Module 6

Design Concepts and the Recommended Approach

▼ Requirements are translated into a blueprint which is translated into what lower level of abstraction?

design representations

▼ What does quality assurance ensure?

- All explicit requirements in the requirements model as well as any implicit requirements from the stakeholders are included.
- The design is clear and understandable to those who generate the code and those who test and support the software.
- The design provides a clear picture of the data, functional, and behavioral domains of the software from an implementation perspective.

▼ What criteria should a design meet?

- Exhibit an architecture that:
  - Has been created using recognizable architectural styles or patterns.
  - Is composed of components that exhibit good design characteristics.
  - Can be implemented in an evolutionary manner that makes implementation and testing easier.
- Be modular so that it can be logically broken down into elements or subsystems.
- Have distinct representations of data, architecture, interface, and components.

- Lead to data structures that are appropriate for the classes to be implemented and are drawn from recognizable data patterns.
- Lead to components that have independent functional characteristics.
- Lead to interfaces that reduce the complexity of connections between components and the external environment.
- Be derived using a repeatable method that is driven by information obtained during software requirements analysis.
- Be represented using a notation that effectively communicates its meaning.

▼ What is abstraction?

Defining a solution in generic terms using the language of the problem environment

▼ What is software architecture?

- Structuring the software in a manner that provides system integrity
- Each structure is made up of software elements, the relations among them, and the properties of both elements and relations

▼ What is a design pattern?

- A formalized best practice to solve a common problem in the system design
- It provides a general, reusable solution to a commonly occurring problem

▼ What is separation of concerns?

The breaking of a complex problem into smaller pieces that can each be solved separately

▼ What is modularity?

Dividing the software into separate named components that are sometimes referred to as modules

▼ What is information hiding?

Hiding the details of an object or function to reduce external complexity and make the object or function easier to use

▼ What is functional independence?

- Creating modules that perform a single task or function.
- A functionally independent module has little to no interaction with other modules.

▼ What is stepwise refinement?

Using a top-down design strategy that allows a software engineer to successively refine levels of procedural detail

▼ What is refactoring?

Using an information technique that simplifies the design of a component without changing its function or behavior

▼ What are design classes?

Adding design detail to the analysis classes to enable the classes to be implemented and support the business solution

▼ How can an agile team can satisfy the need for architectural design?

By creating an architectural prototype (e.g., a walking skeleton) and developing explicit architectural work products to communicate the right information to the necessary stakeholders

▼ What is storyboarding?

This technique **allows the architect to contribute architectural user stories to the project** and works with the product owner to prioritize the architectural stories with the business user stories as “sprints” (work units) are planned

▼ What is responsibility-driven architecture (RDA)?

This technique focuses on when, how, and who should make the architectural decisions on a project team

▼ When does a software development team typically make decisions?

when team members identify key design issues and the rationale behind the architectural solutions chosen

▼ What do comprehensive system architecture decisions include?

- software system organization
- selection of structural elements and their interfaces as defined by their intended collaborations
- the composition of these elements into increasingly larger subsystems

▼ What is included in the context for an architectural design?

- external entities such as other systems, devices
- people that interact with the software and the nature of their interaction

▼ What questions must be answered for a software engineer to create meaningful architectural diagrams?

- Does the diagram show how the system responds to inputs or events?
- What visualizations might there be to help emphasize areas of risk?
- How can hidden system design patterns be made more obvious to other developers?
- Can multiple viewpoints show the best way to refactor specific parts of the system?
- Can design trade-offs be represented in a meaningful way?