Software Architecture for DSD

DSD Team

Overview

- What is software architecture and why is it so important?
- The role of architecture in determining system qualities
- Using architectural structures to achieve design goals
Working Definition

“The software architecture of a program or computing system is the structure or structures of the system, which comprise software components, the externally visible properties of those components, and the relationships among them.”

From *Software Architecture in Practice, Bass, Clements, Kazman*

---

Examples

- **An architecture comprises a set of**
  - Software components
  - Component interfaces
  - Relationships among them

<table>
<thead>
<tr>
<th>Structure</th>
<th>Components</th>
<th>Interfaces</th>
<th>Relationships</th>
</tr>
</thead>
<tbody>
<tr>
<td>Calls Structure</td>
<td>Programs (methods, services)</td>
<td>Program interface and parameter declarations</td>
<td>Invokes with parameters (A calls B)</td>
</tr>
<tr>
<td>Data Flow</td>
<td>Functional tasks</td>
<td>Data types or structures</td>
<td>Sends-data-to</td>
</tr>
<tr>
<td>Process</td>
<td>Sequential program (process, thread, task)</td>
<td>Scheduling and synchronization constraints</td>
<td>Runs-concurrently-with, excludes, precedes</td>
</tr>
</tbody>
</table>
Implications of the Definition

“The software architecture of a program or computing system is the structure or structures of the system, which comprise software components, the externally visible properties of those components, and the relationships among them.” ~ Bass, Clements, Kazman

- Systems typically comprise more than one architecture
  - There is more than one useful decomposition into components and relationships
  - Each addresses different system properties or design goals
- It exists whether any thought goes into it or not!
  - Decisions are necessarily made if only implicitly
  - Issue is who makes them and when
- Many “architectural specifications” aren’t

Is it Architecture?

- Typical (but uninformative) architectural diagram
  - What is the nature of the components?
  - What is the significance of the link?
  - What is the significance of the layout?
“...The earliest artifact that enables the priorities among competing concerns to be analyzed, and it is the artifact that manifests the concerns as system qualities.”

Effects of Architectural Decisions (What?)

- What kinds of system and development properties are affected by the architectures?
- System run-time properties?
- System static properties?
- Production properties? (effects on project)
- Business/Organizational properties?
Effects of Architectural Decisions (What?)

- What kinds of system and development properties are affected by the system structure(s)?
- **System run-time properties**
  - Performance, Security, Availability, Usability
- **System static properties**
  - Modifiability, Portability, Reusability, Testability
- **Production properties? (effects on project)**
  - Work Breakdown Structure, Project cost, time to market
- **Business/Organizational properties?**
  - Lifespan, Versioning, Interoperability, Target market

Effects of Architectural Decisions (Who?)

- There are many interested parties (stakeholders) with many diverse and often conflicting interests
- **Examples of stakeholder interests**
  - Project Management
    - What increments are possible?
    - How soon can I release a useful system?
    - How can I divide the work among teams?
  - Architects
    - What changes may be needed and should be easy to make?
    - How does the system address quality goals like performance, security, etc.?
  - Software Developers
    - What responsibilities must the implementation of this module satisfy?
    - What capabilities will be provided by other parts of the system?
- **Important as their interests often defy mutual satisfaction**
  - There are inherently tradeoffs in most architectural choices
  - E.g. Performance vs. security, initial cost vs. maintainability
Summary

- Earliest set of design decisions – hence, most influential and hardest to change
- Determines a wide range of critical system, production, and business properties
- A product of tradeoffs between conflicting demands by different stakeholders
- Implication: good design is a balance of technical, business, and organizational influences

Designing Architecture for Distributed Teams
Which structures should we use?

- Choice of structure depends on which qualities we want to design for
- Compare to architectural blueprints
  - Different blueprint for load-bearing structures, electrical, mechanical, plumbing
  - Designing for particular qualities requires the right view
    - Process structure for run-time property like performance
    - Module structure for development property like maintainability
- Implication: Schematic Map structures should address major project risks
  - Communication overhead and uncertainty
  - Fixed deadlines

DSD Architectural Design Goals

- Limit the necessity for communication: decompose the system into parts such that
  - Each part can be assigned to a different team and developed independently
  - Parts can be independently verified
  - Properties of the system that are likely to change (e.g. implementation details) are encapsulated
  - Only properties of the system that are unlikely to need to be changed are part of the architecture
  - Role of each part in the overall system is clear (and when together, implement the requirements)
- Support incremental development
  - Can identify a subset of parts that implement a useful subset of the system
  - Can add capability by extending the working system
Which structures should we use?

- What kind of “views” of the system are useful?
- What do we need to be able to see and manipulate?
- How do we communicate important design decisions?

Key Architectural Structures

- **Module Structure**
  - Decomposition of the system into work assignments

- **Module Interface Specs**
  - Define Services Provided
  - Define syntax and semantics for accessing services
  - Define data types, program effects, ...
  - Define test cases
  - Record design decisions and implementation notes

- **Uses Structure**
  - Describes the allowed “uses” relationships between modules
    and limits what other modules the implementer of a module
    may use.
Designing the Module Structure

Goals for the Module Structure
Design Principles

Modularization a la Parnas

- For large, complex software, must divide the development into work assignments (WBS). Each work assignment is called a "module."
- A module is characterized by two things:
  - Its interface: what the module provides to other parts of the systems. Properties other parts of the system may assume and services they can use.
  - Its secrets: what the module hides (encapsulates). Properties other parts of the system may not use and should not depend on.
- Modules are abstract, design-time entities that may or may not directly correspond to programming components like classes/objects
Role of Modules

- Modularization affects broadest set of system qualities
- Communicate/verify
  - Can understand system one component at a time
  - Can review or test individual units before the whole
- Managerial
  - Provides units for work-breakdown, scheduling
  - Allows concurrent development
- Coding
  - Can write modules with little knowledge of other modules
  - Replace modules without reassembling the whole system
- Flexibility/Maintainability/Reuse
  - Anticipated changes affect only a small number of modules (usually one)
  - Can calculate the impact and cost of change
  - Provides unit of reuse

Notional Modules
Top-Level Schematic Map

In-Class Exercise

- A quick review of the Schematic Map design
- Basic question: does it meet our design goals?
  - Limit the necessity for inter-team communication: decompose the system into parts such that
    1. Each part can be assigned to a different team and developed independently
    2. Parts can be independently verified
    3. Properties of the system that are likely to change (e.g., implementation details) are encapsulated
    4. Only properties of the system that are unlikely to need to be change are part of the architecture
    5. Role of each part in the overall system is clear (and when together, implement the requirements)
  - Support incremental development (next week)
- Specifically, describe whether the design meets these goals and why (focus on 1, 3, and 4)
End