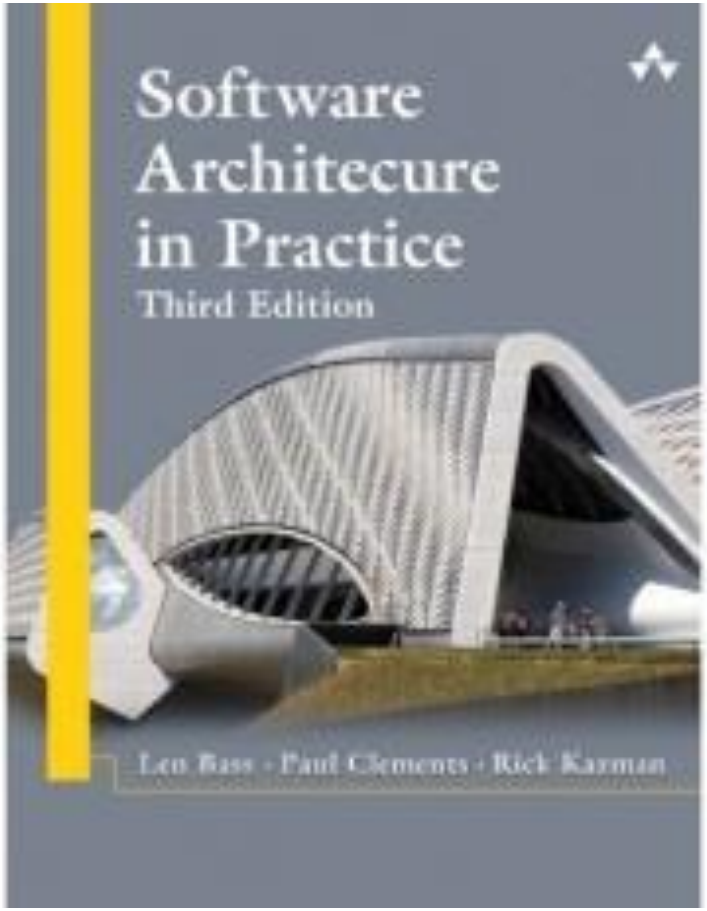
SE 4352
Software Architecture and Design

Fall 2018

Module 4

Chapters 2 & 4


Software
Architecure
in Practice
Third Edition
Len Bass · Paul Clements · Rick Kazman

Why Software Architecture Is Important

1. An architecture will inhibit or enable a system's driving quality attributes.
2. The decisions made in an architecture allow you to reason about and manage change as the system evolves.
3. The analysis of an architecture enables early prediction of a system's qualities.
4. A documented architecture enhances communication among stakeholders.
5. The architecture is a carrier of the earliest and hence most fundamental, hardest-to-change design decisions.
6. An architecture defines a set of constraints on subsequent implementation.
7. The architecture dictates the structure of an organization, or vice versa.
8. An architecture can provide the basis for evolutionary prototyping.
9. An architecture is the key artifact that allows the architect and project manager to reason about cost and schedule.
10. An architecture can be created as a transferable, reusable model that forms the heart of a product line.
11. Architecture-based development focuses attention on the assembly of components, rather than simply on their creation.
12. By restricting design alternatives, architecture channels the creativity of developers, reducing design and system complexity.
13. An architecture can be the foundation for training a new team member.

Inhibiting or Enabling a System's Quality Attributes

- Whether a system will be able to exhibit its desired (or required) quality attributes is substantially determined by its architecture.
- **This is the most important message of this course!**
  - **Performance:** You must manage the time-based behavior of elements, their use of shared resources, and the frequency and volume of inter-element communication.
  - **Modifiability:** Assign responsibilities to elements so that the majority of changes to the system will affect a small number of those elements.
  - **Security:** Manage and protect inter-element communication and control which elements are allowed to access which information; you may also need to introduce specialized elements (such as an authorization mechanism).
  - **Scalability:** Localize the use of resources to facilitate introduction of higher-capacity replacements, and you must avoid hardcoding in resource assumptions or limits.
  - **Incremental subset delivery**: Manage inter-component usage.
  - **Reusability:** Restrict inter-element coupling, so that when you extract an element, it does not come out with too many attachments to its current environment.

Reasoning About and Managing Change

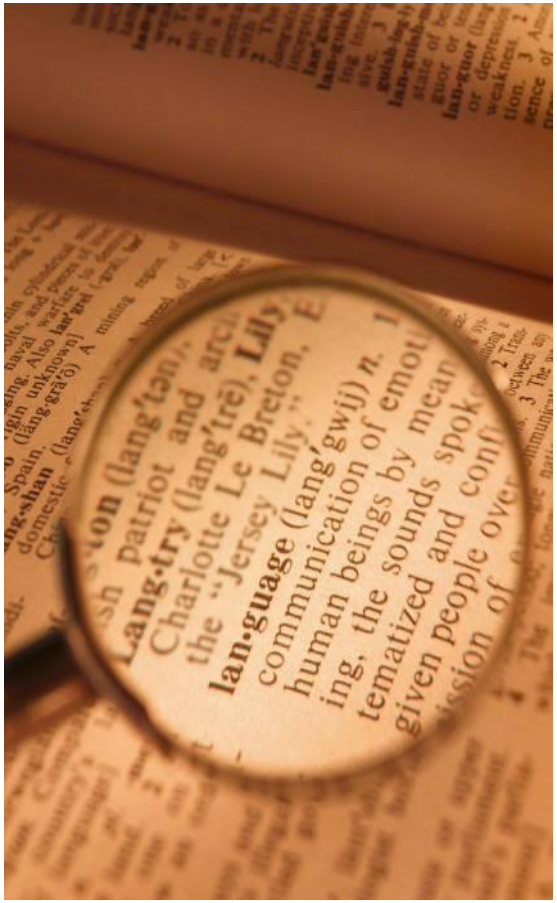
- About 80 percent of a typical software system's total cost occurs after initial deployment
  - accommodate new features
  - adapt to new environments,
  - fix bugs, and so forth.
- Every architecture partitions possible changes into three categories
  - A *local* change can be accomplished by modifying a single element.
  - A *nonlocal* change requires multiple element modifications but leaves the underlying architectural approach intact.
  - An *architectural* change affects the fundamental ways in which the elements interact with each other and will probably require changes all over the system.
- Obviously, local changes are the most desirable
- A good architecture is one in which the most common changes are local, and hence easy to make.

Predicting System Qualities

- If we know that certain kinds of architectural decisions lead to certain quality attributes in a system, we can make those decisions and rightly expect to be rewarded with the associated quality attributes.
- When we examine an architecture we can look to see if those decisions have been made, and confidently predict that the architecture will exhibit the associated qualities.
- The earlier you can find a problem in your design, the cheaper, easier, and less disruptive it will be to fix.

Enhancing Communication Among Stakeholders

- Architecture provides a common language in which different concerns can be expressed, negotiated, and resolved at a level that is intellectually manageable even for large, complex systems.



Earliest Design Decisions

- Architecture is more than just the early decisions
- But they are critical
- Imagine the nightmare of having to change any of these decisions.

Defining Constraints on an Implementation

- e.g. Software Developers may not be aware of the architectural tradeoffs—the architecture (or architect) simply constrains them in such a way as to meet the tradeoffs.

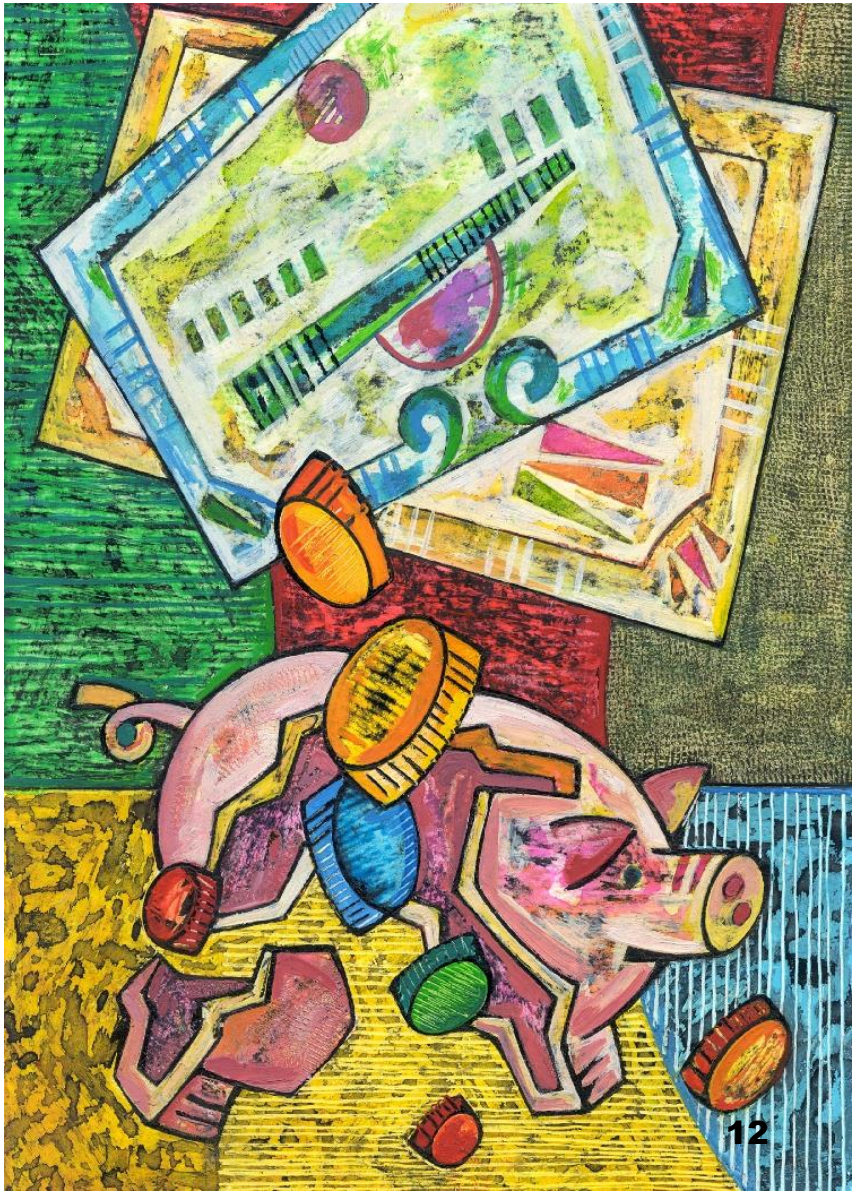
Influencing the Organizational Structure

- Architecture prescribes the structure of the system being developed.
- That structure becomes engraved in the structure of the development project (and sometimes the structure of the entire organization).

Enabling Evolutionary Prototyping

Improving Cost and Schedule Estimates

- One of the duties of an architect is to help the project manager create cost and schedule estimates early in the project life cycle.



Transferable, Reusable Model

- Reuse of architectures provides tremendous leverage for systems with similar requirements.
  - Not only can code be reused, but so can the requirements that led to the architecture in the first place, as well as the experience and infrastructure gained in building the reused architecture.
  - When architectural decisions can be reused across multiple systems, all of the early-decision consequences are also transferred.

Using Independently Developed Components

- Architecture-based development often focuses on components that are likely to have been developed separately, even independently, from each other.
- The architecture defines the elements that can be incorporated into the system.
- Commercial off-the-shelf components, open source software, publicly available apps, and networked services are example of interchangeable software components.

Restricting Design Vocabulary

- As useful architectural patterns are collected, we see the benefit in voluntarily restricting ourselves to a relatively small number of choices of elements and their interactions.
  - We minimize the design complexity of the system we are building.
  - Enhanced reuse
  - More regular and simpler designs that are more easily understood and communicated
  - More capable analysis
  - Shorter selection time
  - Greater interoperability.
- Architectural patterns guide the architect and focus the architect on the quality attributes of interest in large part by restricting the vocabulary of design.
  - Properties of software design follow from the choice of an architectural pattern.

Basis for Training

- The architecture can serve as the first introduction to the system for new project members.

So…13 great reasons!!!

Why Software Architecture Is Important

1. An architecture will inhibit or enable a system's driving quality attributes.
2. The decisions made in an architecture allow you to reason about and manage change as the system evolves.
3. The analysis of an architecture enables early prediction of a system's qualities.
4. A documented architecture enhances communication among stakeholders.
5. The architecture is a carrier of the earliest and hence most fundamental, hardest-to-change design decisions.
6. An architecture defines a set of constraints on subsequent implementation.
7. The architecture dictates the structure of an organization, or vice versa.
8. An architecture can provide the basis for evolutionary prototyping.
9. An architecture is the key artifact that allows the architect and project manager to reason about cost and schedule.
10. An architecture can be created as a transferable, reusable model that forms the heart of a product line.
11. Architecture-based development focuses attention on the assembly of components, rather than simply on their creation.
12. By restricting design alternatives, architecture channels the creativity of developers, reducing design and system complexity.
13. An architecture can be the foundation for training a new team member.