ICSA 2017 Tutorial: Study Foundations

Architecture Styles and Evolution
Robert Heinrich
<table>
<thead>
<tr>
<th>Time</th>
<th>Event</th>
</tr>
</thead>
<tbody>
<tr>
<td>09:00 – 09:10</td>
<td>Welcome and General Introduction</td>
</tr>
<tr>
<td><strong>09:10 – 09:40</strong></td>
<td><strong>Study Foundations</strong></td>
</tr>
<tr>
<td>09:40 – 10:00</td>
<td>Model-based Software Application Monitoring</td>
</tr>
<tr>
<td>10:00 – 10:30</td>
<td>Runtime Architecture Modeling and Visualization</td>
</tr>
<tr>
<td>10:30 – 11:00</td>
<td>Coffee Break</td>
</tr>
<tr>
<td>11:00 – 12:15</td>
<td>Introduction to the ExplorViz, Palladio, and iObserve Approaches with following Tool / Visualization Demos</td>
</tr>
<tr>
<td>12:15 – 12:30</td>
<td>Study Setup</td>
</tr>
<tr>
<td>12:30 – 14:00</td>
<td>Lunch</td>
</tr>
<tr>
<td>14:00 – 15:30</td>
<td>Comprehensibility Study</td>
</tr>
<tr>
<td>15:30 – 16:00</td>
<td>Coffee Break</td>
</tr>
<tr>
<td>16:00 – 16:30</td>
<td>Live Database Trace Visualization in Large Software Landscapes</td>
</tr>
<tr>
<td>16:30 – 17:00</td>
<td>Feedback and Open Discussion</td>
</tr>
</tbody>
</table>
Running Example Scenarios

SCENARIOS
Change Scenarios

Given an existing software system
- Insert new component “database persistence”
- Create new functionality “add billing”
- Update GUI “red → blue button”
ARCHITECTURE PATTERNS
Design vs. architectural patterns

- **Design pattern**
  - Small-scale / low-level solution
  - Usually a number of design patterns is “mixed”

- **Architectural pattern**
  - Large-scale / high-level solution
    (== balance design forces)
  - Dominate the structure of a whole software system

- Architectural patterns and design patterns usually are combined
Architecture Patterns

- Layers
- Client-Server
- Pipe & Filter
- Shared Data
- PAC

Referred to as “architecture style”
Single architecture style applied to a whole system
Architecture Patterns

LAYERS
Layers

- Expresses is-allowed-to-use relation
- Each layer consists of one or several modules
- Any piece of software is allocated to exactly one layer
- A lower layer cannot use a higher layer!
  - (“There is more to layers diagrams than the ability to draw separate parts on top of each other!” [1], p. 78)
- No call-backs
- Forwarding is OK
- Information hiding
- Better changeability
Example Scenarios

- “database persistence”
  - Where to add?
  - Which interface?

- “add billing”
  - All data accessible?
  - GUI + Business Layer + Persistence?

- “red → blue button”
  - 1. Which layer?
  - 2. Which component(s)?

→ Right layer, interfaces between layers, cycle avoidance
Architecture Patterns

CLIENT SERVER
Client Server Architecture

- Distributed system model which shows how data and processing is distributed across a range of components
- Set of stand-alone servers which provide specific services such as printing, data management, etc.
- Set of clients which call on these services
- Network which allows clients to access servers
Client Server: Example

Film and picture library

Client 1

Cataloge Server

Cataloge

Client 2

Video server

Film clip files

Client 3

Picture server

Digitized photographs

Client 4

Hypertext server

Hypertext web

Wide-bandwidth network

Figure: [2]
Client Server Characteristics

Advantages

- Distribution of data is straightforward
- Makes effective use of networked systems.
  - May require cheaper hardware
- Easy to add new servers or upgrade existing servers

Disadvantages

- No shared data model so sub-systems use different data organisation.
  - Data interchange may be inefficient
- Redundant management in each server
- No central register of names and services
  - It may be hard to find out what servers and services are available
Example Scenarios

- “database persistence”
  - Which server?
  - Local / remote?

- “add billing”
  - Client or server?
  - New client type?
  - Common client functionality?

- “red → blue button”
  - Client!
  - Server-side colour schema?

→ interface between server/client, (de-) centralisation criteria

Wide-bandwidth network

Client 1
- Catalogue server
- Video server

Client 2
- Film clip files

Client 3
- Picture server
- Digitized photographs

Client 4
- Hypertext server
- Hypertext web
Architecture Patterns

PIPE AND FILTER
Pipe and Filter (1)

- Elements:
  - Components with in- and out-ports
  - Pipe-Connectors with data-in and data-out roles
- Attached-to relation
- Topology: acyclic
- Example: unix-pipes
  
  `ps efl |grep mozilla |wc -l`

```
C1 -> C2 -> C3
```
Pipe and Filter (2)

- **Filter**
  - Incrementally transform some amount of the data at inputs to data at outputs
  - Stream-to-stream transformations
  - Preserve no state between instantiations
- **Pipe**
  - Move data from a filter output to a filter input
  - Pipes form data transmission graphs
- **Overall Computation**
  - Run pipes and filters (non-deterministically) until no more computations are possible
  - When transformations are sequential, this is a batch sequential model which is extensively used in data processing systems
  - Not really suitable for interactive systems
Pipe and Filter: Example

Invoice processing system

Figure: [2]
Example Scenarios

- “database persistence”
  - Data source or data sink?
  - What are input / output steps?

- “add billing”
  - Which processing steps inside billing?
  - In which sub-chain to add?

- “red → blue button”
  - Suitable architecture?
  - Which are interactive nodes?

→ interface between steps, thinking in terms of clear input / output relation, distinct locations during processing
Architecture Patterns

SHARED DATA
Shared Data (1)

- **Elements:**
  - **Component types:**
    - Shared data repositories
    - Data accessors (sinks and sources)
  - Connector types: data reading and writing
- **Attached-to relation**
- **Topology:** star (bus) or connected stars
Shared Data: Example

Data Oriented Repository (Blackboard)

Direct access

Computation

Memory
Example Scenarios

- "database persistence"
  - Blackboard!
  - Which subcomponent of the blackboard?

- "add billing"
  - New node operating on blackboard?
  - New data structure for blackboard?

- "red $\rightarrow$ blue button"
  - Which are interactive nodes?

$\rightarrow$ interface between nodes and blackboard, hierarchical data storage, strict separation of storage and processing/calculation/import/export, guide for new processing/input/output steps
Architecture Patterns

PAC – HIERARCHICAL SOFTWARE ARCHITECTURE
PAC - Overview

- Presentation: View + Control
- Abstraction: Model
- Control
  - Communication not only via `update()`
    (like in Model View Control, MVC)
- Mediator
PAC

Top Level Agent

Intermediate Level

Bottom Level

Data access

Working leaf

View Coord.

Pie chart

Histogram

Tabular
Example Scenarios

“database persistence”
- Which level?
- New Agent!
- Abstraction node!

“add billing”
- Which agent?
- P, C, and A!

“red → blue button”
- Which agent’s P?

→ Clear hierarchy, strict interfaces between levels and inside agents, repeating interaction patterns, unified extensions via new agents
CONCLUSION
Architecture Patterns

- Layers
- Client-Server
- Pipe & Filter
- Shared Data
- PAC
References
