Software Engineering

[CACS253]

BCA 4th Sem

Er. Sital Prasad Mandal

Mechi Multiple Campus

Bhadrapur, Jhapa, Nepal

https://ctal-softwareeng.blogspot.com/

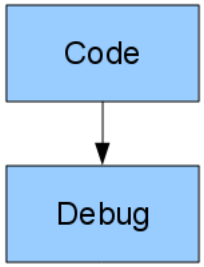
Software Development Process Model Unit-2

1. Software Process
2. Software Process Model:
   - The Waterfall 'Model
   - Evolutionary Development
   - Component-Based Software Engineering (CBSE)
3. Process Iteration:
   - Incremental Delivery
   - Spiral Development
4. Rapid Software Development:
   - Agile Methods
   - Extreme Programming
   - Rapid Application Development
   - Software Prototyping
   - Rational Unified Process (RUP)
5. Computer Aided Software Engineering (CASE):
   - Overview of CASE Approach, Classification of CASE tools

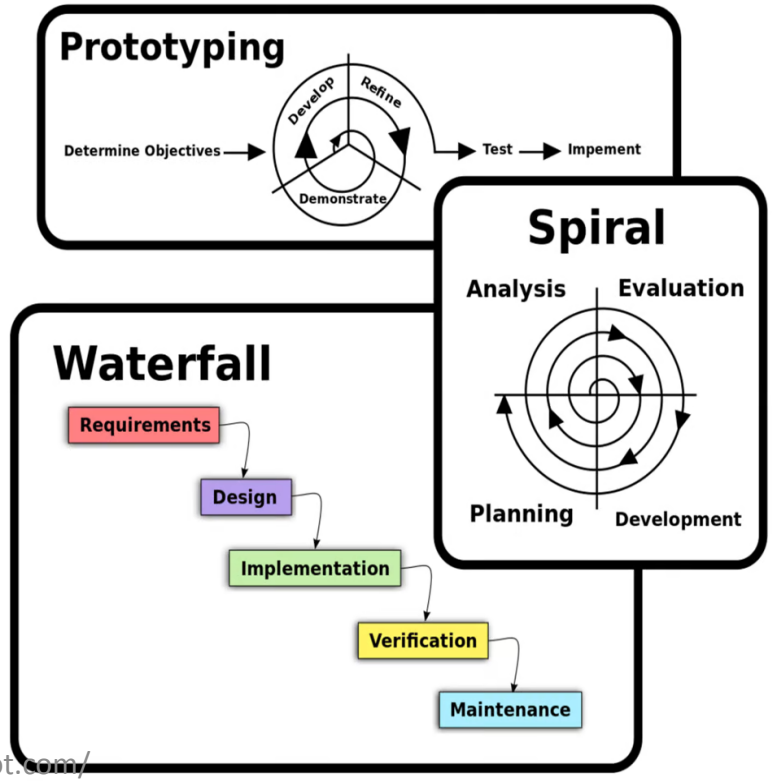
Software Process

- A software process model represents the order in which the activities of software development will be undertaken.
- It describes the sequence in which the phases of the software lifecycle will be performed.

Typical Student Programming Process



Most students are not provided much training in the process of developing software and as a result have a very simplistic procedure they call "programming." There understanding of their own process is quite vague and described with very general terms "coding" and "debugging."



Factors in Choosing a Software Process

1. Customer involvement
2. Stable requirements
3. Team size
4. Developer experience
5. Familiarity with technology
6. Familiarity with domain
7. Severity of impact of incorrect analysis
8. Anticipated changes in technology

Software Process Models

- The term **software** specifies to the set of computer programs, procedures and associated documents (Flowcharts, manuals, etc.) that describe the program and how they are to be used.

- A software process is the set of activities and associated outcome that produce a software product. Software engineers mostly carry out these activities. These are four key process activities, which are common to all software processes. These activities are:

1. **Software specifications**: The functionality of the software and constraints on its operation must be defined.
2. **Software development**: The software to meet the requirement must be produced.
3. **Software validation**: The software must be validated to ensure that it does what the customer wants.
4. **Software evolution**: The software must evolve to meet changing client needs.

Software Process Models

- It is the processes or methodology being selected for the development of project with an aim and goal.
- It specify the various stages in which a development is carried out.
- There are many factors to select a software process models, like project cost, development team, time duration, objective, perspective etc.
- There are various software process models, nowadays, which are as follows:
  - Waterfall Model
  - Rapid Application Development Model
  - Incremental Model
  - Iterative Model
  - Spiral Model
  - Prototype Model
- Selection of right model is very much important.
- Different companies select the model based on their requirements.

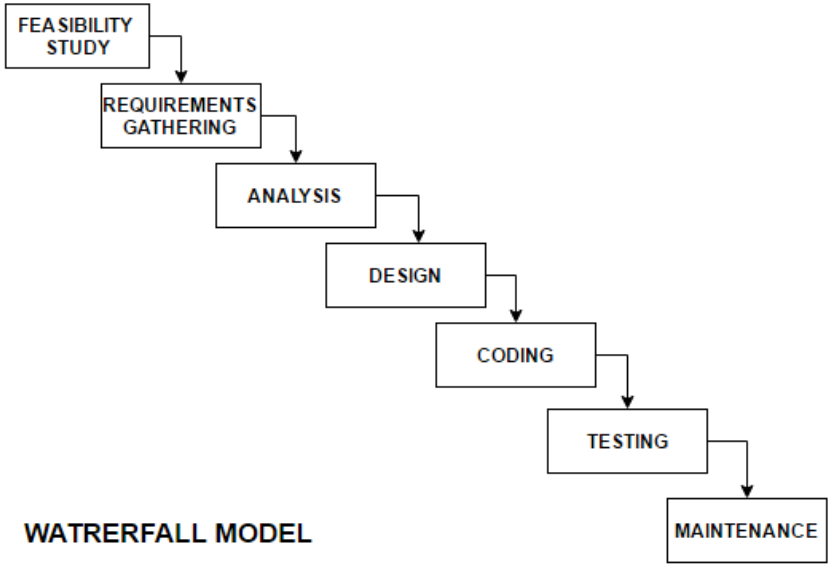
The Waterfall 'Model

- It was the first model introduced.
- It is also known as linear sequential life cycle model.
- It is very simple and understand to use.
- It is basic of all the process development models.
- It is a theoretical model, not to use practically.
- It is called Waterfall because stages goes top to bottom like a natural waterfall.
- When one stage gets completed then only move to next stage.
- Not supposed to come back to previous stage.
- This is considered conventional or classical software life cycle model.

https://www.easyexamnotes.com/p/waterfall-model.html

https://ctal-softwareeng.blogspot.com/

The Waterfall 'Model



WATRERFALL MODEL

- Each phase is carried out completely (for all requirements) before proceeding to the next.
- The process is strictly sequential - no backing up or repeating phases.
- **Advantages:**

  Simple, easy to understand and follow.

  Highly structured, therefore good for beginners.

  After specification is complete, low customer involvement required.
- **Disadvantages:**

  Inflexible - can't adapt to changes in requirements.

The Waterfall 'Model

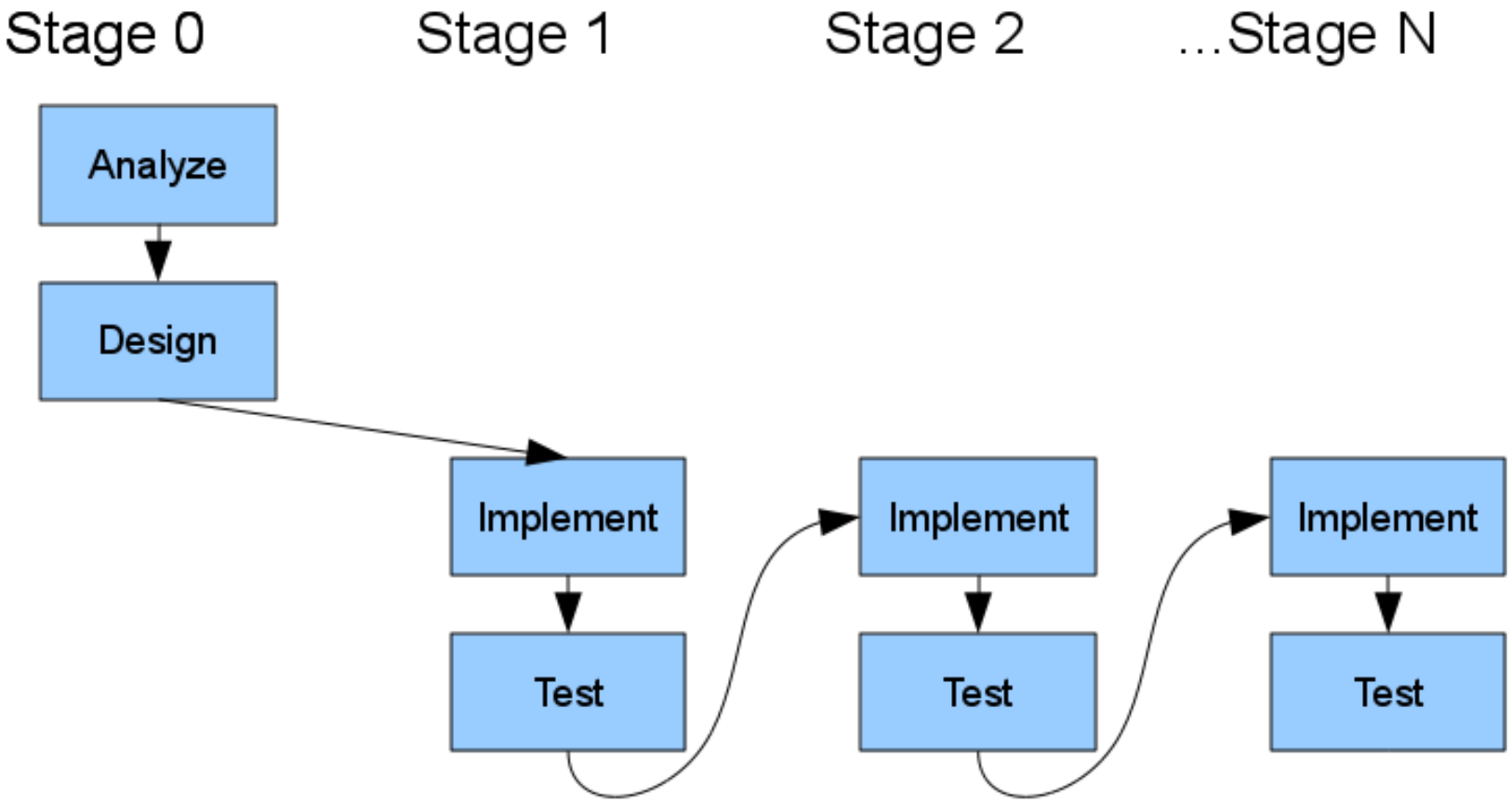
- **When to use Waterfall model:**

  - When the requirements are fixed, understood.
  - Product is understood.
  - Technology is understood.
  - When the project is short.
  - It is costly if project requirements are frequently changing.

- **Why does the Waterfall model sometimes fail?**

  Real projects rarely follows the sequential approach which Waterfall model follows. This creates problem when error occured and we can't go back to the previous stage to correct it.

Evolutionary Model ("Staged Delivery")



Evolutionary Model ("Staged Delivery")

- All analysis and design is done up front.
- Each stage releases some fully functional subset of desired features.
- Emphasis is on high quality releases.
- Can incorporate prototyping.

➢ **Advantages:**

- Partial functionality available early.
- Some flexibility in responding to changing market conditions or customer needs.
- A complete and stable design is produced.
- A compromise between waterfall and agile.

➢ **Disadvantages:**

- Can't easily adapt to entirely new requirements.

Component-Based Software Engineering (CBSE)

- Component Based Software Engineering (CBSE) is a process that focuses on the design and development of computer-based systems with the use of reusable software components.

CBSE Framework Activities

- Framework activities of Component Based Software Engineering are as follows:-
  1. Component Qualification
  2. Component Adaptation
  3. Component Composition
  4. Component Update

Component-Based Software Engineering (CBSE)

- **Component Qualification:**

  This activity ensures that the system architecture define the requirements of the components for becoming a reusable component.Reusable components are generally identified through the traits in their interfaces. It means “the services that are given, and the means by which customers or consumers access these services ” are defined as a part of the component interface.

- **Component Adaptation:**

  This activity ensures that the architecture defines the design conditions for all component and identifying their modes of connection. In some of the cases, existing reusable components may not be allowed to get used due to the architecture’s design rules and conditions. These components should adapt and meet the requirements of the architecture or refused and replaced by other, more suitable components.

Component-Based Software Engineering (CBSE)

- **Component Composition:**

  This activity ensures that the Architectural style of the system integrates the software components and form a working system. By identifying connection and coordination mechanisms of the system, the architecture describes the composition of the end product.

- **Component Update:**

  This activity ensures the updation of reusable components. Sometimes, updates are complicated due to inclusion of third party (the organization that developed the reusable component may be outside the immediate control of the software engineering organization accessing the component currently.

Incremental Delivery

- Incremental model as the name indicated produces many versions.
- Each version with a new functionality.
- In it whole requirements is divided into various builds.
- Each new build carry new functionality over previous build.
- It seems like a multi “Waterfall Model cycle”.
- In this each module passes through requirements, analysis, design, coding, testing, deployment, maintenance etc.
- A first working software is produced in first release of this model.
- Development of new builds continues until the complete system is released.
- In this whole product got ready step by step.

Incremental Delivery

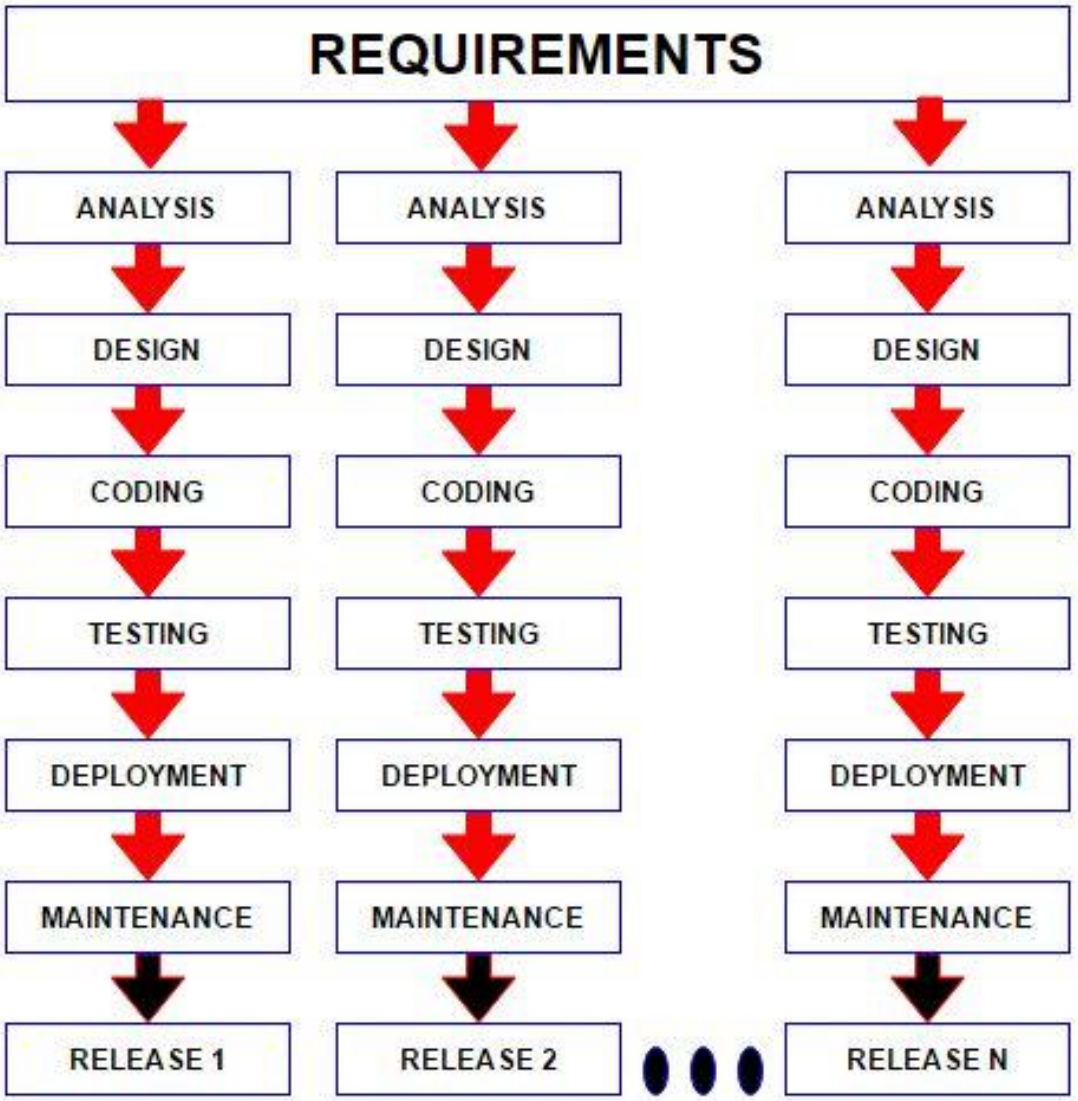


Fig: Illustration of Incremental Model

Incremental Delivery

Advantages of Incremental Model:

- Generates working software quickly.
- More flexible.
- Less costly.
- Gives time to think new function.
- Able to fulfill all and the new requirements of end users without stopping application use.
- Easier to test and debug during smaller iteration.
- Customer can respond to each build.
- Easier to manage.

Disadvantages of Incremental Model:

- Total cost is higher than Waterfall model.
- Each new build requires with a new functionality.
- Requirements can go beyond the scope decided.
- Needs good planning and design.

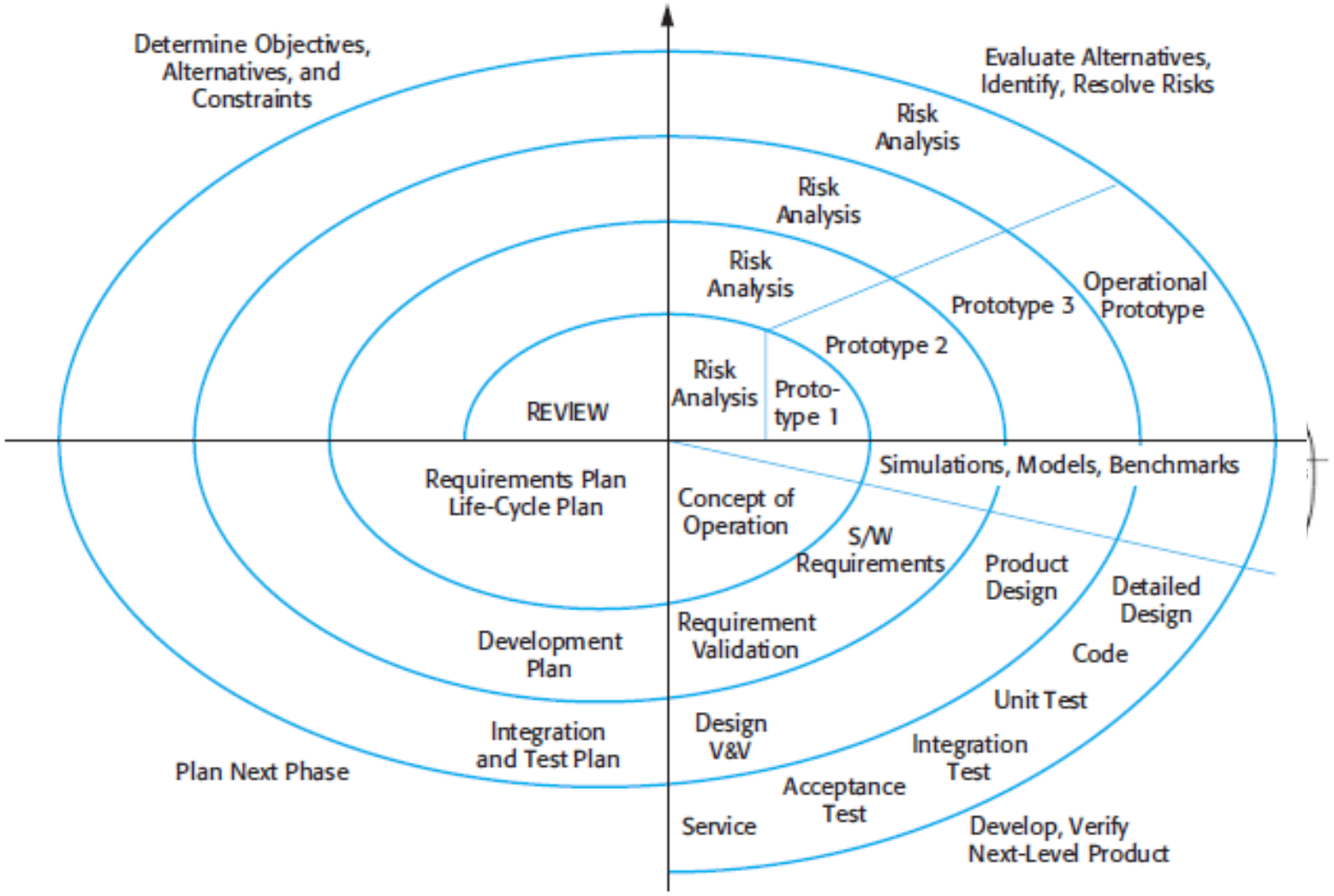
When to use Incremental Model:

- When a new technology is being used.
- When there are some high risk features and goals.
- When there is a need to get the product early in the market.
- When some requirements can be evolve with time.
- When requirements of the system easily understood and well defined.

Spiral Model

- Spiral model is an evolutionary software model.
- Spiral model may be viewed as a Meta model, because it can accommodate any process model.
- Spiral model focuses on identifying and eliminating high risk problems.

Spiral Model



The Spiral Model

https: mode

https://ctai-softwareeng.blogspot.com/

Spiral Model

- First Quadrant: It determine the objective and alternative solution possible for the phase under consideration.
- Second Quadrant: We evaluate different alternatives based on objective and constraint. To resolve risk.
- Third Quadrant: It emphasises development of strategies to resolve the uncertainties and risks.
- Fourth Quadrant: We determine the objective that should be full filled in next cycle to get complete system.

Characteristics of Spiral Model:

- It is cyclic not linear like Waterfall model.
- Each cycle of Spiral Model consist of four stages.
- Each stage is represented by quadrant of Cartesian Diagram.
- Radius of Spiral represent cost accumulated so far in the process.
- Angular dimension represent progress in process.

Spiral Model

- **Advantages of Spiral Model:**
  - It is risk driven model.
  - It is very flexible.
  - Less documentation is needed.
  - It uses prototyping.
  - It is more realistic model for software development.

- **Disadvantages of Spiral Model:**
  - Not suitable for small projects.
  - Cost is very high.
  - Rely on risk assessment expertise.
  - Excellent management skills needed.
  - Involvement of different persons makes it complex too.

Spiral Model

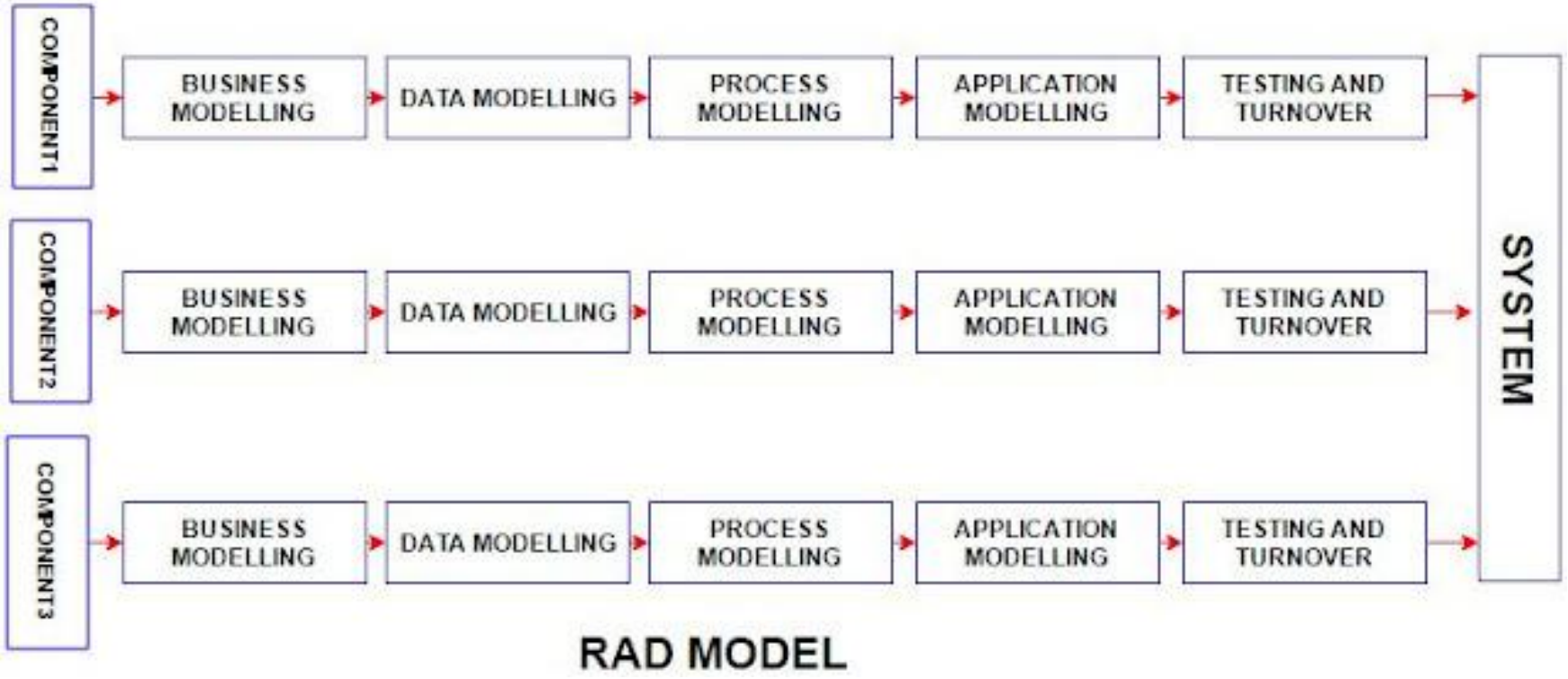
- **Limitations of Spiral Model:**
  - Software development has no strict standard.
  - Particular phase has no particular beginning and end.

- **When to use Spiral Model:**
  - Spiral model is used when experimenting on technology.
  - When trying out new skills.
  - When the user is not able to offer requirements in clear terms.
  - When system is very complex with lot of functions and facilities.
  - When requirements is not clear.
  - When the intended solution has multi user, multi functions, multi features, multi locations applications to be used on multiple platforms.

RAD MODEL:

- Rapid Application Development Model.
- It is a type of Incremental Model.
- It is high speed adaptation of Waterfall model.
- In it projects are developed in component as mini projects.
- Than mini projects are assembled in a single project.
- It takes very short period of time to construct a project.
- Customers can give feedback easily on mini projects as well as on complete project.

RAD MODEL:



RAD MODEL

RAD MODEL:

Advantages of RAD Model:

- Increases re-usability of components.
- Less development time.
- Quick initial responses.
- Encourages customer feedback's.
- Involvement of end users from very beginning solves lot of development issues.

Disadvantages of RAD Model:

- Projects which can be developed into mini projects/components can use RAD Model.
- Requires strong and skilled developers team.
- Cost is very high..

When to use RAD Model:

- When there is need to develop system within short period of time (2-3 months).
- When there is high availability of developers.
- When budget is high enough to afford the development cost.

Agile Methods

What is Agile?

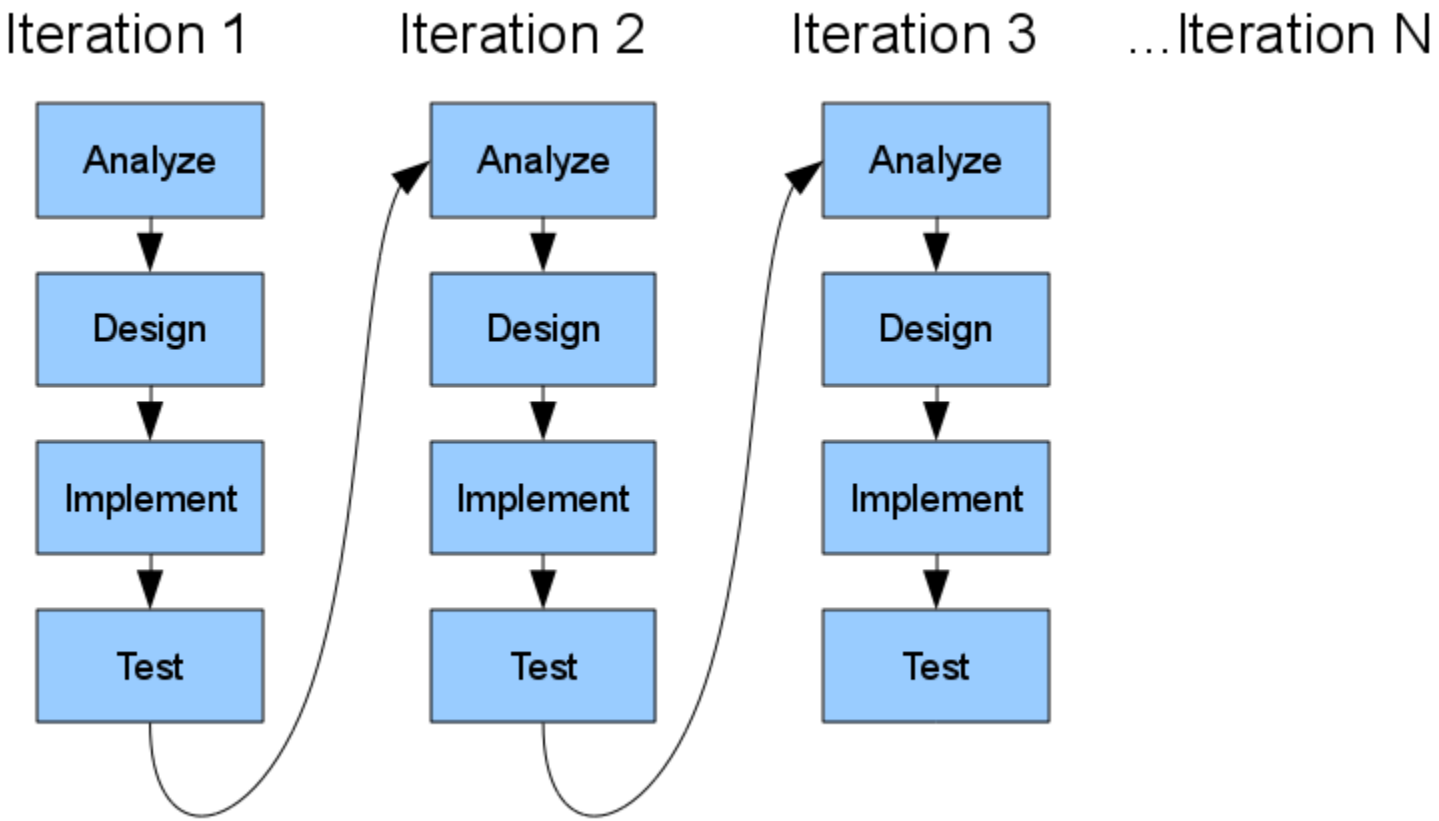
The word ‘agile’ means −

- Able to move your body quickly and easily.
- Able to think quickly and clearly.

Ref: Cambridge Dictionaries online:

In software development, the term ‘agile’ is adapted to mean ‘the ability to respond to changes − changes from Requirements, Technology and People.’

Agile Methods



Agile Methods

An iterative approach. During each iteration a single feature or small set of features are chosen and implemented completely.

Advantages:

- Can adapt to changing requirements because you haven't committed to big design that encompasses everything.
- Easy to change direction to adapt to dynamic market conditions.

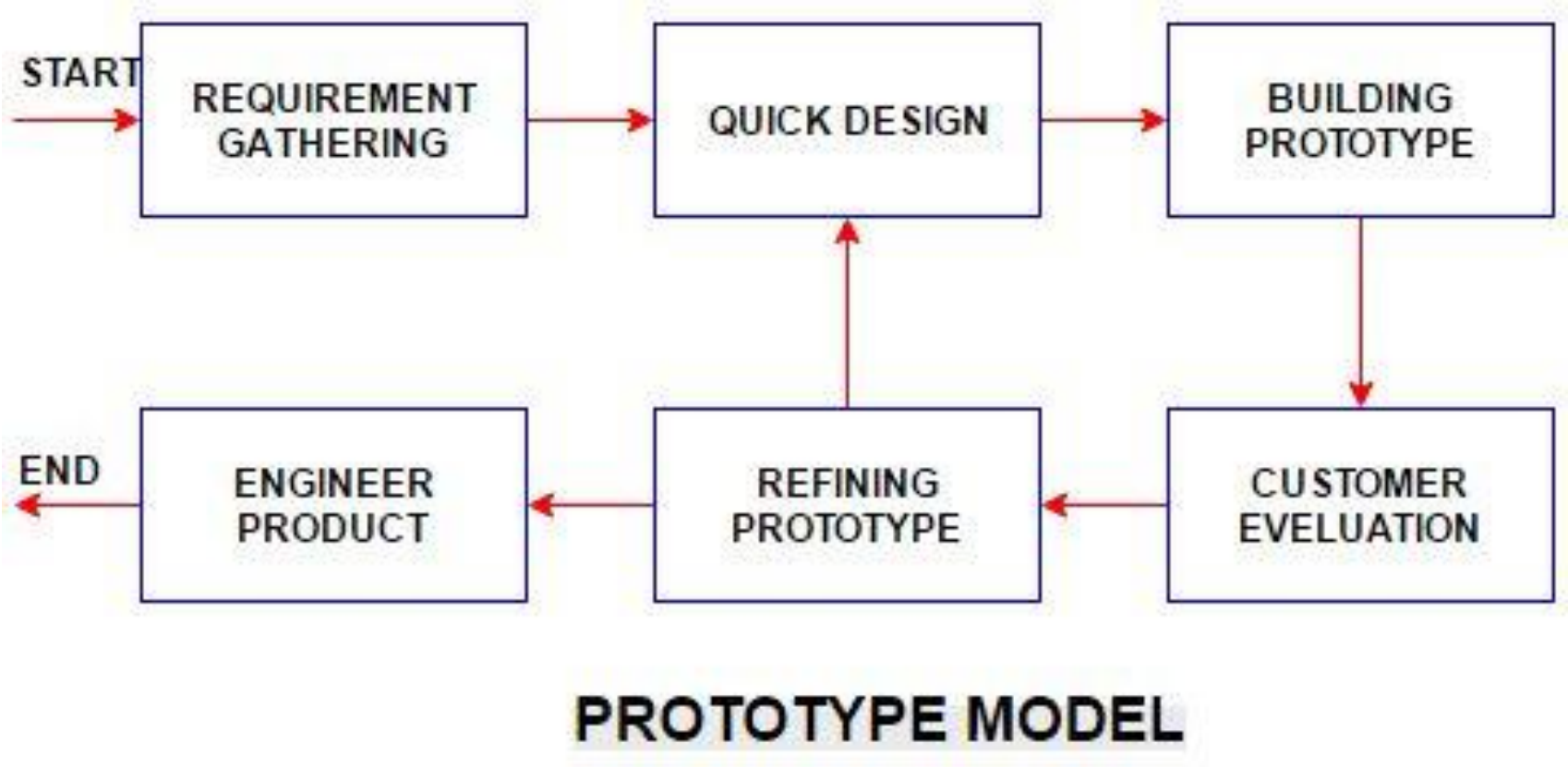
Disadvantages:

- Used as an excuse for hacking - proceeding without a plan.
- Substantial refactoring or redesign may be needed between iterations.
- Not suitable for large projects or large teams.
- Requires huge customer involvement, which is unusual to find.

Prototype Model

- It requires that before carrying out actual software its prototype (model) must be created.
- Prototype model is a toy implementation of system.
- A prototype usually a demo version of actual system, possibly with limited functionality, low reliability, and inefficient performance compare to actual system.
- Detailed information is not available in it.
- Idea behind it is to create a prototype to gather the basic requirements.
- Prototype is built on the basis of current available requirements.
- It gives the “actual feel” of the system.
- Prototype is not complete system many of the details are not built into the prototype.
- The goal is to provide system with overall functionality.

Prototype Model



PROTOTYPE MODEL

Prototype Model

Advantages of Prototype Model:

- Working model of system is provided, so user gets a better understanding.
- Errors can be detected and corrected easily.
- User feedback quickly and easily available which leads to a better solution.
- Missing functionality can be easily identified.
- Users are actively involved in development of system.

Disadvantages of Prototype Model:

- Leads to complexity of the system, as scope of the system may expand beyond the original plan,
- Incomplete application sometimes leads to confusion of actual system working.

When to use Prototype Model:

- When lot of interaction with the user is needed.
- In case of online system, web interfaces, where end user interaction is very high. Prototype model is best suited.
- When consistent feedback from user is required.

RUP:

- Rational Unified Processes.
- It is a software engineering process management tool encapsulating best practices in software development and maintenance.RUP includes following some of the best practices:
  - Iterative and incremental development of software.
  - Requirements management.
  - Software components reuse.
  - Visual Modelling.
  - Software quality verification.
  - Software configuration.
  - Software change management.
- RUP consists of well defined sequential phases for iterative development of software.
- RUP includes following phases:

  Inception, Elaboration, Construction & Transition

RUP:

1. Inception Phase:

- During it development team develop scope of project.
- Stakeholder of the project are identified.
- Stakeholders interfaces with project are identified.
- All use cases are identified.
- Risk management plan is developed.
- Cost as well as schedule estimations are developed.
- Its main outputs are:
  - Vision and scope documents.
  - Initial risk management plan.
  - Initial use case model.
  - Initial prototype.

RUP:

2. Elaboration Phase:

- Use model is formalized.
- Executable prototype is implemented.
- Its main outputs are:
  - Use case model
  - Non functional requirements specification.
  - Revised and complete project plan.
  - Revised risk management plan.
  - More accurate estimation of the cost and time needed in project development.
  - An executable prototype.

RUP:

3. Construction Phase:

Software is designed, integrated and implemented.

Its main outputs are:

- Software design document.
- Software Product.
- Installation document.
- Manual document.
- All other documents which required.

4. Transition Phase:

- The software is deployed.
- The software is made ready for the use of user.
- Product is validated.
- Necessary database conversions are performed.
- Training for software use is provided.
- Software is distributed to the users via available channels or marketing.

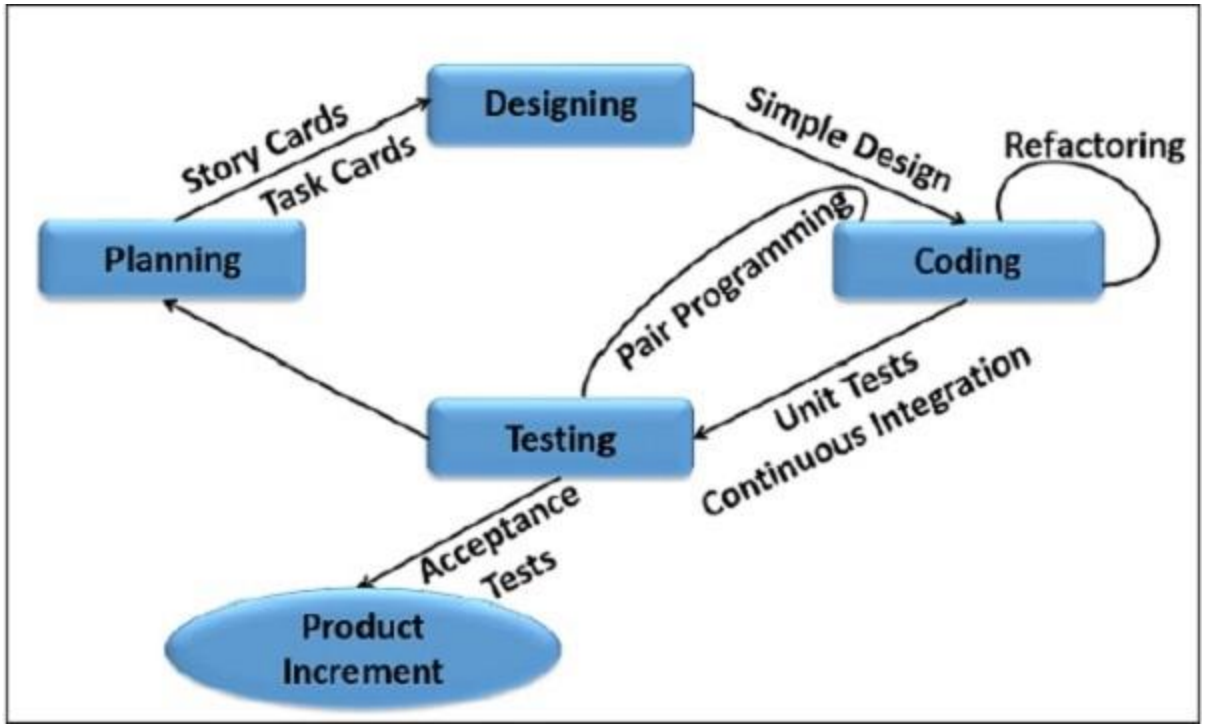
RUP:

RUP Workflows:

- Business Modelling
- Requirements
- Analysis and Design
- Implementation
- Testing
- Deployment
- Project management
- Configuration and change management
- Environment

Extreme Programming (XP)

- XP is a lightweight, efficient, low-risk, flexible, predictable, scientific, and fun way to develop a software.
- Extreme Programming is one of the Agile software development methodologies.

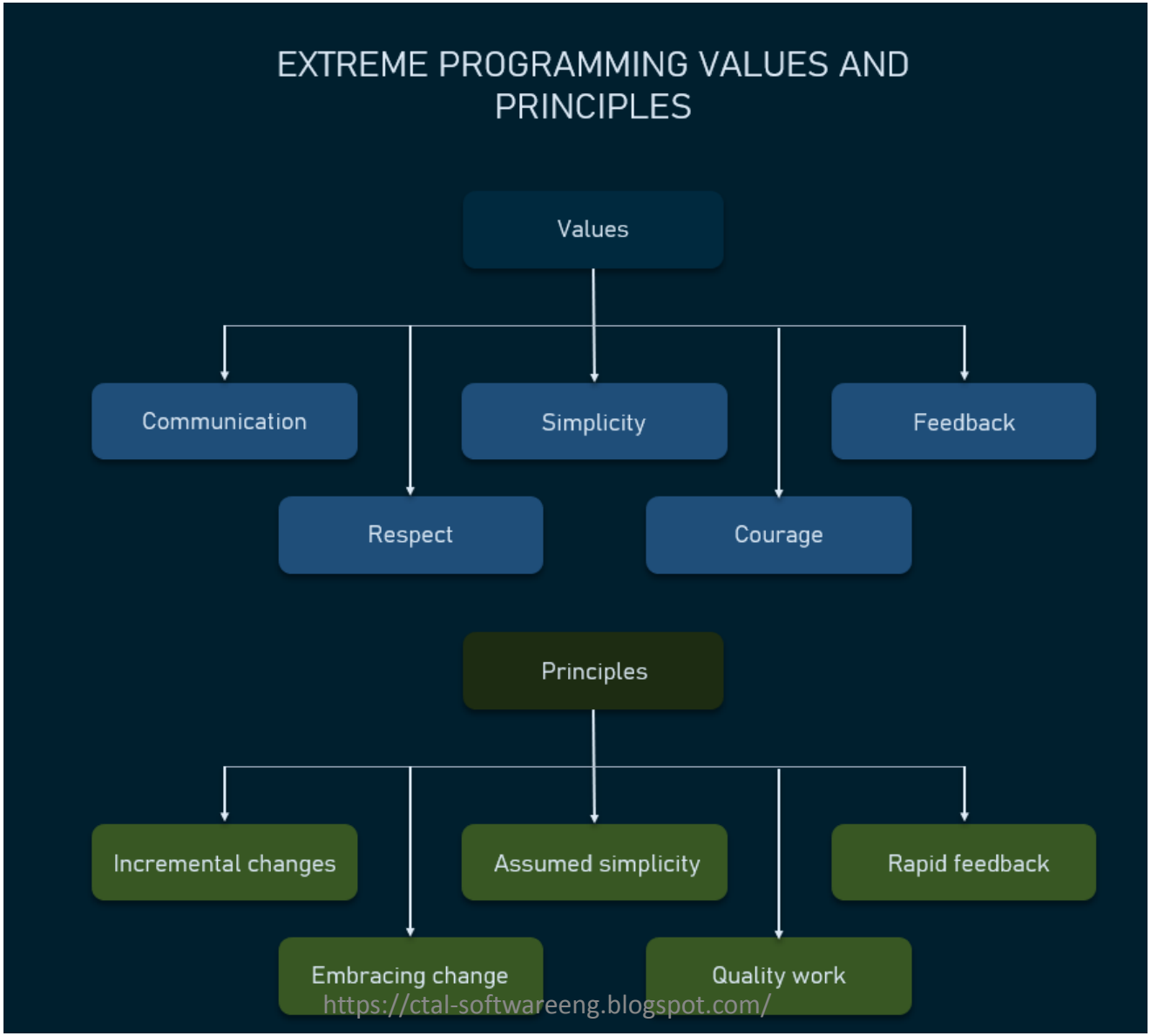


XP framework normally involves 5 phases or stages of the development process that iterate continuously.

Extreme Programming (XP)

- It is used to improve software quality and responsive to customer requirements.
- The extreme programming model recommends taking the best practices that have worked well in the past in program development projects to extreme levels.
- Extreme programming puts people in the center of the system, emphasizing the value and importance of such social skills as communication, cooperation, responsiveness, and feedback.

Extreme Programming (XP)



Extreme Programming (XP)

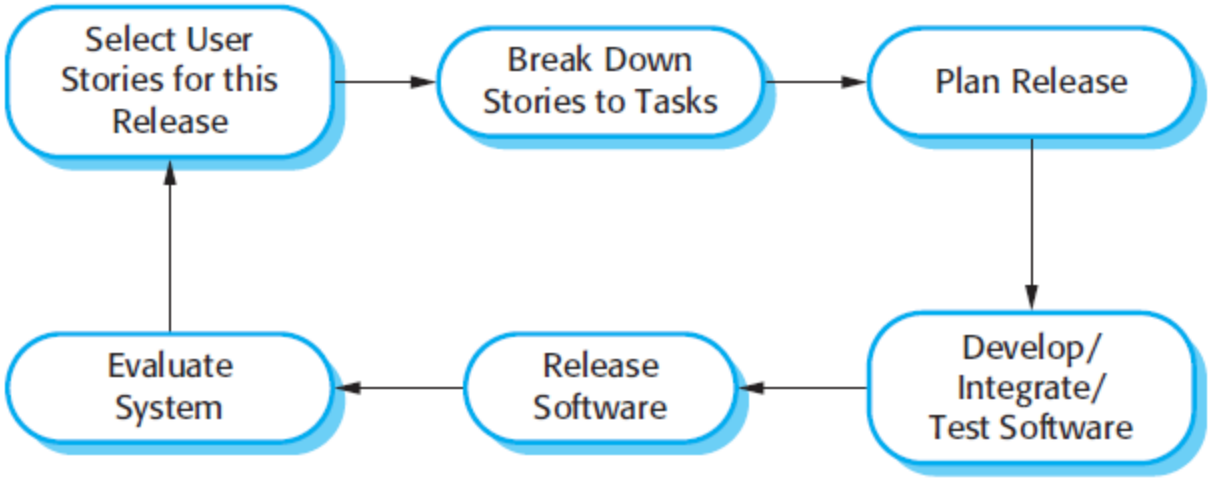


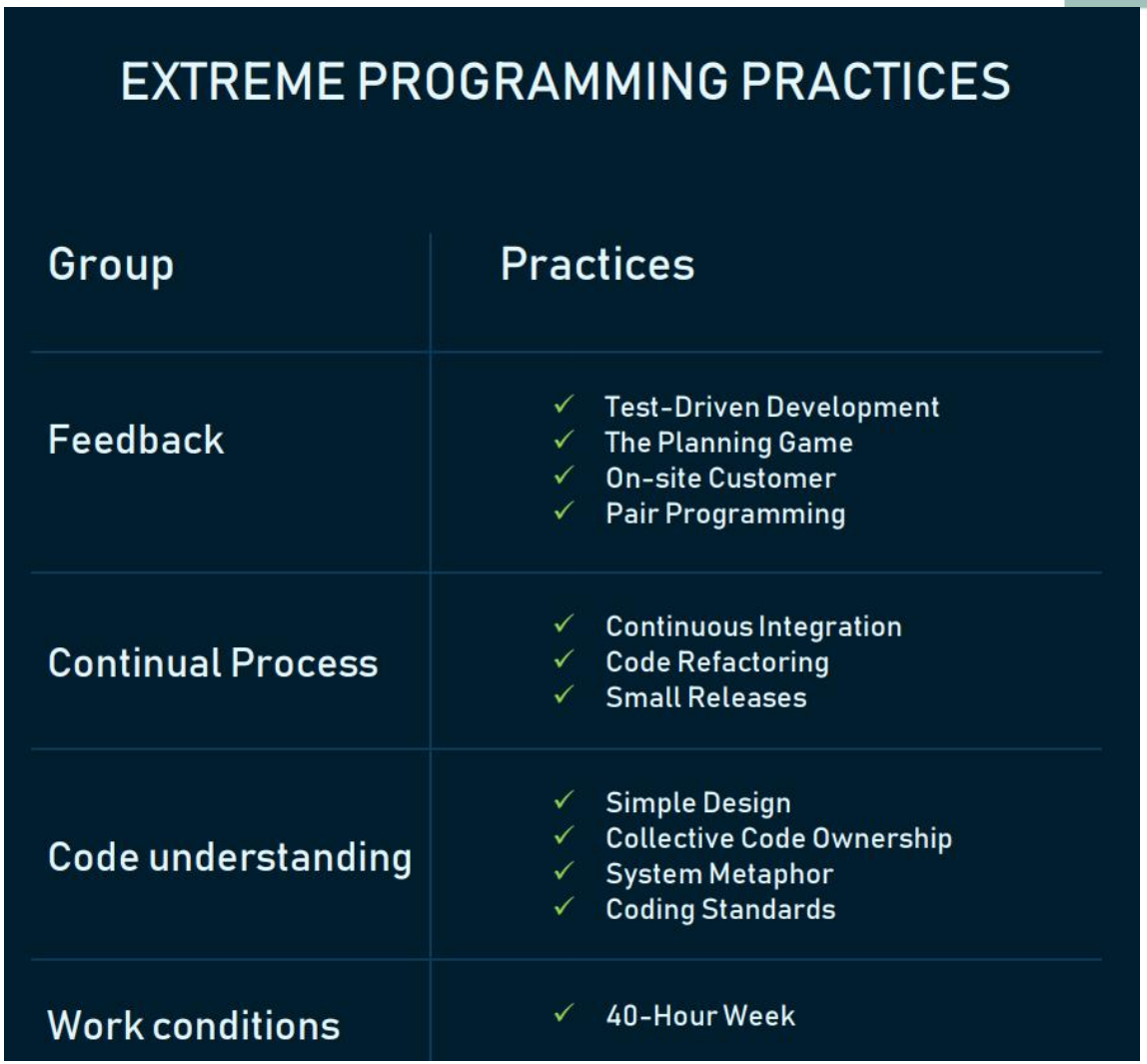
Figure: The extreme programming release cycle

Extreme Programming (XP)

good practices are	**pushed to the *extreme*.**
Code reviews	review code all the time (pair programming)
Testing	everybody will test all the time (unit testing) even the customers (functional testing)
Design	make it part of everybody's daily business (refactoring)
Simplicity	always leave the system with the simplest design that supports current functionality (simplest thing that could possibly work)
Architecture	everybody will work defining and refining the architecture all the time (metaphor)
Integration testing	integrate and test several times a day (continuous integration)
Short iterations	make iterations really short-seconds, minutes, hours not weeks, months, years (the planning game)

Table: Why it's called eXtreme

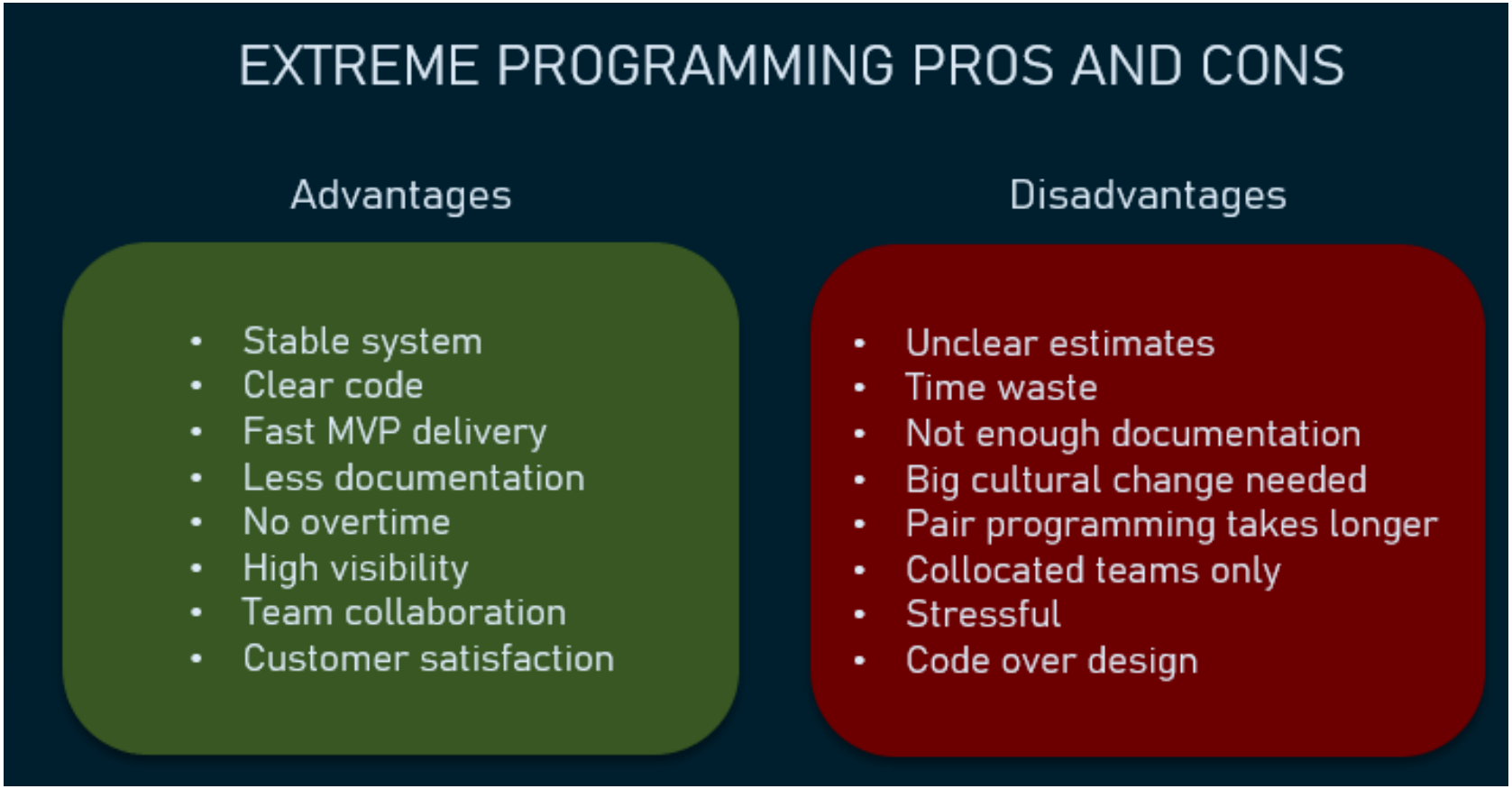
Extreme Programming (XP)



Extreme programming practices

https://ctal-softwareeng.blogspot.com/

Extreme Programming (XP)



minimum viable product (MVP)

https://ctal-softwareeng.blogspot.com/

Extreme programming

Extreme Programming (XP) is an agile software development framework that aims to produce higher quality software, and higher quality of life for the development team.

XP is the most specific of the agile frameworks regarding appropriate engineering practices for software development.

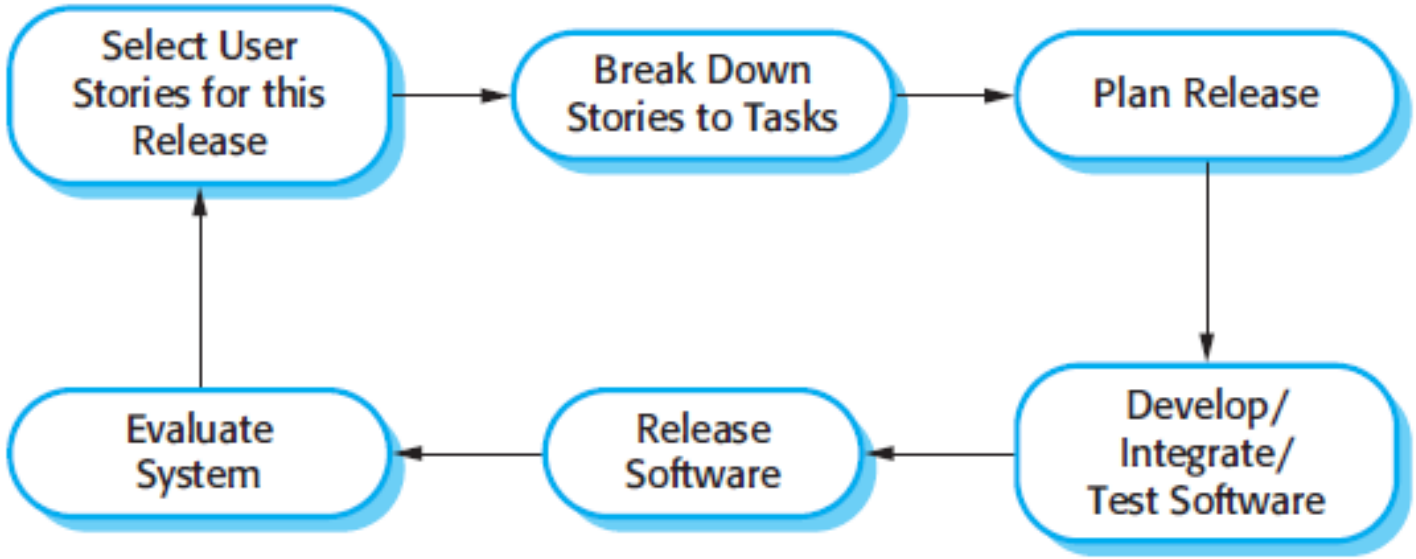
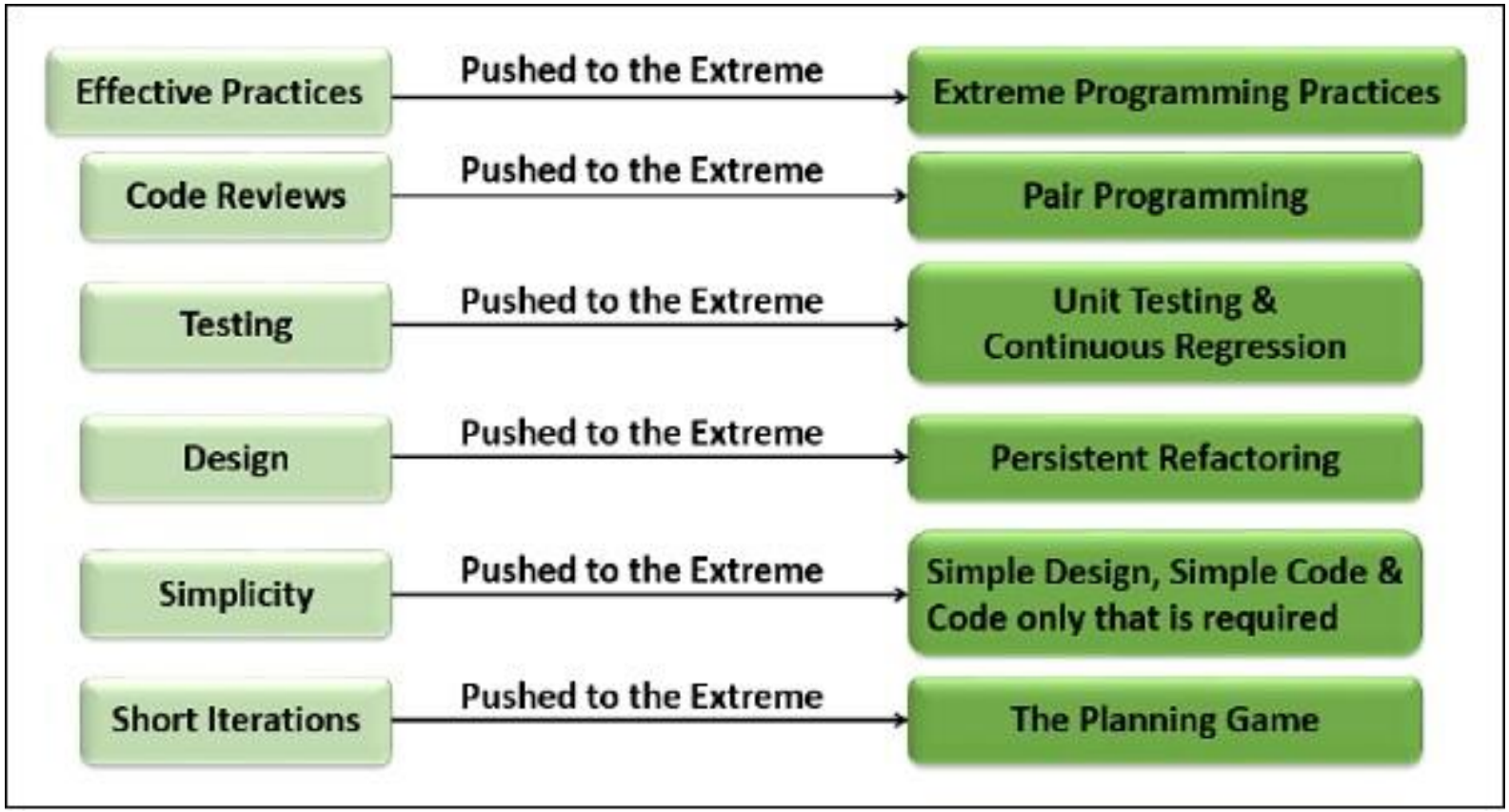


Figure : The extreme programming release cycle

Why is it called “Extreme?”

Extreme Programming takes the effective principles and practices to extreme levels.



Why is it called “Extreme?”

Extreme Programming takes the effective principles and practices to extreme levels.

- ✓ Code reviews are effective as the code is reviewed all the time.
- ✓ Testing is effective as there is continuous regression and testing.
- ✓ Design is effective as everybody needs to do refactoring daily.
- ✓ Integration testing is important as integrate and test several times a day.
- ✓ Short iterations are effective as the planning game for release planning and iteration planning.

Computer Aided Software Engineering (CASE):

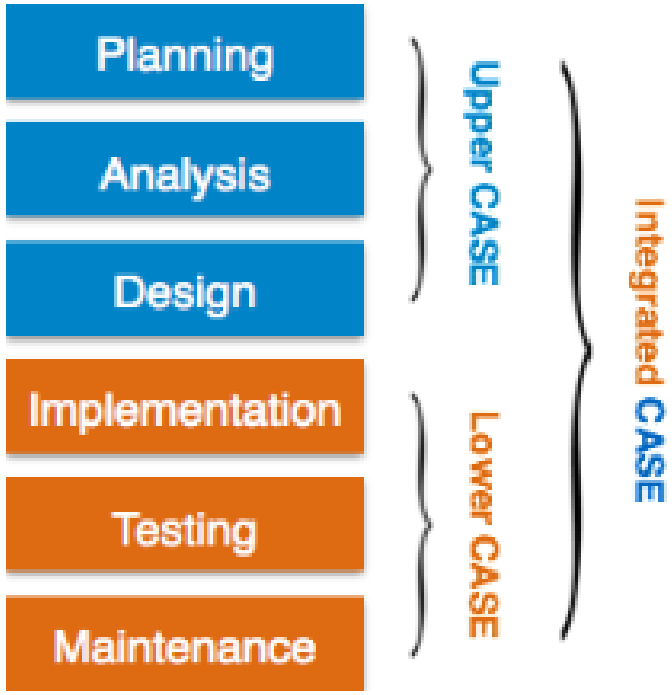
CASE stands for Computer Aided Software Engineering. It means, development and maintenance of software projects with help of various automated software tools.

CASE Tools

- CASE tools are set of software application programs, which are used to automate SDLC activities. CASE tools are used by software project managers, analysts and engineers to develop software system.
- There are number of CASE tools available to simplify various stages of Software Development Life Cycle such as Analysis tools, Design tools, Project management tools, Database Management tools, Documentation tools are to name a few.

Computer Aided Software Engineering (CASE):

- Software design is usually completed with at least some assistance from Computer-Aided Software Engineering tools, or CASE tools.
- CASE is basically the use of computer-based support by developers to develop and maintain software, especially on larger scale, or for more complex projects.



https://ctal-softwareeng.blogspot.com/

Computer Aided Software Engineering (CASE):

CASE stands for Computer Aided Software Engineering. It means, development and maintenance of software projects with help of various automated software tools.

CASE Tools

- CASE tools are set of software application programs, which are used to automate SDLC activities. CASE tools are used by software project managers, analysts and engineers to develop software system.
- There are number of CASE tools available to simplify various stages of Software Development Life Cycle such as Analysis tools, Design tools, Project management tools, Database Management tools, Documentation tools are to name a few.

Computer Aided Software Engineering (CASE):

CASE Classification / Components of CASE Tools

1. ***Upper Case Tools*** *- Upper CASE tools are used in planning, analysis and design stages of SDLC.*
2. ***Lower Case Tools*** *- Lower CASE tools are used in implementation, testing and maintenance.*
3. ***Integrated Case Tools*** *- Integrated CASE tools are helpful in all the stages of SDLC, from Requirement gathering to Testing and documentation.*
4. ***Central Repository*** - CASE tools require a central repository, which can serve as a source of common, integrated and consistent information. Central repository is a central place of storage where product specifications, requirement documents, related reports and diagrams, other useful information regarding management is stored. Central repository also serves as data dictionary.