Acknowledgment
Some of the material in this course are developed by David Garlan, Tony Lattanze, Nenad Medvidovic, Joao Sousa, and Sam Malek and used with their permission.
Outline

- what is software architecture?
  - relation to software engineering
    - relation to engineering
Every engineered artifact has an architecture just like every building has an architecture and someone who acted as an architect.
Sometimes refereed to as the system’s blueprint

Here is the blueprint of a car
Importance of architecture in engineering disciplines

- An architecture may be
  - elegant and effective, or
  - clumsy and dysfunctional

- Architectural models provide the means to represent and communicate the engineered artifact’s properties

- Facilitates communication among the stakeholders
Examples of software architecture
More examples

FIGURE 7. Flight Computer Operating System (The FCOS dispatcher coordinates and controls all work performed by the on-board computers.)

More examples
More examples

<table>
<thead>
<tr>
<th>Client Layer*</th>
</tr>
</thead>
<tbody>
<tr>
<td>Access domain management</td>
</tr>
<tr>
<td>Buffering and record-level I/O</td>
</tr>
<tr>
<td>Transaction coordination</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Agent Layer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Implementation of standard server interface</td>
</tr>
<tr>
<td>Logger, agent, and instance tasks</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Helix Directories</th>
</tr>
</thead>
<tbody>
<tr>
<td>Path name to FID mapping</td>
</tr>
<tr>
<td>Single-file (database) update by one task</td>
</tr>
<tr>
<td>Procedural interface for queries</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Object (FID directory)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Identification and capability access (via FIDs)</td>
</tr>
<tr>
<td>FID to tree-root mapping; table of (FID, root, ref_count)</td>
</tr>
<tr>
<td>Existence and deletion (reference counts)</td>
</tr>
<tr>
<td>Concurrency control (file interlocking)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Secure Tree</th>
</tr>
</thead>
<tbody>
<tr>
<td>Basic crash-resistant file structure</td>
</tr>
<tr>
<td>Conditional commit</td>
</tr>
<tr>
<td>Provision of secure array of blocks</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>System</th>
</tr>
</thead>
<tbody>
<tr>
<td>Commit and restart authority</td>
</tr>
<tr>
<td>Disk space allocation</td>
</tr>
<tr>
<td>Commit domains</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Cache</th>
</tr>
</thead>
<tbody>
<tr>
<td>Caching and performance optimization</td>
</tr>
<tr>
<td>Commit support (flush)</td>
</tr>
<tr>
<td>Frame allocation (to domains)</td>
</tr>
<tr>
<td>Optional disk shadowing</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Canonical Disk</th>
</tr>
</thead>
<tbody>
<tr>
<td>Physical disk access</td>
</tr>
</tbody>
</table>

*Also called client helix.

Figure 2. Abstraction layering.

More examples

Figure 1. The NIST/ECMA reference model.
More examples
More examples

**Figure 2.1**

*JMX Management Architecture.*
description of a system’s architecture

- usually informal prose plus box-and-line diagram
- lots of appeal to intuition
- rarely formal, little precision
unlike buildings and cars, software is hard to “see”

- architectures produced by different people looking at the same code
good architecture bestows:

- strength
  - durable
  - safe

- beauty
  - balanced proportions

- utility
  - fit planned uses

[Vitruvius, ~40 BCE]

- reliable
- fault-tolerant
- ... 
- usable
- intelligible
- maintainable
- ...
- desired features
- quality of service
the challenge

- develop systems “architecturally”
  - apply codified architectural design expertise
  - build systems by assembling existing parts
  - use standards for integration
  - assure that the system has the desired properties
  - reduce development and evolution costs

- turn Software Architecture into an engineering discipline
  - from ad hoc definition to codified principles
What is the nature of software construction?

[somewhat conflicting] quotes:

- **software engineering is, in fact, a form of engineering**
  
  David Parnas (modules, hiding, & OO)

- **programming is an art and a science**
  
  Donald Knuth (computational complexity, analysis of algorithms)
what is engineering?
definitions abound. They have in common:

*Creating cost-effective solutions* ...
... to practical problems ...
... by applying scientific knowledge ...
... building things ...
... in the service of mankind

engineering enables ordinary people
to do things that formerly required virtuosos
engineering entails making decisions under the constraints of limited time, knowledge, and resources
**what is engineering?**

- Before the 19th century, the kinds of systems that engineers studied and built had little technological or mechanical aspects.

- By 1920, AT&T had job titles for System Engineers, who built and maintained *The System*: their national network.

many angles into software engineering

- managing software production
  - who does what, when – lifecycle/process/risk management
  - methods, tools, skills

- best practices/tools of the trade
  - systematic testing/peer reviews
  - separation of concerns/aspects
  - rapid prototyping/configuration management/evolution
  - product lines/design patterns

- correlating structure and properties
  - software economics: size/complexity <-> cost
  - Software Architecture:
    - design styles & patterns <-> *-ility tradeoffs
**software architect:**
a recognized trade within SE

- human-computer interaction
- applications
- development/execution envs.
- middleware
- OS/storage management
- networking
- hardware

interfaces with:
- psychology, ergonomics, human factors

---

- HCI designer
- manager
- tester/QA
- developer
- methods/tools/process

---

CS
EE

SWE 443 - Software Architecture
what is software architecture?

Definitions abound. They have in common:

- the structure
- or structures of the system
- which comprise elements
- their externally-visible properties
- and the relationships among them
SA & programming address different issues

<table>
<thead>
<tr>
<th>Architecture</th>
<th>Programs</th>
</tr>
</thead>
<tbody>
<tr>
<td>interactions among parts</td>
<td>implementations of parts</td>
</tr>
<tr>
<td>system-wide views</td>
<td>local views</td>
</tr>
<tr>
<td>declarative</td>
<td>operational</td>
</tr>
<tr>
<td>more stable components</td>
<td>dynamic allocation</td>
</tr>
<tr>
<td>system-level performance</td>
<td>algorithmic performance</td>
</tr>
</tbody>
</table>

- software engineering is, in fact, a form of engineering
  David Parnas

- programming is an art and a science
  Donald Knuth
SA makes a difference

- design Reviews at ATT
  - Architecture Review Board created 1988
  - facilitates architectural reviews for projects

- results
  - reviewed over 350 projects from 1989-95
  - “a correct architecture has the largest single impact on cost and quality of the product” (Maranzano 1995)
  - “based on experience ... we found an average savings of at least one-half staff year per project reviewed, and substantially larger savings on several projects reviewed.”
  - estimated cumulative savings: approx 175 staff years
SA in the 2000’s

- incorporation of architectural notions into mainstream *design languages* (e.g., UML-2), and *tools* (e.g., Rose)

- *methods* based on architectural design and refinement (e.g., Model-Driven Architecture – MDA)

- some architecture *analysis tools*

- architectural *standards* for enterprise systems (e.g., CORBA, J2EE,...)
SA correlates design patterns & styles with quality attribute tradeoffs

- using formal/quantitative models of systems and environment