Foundations of Software Engineering

Lecture 8: Software Architecture II
Christian Kästner

(with slides by Ivan Ruchkin)
Learning Goals

• Understand key parts of architectural process
• Use architectural styles and tactics for design decisions
• Make justified architectural decisions for new systems and within existing systems
• Review a proposed architecture
### Business Requirements Document

<table>
<thead>
<tr>
<th>Feature</th>
<th>Definition</th>
<th>Requirement Shopping</th>
</tr>
</thead>
<tbody>
<tr>
<td>Standard based and interoperable messaging protocol</td>
<td>Messaging protocol must be based on industry standards to enable interoperability</td>
<td></td>
</tr>
<tr>
<td>Send Only</td>
<td>Also called Push MEP is simple one-way messaging where a message is sent with no expectation response.</td>
<td></td>
</tr>
<tr>
<td>Receive only</td>
<td>Also called Pull MEP is a message pattern where a non-addressable sender supports the ability to explicitly obtain messages from another application. This can be used for exchanges</td>
<td></td>
</tr>
<tr>
<td>Request/Response exchange</td>
<td>Message pattern consists of one or more request/response pairs. The correlation between request and a response is well defined. In this response maybe deferred and the requesting application may or may not block application processing until a response is received</td>
<td></td>
</tr>
<tr>
<td>Diagnostics</td>
<td>Authentication, diagnostic, logging &amp; routing information should be included in the message and not the payload</td>
<td></td>
</tr>
<tr>
<td>Reliability</td>
<td>Protocol capability to support assured and single delivery to the receiving application with no loss</td>
<td></td>
</tr>
</tbody>
</table>

So far in course

- **Requirements**
- **Architecture**
- **Implementation**
Levels of abstraction

- Requirements
  - high-level “what” needs to be done

- Architecture (High-level design)
  - high-level “how”, mid-level “what”

- OO-Design (Low-level design, e.g. design patterns)
  - mid-level “how”, low-level “what”

- Code
  - low-level “how”
What is architecture?

Architecture as structures and relations (the actual system)

Architecture as documentation (representations of the system)

Architecture as process (activities around the other two)
Architectural Styles and Tactics
Architectural style (pattern)

- Broad principle of system organization
- Describes computational model
  - E.g., pipe and filter, call-return, or publish-subscribe
- Related to one of common view types
  - Static, dynamic, physical
Architectural style (pattern)

- Broad principle of system organization
- See reading
Architectural style (pattern)
Client-server pattern

• Separation of clients and servers
  – Servers provide services; known and “stable”
  – Clients request services; come and go
• Varieties: synchronous/asynchronous
• Impact on security, performance, scalability
• Examples: TCP, HTTP, X11
Layered system
Client-server style

Source: wikimedia commons
Tiered architecture
Where to validate user input?

Example: Yelp App

Example: There are a few times in life when a meal is so expertly crafted and planned that it is nothing short of genius. Last night, I had one of those meals - the Mahi Mahi.

The dish was excellently prepared. Grilled, juicy, and fresh without a hint of fishiness. A glaze of tangerine sauce brought a hint of tart sweetness. The fish was placed on a mound of sweet plantain rice. The combination of the fish and rice alone was to die for!
Services?
Architectural Styles

• Pipes and Filters
• Object-Oriented Organization, Services
• Event-Based, Implicit Invocation
• Layered System
• Repositories
• ...

Tactics

• Architectural techniques to achieve qualities
• Smaller scope than architectural patterns
  – Problem solved by patterns: “How do I structure my (sub)system?”
  – Problem solved by tactics: “How do I get better at quality X?”
• Collection of common strategies and known solutions
Many tactics described in Chapter 5

Brief high-level descriptions (about 1 paragraph per tactic)

Second and more detailed third edition available as ebook through CMU library.
Case Study: ROS
"Robot Operating System", open source

The philosophical goals of ROS can be summarized as:

- Peer-to-peer
- Tools-based
- Multi-lingual
- Thin
- Free and Open-Source
Fig. 1. A typical ROS network configuration

Quality Goals?
"A Distributed, Modular Design"

• users can use as much or as little of ROS as they desire
• modularity of ROS allows you to pick and choose which parts are useful for you and which parts you'd rather implement yourself
• large community of user-contributed packages (3000 packages)
Architectural Style?

- Pipes and Filters
- Object-Oriented Organization, Services
- Event-Based, Implicit Invocation
- Layered System
- Repositories
- ...
ROS Communication Infrastructure

- **Message Passing**
  - Publish/subscribe for channels
  - Messages interfaces through IDL (cross-language)
- **Recording and Playback of Messages**
- **Remote procedure calls**
- **Share configuration through global key-value store**
Tradeoff discussion

- Decoupling
- Reuse, Extensibility
- Reliability
- Understandability
- Performance
- Community contributions
Architecture Design Process
What is architecture?

Architecture as structures and relations
(the actual system)

Architecture as documentation
(representations of the system)

Architecture as process
(activities around the other two)
Architecture design process

• Choose part or whole system to focus on
• Understand relevant requirements
• Choose a notation
  – Type of view, vocabulary of elements
• Create a design
  – Patterns, tactics
• Evaluate
• Go vs no-go
  – Issues feed back into process
Architecture design process

Choose scope ➔ Understand relevant requirements ➔ Choose a notation ➔ Create/refine a design ➔ Evaluate ➔ “Go”

Source: ACDM, ADD
Architectural decisions

• Heart of architecture – deciding which path to go
• Involve tradeoff analysis
• Representing the alternatives clearly – half of work
Architectural decisions

• Software architecture is *design*
  
  “Engineering design is [...] a decision-making process (often iterative), in which the basic sciences, mathematics, and engineering sciences are applied to convert resources optimally to meet a stated objective” – ABET

• A decision is a step in the process
  
  – **Record rationale!** (not just diagrams)
  – Tradeoffs
  – Backtracking
Architecture evaluation

• Goal: does the architecture satisfy requirements?
• ATAM – Architecture Tradeoff Analysis Method
  – Present requirements
  – Present architecture
  – Analyze architecture
  – Present results – risks and non-risks
Utility

Performance
- Data Latency
  - (M,L) Minimize storage latency on customer DB to 200 ms.
  - (H,M) Deliver video in real time
- Transaction Throughput

Modifiability
- New product categories
  - (L,H) Add CORBA middleware in < 20 person-months
- Change COTS
  - (H,L) Change web user interface in < 4 person weeks

Availibility
- H/W failure
  - (M,M) Restart after disk failure in < 5 mins
- COTS S/W failures
  - (H,L) Network failure is detected and recovered in < 1.5 mins

Security
- Data confidentiality
  - (L,H) Credit card transactions are secure 99.999% of time
- Data integrity
  - (L,H) Customer database authorization works 99.999% of time

Source: arnon.me
Athena – code review system (original design)
Athena – code review system

Source: Jansen and Bosch 2005
Source: Jansen and Bosch 2005
Architecture design process

Choose scope → Understand relevant requirements → Choose a notation → Create/refine a design → Evaluate

Source: ACDM, ADD
Challenges of architecting

• Describe the system that is not built yet
• Domain knowledge is essential
• Huge space of options
• Heavily reliant on judgment
Learning Goals

• Architectural patterns provide common abstractions of system structures with known tradeoffs

• Tactics provide common strategies to achieve quality goals, often useful during design to explicitly discuss alternatives

• Architecture evaluation provides early quality assurance
Further Readings