Software Process Models

Instructor: Mehroze Khan

Software Life Cycle

When the process involves the building of some product, we sometimes refer to the process as a **life cycle**. Thus, the software development process is sometimes called the **software life cycle**, because it describes the life of a software product from its conception to its implementation, delivery, use, and maintenance.

Software Life Cycle

Software development usually involves the following stages:

- Requirements analysis and definition
- System design
- Program design
- Writing the programs (program implementation)
- Unit testing
- Integration testing
- System testing
- System delivery
- Maintenance

Process

- A process is defined as a collection of work activities, actions, and tasks that are performed when some work product is to be created.
- Each of these activities, actions, and tasks reside within a framework or model that defines their relationship with the process and with one another.
- A *software process* is a framework for the activities, actions, and tasks that are required to build high-quality software.

Generic Process Framework

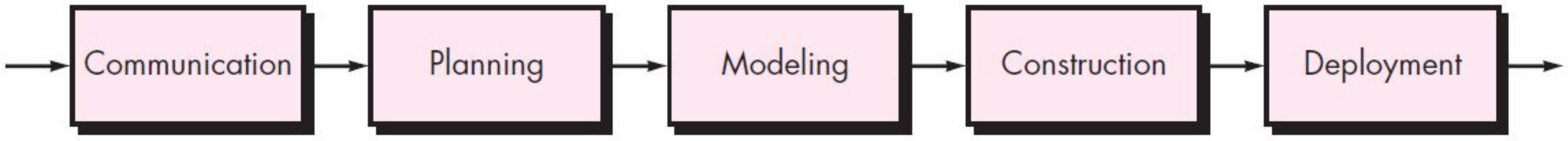
A generic process framework for software engineering defines five framework activities:

- Communication
- Planning
- Modeling
- Construction
- Deployment

Process Flow

- ***Process flow***—describes how the framework activities and the actions and tasks that occur within each framework activity are organized with respect to sequence and time.

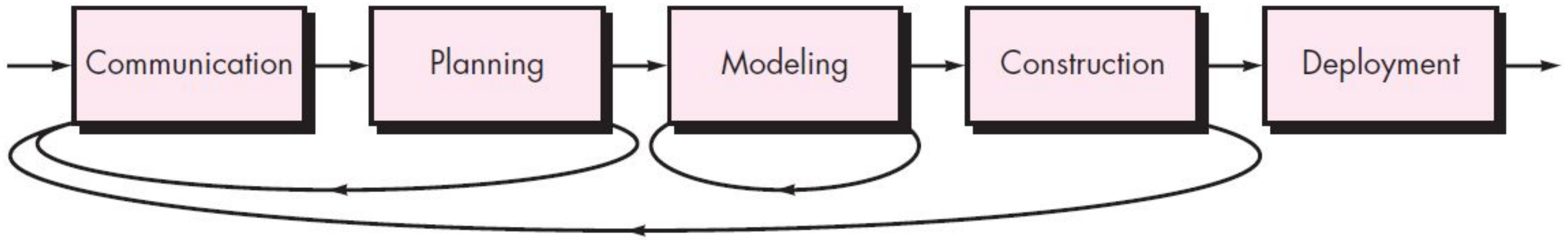
- A ***linear process flow*** executes each of the five framework activities in sequence, beginning with communication and culminating with deployment.



(a) Linear process flow

Process Flow

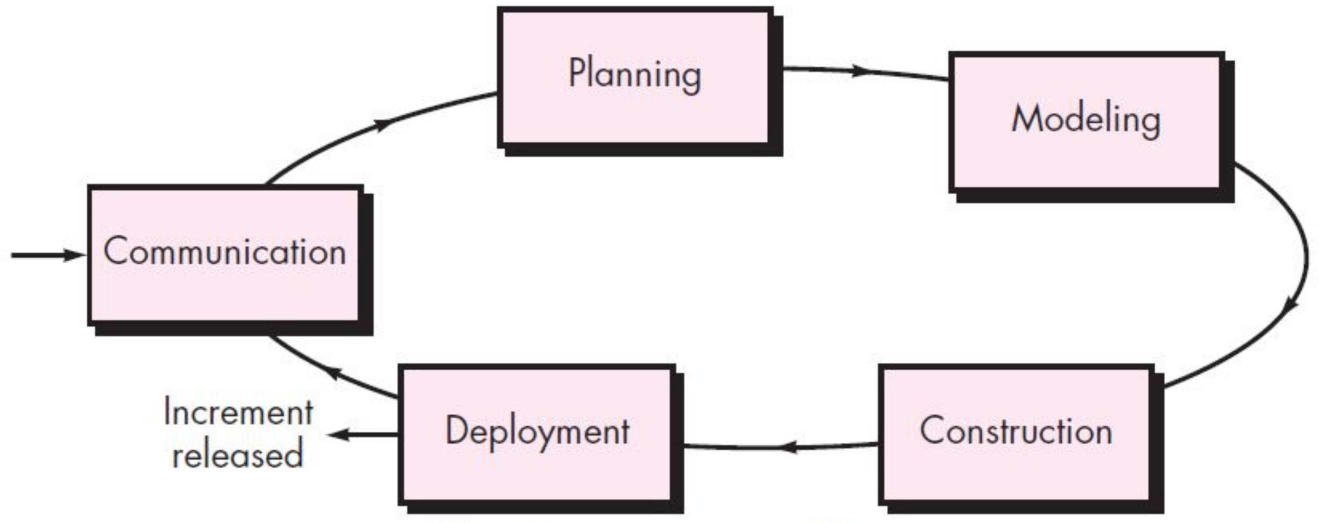
- An ***iterative process flow*** repeats one or more of the activities before proceeding to the next.



(b) Iterative process flow

Process Flow

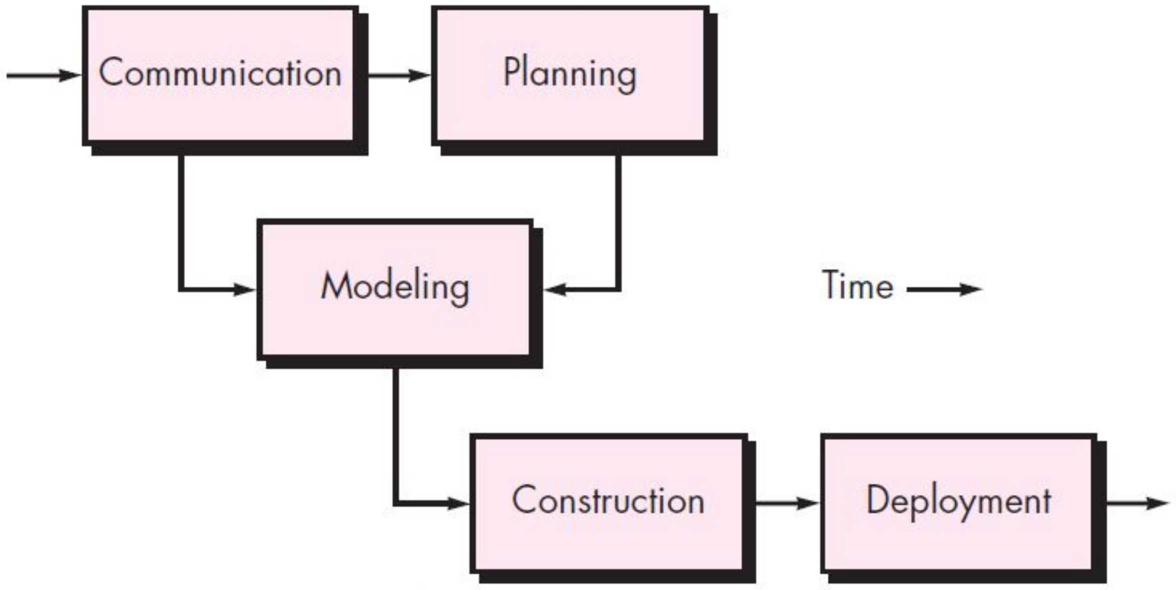
- An ***evolutionary process flow*** executes the activities in a “circular” manner. Each circuit through the five activities leads to a more complete version of the software.



(c) Evolutionary process flow

Process Flow

- A ***parallel process flow*** executes one or more activities in parallel with other activities (e.g., modeling for one aspect of the software might be executed in parallel with construction of another aspect of the software).



(d) Parallel process flow

Testing Types

- **Unit Testing:** It focuses on the smallest unit of software design. In this, we test an individual unit or group of interrelated units.
- **Integration Testing:** The objective is to take unit-tested components and build a program structure that has been dictated by design. Integration testing is testing in which a group of components is combined to produce output.
- **Regression Testing:** Every time a new module is added leads to changes in the program. This type of testing makes sure that the whole component works properly even after adding components to the complete program.
- **Smoke Testing:** This test is done to make sure that the software under testing is ready or stable for further testing.

Testing Types

- **Alpha Testing:** This is a type of validation testing. It is a type of *acceptance testing* which is done before the product is released to customers.
- **Beta Testing:** The beta test is conducted at one or more customer sites by the end-user of the software. This version is released for a limited number of users for testing in a real-time environment.
- **System Testing:** The software is tested such that it works fine for the different operating systems.
- **Acceptance Testing:** Acceptance testing is done by the customers to check whether the delivered products perform the desired tasks or not, as stated in requirements.

Task Set

For a small software project requested by one person (at a remote location) with simple, straightforward requirements, the communication activity might encompass little more than a phone call with the appropriate stakeholder. Therefore, the only necessary action is *phone conversation,* and the work tasks (the *task set*) that this action encompasses are:

1. Contact stakeholder via telephone.
2. Discuss requirements and take notes.
3. Organize notes into a brief written statement of requirements.
4. E-mail to stakeholder for review and approval.

Task Set

If the project was considerably more complex with many stakeholders, each with a different set of (sometime conflicting) requirements, the communication activity might have six distinct actions:

- *Inception*
- *Elicitation*
- *Elaboration*
- *Negotiation*
- *Specification*
- *Validation.*

Software Development Process Models

- Waterfall Model
- V Model
- Incremental Model
- Prototyping Model
- Evolutionary Process Model
  - Spiral
- Unified Process Model
- Rapid Application Development (RAD)
- Agile Models
  - XP
  - Scrum
  - Kanban

The Waterfall Model

- There are times when the requirements for a problem are well understood—when work flows from **communication** through **deployment** in a reasonably linear fashion.
- This situation is sometimes encountered when well-defined adaptations or enhancements to an existing system must be.
- The *waterfall model,* sometimes called the ***linear sequential model***, suggests a systematic, sequential approach to software development that begins with customer specification of requirements and progresses through planning, modeling, construction, and deployment.

Problems with Waterfall Model

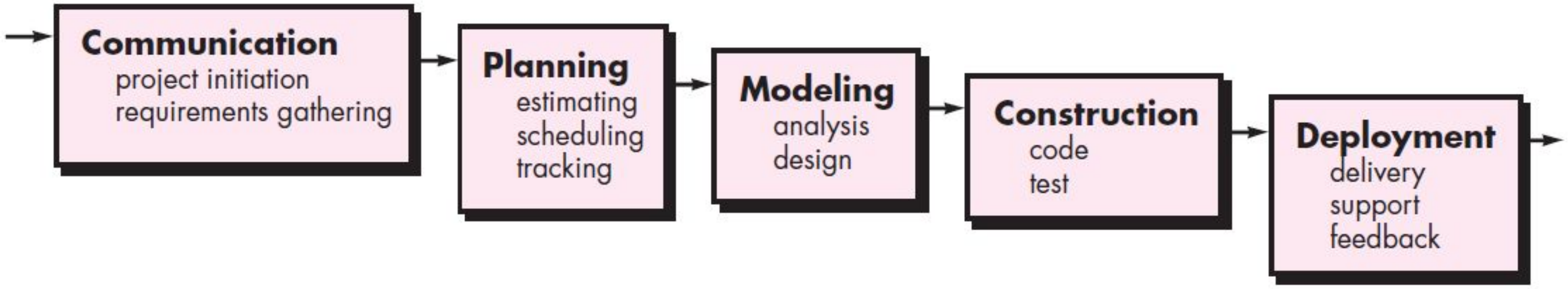
Real projects rarely follow the sequential workflow that the model proposes.

It is often difficult for the customer to state all requirements explicitly at the beginning of most projects.

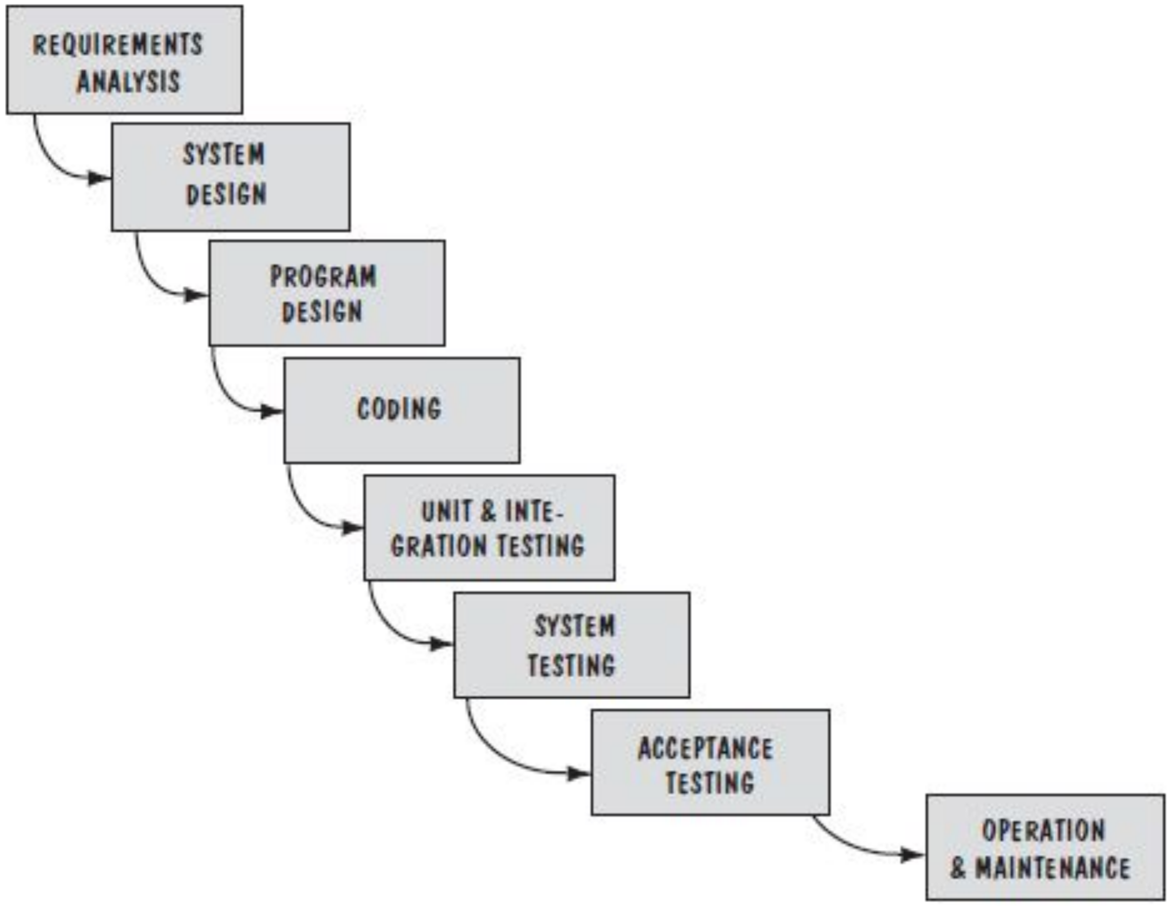
The customer must have patience because a working version of the program(s) will not be available until late in the project time span.

Major blunders may not be detected until the working program is reviewed.

The Waterfall Model



The Waterfall Model



The Waterfall Model

Pros:

- It is easy to understand and plan.
- It works for well-understood small projects.
- Analysis and testing are straightforward.

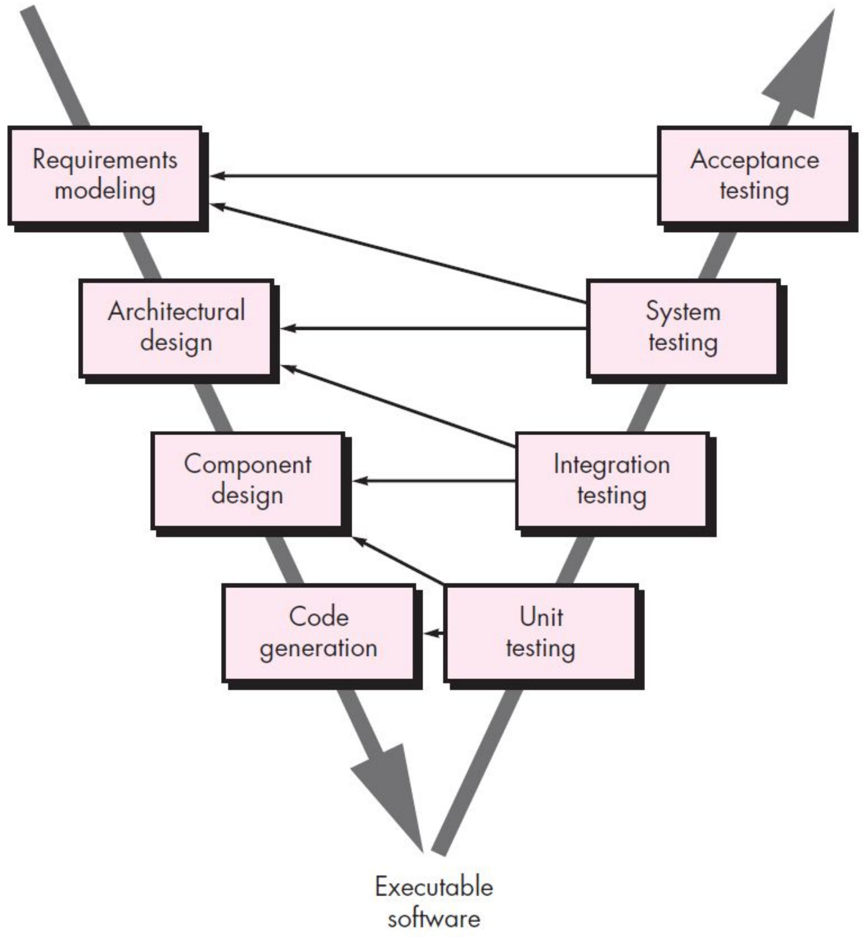
Cons:

- It does not accommodate change well.
- Testing occurs late in the process.
- Customer approval is at the end.

The V-Model

- A variation in the representation of the waterfall model is called the ***V-model*.**
- The V-model depicts the relationship of quality assurance actions to the actions associated with communication, modeling, and early construction activities.
- As a software team moves down the left side of the V, basic problem requirements are refined into progressively more detailed and technical representations of the problem and its solution.
- Once code has been generated, the team moves up the right side of the V, essentially performing a series of tests (quality assurance actions) that validate each of the models created as the team moved down the left side.

The V-Model



Requirements modeling
Acceptance testing
Architectural design
System testing
Component design
Integration testing
Code generation
Unit testing
Executable software

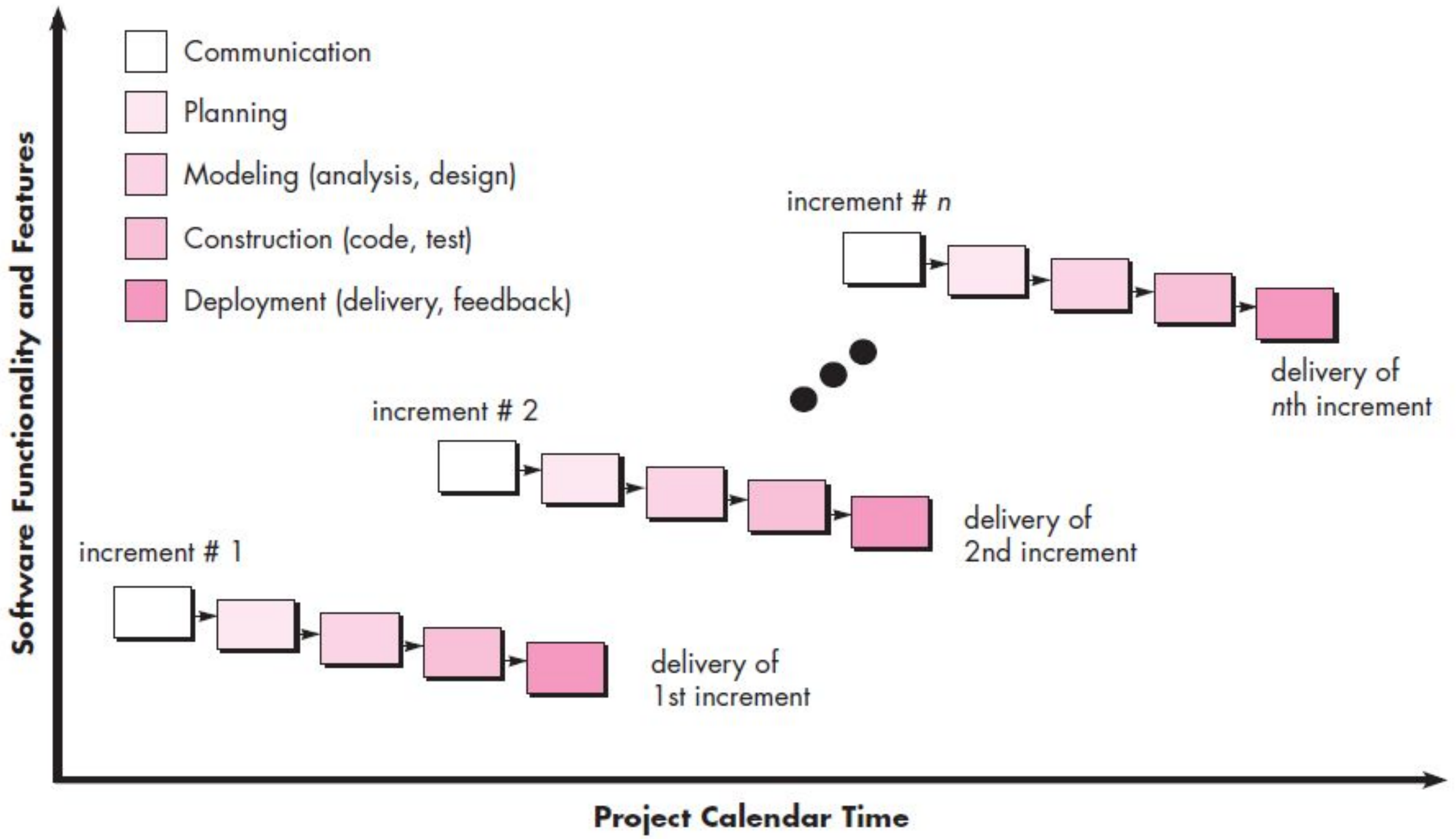
The Incremental Model

- There may be a compelling need to provide a limited set of software functionality to users quickly and then refine and expand on that functionality in later software releases. In such cases, you can choose a process model that is designed to produce the software in increments.
- The incremental model applies linear sequences in a staggered fashion as calendar time progresses. Each linear sequence produces deliverable “increments” of the software.

Example of Word Processor

- Word-processing software developed using the incremental paradigm might deliver:
  - Basic file management, editing, and document production functions in the first increment.
  - More sophisticated editing and document production capabilities in the second increment.
  - Spelling and grammar checking in the third increment.
  - Advanced page layout capability in the fourth increment.

The Incremental Model



Communication
Planning
Modeling (analysis, design)
Construction (code, test)
Deployment (delivery, feedback)
increment # 1
delivery of 1st increment
increment # 2
delivery of 2nd increment
increment # *n*
delivery of *n*th increment
Software Functionality and Features
Project Calendar Time

The Incremental Model

Pros:

- Early increments can be implemented with fewer people.
- Develop high risk or major functions first.
- Customer can respond to each build.
- Risk of changing requirements is reduced.

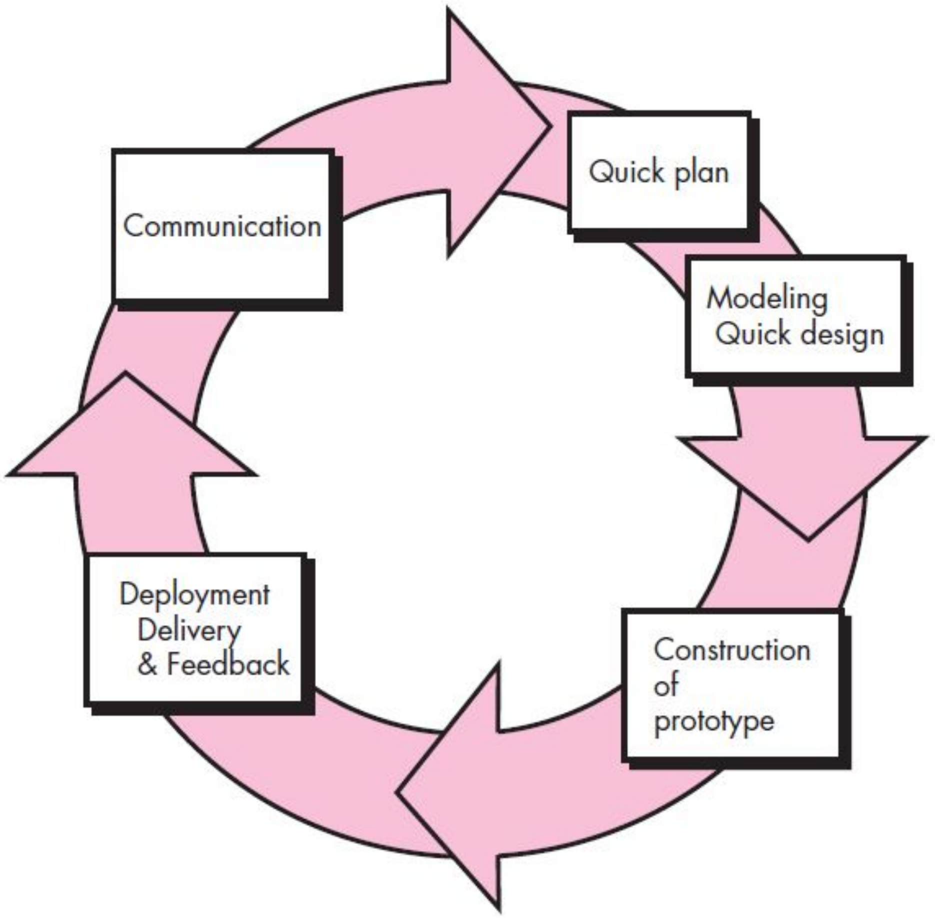
Cons:

- Requires good planning and design.
- Well defined module interfaces are required.

Prototyping Model

- Often, a customer defines a set of general objectives for software but does not identify detailed requirements for functions and features.
- The prototyping paradigm assists you and other stakeholders to better understand what is to be built when requirements are fuzzy.

Prototyping Model



Prototyping Model

Pros:

- There is a reduced impact of requirement changes.
- The customer is involved early and often.
- It works well for small projects.
- There is reduced likelihood of product rejection.

Cons:

- Customer involvement may cause delays.
- There may be a temptation to “ship” a prototype.
- Work is lost in a throwaway prototype.
- It is hard to plan and manage.

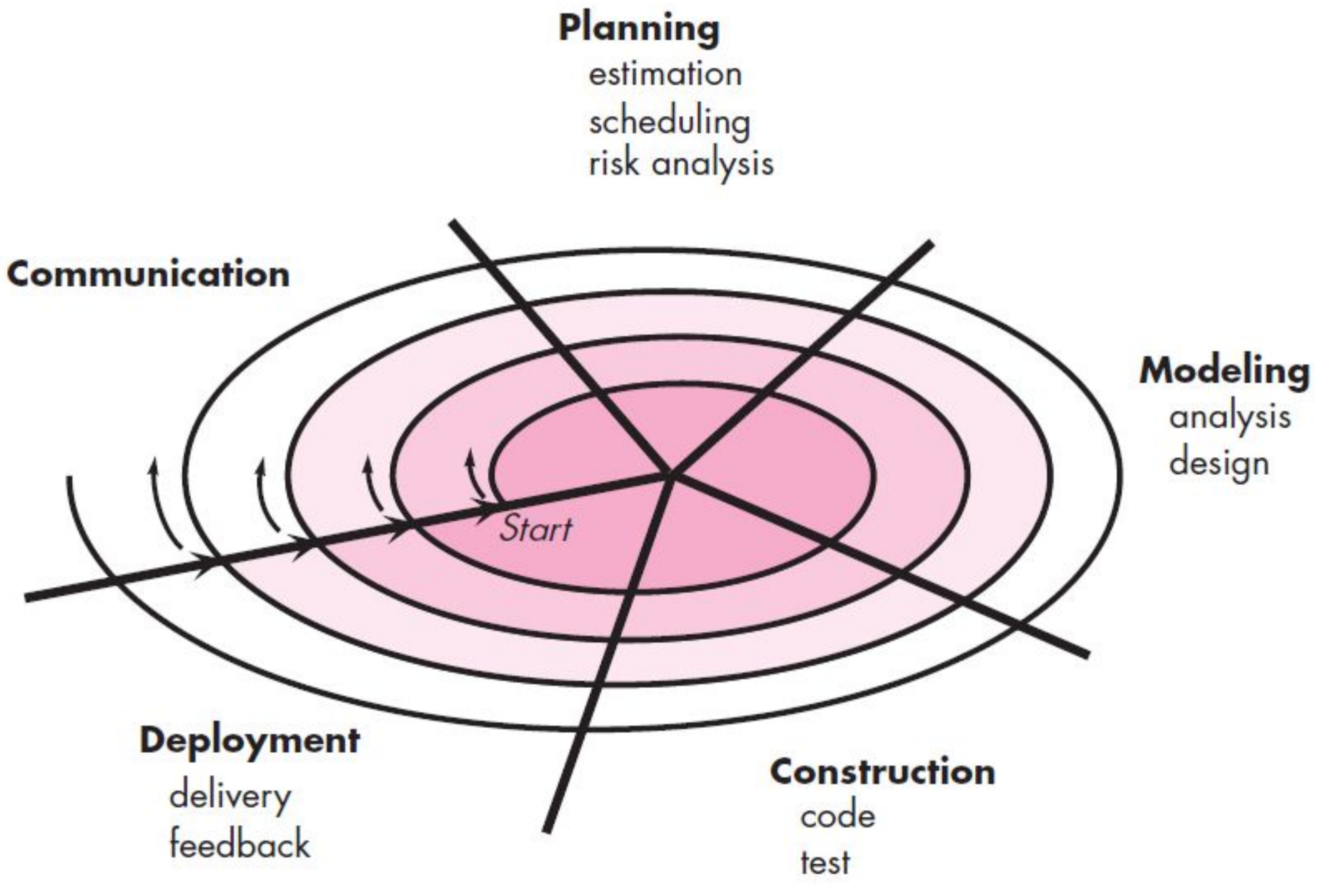
Evolutionary Process Model: The Spiral Model

- The *spiral model* is an evolutionary software process model that couples the iterative nature of prototyping with the controlled and systematic aspects of the waterfall model.
- Using the spiral model, software is developed in a series of evolutionary releases. During early iterations, the release might be a model or prototype. During later iterations, increasingly more complete versions of the engineered system are produced.
- Each of the framework activities represent one segment of the spiral path. The software team performs activities that are implied by a circuit around the spiral in a clockwise direction, beginning at the center.

Evolutionary Process Model: The Spiral Model

- **Risk** is considered as each revolution is made.
- ***Anchor point*** *milestones*—a combination of work products and conditions that are attained along the path of the spiral—are noted for each evolutionary pass.
- The first circuit around the spiral might result in the development of a product specification
- Subsequent passes around the spiral might be used to develop a prototype
- Then progressively more sophisticated versions of the software.

Evolutionary Process Model: The Spiral Model



Spiral Model

Pros:

- There is continuous customer involvement.
- Development risks are managed.
- It is suitable for large, complex projects.
- It works well for extensible products.

Cons:

- Risk analysis failures can doom the project.
- The project may be hard to manage.
- It requires an expert development team.

Unified Process Model

Phases of UP:

 Inception:

- Fundamental business requirements are described through a set of preliminary use cases.
- Planning identifies resources, assesses major risks, and defines a preliminary schedule for the software increments.

 Elaboration:

- Refines and expands the preliminary use cases and creates an architectural baseline.
- Modifications to the plan are often made at this time.

Unified Process Model

▯ **Construction:**

- All necessary and required features and functions for the software increment (i.e., the release) are implemented in source code.
- As components are being implemented, unit tests are designed and executed for each.
- Integration activities (component assembly and integration testing) are conducted.
- Use cases are used to derive a suite of acceptance tests.

▯ **Transition:**

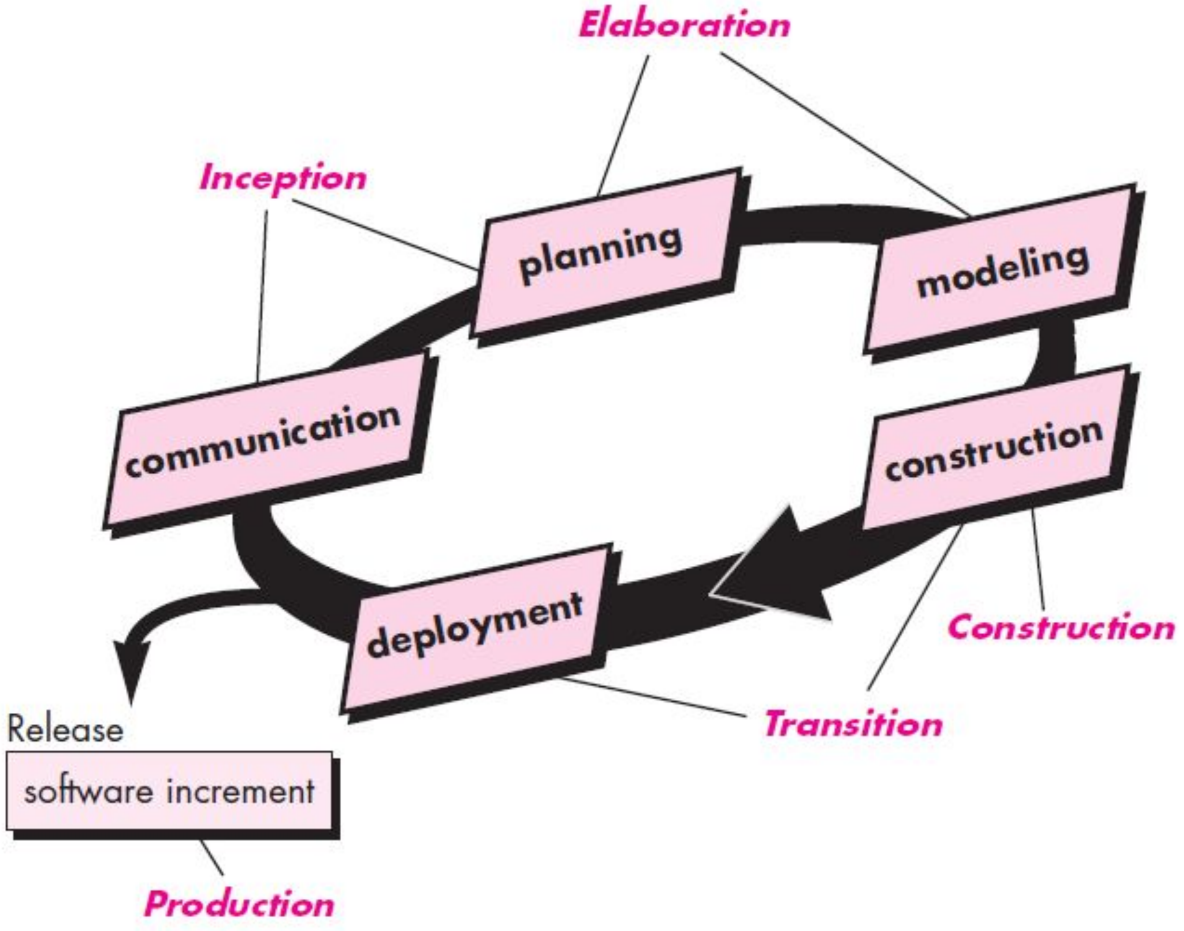
- Software and supporting documentation is given to end users for beta testing
- User feedback reports both defects and necessary changes.

Unified Process Model

Production:

- Ongoing use of the software is monitored.
- Support for the operating environment is provided
- Defect reports and requests for changes are submitted and evaluated.

Unified Process Model



Unified Process Model

Pros:

- Quality documentation is emphasized.
- There is continuous customer involvement.
- It accommodates requirements changes.
- It works well for maintenance projects.

Cons:

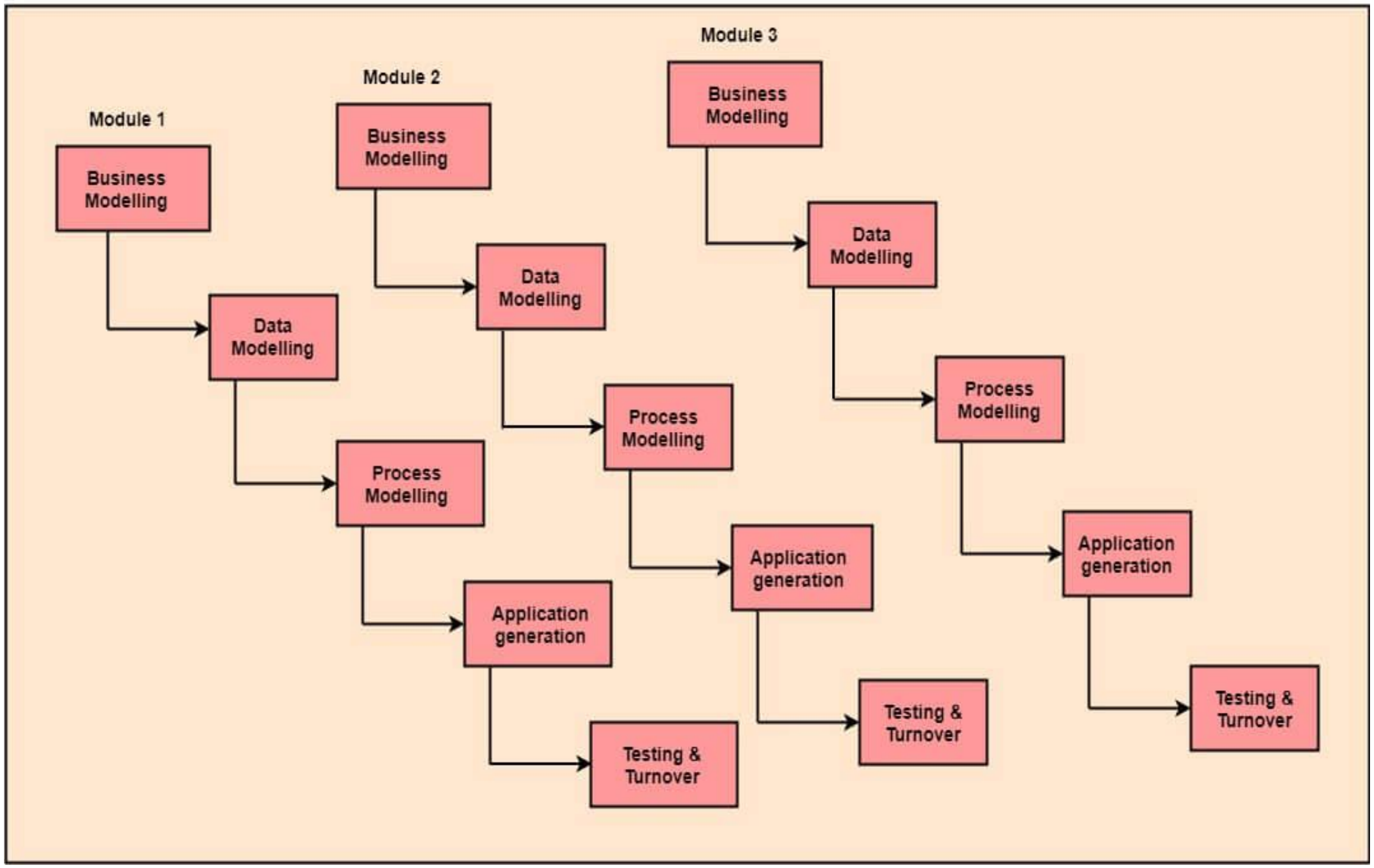
- Use cases are not always precise.
- Overlapping phases can cause problems.
- It requires an expert development team.

Rapid Application Development (RAD) Model

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

RAD Model

Fig: RAD Model



RAD Model

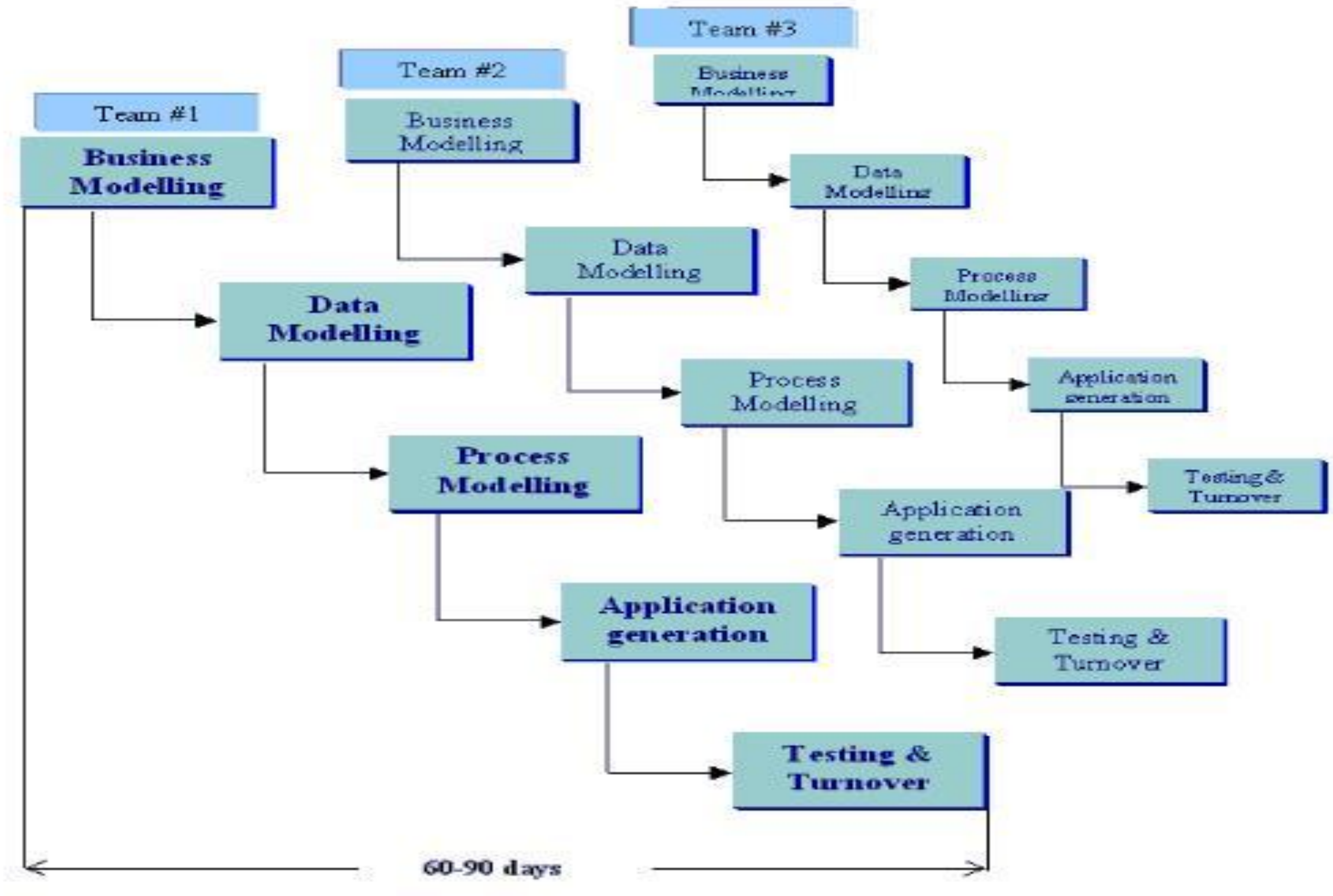


Figure 1.5 – RAD Model

Phases of RAD Model

1. **Business Modelling:** The business model for the product under development is designed in terms of flow of information and the distribution of information between various business channels. A complete business analysis is performed to find the vital information for business.
2. **Data Modelling:** The data collected from business modeling is refined into a set of data objects (entities) that are needed to support the business. The attributes (character of each entity) are identified, and the relation between these data objects (entities) is defined.
3. **Process Modelling:** The information object defined in the data modeling phase are transformed to achieve the data flow necessary to implement a business function. Processing descriptions are created for adding, modifying, deleting, or retrieving a data object.

Phases of RAD Model

4. **Application Generation:** The actual system is built and coding is done by using automation tools to convert process and data models into actual prototypes.
5. **Testing & Turnover:** The overall testing time is reduced in the RAD model as the prototypes are independently tested during every iteration. However, the data flow and the interfaces between all the components need to be thoroughly tested with complete test coverage. Since most of the programming components have already been tested, it reduces the risk of any major issues.

When to use RAD Model

- When the system should need to create the project that modularizes in a short span time (2-3 months).
- When the requirements are well-known.
- When the technical risk is limited.
- Progress needs to be made visible

RAD Model

Pros:

- This model is flexible for change.
- Encourages and priorities customer feedback.
- It reduced development time.
- It increases the reusability of features.

Cons:

- It requires highly skilled designers.
- Only suitable for projects which have a small development time.
- Only systems which can be modularized can be developed using Rapid application development.
- On the high technical risk, it's not suitable.
- Required user involvement.