Software Architecture

Jonathan Aldrich, Ph.D.
Associate Professor
Institute for Software Research
School of Computer Science
Carnegie Mellon University

January 2014

© 2014 Jonathan Aldrich  These materials may not be duplicated or distributed without advance written permission from the author.
Outline

• What is software architecture?
• What are its benefits?
• How to develop a software architecture?
• How to document a software architecture?
• Conclusion and takeaways
What is (Building) Architecture? And why is it useful?
What is Software Architecture?

Software architecture represents the high-level design of a software system, showing how desired system properties are achieved.
Where Architecture Fits

- **Requirements**
  - What the system should do
  - What properties it should have

- **Architecture**
  - High-level design, how properties are achieved

- **Detailed design**
  - Lower-level design, how system functions

- **Code**
  - How the system actually works
Two Architectures for Web Search

How does architecture affect system properties?
- Modifiability / ease of change
- Consistency of results
- System cost
- Scalability of system
- Reliability of system
Two Architectures for Sending Email

Which architecture was better in 1980? Which was better in 2000?

Factors to consider

- Simplicity
- Efficiency
- Security
Two Architectures for Sending Email

- Sendmail was the dominant email client from 1982 until 2000.
- In 1988 the Morris worm, the first internet worm, took advantage of a sendmail vulnerability; many other vulnerabilities have been found since.
- By 2000 sendmail had begun a steep and permanent decline, and qmail was growing exponentially.
Architecture is an Abstraction

- Focus on principal design decisions
  - **Structure** – components and connections
  - **Behavior** – responsibilities of each component, high level algorithms
  - **Interaction** – rules governing how components communicate
  - **Quality attributes** – strategy for achieving
  - **Implementation** – language, platform, libraries, etc.

  - *Any decision that impacts key stakeholder concerns or has global impact on the program*

- Elide unimportant details
  - Decisions that are **internal to a component**
    - i.e. which other components cannot depend on
    - e.g. internal algorithms, data structures, local design patterns
  - AND do not impact **key stakeholder concerns**

*Architecture is design, but not all design is architectural*
Outline

• What is software architecture?
• **What are its benefits?**
• How to develop a software architecture?
• How to document a software architecture?
• Conclusion and takeaways
Architecture Benefits: System Properties

• Architecture is not about a system’s function, but rather the system’s properties

• Some properties and their consequences
  – **Fitness**: performance, reliability, security → competitive advantage
  – **Modifiability/ease of changing**: business agility
  – **Reuse** of code → reduced cost
Business Case: Cell Phones [M. Bass]

- Market is driven by killer products
  - e.g. Razr, iPhone
- Most profit is made at initial release
  - Premium charged on initial sales
  - Drops rapidly when copycats arrive
- Business model
  - Be first to market with new features
- Software quality attributes
  - Ability to change rapidly and at low cost
- True story: effect of architecture
  - Leading cell phone manufacturer
    - not enough new products
    - starts to lose market share, decides to release faster
    - leads to trouble: e.g. tone so loud it damages hearing ➔ recalls
  - Analysis
    - software structure did not enable rapid change
    - too costly to rewrite software from scratch
    - eventually left cell phone business entirely
Telecom Architecture Scenario

• Context: telecommunications wholesaler
  – Provides services both to end users and resellers
  – 8 legacy applications built with different interfaces, technologies

• Challenges
  – Duplicate functionality between end user / reseller channels
  – Several manual steps in process; difficult to automate
  – Difficult to roll out new services
  – Need to free reserved resources when an operation is canceled

• What would you do?
Telecom Architecture Solution

• Service-Oriented Architecture
  – Wrap legacy applications with a standard web services interface
  – Automate tasks using scripting (BPEL)
  – Share common operations, services between the different channels
  – Incorporate undoing reservations into the script

• Impacts
  – Common interface enabled automation $\rightarrow$ lower cost
  – Also facilitates replacing components $\rightarrow$ agility
  – Scripts make business operation changes easier $\rightarrow$ agility
  – Reuse of common components $\rightarrow$ lower cost
  – Built-in undo avoids wasting resources $\rightarrow$ reliability, lower cost
Outline

- What is software architecture?
- What are its benefits?
- **How to develop a software architecture?**
- How to document a software architecture?
- Conclusion and takeaways
How to Develop a Software Architecture

• Investment driven by **complexity** and **scale**
• Fitness evaluated by key **risks**
• Design appropriate for the **domain**
• Structure aligned with the **organization**
Tradeoffs in Architecture Investment

Source: Boehm, Valerdi, Honour 2008
Driving Architecture via Risks

• Low risk \(\rightarrow\) little investment needed
  – Typically use a reference architecture (e.g. 3-tier web)
  – Reference architectures capture (“hoist”) known domain risks

• Otherwise, evaluate architecture fitness using risks

• Major risks are architectural drivers
• Example drivers and architectural analysis approaches
  – Maintainability/Reuse: variation, interface standards
  – Performance: queuing theory, real-time analysis
  – Security: threat modeling
  – Distributed development: interfaces between teams
Domain-Specific Architectures

- Pattern: A reusable solution to a recurring architecture design problem

- Example: **3-tier web applications**
  - Data tier stores data in a database
  - Logic tier implements business logic
  - Presentation tier handles web requests
  - *Benefits?*
Domain-Specific Architectures

• Pattern: A reusable solution to a recurring architecture design problem

- Example: **3-tier web applications**
  - Data tier stores data in a database
  - Logic tier implements business logic
  - Presentation tier handles web requests
  - Benefits include **modifiability, scalability**
Architecture-Organization Alignment

• Conway’s Law
  Any organization that designs a system...will inevitably produce a design whose structure is a copy of the organization's communication structure (Conway, 1968)

• Case example: product line
  – Applications initially developed independently
  – Desired reusable library to reduce cost, increase agility
  – Failed to build library using existing teams
  – Success required a team dedicated to the core library.
Outline

- What is software architecture?
- What are its benefits?
- How to develop a software architecture?
- How to document a software architecture?
- Conclusion and takeaways
Architectural Views

• Many possible “views” of architecture
  – Implementation structures
    • Modules, packages
    • Modifiability, Independent construction, …
  – Run-time structures
    • Components, connectors
    • Interactions, dynamism, reliability, …
  – Deployment structures
    • Hardware, processes, networks
    • Security, fault tolerance, …
Why Document Architecture?

• Blueprint for the system
  – Artifact for early analysis
  – Primary carrier of quality attributes
  – Key to post-deployment maintenance and enhancement

• Documentation speaks for the architect, today and 20 years from today
  – As long as the system is built, maintained, and evolved according to its documented architecture
What is Wrong Today?

• In practice today’s documentation consists of
  – Ambiguous box-and-line diagrams
  – Inconsistent use of notations
  – Confusing combinations of viewtypes
• Many things are left unspecified:
  – What kind of elements?
  – What kind of relations?
  – What do the boxes and arrows mean?
  – What is the significance of the layout?
What could the arrow mean?
What could the arrow mean?

- Many possibilities
  - A passes control to B
  - A passes data to B
  - A gets a value from B
  - A streams data to B
  - A sends a message to B
  - A creates B
  - ...

© 2014 Jonathan Aldrich
Representing C&C Views

- **system**
- **component**
- **connector**
- **port**
- **role**
Guidelines: Avoiding Ambiguity

- Always include a legend
- Define precisely what the boxes mean
- Define precisely what the lines mean
- Don’t mix viewtypes unintentionally
  - Recall: Module (classes), C&C (components)
- Supplement graphics with explanation
  - Very important: rationale (architectural intent)
- Do not try to do too much in one diagram
  - Each view of architecture should fit on a page
  - Use hierarchy
Technique: Hierarchy

• Use hierarchy to define elements in more detail in separate views
• Helps keep an architectural description manageable
Top-level C&C View

Legend

- Web Component
- LDAP Directory
- RDBMS
- Direct Adapter
- Indirect Adapter
- Controller
- Viewer
- Interface
- SOAP Connector & roles
- LDAP Connector & roles
- DB Connector & roles
- RMI Connector & roles
- Event Bus Connector & roles
- System Boundary

Administrator Console

Join Engine

Meta Viewer

Integrated Data Rep

Façade Component

Rule & Configuration DB

Transaction Log

Change Log

External LDAP1

External LDAP2

External DB1

External DB1

Adapter Registry

Adapter Manager

Direct Adapter1

Direct Adapter2

Indirect Adapter1

Indirect Adapter2

Façade Component

Viewer

System Boundary
Showing Details of Component
Outline

• What is software architecture?
• What are its benefits?
• How to develop a software architecture?
• How to document a software architecture?
• Conclusion and takeaways
Conclusion: Key Takeaways

• Architecture captures **high-level design** of software
  – Structure and communication
  – Key design decisions

• Enables desired **properties** of system
  – **Reuse** → reduce cost
  – **Modifiability** → business agility
  – **Fitness for use** → competitive advantage
Extra: Architecture Research at CMU

- Architecture modeling and analysis
  - Verify security, performance properties
  - Ensure an architecture is realizable

- Architecture adaptation models
  - React to breakdowns, security breaches
  - Adapt to changing resources (e.g. network bandwidth)

- Architecture-based development
  - Synchronizing code and architecture
  - Verifying constraints at architectural interfaces
References and Further Reading

**References**


**Further Reading**
