Architecture? Architecting!

André van der Hoek
University of California, Irvine
Department of Informatics & Institute for Software Research

andre@ics.uci.edu
Repositioning the Workshop Theme

• Software Architecture Challenges in the 21st Century

• Software Architecting Challenges in the 21st Century
Architecting

• An activity

• A design activity

• A crucial design activity

• What do software architects actually do and how can we better assist them?
Architecting
Architecting

- Performance
  - Response time < 1sec
  - Ballistics kernel computation
  - Inter-node message transfer

- Utility
  - Modifiability
    - New msg format < 1 p.m.
    - New msg data type < 1 p.w.
    - Change from JVM to EJVM
    - Change web UI < 2 p.w.

- Availability
  - HW failure MTTR < 5 min.
  - Diskless operation
  - Failure of commander node
  - Survive single network failure

Scenario: S12 (Detect and recover from HW failure of main switch.)
Attribute: Availability
Environment: normal operations
Stimulus: CPU failure
Response: 0.999999 availability of switch

Architectural decisions

<table>
<thead>
<tr>
<th>Backup CPU(s)</th>
<th>Risk</th>
<th>Sensitivity</th>
<th>Tradeoff</th>
</tr>
</thead>
<tbody>
<tr>
<td>No backup Data Channel</td>
<td>R9</td>
<td>S3</td>
<td>T3</td>
</tr>
<tr>
<td>Watchdog</td>
<td>S4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Heartbeat</td>
<td>S5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Failover routing</td>
<td>S6</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Reasoning:
- ensures no common mode failure by using different hardware and operating system (see Risk 8)
- worst-case rollover is accomplished in 4 seconds as computing state takes ...
- guaranteed to detect failure with 2 seconds based on rates of heartbeat and watchdog ...
- watchdog is simple and proven reliable
- availability requirement might be at risk due to lack of backup data channel … (see Risk 9)

Architecture diagram:

Primary CPU (OS1)

Backup CPU w/watchdog (OS2)

Heartbeat (1 sec)

Switch CPU (OS1)
Architecting
Some Questions

• If an architecture is (components, connectors, configurations), what are the above?

• If an architecture is (the set of principal design decisions), where are those decisions recorded and how are they kept track of?

• How do architects make effective architectural decisions?

• How do those decisions lead to an eventual architecture?
A Personal Research Agenda

- Study of general design literature
- Study software designers “in action”
- Tool support
- Education

Understand software design
General Design Literature

Etc., etc., etc...
General Design Literature

• Designers...
  – ...employ a mix of opportunistic and rationalistic reasoning
  – ...exploit freedom of expression
  – ...mix different representations
  – ...leverage the existence of different kinds of constraints
  – ...engage in a reflective conversation with materials
  – ...understand the role and pitfalls of communication
  – ...

• This seems to hold true for software designers as well [Zannier] [Dekel] [...]

6/8/2009 André van der Hoek, University of California, Irvine 11
Study Software Designers in Action

• Traffic simulator prompt given to experienced designers and architects in highly respectable organizations (Adobe, Intuit, Amberpoint, Callfire, ...)

• Two hours to design, at the whiteboard, video taped and interviewed afterwards
Video
Analysis
Preliminary Findings/Intuitions

• Good designers...
  – ...are aware of the process and explicitly manage it
  – ...know when to set items aside
  – ...know when to stop drilling down
  – ...continuously look for issues with the current design
  – ...

• Surprise (?): not every experienced designer is a good designer
Preliminary Findings/Intuitions

• Five types of design
  – application design
  – interaction design
  – architecture design
  – implementation design
  – (maintenance design)

• Designers move across these fluidly and constantly
  – even when their job titles signify a single specialty

• We have encountered architects who code, who read code from others, who solely talk to customers, ...
A Personal Research Agenda

- Study of general design literature
- Study software designers “in action”
- Understand software design
- Tool support
- Education
Theory of the interdisciplinary product-process framework of design
Understand Software Design

Theory of the interdisciplinary product-process framework of design
Tool Support
Approach
Calico Demo
# Calico Features

<table>
<thead>
<tr>
<th>Need</th>
<th>Features</th>
</tr>
</thead>
<tbody>
<tr>
<td>Low detail</td>
<td>Freehand drawing, uncluttered workspace</td>
</tr>
<tr>
<td>Ambiguity</td>
<td>Uninterpreted strokes, amorphic scraps</td>
</tr>
<tr>
<td>Shifting focus</td>
<td>Grid, low-cost revising</td>
</tr>
<tr>
<td>Variety of representations</td>
<td>Palette, minimal semantics</td>
</tr>
</tbody>
</table>
Evaluation
Evaluation

• All features of Calico were used, but not all teams used all features
• Grid ubiquitously valued
• Scraps, when used, were used effectively, but they also posed some hurdles
  – scraps take on a type as identity, which brings with it expectations of interactions
  – scraps effectively deaden an otherwise usable space around their borders
• Physical set up matters
A Personal Research Agenda

- Study of general design literature
- Study software designers “in action”

Understand software design

- Tool support
- Education
• How to bring all this into the classroom?
Education

• How to bring all this into the classroom?

• Informatics 121 “Software Design I”
  – design as a creative endeavor

• Informatics 122 “Software Design II”
  – design as we know it
Summary & Challenges

• Whirlwind tour of a personal research agenda

• Much of this work is ongoing

• Our perspective muddies design and architecture (or perhaps clears it up?)
  – software engineering as a design discipline
  – design decisions are made throughout
  – architecture pertains to the design decisions defining a system’s key internal structures

• Our perspective is rooted in what we have learned from designers in the real world
Summary & Challenges

• Challenge 1: Build an understanding of how designers (architects) perform their work, especially how good designers perform their work

• Challenge 2: Realize a decision-oriented tool set that amplifies good designerly behavior
  – right tool at right time
  – early decisions guide later decisions
  – later decisions as important as early decisions
  – key: a design emerges, and continuously changes

• Challenge 3: Bring a design focus into our software engineering classrooms

• Challenge 4: Does this matter?