UNIT - IV

Model based software architectures: A Management perspective and technical perspective.


7. Model based software architecture

7.1 ARCHITECTURE: A MANAGEMENT PERSPECTIVE

The most critical technical product of a software project is its architecture: the infrastructure, control, and data interfaces that permit software components to cooperate as a system and software designers to cooperate efficiently as a team. When the communications media include multiple languages and intergroup literacy varies, the communications problem can become extremely complex and even unsolvable. If a software development team is to be successful, the inter project communications, as captured in the software architecture, must be both accurate and precise.

From a management perspective, there are three different aspects of architecture.

1. An architecture (the intangible design concept) is the design of a software system this includes all engineering necessary to specify a complete bill of materials.
2. An architecture baseline (the tangible artifacts) is a slice of information across the engineering artifact sets sufficient to satisfy all stakeholders that the vision (function and quality) can be achieved within the parameters of the business case (cost, profit, time, technology, and people).
3. An architecture description (a human-readable representation of an architecture, which is one of the components of an architecture baseline) is an organized subset of information extracted from the design set model(s). The architecture description communicates how the intangible concept is realized in the tangible artifacts.

The number of views and the level of detail in each view can vary widely.

The importance of software architecture and its close linkage with modern software development processes can be summarized as follows:

- Achieving a stable software architecture represents a significant project milestone at which the critical make/buy decisions should have been resolved.
- Architecture representations provide a basis for balancing the trade-offs between the problem space (requirements and constraints) and the solution space (the operational product).
- The architecture and process encapsulate many of the important (high-payoff or high-risk) communications among individuals, teams, organizations, and stakeholders.
- Poor architectures and immature processes are often given as reasons for project failures.
- A mature process, an understanding of the primary requirements, and a demonstrable architecture are important prerequisites for predictable planning.
- Architecture development and process definition are the intellectual steps that map the problem to a solution without violating the constraints; they require human innovation and cannot be automated.

7.2 ARCHITECTURE: A TECHNICAL PERSPECTIVE

An architecture framework is defined in terms of views that are abstractions of the UML models in the design set. The design model includes the full breadth and depth of information. An architecture view is an abstraction of the design model; it contains only the architecturally significant information. Most real-world systems require four views: design,
process, component, and deployment. The purposes of these views are as follows:

- **Design**: describes architecturally significant structures and functions of the design model
- **Process**: describes concurrency and control thread relationships among the design, component, and deployment views
- **Component**: describes the structure of the implementation set
- **Deployment**: describes the structure of the deployment set

Figure 7-1 summarizes the artifacts of the design set, including the architecture views and architecture description.

The requirements model addresses the behavior of the system as seen by its end users, analysts, and testers. This view is modeled statically using use case and class diagrams, and dynamically using sequence, collaboration, state chart, and activity diagrams.

- The **use case view** describes how the system's critical (architecturally significant) use cases are realized by elements of the design model. It is modeled statically using use case diagrams, and dynamically using any of the UML behavioral diagrams.
- The **design view** describes the architecturally significant elements of the design model. This view, an abstraction of the design model, addresses the basic structure and functionality of the solution. It is modeled statically using class and object diagrams, and dynamically using any of the UML behavioral diagrams.
- The **process view** addresses the run-time collaboration issues involved in executing the architecture on a distributed deployment model, including the logical software network topology (allocation to processes and threads of control), interprocess communication, and state management. This view is modeled statically using deployment diagrams, and dynamically using any of the UML behavioral diagrams.
- The **component view** describes the architecturally significant elements of the implementation set. This view, an abstraction of the design model, addresses the software source code realization of the system from the perspective of the project's integrators and developers, especially with regard to releases and configuration management. It is modeled statically using component diagrams, and dynamically using any of the UML behavioral diagrams.
- The **deployment view** addresses the executable realization of the system, including the allocation of logical processes in the distribution view (the logical software topology) to physical resources of the deployment network (the physical system topology). It is modeled statically using deployment diagrams, and dynamically using any of the UML behavioral diagrams.

Generally, an architecture baseline should include the following:

- **Requirements**: critical use cases, system-level quality objectives, and priority relationships among features and qualities
- **Design**: names, attributes, structures, behaviors, groupings, and relationships of significant classes and components
- **Implementation**: source component inventory and bill of materials (number, name, purpose, cost) of all primitive components
- **Deployment**: executable components sufficient to demonstrate the critical use cases and the risk associated with achieving the system qualities
8. Workflow of the process

The requirements set may include UML models describing the problem space.

The design set includes all UML design models describing the solution space.

The design, process, and use case models provide for visualization of the logical and behavioral aspects of the design.

The component model provides for visualization of the implementation set.

The deployment model provides for visualization of the deployment set.

An architecture is described through several views, which are extracts of design models that capture the significant structures, collaborations, and behaviors.

Architecture Description Document
- Design view
- Process view
- Use case view
- Component view
- Deployment view
- Other views (optional)
- Other material:
  - Rationale
  - Constraints

Figure 7-1. Architecture, an organized and abstracted view into the design models
8.1 SOFTWARE PROCESS WORKFLOWS

The term WORKFLOWS is used to mean a thread of cohesive and mostly sequential activities. Workflows are mapped to product artifacts. There are seven top-level workflows:

1. Management workflow: controlling the process and ensuring win conditions for all stakeholders
2. Environment workflow: automating the process and evolving the maintenance environment
3. Requirements workflow: analyzing the problem space and evolving the requirements artifacts
4. Design workflow: modeling the solution and evolving the architecture and design artifacts
5. Implementation workflow: programming the components and evolving the implementation and deployment artifacts
6. Assessment workflow: assessing the trends in process and product quality
7. Deployment workflow: transitioning the end products to the user

Figure 8-1 illustrates the relative levels of effort expected across the phases in each of the top-level workflows.

Table 8-1 shows the allocation of artifacts and the emphasis of each workflow in each of the life-cycle phases of inception, elaboration, construction, and transition.
<table>
<thead>
<tr>
<th>WORKFLOW</th>
<th>ARTIFACTS</th>
<th>LIFE-CYCLE PHASE EMPHASIS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Management</td>
<td>Business case</td>
<td>Inception: Prepare business case and vision</td>
</tr>
<tr>
<td></td>
<td>Software development plan</td>
<td>Elaboration: Plan development</td>
</tr>
<tr>
<td></td>
<td>Status assessments</td>
<td>Construction: Monitor and control development</td>
</tr>
<tr>
<td></td>
<td>Vision</td>
<td>Transition: Monitor and control deployment</td>
</tr>
<tr>
<td></td>
<td>Work breakdown structure</td>
<td></td>
</tr>
<tr>
<td>Environment</td>
<td>Environment</td>
<td>Inception: Define development environment and change management infrastructure</td>
</tr>
<tr>
<td></td>
<td>Software change order database</td>
<td>Elaboration: Install development environment and establish change management database</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Construction: Maintain development environment and software change order database</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Transition: Transition maintenance environment and software change order database</td>
</tr>
<tr>
<td>Requirements</td>
<td>Requirements set</td>
<td>Inception: Define operational concept</td>
</tr>
<tr>
<td></td>
<td>Release specifications</td>
<td>Elaboration: Define architecture objectives</td>
</tr>
<tr>
<td></td>
<td>Vision</td>
<td>Construction: Define iteration objectives</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Transition: Refine release objectives</td>
</tr>
<tr>
<td>Design</td>
<td>Design set</td>
<td>Inception: Formulate architecture concept</td>
</tr>
<tr>
<td></td>
<td>Architecture description</td>
<td>Elaboration: Achieve architecture baseline</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Construction: Design components</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Transition: Refine architecture and components</td>
</tr>
<tr>
<td>Implementation</td>
<td>Implementation set</td>
<td>Inception: Support architecture prototypes</td>
</tr>
<tr>
<td></td>
<td>Deployment set</td>
<td>Elaboration: Produce architecture baseline</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Construction: Produce complete componentry</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Transition: Maintain components</td>
</tr>
<tr>
<td>Assessment</td>
<td>Release specifications</td>
<td>Inception: Assess plans, vision, prototypes</td>
</tr>
<tr>
<td></td>
<td>Release descriptions</td>
<td>Elaboration: Assess architecture</td>
</tr>
<tr>
<td></td>
<td>User manual</td>
<td>Construction: Assess interim releases</td>
</tr>
<tr>
<td></td>
<td>Deployment set</td>
<td>Transition: Assess product releases</td>
</tr>
<tr>
<td>Deployment</td>
<td>Deployment set</td>
<td>Inception: Analyze user community</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Elaboration: Define user manual</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Construction: Prepare transition materials</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Transition: Transition product to user</td>
</tr>
</tbody>
</table>

8.2 ITERATION WORKFLOWS
Iteration consists of a loosely sequential set of activities in various proportions, depending on where the iteration is located in the development cycle. Each iteration is defined in terms of a set of allocated usage scenarios. An individual iteration’s workflow, illustrated in Figure 8-2, generally includes the following sequence:

- **Management:** iteration planning to determine the content of the release and develop the detailed plan for the iteration; assignment of work packages, or tasks, to the development team
- **Environment:** evolving the software change order database to reflect all new baselines and changes to existing baselines for all product, test, and environment components

![Figure 8-2. The workflow of an iteration](image)

- **Requirements:** analyzing the baseline plan, the baseline architecture, and the baseline requirements set artifacts to fully elaborate the use cases to be demonstrated at the end of this iteration and their evaluation criteria; updating any requirements set artifacts to reflect changes necessitated by results of this iteration’s engineering activities
- **Design:** evolving the baseline architecture and the baseline design set artifacts to elaborate fully the design model and test model components necessary to demonstrate against the evaluation criteria allocated to this iteration; updating design set artifacts to reflect changes necessitated by the results of this iteration’s engineering activities
- **Implementation:** developing or acquiring any new components, and enhancing or modifying any existing components, to demonstrate the evaluation criteria allocated to this iteration; integrating and testing all new and modified components with existing baselines (previous versions)
• Assessment: evaluating the results of the iteration, including compliance with the allocated evaluation criteria and the quality of the current baselines; identifying any rework required and determining whether it should be performed before deployment of this release or allocated to the next release; assessing results to improve the basis of the subsequent iteration's plan.

• Deployment: transitioning the release either to an external organization (such as a user, independent verification and validation contractor, or regulatory agency) or to internal closure by conducting a post-mortem so that lessons learned can be captured and reflected in the next iteration.

Iterations in the inception and elaboration phases focus on management, requirements, and design activities. Iterations in the construction phase focus on design, implementation, and assessment. Iterations in the transition phase focus on assessment and deployment. Figure 8-3 shows the emphasis on different activities across the life cycle. An iteration represents the state of the overall architecture and the complete deliverable system. An increment represents the current progress that will be combined with the preceding iteration to form the next iteration. Figure 8-4, an example of a simple development life cycle, illustrates the differences between iterations and increments.
Progress can be measured as the % of components under configuration control, the % of demonstrable use cases, etc.

Figure 8-4. A typical build sequence associated with a layered architecture.