SATURN 2017
Visual Architecting
Ruth Malan
Talk Outline
1. Introduction

“—Mr. Beck, what is software architecture?” asked a participant at an OOPSLA workshop in Vancouver in the fall of 1992. “—Software architecture?” replied Kent, now famous for being the father of XP (eXtreme Programming, not the O.S.), “well, it is what software architects do.” (Chuckles in the audience.) “—So then, what is an architect?” “—Hm, ‘software architect’ it’s a new pompos title that programmers demand to have on their business cards to justify their sumptuous emoluments.”

In the following four years I was going to wear a rather large team of software architects and I often ask myself that very question: “what do architects really do?” and was also asked by my management and our customers. Since then I have seen many architecture teams in many countries, in companies of all sizes and various domains, and I have witnessed a wide range of good, not-so-good, and really bad answers to this question.
Who Needs an Architect?

Martin Fowler

We are all familiar with the idea of an architect: someone who creates and plans the design of buildings, bridges, and other structures. But do we really need an architect in the world of software development? Let’s explore this question.

Martin Fowler

The architect’s role

So if architecture is the important stuff, then the architect is the person (or people) who worry about the important stuff. And here we get to the essence of the difference between Martin’s refactored species of architecture and the style that Dave Larabee exemplifies.

Architecture for architects: the view of the architect is one of powerful, influential architects who can make or break a project. They are the ones who set the direction and make decisions that have far-reaching consequences.

Architecture for developers: the view of the architect is more modest. They are the ones who provide guidance and direction, but they are not the ones who make the final decisions.

So, this leads us to the important question: “Tell us what is important!” This is because the architect’s role is to set the direction and ensure that the project is on the right track.

What makes a component significant?

It is significant because the expert developers say so.

In a successful software project, the expert developers are the ones who know what to keep and what to throw away. They are the ones who have the experience and expertise to make those decisions.

To describe their architecture, they might say things like:

“Architecture is important because it

What needs to get some things right early in the project? The answer, obviously, is because they prevent these things as hard to change. So you might end up defining architecture things that people prefer as hard to change.”

It’s commonly believed that if you are building an enterprise application, you must get the database schema right early on because it’s hard to change the database schema—particularly once you have gone live. On one of our projects, the database administrator, Prakash Bajaj, devised a system that let us change the database schema (and migrate the data easily with http://martin/fowler.com/wf/sdlv.html).
"Architecture represents the significant design decisions that shape a system"

— Grady Booch, blog post, March 2, 2006
Decision Template

**Title:** short noun phrase

**Context:** describe the forces at play, probably in tension

**Decision:** describe our response to these forces

**Status:** proposed, accepted, deprecated or superseded

**Consequences:** describe the resulting context, after applying the decision

— Michael Nygard, Documenting Architecture Decisions, Nov 2011
Architecturally Significant Decisions!

"What decisions does the architect make?"
Architecturally significant ones.
"What is architecturally significant?"
The architect decides!
"Architecture represents the significant design decisions that shape a system, where significant is measured by cost of change."

— Grady Booch, blog post, March 2, 2006
“If you think good architecture is expensive, try bad architecture”
– Brian Foote
Big Ball of Mud Architecture

“Big Ball of Mud”, Brian Foote and Joseph Yoder
http://www.laputan.org/mud/
Big Ball of Mud Architecture

“You reach for the banana, and get the entire gorilla”
– Michael Stahl
Actually, it looks more like this
Modular Structure(s): ↓ Cost of Change!

- Isolate impact of change
- Isolate arenas of uncertainty and experiment
- Increase reversibility, replaceability,
- Increase responsiveness/adaptability
- Reduce complexity
  - Divide and conquer
  - we have to keep it crisp, disentangled, and simple if we refuse to be crushed by the complexities of our own making...” – Dijkstra
Who Needs an Architect?

Martin Fowler

It striking how out of order a web page can look, like a colleague Dave rice in a particularly grumpy mood. My brief question about a violent statement, “We should invert our priorities to have an architect on our resume.” At first blush, this was an odd turn of phrase, but because we usually introduce Dave as one of our leading architects.

The reason for his title schizophrenia is the fact that, even by our industry standards, “architect” and “architecting” are many different things, helping explain the role of the term “software architect” in particular with the now prominent image of the role of the engineer. The respect you thought of as different, and it is a critical decision.

What is an architect?

When I was formulating the title for my Patterns of Enterprise Application Architecture (Addison-Wesley, 2002), everyone who reviewed it agreed that “architect” belonged in the title. Why we felt uncomfortable defining the word. Because it was my book, I felt compelled to take a stab at defining it.

My best move was to avoid it, and just let my cynicism hang out there. In a sense, I define architecture as what we want to call a web site does, to be simple enough to be understood, and yet complex enough to be challenging. Yes, you can imagine a similar phenomenon for architectural. However, as so often occurs, I found that I had not been thinking about the concept correctly.

The architect is not a person who has a particular architectural sense, not a person who has a particular architectural sense, but an architect who has a particular architectural sense. If architecture is a single mind, then the person who makes the architect does not see that the single mind is needed to ensure a system’s conceptual integrity, and perhaps because the architect doesn’t think that the team members are sufficiently skilled to make their level so that they can take on more complex issues. Improving the developer team’s ability gives an architect much greater leverage than being able to decide what the architect means to do, even if the architect might be more focused on developing the technical aspects of the project. Architectures are inherently difficult to define, and architect means to ensure that the team members are sufficiently skilled to make their level so that they can take on more complex issues. Improving the development team’s ability gives an architect much greater leverage than being able to decide what the architect means to do, even if the architect might be more focused on developing the technical aspects of the project. Architectures are inherently difficult to define, and architect means to ensure that the team members are sufficiently skilled to make their level so that they can take on more complex issues. Improving the development team’s ability gives an architect much greater leverage than being able to decide what the architect means to do, even if the architect might be more focused on developing the technical aspects of the project. Architectures are inherently difficult to define, and architect means to ensure that the team members are sufficiently skilled to make their level so that they can take on more complex issues. Improving the development team’s ability gives an architect much greater leverage than being able to decide what the architect means to do, even if the architect might be more focused on developing the technical aspects of the project. Architectures are inherently difficult to define, and architect means to ensure that the team members are sufficiently skilled to make their level so that they can take on more complex issues.

What makes a component significant? It is significant because the expert developers say so.

I was reviewing the data I had collected, and I had a hunch that this new study was important. Remember Johnson’s second definition? “Architecture is the decisions that you make at your earliest stage in a project.” Why do people feel the need to get some things right in the earliest stages of a project? The answer, of course, is because they practice those things as hard as they can. So you might end up defining the architecture as “things that people perceive as hard to change.” It’s commonly believed that if you are using an enterprise application, you must get the database schema right early on because it’s hard to change. The database schema—particularly once you have gone live. On one of our projects, the database administrator, Pramod Kandula, delayed a system that let us change the database schema (and migrate the data) easily (http://www.theservers.com/ted/2003).
Foundations for the Study of Software Architecture

Dewayne E. Perry
AT&T Bell Laboratories
600 Mountain Avenue
Murray Hill, New Jersey 07974
dep@research.att.com

ACM SIGSOFT SOFTWARE ENGINEERING NOTES vol 17 no 4 Oct 1992 Page 40

Abstract

The purpose of this paper is to build the foundation for software architecture. We first develop an intuition for software architecture by appealing to several well-established architectural disciplines. On the basis of this intuition, we present a model of software architecture that consists of three components: elements, form, and rationale. Elements are either processing, data, or connecting entities. Form is defined in terms of properties, and the relationships among the elements that is, the constraints on the elements. The rationale provides the underlying basis for the architecture in terms of the system constraints, which must often derive from the system requirements. We discuss the components of the model in terms of both architectural and implementation styles and present an extended example to illustrate some important architecture and style considerations. We conclude by presenting some of the benefits of our approach to software architecture, summarizing our contributions, and relating our approach to earlier work.

1 Introduction

Software design received a great deal of attention by researchers in the 1970s. This research arose in response to the unique problems of developing large-scale software systems recognized in the 1960s [4]. The premise of the research was that design is an activity separate from implementation, requiring special notations, techniques, and tools [3, 9, 12]. The results of this software design research have now begun to make important contributions to software architecture and software development.

Software architecture is concerned with the selection of architectural elements, their interactions, and the constraints on those elements and their interactions.
Software architecture refers to the high level structures of a software system [...] Each structure comprises software elements, relations among them, and properties of both elements and relations.

— wikipedia/Clements et al
“Everything that needs to be said has already been said. But since no one was listening, everything must be said again.

― André Gide
"All architecture is design but not all design is architecture. Architecture represents the significant design decisions that shape a system, where significant is measured by cost of change."

-- Grady Booch, blog post, March 2, 2006
Design?

"Everyone designs who devises courses of action aimed at changing existing situations into preferred ones."

-- Herbert Simon
We design to get more what we want!
So about those elements and relations and those (much maligned) box and line diagrams...

How do we design (Better) Boxes?
Finding the (Natural) Shape

“I go along with the natural makeup”...

“when I come to the tricky parts, I slow down”

— Chuang Tzu: “The Dexterous Butcher”
“Design things to make their performance as insensitive to the unknown or uncontrollable external influence as practical.” — Eb Rechtin

ABATIS, n. [1.] Rubbish in front of a fort, to prevent the rubbish outside from molesting the rubbish inside.

— Ambrose Bierce, Devil’s Dictionary

Image: Engineering and the Mind’s Eye
Hexagonal Architecture

Separation of Concerns

Ports and adapters

Image: alistair.cockburn.us/Hexagonal+architecture
Abstractions

As programmers we deal with abstractions all the time and we have to invent them in order to solve our problems

— Michael Feathers
Bounded Contexts in DDD

Separation of Concerns

Image: martinfowler.com/bliki/BoundedContext.html
Components and Responsibilities
Factor and Refactor

Does this component have a cohesive identity or purpose — a single responsibility at the level of abstraction of the abstraction?

“The responsibility of architecture is the architecture of responsibility.”
— Jan van Til/Tom Graves
The architect’s SCARS:

• Separation of Concerns
• crisp and resilient Abstractions
• balanced distribution of Responsibilities
• strive to Simplify — Grady Booch
“disorder is easier and more permanent than order, which is difficult and temporary”

— Jeremy Campbell
Boundaries: Conway’s Law

“Any organization that designs a system (defined broadly) will produce a design whose structure is a copy of the organization's communication structure.”

—Melvyn Conway (in 1968!)

Keep Concerns Separate
"All architecture is design but not all design is architecture. Architecture represents the significant design decisions that shape a system, where significant is measured by cost of change."

-- Grady Booch, blog post, March 2, 2006
“The defining properties of any system, are properties of the whole, which none of the parts have. If you take the system apart, it loses its essential properties”
— Russell Ackoff
Parts Flying in Formation

"trying to be an airplane"

— Wim Roelandts
Minimalist Architecture

Decisions that *must* be made from a system perspective

- system outcomes
- across boundaries
Structure and Behavior

Posit structure

Explore behavior

Revise structure
Structure and Behavior

Posit structure

Explore behavior

Revise structure

How will this work?

What is it made (up) of?

How does this contribute to/inhibit desired properties?
Remember! Responsibilities!

- Patterns
- Metaphors
- Experience
- Existing stuff

Posit responsibilities
Refine as explore behavior and different views
Visual models

• sketch prototype
• try alternatives on the cheap
• “mob modeling”
• “test drive”

“at least they’re looking at it”
“Architects must have self-repairing egos”
— Dana Bredemeyer
“A change of perspective is worth 80 IQ points”
— Alan Kay
“You don't understand something until you understand it more than one way”
— Marvin Minsky

Image:
en.wikipedia.org/wiki/Marvin_Minsky#/media/File:Marvin_Minsky_at_OLPCb.jpg
Rule of Three

“If you haven’t thought of three possibilities, you haven’t thought enough.”
– Jerry Weinberg
ARCHITECTURE

STRUCTURE

What

Conceptual

Logical

Execution
Design across

“Design is not just what it looks like and feels like. Design is how it works.”
— Steve Jobs
ARCHITECTURE

BEHAVIOR

How

STRUCTURE

What

Conceptual elements and relationships

Logical interfaces
“Design quality is not a property of the code. It's a joint property of the code and the context in which it exists.” – Sarah Mei
Architecturally significant? The demands (forces, properties, ...) on the system that are challenging, push the limits, require design attention
LMAX Disruptor Mechanism

Challenges:
A trading platform needs very low latency - trades have to be processed quickly because the market is moving rapidly. A retail platform adds complexity because it has to do this for lots of people.

Source: Martin Fowler
http://martinfowler.com/articles/lmax.html
“The Federalist Papers are arguments that support different parts of the design of the Constitution.”

– Alan Kay, 1995
Federalist 51

addresses

• separation of powers
• checks and balances
“PROGRAMMING AS THEORY BUILDING”

Introduction
The present discussion is a contribution to the understanding of what programming is. It suggests that programming properly should be regarded as an activity by which the programmers form or achieve a certain kind of insight, a theory, of the matters at hand. This suggestion is in contrast to what appears to be a more common notion, that programming should be regarded as a production of a program and certain other texts.

Some of the background of the views presented here is to be found in certain observations of what actually happens to programs and the teams of programmers dealing with them, particularly in situations arising from unexpected and perhaps erroneous program executions or reactions, and on the occasion of modifications of programs. The difficulty of accommodating such observations in a
Take note(s)!

Develop and share theory

• of operation (interactions, resolution of forces, outcomes)
• of relation of structure to function/properties

Leonardo da Vinci’s notebooks
Static structure
from Pollock
to Kandinsky?
Your Code as a Crime Scene

http://www.adamtornhill.com
Your Code as a Crime Scene

2.6% of the code, 15.5% effort
ASP.NET MVC 200 kLoC

2.5% of the code, 11.0% effort
Mono 6.7 MLoC

2.4% of the code, 13.3% effort
Docker 500 kLoC

http://www.adamtornhill.com
<table>
<thead>
<tr>
<th>System Design Intention (what should be)</th>
<th>System Design Reflection (what is)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CON Text</strong></td>
<td><strong>Organizational</strong></td>
</tr>
<tr>
<td>Why</td>
<td>Who</td>
</tr>
<tr>
<td>What (user view)</td>
<td></td>
</tr>
<tr>
<td>How well</td>
<td></td>
</tr>
<tr>
<td><strong>Context</strong></td>
<td><strong>System</strong></td>
</tr>
<tr>
<td><strong>Capabilities</strong></td>
<td><strong>Static</strong></td>
</tr>
<tr>
<td>Functional</td>
<td>What</td>
</tr>
<tr>
<td>Qualities</td>
<td>How</td>
</tr>
<tr>
<td><strong>System Architecture</strong></td>
<td><strong>Dynamic</strong></td>
</tr>
<tr>
<td>Behavioral</td>
<td>What</td>
</tr>
<tr>
<td>Structural</td>
<td>How</td>
</tr>
<tr>
<td><strong>Code</strong></td>
<td><strong>Organization</strong></td>
</tr>
<tr>
<td>Why</td>
<td>Who</td>
</tr>
</tbody>
</table>
"Always design a thing by considering it in its next larger context"

—Eliel Saarinen
"Architecture represents the significant design decisions that shape a system, where significant is measured by cost of change."

— Grady Booch, blog post, March 2, 2006
Architecturally Significant — also:

What is make or break?

What impacts how we compete?

Structurally significant
• Organizing structure
• Architecturally significant mechanisms
• Structural integrity and sustainability

Strategically significant
• game shapers and game changers

"I wasn't the one pushing things in the wrong direction, but I should have been the one to stop it." — Chad Fowler
Just enough

• Not “big design” that we just spread out over time
• using judgment, assessing what’s architecturally significant
  • where do we need “cognitive assist”
    • to “see,” to draw out (assumptions, relationships, …)
    • try out alternatives cheaply to decide where to run code experiments
to test out ideas
• where do we need to work together
  • involve others, convey and persuade, inform and coach by showing
    how we resolve interplay of forces and context to create solutions, …
SATURN 2017

Visual Architecting
@RuthMalan
Bredemeyer Consulting