Architectural Blueprints – The “4+1” View Model of Software Architecture


Slides By: Jack ZhenMing Jiang
Outline

- Problems and Solutions
- “4 + 1” Views
  - The Logical Views
  - The Process Views
  - The Development Views
  - The Physical Views
  - The Use Case Views
- Conclusion
Problems

- Ambiguities in Boxes-and-Arrows Diagrams
  - Boxes can be programs, chunks of source code, physical computers, logical groupings of functionalities, …
  - Arrows can be data flow, control flow, or both.
- Architecture documents over-emphasize one aspect of software development or
- Architecture documents do not address the concerns of all stakeholders
Solutions

Different notation addresses separate concerns of the stakeholders and aspects of the software developments.

- Stakeholders:
  - end-users
  - developers
  - system engineers
  - project managers

- An Architectural Model – The "4 + 1" Views
  - Logical View
  - Development View
  - Process View
  - Physical View
  - Scenarios

Note: in the paper, terms like "architecture", "views", "blueprints" are used.
Logical Views – The Object-Oriented Decomposition

- Concerns
  - Primarily supports the functional requirements (services to users)

- Representation
  - Class diagrams (classes and logical relationships)
  - Class categories
  - Class utilities
An Example of Logical Architecture - PABX
A Bigger Example of the Logical Architecture – An Air Traffic Control System

- Display & User Interface
- External Interfaces - Gateways
- Simulation and Training
- Flight Management
- Air Traffic Management
- Aeronautical Information
- Basic Elements
- Mechanisms Services
The Process Architecture – The Process Decomposition

- Concerns
  - Nonfunctional requirements (concurrency, performance, availability, etc.)

- Representation
  - Different levels of abstractions
    - Processes and Threads
    - Major Tasks, Minor Tasks
    - Communication Mechanisms
      - Major tasks uses synchronous and asynchronous message communications, RPC, and event broadcasts, etc.
      - Minor tasks uses rendezvous or shared memory
An Example of the Process Architecture - PABX
An Example of the Process Architecture (Continued)
The Development Architecture – Subsystem Decomposition

- Concerns
  - Actual software module organization on the software development environment

- Representation
  - Layered Style (depends on same levels or layers below)
An Example of the Development Architecture

- 72 subsystems across 5 layers
- each layer about 10 to 50 modules
The Physical Architecture – Mapping the Software to Hardware

- **Concerns**
  - Primarily the nonfunctional requirements of the systems (like availability, reliability, scalability)

- **Representation**
  - Various forms (words, notations) over the process view
C, F, K are three types of computers - with different capacity - supporting different executables.
The Use Case View – Putting It All Together

- **Concerns**
  - Redundant with other views (thus “+1”)
  - Drivers to discover architectural elements
  - Validation and illustration to show the design is complete

- **Representation**
  - Similar to the logical view but a few variations
An Example of the Scenarios - PABX

1. Off-Hook
2. Dial tone
3. Digit
4. Digit
5. Open conversation

Joe: Controller
Joe: Terminal
Numbering plan
Class
Messages
Class Utilities

:Conversation
Correspondence Between the Views

- End-user Functionality
- Programmers Software management
- Logical View
- Development View
- Scenarios
- Process View
- Physical View
- Integrators
  - Performance
  - Scalability
- System engineers
  - Topology
  - Communications
Conclusions

<table>
<thead>
<tr>
<th>View</th>
<th>Logical</th>
<th>Process</th>
<th>Development</th>
<th>Physical</th>
<th>Scenarios</th>
</tr>
</thead>
<tbody>
<tr>
<td>Components</td>
<td>Class</td>
<td>Task</td>
<td>Module, Subsystem</td>
<td>Node</td>
<td>Step, Scripts</td>
</tr>
<tr>
<td>Connectors</td>
<td>association, inheritance, containment</td>
<td>Rendez-vous, Message, broadcast, RPC, etc.</td>
<td>compilation dependency, “with” clause, “include”</td>
<td>Communication medium, LAN, WAN, bus, etc.</td>
<td></td>
</tr>
<tr>
<td>Containers</td>
<td>Class category</td>
<td>Process (library)</td>
<td>Subsystem (library)</td>
<td>Physical subsystem</td>
<td>Web</td>
</tr>
<tr>
<td>Stakeholders</td>
<td>End-user</td>
<td>System designer, integrator</td>
<td>Developer, manager</td>
<td>System designer</td>
<td>End-user, developer</td>
</tr>
<tr>
<td>Concerns</td>
<td>Functionality</td>
<td>Performance, availability, S/W fault-tolerance, integrity</td>
<td>Organization, reuse, portability, line-of-product</td>
<td>Scalability, performance, availability</td>
<td>Understandability</td>
</tr>
<tr>
<td>Tool support</td>
<td>Rose</td>
<td>UNAS/SALE DADS</td>
<td>Apex, SoDA</td>
<td>UNAS, Openview DADS</td>
<td>Rose</td>
</tr>
</tbody>
</table>
Conclusions

- Different views address different concerns
- Not all views are necessary
- Lots of efforts needed to maintain these concurrent views, especially as the software system evolves
  - inconsistency, inaccurate
## Extra – IBM Introduction to “4+1 Views"

<table>
<thead>
<tr>
<th>Views</th>
<th>Notations</th>
</tr>
</thead>
<tbody>
<tr>
<td>The Logical View</td>
<td>Class Diagrams, Sequence Diagrams, Collaboration Diagrams</td>
</tr>
<tr>
<td>The Development View</td>
<td>Package Diagram</td>
</tr>
<tr>
<td>Process View</td>
<td></td>
</tr>
<tr>
<td>Physical View</td>
<td>Deployment Diagram</td>
</tr>
<tr>
<td>Use Case View</td>
<td>Case Diagram and Use Case Specifications</td>
</tr>
</tbody>
</table>