Conceptual, Concrete & Reference Architecture

“Linux Case Study”

“A Reference Architecture for Web Servers”

CS 446 / 646 ECE452
May 27th, 2011
Conceptual Architecture

Intent

• “direct attention at an appropriate decomposition of the system without delving into the details of interface specification”[1]

• appropriate decomposition
  – focus on key constructs & abstractions rather than on technical/implementation details

Conceptual Architecture

Includes

- what a system does (requirements / functionality)
- identification of significant components & connectors
  - component responsibilities
  - component interactions
  - control & data flow
- identification of architectural (non-functional) concerns
  - e.g. security
  - other crosscutting concerns
Conceptual Architecture

Provides

- communicating architectural details with
  - stakeholders (technical & non-technical)

Input artifacts

- reference architecture (more on this later)
- requirements (functional & non-functional)
- documentation & code
  - Hmm...these two seem out of place?
Conceptual Architecture

Activities

- capture system functionality
  - requirements to use case descriptions
- capture system properties
  - evolution, system load, portability
- capture system constraints
  - legacy components, third party components
  - resources, time
  - technical capabilities
Concrete Architecture

Intent

- implementation specific architecture
  - decomposition into (implementation specific components)
  - identification of actual relationships
- actual
  - e.g. identify third party, COTS components
Concrete Architecture

Includes

• what a system does (requirements / functionality)
• how will it do it
• identification of significant components & connectors
  – component responsibilities
  – component interactions
  – control & data flow
• realization of architectural non-functional concerns
Concrete Architecture

Input Artifacts

- ask the class
- can we extract concrete architecture from code
  - isn't that reverse engineering?
  - choice of tools
Example

Linux

- lack of formal architecture
  - conceptual or concrete
- considerable code size
- fragmented documentation
  - individual systems well defined but not the overall
- based on
Example

Linux Conceptual Architecture

- input artifacts
  - Unix & Minix descriptions
  - Linux documentation
Observations

- seven kernel sub-systems
- dependencies
  - initialization depends on all
  - library sub-system forms the core
Linux Concrete Architecture

General technique

- **inspection**
  - source (directory structures, packages, file naming etc)
  - compiled artifacts
- **clustering**
  - grouping of components
- **discovery**
  - inter-component dependency
Linux Concrete Architecture

Legend: Subsystem — extracted dependency
Comparison

Conceptual Architecture

Concrete Architecture

Legend: Subsystem depends on

Legend: Subsystem extracted dependency
Observations

Structural

- same subsystem decomposition
  - despite different input artifacts

Dependency Relationships

- higher than expected inter-component dependency found
- why would this be?
  - “avoided existing interfaces for better efficiency”
Architectural Drift & Erosion

Architectural erosion

- *conceptual architectural violations*

Architectural drift

- *concrete architecture shifts away from conceptual architecture*
Reference Architecture

Definitions

- “A reference architecture for a domain defines the fundamental components of the domain and the relations between them”

- “A reference architecture for a domain is an architectural template for all the software systems in the domain”
Example 1

Automobile

- attributes
  - transportation
  - wheels
  - steering
  - speed / gears
Is This an Automobile?

But this meets the attribute list
Properties

Domain

- what is a domain?
- reference architecture covers a whole domain

Fundamental components

- universal abstractions
  - applicable across the domain
- interaction of these abstractions

Template

- a *product* architecture is an instantiation of the reference
Properties

Availability

- well known for mature domain
  - compilers, operating systems
- absent for new domains
  - web servers
Benefits

Documentation

• captures the main ideas and components across domain
• provides a higher level abstraction for architecture itself
  – we don't have to reinvent the wheel or the architecture

Communication

• provides a common vocabulary
  – the *wheel* is too big
  – *braking* distance of the car is reasonable
  – 0 to 60 in 10 seconds
Benefits

Evaluation

- aids in the comparison of the different product architectures in the same domain
  - electric vs. hybrid
  - sedan vs. coupe
Web Server Reference Architecture

Domain:

- web servers, application servers

Web servers

- Apache
- AOL Server
- Jigsaw
# Summary of Web Servers

<table>
<thead>
<tr>
<th>Web Server</th>
<th>Dev Type</th>
<th>1st Release</th>
<th>Code Size (KLOC)</th>
<th>Impl</th>
<th>Arch stable for (years)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Apache</td>
<td>Open source</td>
<td>April 1995</td>
<td>80</td>
<td>C</td>
<td>5</td>
</tr>
<tr>
<td>AOL Server</td>
<td>Commercial</td>
<td>May 1995</td>
<td>164</td>
<td>C &amp; TCL</td>
<td>-</td>
</tr>
<tr>
<td>Jigsaw</td>
<td>Educational</td>
<td>May 1996</td>
<td>106</td>
<td>Java</td>
<td>2.5</td>
</tr>
</tbody>
</table>
Deriving Reference Architecture

Process

- **step 1**: derive a conceptual architecture for each
  - **propose** a conceptual architecture
    - using domain knowledge and available documentation
  - **refine** the conceptual architecture
    - using the concrete architecture

- Q: did we not say that a reference architecture should be an input artifact to a conceptual architecture?
Deriving Reference Architecture

Process

- step2: derive a reference architecture from step 1
  - **propose** a reference architecture based on
    - domain knowledge
    - common structure between the conceptual architecture
  - **refine** the reference architecture
    - using the conceptual architecture (from step 1)
Web Server Reference Architecture

Architectural Style

- follows pipes & filter architectural style
  - hmmm.... does it really?
  - what other architectural styles better define web servers?
Flexibility of Reference Architecture

Intent

- “To be useful a reference architecture must be flexible enough to encompass many product architectures”

- what does flexibility mean?
  - security flexibility?
  - concurrency flexibility?
Apache

Reference & Conceptual Mapped

Legend:

Subsystem

control flow
'all depend on'

<table>
<thead>
<tr>
<th>WATERLOO</th>
</tr>
</thead>
<tbody>
<tr>
<td>CHERITON SCHOOL OF</td>
</tr>
<tr>
<td>COMPUTER SCIENCE</td>
</tr>
</tbody>
</table>

2011-05-04
CS446/646 ECE452

30
AOL

reception and request analysis

Communication Driver

Daemon - Core

resource handler

URL Handle

access control

NSPerm

record transaction

NSLog

util

OSAL

Timer

Util

Database Interface

TCL Interpreter

NSThread

WATERLOO

CHERITON SCHOOL OF COMPUTER SCIENCE
Observations

Structural

• subsystem *organization* is the main difference and not the subsystem *responsibility*

• reference architecture is abstract
  – does not depend on
    • development methodology
    • platform
    • implementation concerns