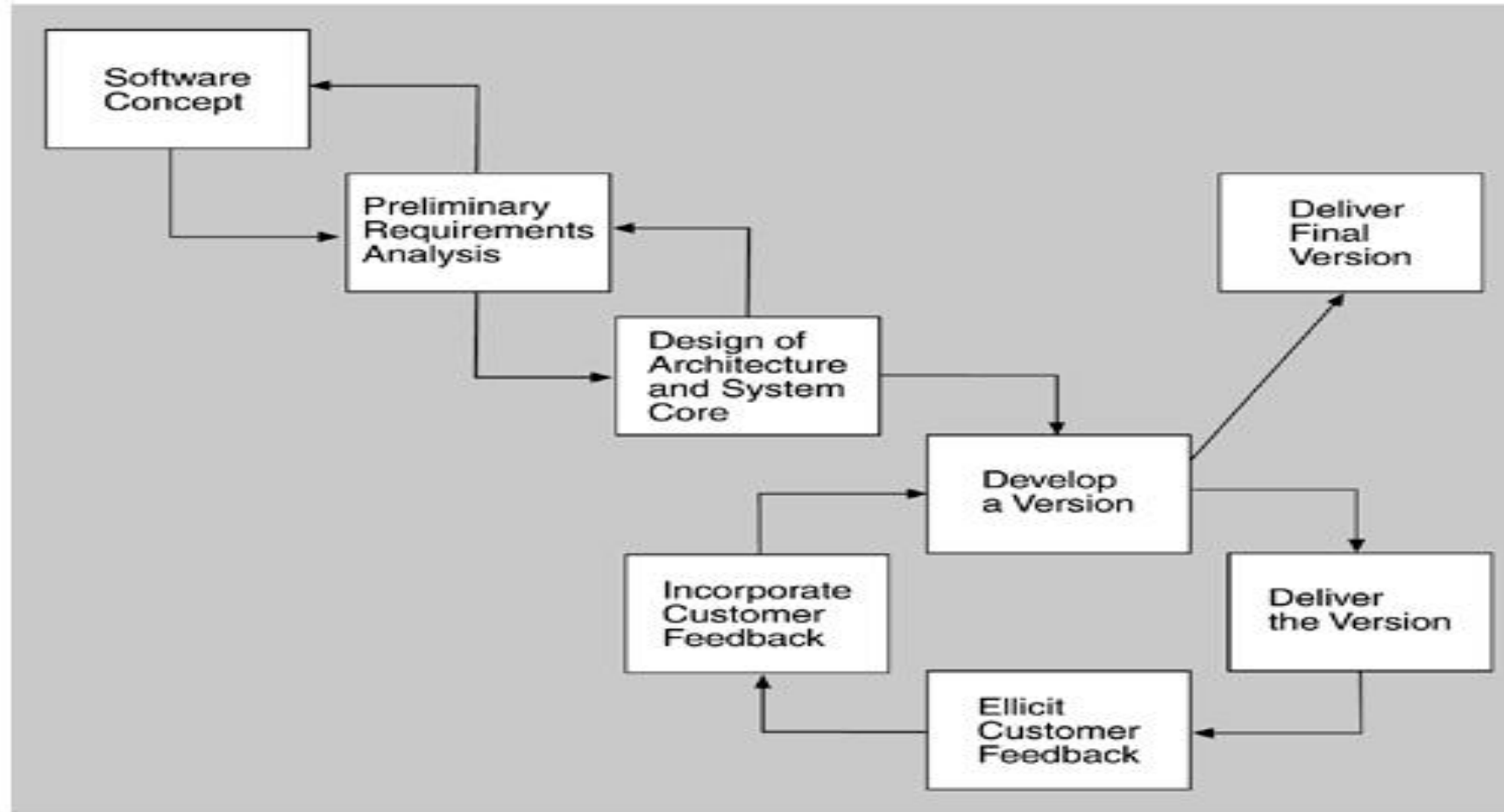


Architecture in the Life Cycle

introduction

- u Any organization that embraces architecture as a foundation for its software development processes needs to understand its place in the life cycle.
- u Several life-cycle models exist in the literature, but one that puts architecture squarely in the middle of things is the Evolutionary Delivery Life Cycle model.
- u The intent of this model is to get user and customer feedback and iterate through several releases before the final release.
- u The model also allows the adding of functionality with each iteration and the delivery of a limited version once a sufficient set of features has been developed.

Cont..



WHEN CAN I BEGIN DESIGNING?

- u The life-cycle model shows the design of the architecture as iterating with preliminary requirements analysis. Clearly, you cannot begin the design until you have some idea of the system requirements. On the other hand, it does not take many requirements in order for design to begin.
- u An architecture is "shaped" by some collection of functional, quality, and business requirements. We call these shaping requirements architectural drivers and we see examples of them in our case studies.
- u The architecture of the A-7E is shaped by its modifiability and performance requirements.

Cont..

- u To determine the architectural drivers, identify the highest priority business goals. There should be relatively few of these.
- u Turn these business goals into quality scenarios or use cases. From this list, choose the ones that will have the most impact on the architecture. These are the architectural drivers, and there should be fewer than ten.
- u Once the architectural drivers are known, the architectural design can begin

Designing the Architecture

- u In this section we describe a method for designing an architecture to satisfy both quality requirements and functional requirements. We call this method Attribute-Driven Design (ADD).
- u ADD takes as input a set of quality attribute scenarios and employs knowledge about the relation between quality attribute achievement and architecture in order to design the architecture.
- u The ADD method can be viewed as an extension to most other development methods, such as the Rational Unified Process. The Rational Unified Process has several steps that result in the high-level design of an architecture but then proceeds to detailed design and implementation.

Cont..

- ▮ ADD is an approach to defining a software architecture that bases the decomposition process on the quality attributes the software has to fulfill.
- ▮ It is a recursive decomposition process where, at each stage, tactics and architectural patterns are chosen to satisfy a set of quality scenarios and then functionality is allocated to instantiate the module types provided by the pattern.
- ▮ ADD is positioned in the life cycle after requirements analysis and, as we have said, can begin when the architectural drivers are known with some confidence.

Cont...

- u We demonstrate the ADD method by using it to design a product line architecture for a garage door opener within a home information system. The opener is responsible for raising and lowering the door via a switch, remote control, or the home information system. It is also possible to diagnose problems with the opener from within the home information system
- u **ADD Steps**
- u We begin by briefly presenting the steps performed when designing an architecture using the ADD method. We will then discuss the steps in more detail.
- 1. Choose the module to decompose. The module to start with is usually the whole system. All required inputs for this module should be available (constraints, functional requirements, quality requirements).
- 2. Refine the module according to these steps:
 - a. Choose the architectural drivers from the set of concrete quality scenarios and functional requirements. This step determines what is important for this decomposition.
 - b. Choose an architectural pattern that satisfies the architectural drivers. Create (or select) the pattern based on the tactics that can be used to achieve the drivers. Identify child modules required to implement the tactics.

Cont...

- c. Instantiate modules and allocate functionality from the use cases and represent using multiple views.
 - d. Define interfaces of the child modules. The decomposition provides modules and constraints on the types of module interactions. Document this information in the interface document for each module.
 - e. Verify and refine use cases and quality scenarios and make them constraints for the child modules. This step verifies that nothing important was forgotten and prepares the child modules for further decomposition or implementation.
3. Repeat the steps above for every module that needs further decomposition.

Cont...

- u Choose the Module to Decompose
- u Choose the Architectural Drivers
- u Choose an Architectural Pattern
- u Instantiate Modules and Allocate Functionality Using Multiple Views
- u Define Interfaces of the Child Modules
- u Verify and Refine Use Cases and Quality Scenarios as Constraints for the Child Modules

The background of the slide features abstract, overlapping geometric shapes in various shades of blue, ranging from light sky blue to deep navy blue. These shapes are primarily located on the right side and bottom of the slide, creating a modern, dynamic feel.

u Reading assignment:

Creating a Skeletal System

Forming the Team Structure