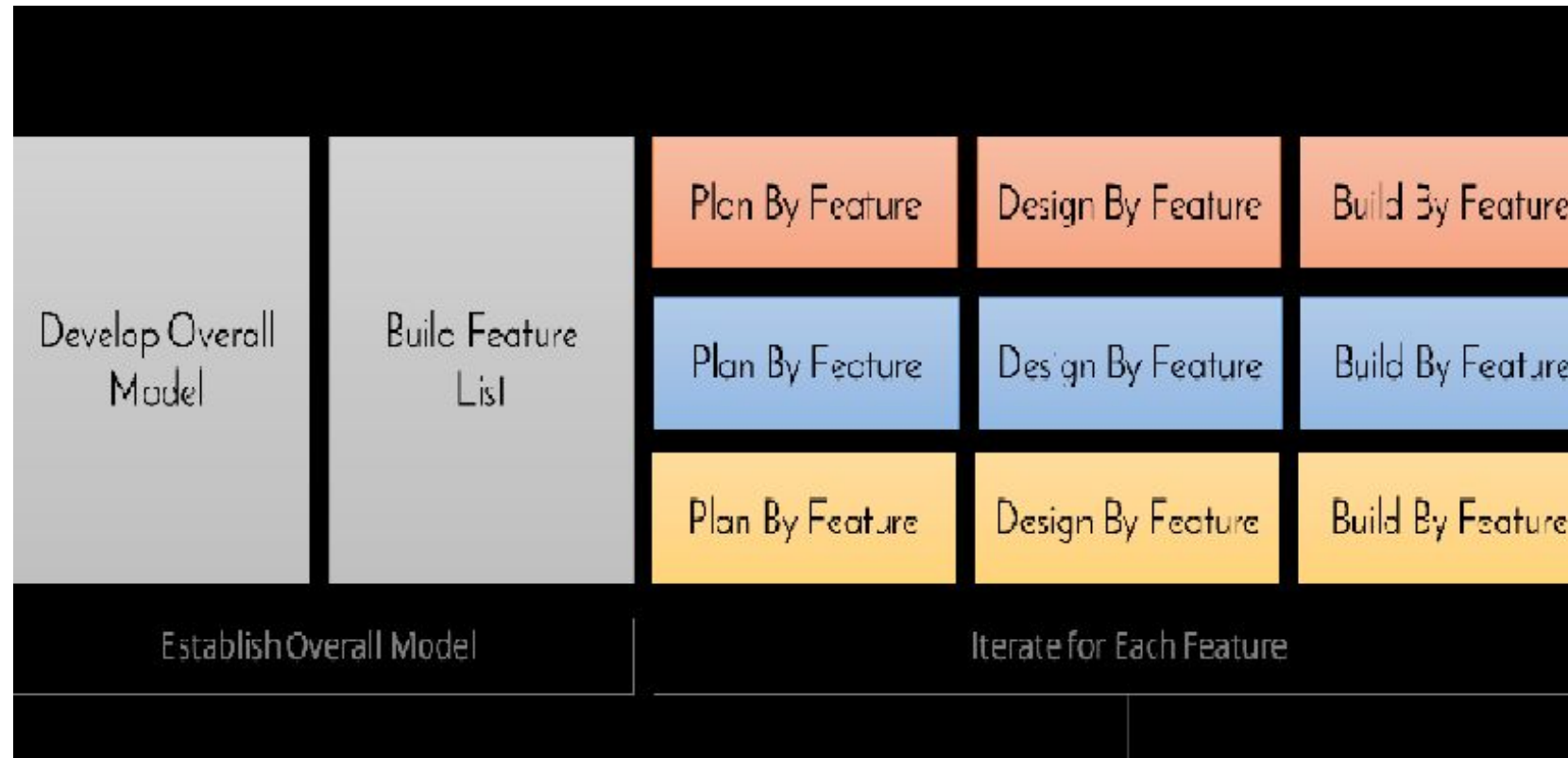
# FDD LIFECYCLE

## FDD Lifecycle

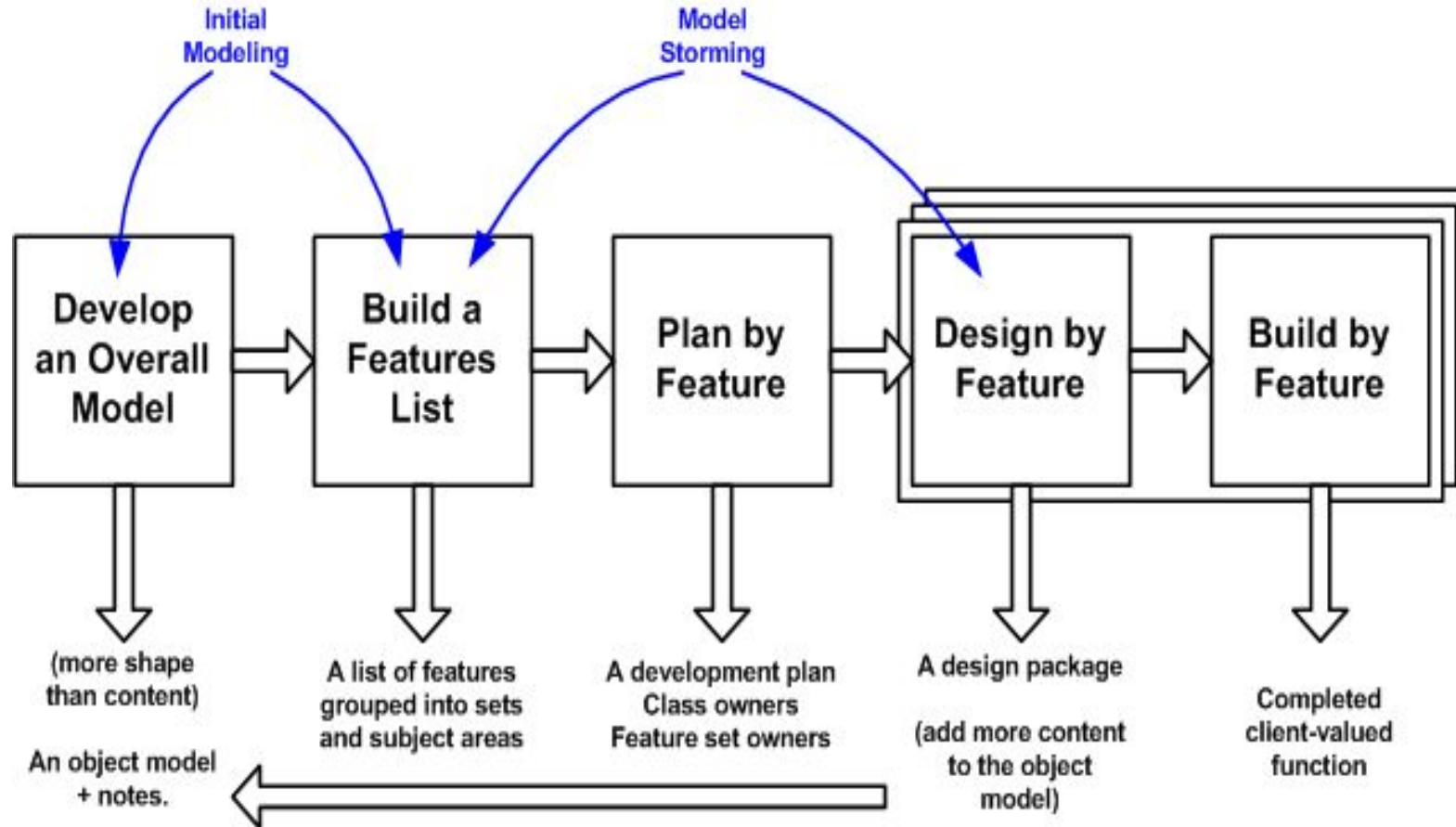
1. Build overall model
2. Build feature list
3. Plan by feature
4. Design by feature
5. Build by feature



# FEATURE-DRIVEN DEVELOPMENT.



# FEATURE-DRIVEN DEVELOPMENT.



Copyright 2002-2005 Scott W. Ambler  
Original Copyright S. R. Palmer & J.M. Felsing





# FEATURE-DRIVEN DEVELOPMENT.

- Advantages of FDD

1. Simple five-step process allows for more rapid development
2. Reporting at all levels leads to easier progress tracking.
3. FDD provides continuous success for larger size of teams and projects
4. Breaks feature sets into smaller chunks and regular iterative releases
5. Which makes it easier to track and fix coding errors, reduces risk



# FEATURE-DRIVEN DEVELOPMENT.

- Disadvantages of FDD

1. This agile practice is not good for smaller projects.
2. There is high dependency on lead programmers, designers and mentors.
3. There is lack of documentation which can create an issue afterwards.



# COMPARISON OF AGILE MODEL

Criteria	Extreme Programming (XP)	Scrum	Crystal	DSDM
Principles	Programmer-oriented Based on five values: communication, simplicity, feedback, courage, and respect Small and medium scale project	Project- management oriented Facilitates tracking activities Small and large scale project	Problem-solving for well-defined problem Allow adjusting to project size and criticality Small and medium scale project	Business value is imperative Project that are requirements are flexible Small and large scale project
Methods	Iterative and incremental development planning game, pair programming, refactoring, simple design, continuous integration, test-first programming, collective ownership, coding standards, short releases, metaphor, sustainable pace, on-site customer	Incremental phases adjustment Product backlog, sprint, sprint goal, sprint backlog, daily meeting, sprint review meeting	Iterative and incremental development Project interview, team workshop, reflection workshop	Iterative and incremental development Timeboxing, prioritise requirements, prototyping, regular meeting
Tools	Automated testing and configuration tool	Project management tool	Automated regression testing tool	Requirements analysis, system prototyping, testing and configuration management



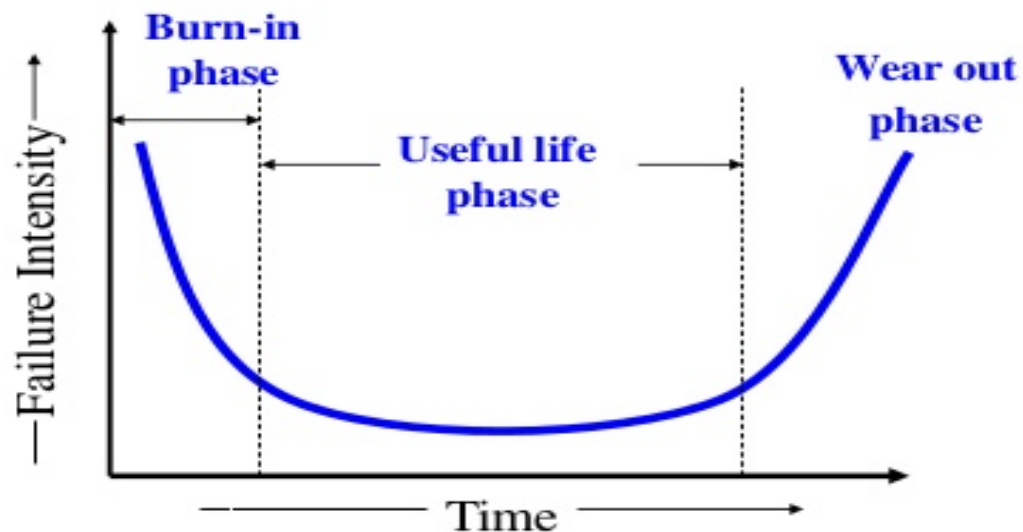
# Software Characteristics

Different individuals judge software on different basis. This is because they are involved with the software in different ways. For example, users want the software to perform according to their requirements. Similarly, developers involved in designing, coding, and maintenance of the software evaluate the software by looking at its internal characteristics, before delivering it to the user. Software characteristics are classified into six major components.

- • **Functionality:** Refers to the degree of performance of the software against its intended purpose.
- • **Reliability:** Refers to the ability of the software to provide desired functionality under the given conditions.
- • **Usability:** Refers to the extent to which the software can be used with ease.
- • **Efficiency:** Refers to the ability of the software to use system resources in the most effective and efficient manner.
- • **Maintainability:** Refers to the ease with which the modifications can be made in a software system to extend its functionality, improve its performance, or correct errors.
- • **Portability:** Refers to the ease with which software developers can transfer software from one platform to another, without (or with minimum) changes. In simple terms, it refers to the ability of software to function properly on different hardware and software platforms without making any changes in it.
- In addition to the above mentioned characteristics, robustness and integrity are also important. **Robustness** refers to the degree to which the software can keep on functioning in spite of being provided with invalid data while **integrity** refers to the degree to which unauthorized access to the software or data can be prevented.

# Software Characteristics

✓ Software does not wear out.



# Software Characteristics

- ✓ Flexibility of Software
- ✓ Reusability of Software

