Software Engineering

Lecture 5

Mr. Noor Ul Arfeen

Evolutionary Models

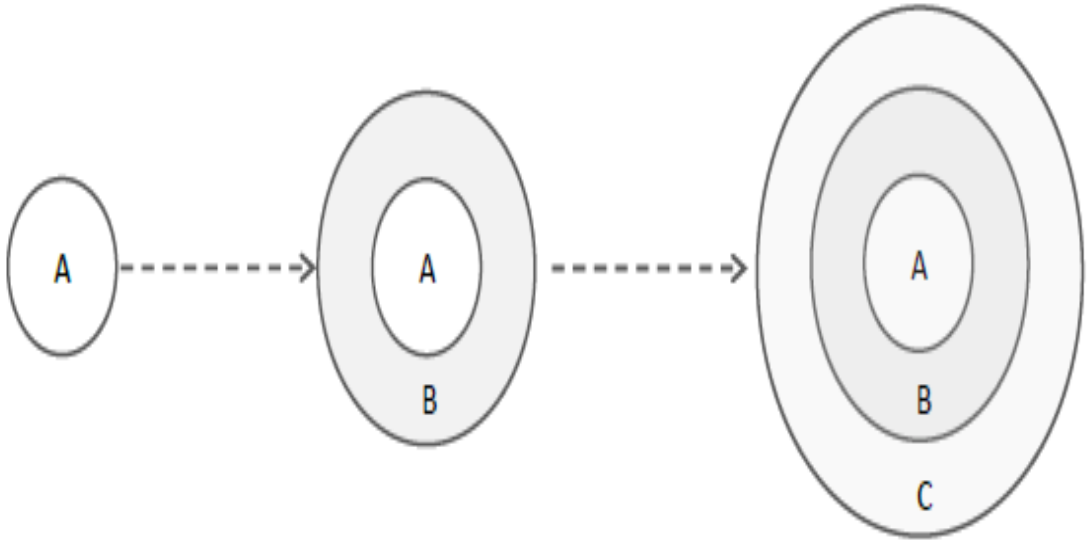
Evolutionary model is a combination of Iterative and Incremental model

Evolutionary development model divides the development cycle into smaller, incremental waterfall models in which users are able to get access to the product at the end of each cycle

Produce an increasingly more complete version of the software with each iteration

Two types

- ❑Prototyping
- ❑Spiral Model



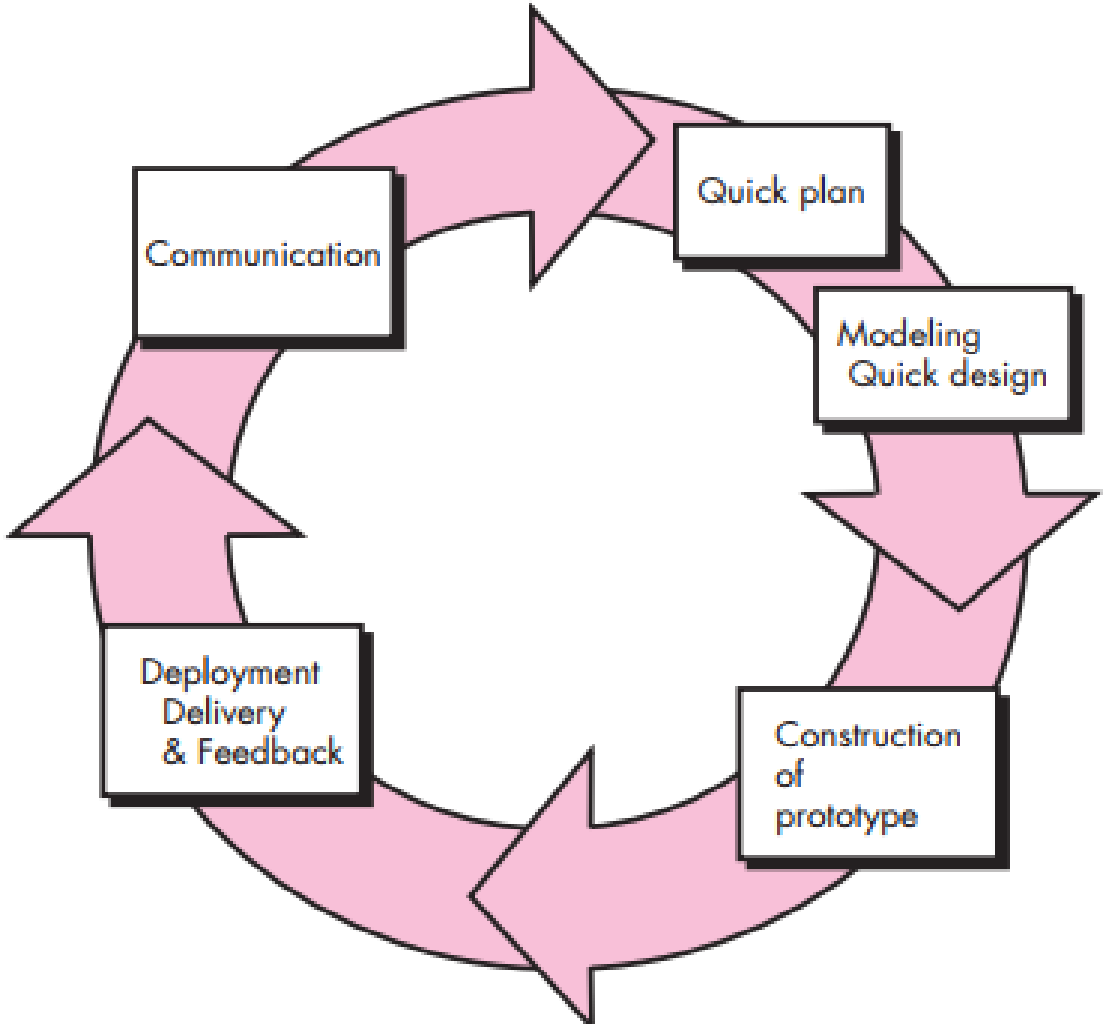
Prototyping Model

Consider the following two different situations:

Often, a customer defines a set of general objectives for software, but does not identify a detailed specifications like input, processing and output requirements. It's a **problem in customer side**

Next is, **problems in developer side**; a developer may unsure about the efficiency of an algorithm; the adoptability of an operating system; problems in human-computer interaction etc

In these non-stabilized cases, the prototyping model may offer a best solution



Prototyping Model

Prototyping paradigm begins with communication. You meet with stakeholders to define the overall objectives for the software, identify whatever requirements are known, and outline areas where further definition is mandatory

Prototyping iteration is planned quickly, and modeling (in form of a “quick design”) occurs. A quick design focuses on a representation of those aspects of software that will be visible to end users (e.g., human interface layout)

Quick design leads to the construction of a prototype. The prototype is deployed and evaluated by stakeholders, who provide feedback that is used to further refine requirements. Iteration occurs as prototype is tuned to satisfy needs of various stakeholders, while at same time enabling you to better understand what needs to be done

Ideally, the prototype serves as a mechanism for identifying software requirements

Prototyping Model

A software prototype can be used in a software development process to help anticipate changes that may be required:

1. In the requirements engineering process, a prototype can help with the elicitation and validation of system requirements.

2. In the system design process, a prototype can be used to explore particular software solutions and to support user interface design.

Prototyping Model

Developers are sometimes pressured by managers to deliver throwaway prototypes, particularly when there are delays in delivering the final version of the software. However, this is usually unwise:

1. It may be impossible to tune the prototype to meet non-functional requirements, such as performance, security, robustness, and reliability requirements, which were ignored during prototype development.

2. Rapid change during development inevitably means that the prototype is undocumented. The only design specification is the prototype code. This is not good enough for long-term maintenance.

3. The changes made during prototype development will probably have degraded the system structure. The system will be difficult and expensive to maintain.

4. Organizational quality standards are normally relaxed for prototype development.

Prototyping Model - Advantages

Active involvement of user in the development process

Errors can be detected much earlier

Quicker user feedback is available leading to better solution

Missing functionality can be identified easily

Confusing or difficult functions can be identified

Prototyping Model - Problems

Customer gets disappointed and lodge more complaints about product after seeing an initial version of working model, because they see a temporary patch. The customer sees what appears to be a working version of the software, unaware about the prototype. When a developer informed that the product must be rebuilt so that high quality can be maintained, the customer demands that “a few fixes” be applied to make prototype a working product

The developer makes implementation compromises in order to get a working prototype. An inappropriate program / inefficient algorithm / easy tool may be used to demonstrate the capability of working model. Later, the developer become comfortable with these choices and forget all the reasons why they were inappropriate. The less-than-ideal choice has now become an integral part of the system

Spiral Model

Combines iterative nature of prototyping with the controlled and systematic aspects of the waterfall model

Using spiral, software developed in as series of evolutionary release

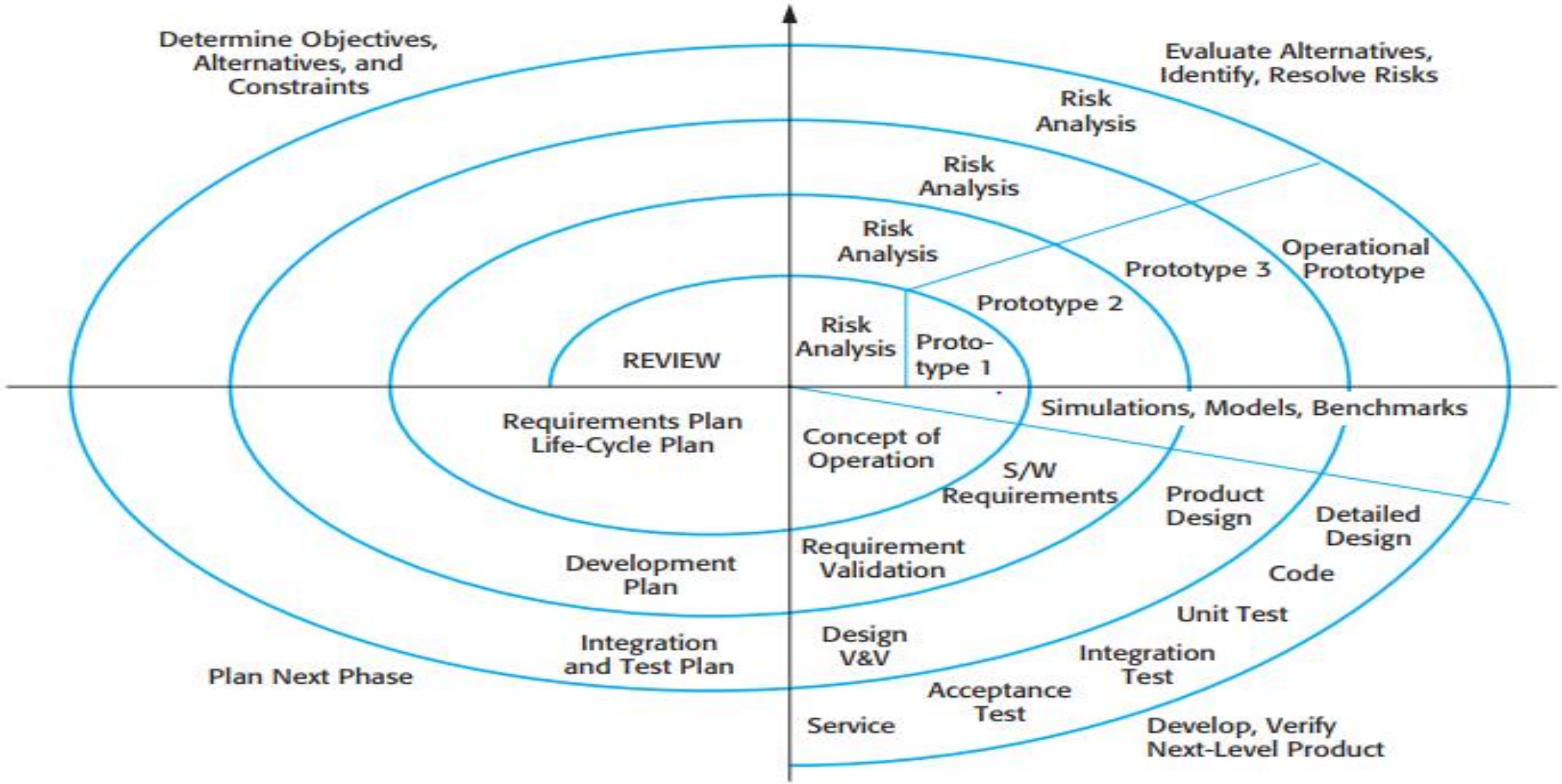
- Early iteration, release might be on paper or prototype
- Later iteration, more complete version of software

Divided into a set of framework activities. Each activity represent one segment of spiral path

It provides the potential for rapid development of increasingly more complete versions of the software

Also called as risk-driven software development process model

Spiral Model



Boehm's spiral model of the software process

Spiral Model

Each loop in the spiral is split into four sectors:

1. *Objective setting* Specific objectives for that phase of the project are defined. Constraints on the process and the product are identified and a detailed management plan is drawn up. Project risks are identified. Alternative strategies, depending on these risks, may be planned.

2. Risk assessment and reduction For each of the identified project risks, a detailed analysis is carried out. Steps are taken to reduce the risk. For example, if there is a risk that the requirements are inappropriate, a prototype system may be developed

Spiral Model

3. Development and validation After risk evaluation, a development model for the system is chosen. For example, throwaway prototyping may be the best development approach if user interface risks are dominant. If safety risks are the main consideration, development based on formal transformations may be the most appropriate process, and so on. If the main identified risk is sub-system integration, the waterfall model may be the best development model to use.

4. Planning The project is reviewed and a decision made whether to continue with a further loop of the spiral. If it is decided to continue, plans are drawn up for the next phase of the project.

Spiral Model - Applications

When the project is large

Where the software needs continuous risk evaluation

Requirements are a bit complicated and require continuous clarification

Significant changes are expected in the product during the development cycle

Spiral Model - Advantages

Risk Handling: The projects with many unknown risks that occur as the development proceeds, in that case, Spiral Model is the best development model to follow due to the risk analysis and risk handling at every phase

Good for large projects: It is recommended to use the Spiral Model in large and complex projects

Flexibility in Requirements: Change requests in the Requirements at later phase can be incorporated accurately by using this model

Customer Satisfaction: Customer can see the development of the product at the early phase of the software development and thus, they habituated with the system by using it before completion of the total product

Spiral Model - Disadvantages

Complex: The Spiral Model is much more complex than other SDLC models

Expensive: Spiral Model is not suitable for small projects as it is expensive

Too much dependability on Risk Analysis: The successful completion of the project is very much dependent on Risk Analysis. Without very highly experienced experts, it is going to be a failure to develop a project using this model

Difficulty in time management: As the number of phases is unknown at the start of the project, so time estimation is very difficult

Motivation

"Every error is a hidden lesson. Debugging isn't just fixing problems; it's uncovering the pathways to deeper understanding and mastery"

"Every line of code you write is a step toward solving real-world problems. Embrace the challenges of software engineering, for in every bug lies an opportunity to learn, grow, and create something extraordinary"

RAD (Rapid Application Development) Model

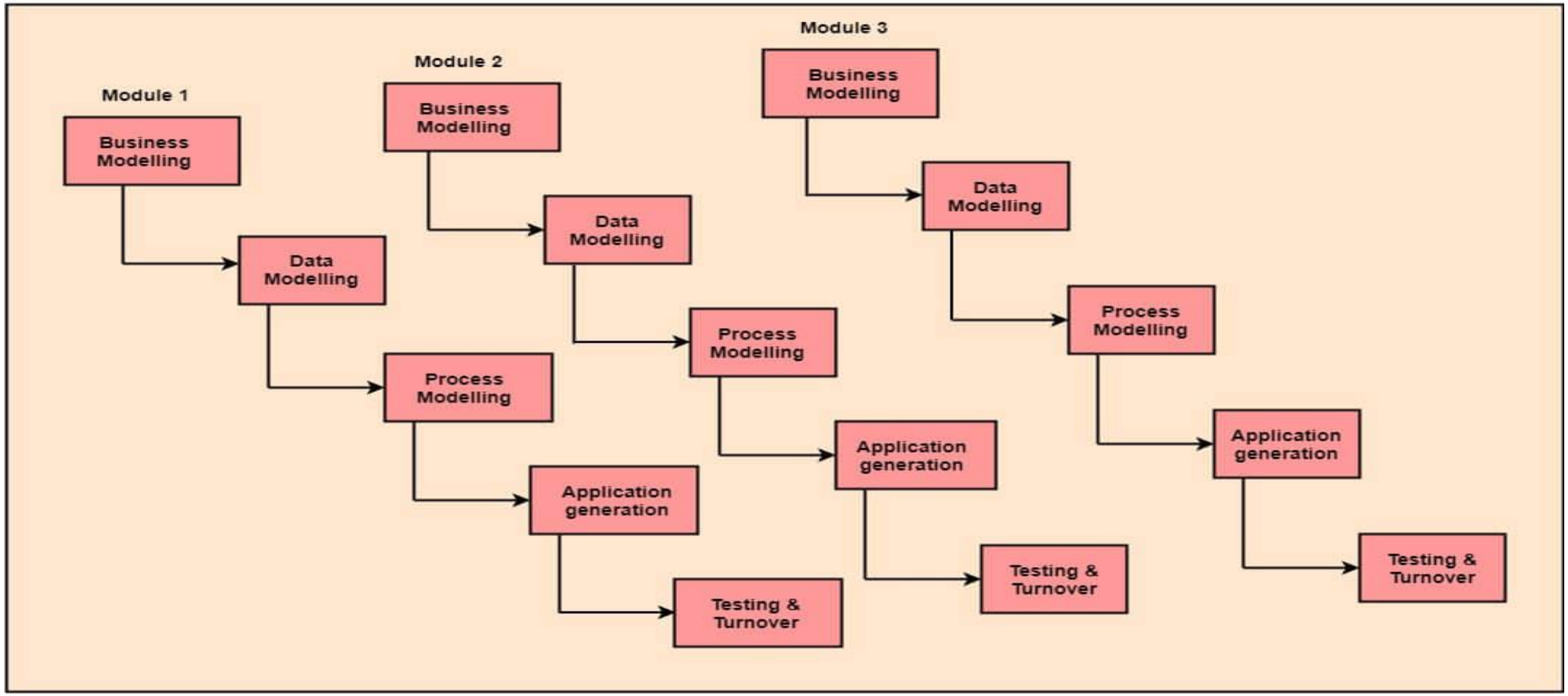
- RAD is a linear sequential software development process model that emphasizes a concise development cycle using an element based construction approach. If the requirements are well understood and described, and the project scope is a constraint, the RAD process enables a development team to create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

RAD (Rapid Application Development) Model

- RAD (Rapid Application Development) is a concept that products can be developed faster and of higher quality through:
  - Gathering requirements using workshops or focus groups
  - Prototyping and early, reiterative user testing of designs
  - The re-use of software components
  - A rigidly paced schedule that refers design improvements to the next product version
  - Less formality in reviews and other team communication

RAD (Rapid Application Development) Model

Fig: RAD Model



Phases of RAD Model

- **1.Business Modelling:** The information flow among business functions is defined by answering questions like what data drives the business process, what data is generated, who generates it, where does the information go, who process it and so on.
- **2. Data Modelling:** The data collected from business modeling is refined into a set of data objects (entities) that are needed to support the business. The attributes (character of each entity) are identified, and the relation between these data objects (entities) is defined.
- **3. Process Modelling:** The information object defined in the data modeling phase are transformed to achieve the data flow necessary to implement a business function. Processing descriptions are created for adding, modifying, deleting, or retrieving a data object.

Phases of RAD Model

- **4. Application Generation:** Automated tools are used to facilitate construction of the software; even they use the 4th GL techniques.
- **5. Testing & Turnover:** Many of the programming components have already been tested since RAD emphasis reuse. This reduces the overall testing time. But the new part must be tested, and all interfaces must be fully exercised

When to Use RAD Model?

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

RAD Model - Advantages

- This model is flexible for change.
- In this model, changes are adoptable.
- Each phase in RAD brings highest priority functionality to the customer.
- It reduced development time.
- It increases the reusability of features.

RAD Model - Disadvantages

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

Motivation

Understanding SDLC is crucial to becoming a proficient and versatile software developer. It's more than just a framework, it's the key to managing complexity, improving quality, and succeeding in both individual projects and collaborative environments. By mastering SDLC, you prepare yourself for a successful career in technology and beyond.