DESIGN VISUALIZATION: SMOKE AND MIRRORS

A rethinking of software architecture, its place in software design, and how we approach it

Going to bat for wisdom, and

Putting visualization back in the tool belt

by:
Ruth Malan
Bredemeyer Consulting

October 14, 2015
Introduction

Introduction
Theme of talk
Outline of talk
These are the slides and notes from my Design Visualization: Smoke and Mirrors talk at the Software Architect Conference in London in October, 2015.

This is the part where I tell you "This flight is going to Indiana by way of South Africa. If you’re on the wrong flight, this would be a good time to attend one of the awesome talks down the hall."

My name is Ruth Malan -- I don’t know how to pronounce it either, so any way you say it is fine.

Well, as long as you’re staying, let me tell you that Dana Bredemeyer says “There is a silver bullet in software engineering, and it is relationships of goodwill and a commitment to objectivity.” There’s a catch though. Those relationships take effort and attention to build and to sustain and grow.

So here I am, relying on you to extend a little goodwill on credit to me, and as we proceed, I hope I’ll earn a little more, as we talk about some ways we can work on objectivity.
Imagine you are looking at M.C. Escher’s *Day and Night.*

The M.C. Escher Company holds and protects their copyrights to M.C. Escher’s work quite tenaciously. Not being able to reproduce *Day and Night* for you here, presents a little opportunity to demonstrate a point at the heart of this talk.

Do you see it in your mind’s eye? It is the one where there is a town in the light, and a mirror image in the dark of night. A river runs by, in the light, and the night. There are fields that become abstractions that become duck (or geese), flying <gestures a mesh with interwoven fingers of both hands> from the light, others in mirror image out of the night.

When you called it to mind, perhaps at first you were thinking of another Escher, or perhaps it struck you differently and you remembered different features.

And that’s the point, really. We don’t know if what I call to mind, and see in my mind’s eye is quite what you see, even if the original was the same image. Our software systems are so large, we can never see them in their entirety. Our mental models are central. But idiosyncratic and fallible.

*It is worth noticing how different our mental models (recollection of “fact” and making of meaning) can be, even when we are referring to a picture held in memory. Our software systems are so large, we can never see them in their entirety. Our mental models are central. But idiosyncratic and fallible.*
That is where we are headed -- to talk about design as intention, and reflection of design as realized, and how we express and envision our designs. But first I would like to revisit how we think of software architecture, and the role visual expression plays in design.
Software Architecture

Intention and Reflection
Software Architecture
Decisions
Design
Significance

Technical Wisdom
Structurally and Strategically Significant
Minimalist
Intentional and Emergent
**Intention and Reflection.**

**Or Wizardry.**

I playfully subtitled the talk “Smoke and Mirrors” -- for it can seem like we practice magic and sleights of hand when we don’t expressly communicate our design intention, or reflect on the design as realized, in order to iterate on and improve our understanding of what we have, and intend.
The Devil’s Dictionary on Intention and Reflection

Here we have the definitions of intention and reflection from Ambrose Bierce’s *Devil’s Dictionary*. If you’re not familiar with the Devil’s Dictionary, it gives definitions a cynical side-eye, helping us understand something better by looking at the (negative) space left by contact of the concept with reality, and what that defines. Intention, then, is what we get, despite our intention -- an involuntary act. It is a big wink at our bias and foible and self delusions and intellectual aggrandizements.

As for reflection: "An action of the mind whereby we obtain a clearer view of our relation to the things of yesterday and are able to avoid the perils that we shall not again encounter." That so well captures the weltanschauung of our day and field -- we forget, of course, that it was Heraclitus who said "Change is the only constant."

Now, when it comes to definitions I’m on the same page as Richard Feynman when he said: ‘We can’t define anything precisely. If we attempt to, we get into that paralysis of thought that comes to philosophers… one saying to the other: “you don’t know what you are talking about!”’. The second one says: "what do you mean by talking? What do you mean by you? What do you mean by know?"

Undeterred, we’re going to take a look at the definition of software architecture.
Software Architecture

This is the definition from the touchstone of our era, *wikipedia*.

When I see a definition, especially in a context like this, I tend to see it as words words words... words words. So I helped a little to lead attention from software architecture to system.

What do we mean by system?

---

**Software architecture** refers to the high level structures of a software system, the discipline of creating such structures, and the documentation of these structures. It is the set of structures needed to reason about the software system. Each structure comprises software elements, relations among them, and properties of both elements and relations.

-- *wikipedia/Clements et al*
System

Dana Meadows characterized a system as "an interconnected set of elements that is coherently organized in a way that achieves something."

"A system is an interconnected set of elements that is coherently organized in a way that achieves something"
-- Donella Meadows
Software Architecture

Returning to Paul Clements and colleagues definition on wikipedia, we note "high level structures" -- comprising elements and relations, along with properties.

High level?
Software Architecture – Decisions!

Grady Booch memorably observed "All architecture is design, but not all design is architecture. Architecture represents the significant design decisions"

Decisions?!
Decisions!

We know what to do with decisions! We’ll name them, describe them, identify what problem the decision addresses and the forces we’re weighing to harmonize, resolve and balance. We’ll keep track of assumptions we’re making, explicitly identifying what in the context we presume to be stable and what to watch.

We’ll connect the dots to the business rationale and technical goals we’re trying to achieve, or the hard- or (l)earned scars of experience we’re seeking to avoid this time round. We’ll outline alternatives we considered but ruled out, so we don’t have to revisit those arguments again and again. And we’ll note implications so we -- our teams and those we collaborate with -- can be prepared. Then we’ll consider what further requirements are implied by, or derive from this decision we’re making and documenting. And identify related decisions we need to make next...
Which decisions?

Which decisions? Design decisions.

Design!
Design?

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

-- Herbert Simon

Herbert Simon, according to Jabe Bloom, is "ground zero in design discourse." And this is quintessential Herb: "Everyone designs who devises courses of action aimed at changing existing situations into preferred ones."
Design?

"The engineer, and more generally the designer, is concerned with how things ought to be - how they ought to be in order to attain goals, and to function."

-- Herbert Simon

More Herb on Design

A variation on the theme: "The engineer, and more generally the designer, is concerned with how things ought to be - how they ought to be in order to attain goals, and to function."
Design?

We design to get more what we* want.

* For some value of we (tbd)

And that is just one of those so precious moments when the world is full of simple wonder again -- we design to get more what we want.
Returning to the eminent Mr. Booch:
"Architecture represents the significant design decisions"

Significant!

"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
Architecturally Significant?

"What decisions does the architect make?"
Architecturally significant ones.
"What is architecturally significant?"
The architect decides!

Architecturally Significant?

"What decisions does the software architect make?"
The architecturally significant ones.
"What is architecturally significant?"
The architect decides!
The architect decides. That sounds like a tautology, but it really is the crux of the matter.
Is this the part where we get to talk about Technical Wisdom?!

[insert obligatory Yoda image]
Is this the part where we get to talk about *Technical Wisdom*?! 

No, this is the part where we get to say “Awww”

Just kidding.
Architecturally Significant?

"the architecture of a system is the naming of the most significant design decisions that shape a system, where we measure significant by cost of change and by impact upon use."

– @Grady_Booch
What decisions are architecturally significant?

Ok, so if architecture is a set of decisions, but not all decisions, and we're asking which belong in the architecture, we're looking for those that shape, that

- give form to the system
- set direction
- constrain
- bring integrity and consistency

And those that have the highest cost of change. Decisions which would incur substantial

- resources and time
- operational downtime
- deferral of value
- emotions and politics
to change or revert and rework, are architecturally significant.

Also, if a decision meaningfully reduces the cost of making changes, enabling our business to be more fleet, adapting and extending its services or product set as the market shifts, it is architecturally significant.
Strategically and Structurally Significant

Those are important insights, but I would like to add: architecture decisions are *structurally significant*. They deal with the organizing structure of the system and the design of architecturally significant mechanisms to yield desired system outcomes, including system properties, while addressing inherent challenges. Structurally significant. You know, make or break.

And *strategically significant*. "Software is eating the world" (Marc Andreessen). Software enables. And more, across industries, software is increasingly a source, if not the source, of differentiation. It is game shaping and game changing.

**Architecture decisions are those that are:**
- Make or break!
- Game shapers and game changers!
Creating Ground Under the Feet

Software systems are in place -- for 3 months, 3 years, 10 years. 20. Because they enable something our business depends on. But, increasingly, and more as systems age and the architecture erodes under the weight of accommodations and agglomerations, these systems constrain the business, impede agility and adaptability and responsiveness to an ever shifting context.

And architecture, the critical decisions that hold the system up and tie it down, in turn enables and constrains the code (and those who write it).

Architecture decisions create the context for further decisions, reducing -- cleaving -- the decision space. This is good. It reduces the overload of overwhelming ambiguity and uncertainty, creating "ground under the feet" (Dana Bredemeyer) that we can move forward on. Critical decisions take time to make attentively and can be fraught with downstream consequences if made inattentively and without foresight. And this is bad, if it takes decisions away, reduces empowerment or degrees of freedom and motivation, where it matters. So we seek to keep our architecture decisions, in Dana Bredemeyer’s urging, to a minimal set.
To Do Big Things, When We Need To

To reach a helpful notion of what, then, we consider architecturally significant, Dana points to Daniel Day-Lew--er(r) Abraham Lincoln:

"The legitimate object of government is to do for a community of people whatever they need to have done, but cannot do at all, or cannot so well do, for themselves — in their separate, and individual capacities."

— Abraham Lincoln
We Architect Across

Architecture decisions are those that impact system outcomes. That is, outcomes that can’t be ascribed to locally-scoped parts of the system. Intentional architecture decisions are those, from a standpoint of experience, we deem must be made from a system perspective, to get more what we want from the system. More than we would get, if we left the decision to be made at a more narrow scope with only local information about the forces that impinge upon, and outcomes that are contingent on, the decision.

The Arc of Architecture

That is, architecture decisions are those that need to be made across boundaries - - the system boundary, and boundaries within the system, in order to achieve desired system outcomes -- to meet system goals with more the system properties we want. Properties like usability or performance that the user cares about. Properties like resilience and scalability that the ops team cares about.. Properties like understandability and adaptability that the dev team cares about (or will, as these properties are compromised). Properties that emerge from interactions and relations among elements, rather than localized concerns.

The architect has to take a broader perspective -- across the system, across stakeholders, and a longer time horizon. But we do just enough, to achieve concert and coherence, structural and design integrity, to get more the outcomes we seek.
Some part intention,
Some part emergence

Of course, despite our best intentions, some implicit decisions will prove to be architecturally significant.

The point is, do we want to leave matters of system integrity and strategic import to accident, or do we want to bring what we can to bear, to get more what we want?
What's next?

Visual Design

- Sketching in Engineering: Leonardo da Vinci
- Visualization in Other Fields
- What's Missing from Code
- Why We Model
- A Change in Perspective
- Mechanisms
- Wisdom and Fallibility
- Extraordinary Moment Principle
Part II: Visual Design

Coming soon

Thanks for your encouragement – it makes a difference!
Thanks

Thanks to Dana Bredemeyer for endless generosity in sharing ideas and insight, as well as encouragement and support.

Thanks also to Duarte for the SlideDocs idea, and the template that I started with. See http://www.duarte.com/slidedocs/

About me:

Ruth Malan
Bredemeyer Consulting
Web: bredemeyer.com

Twitter: @RuthMalan
Web: ruthmalan.com
Journal: ruthmalan.com/Journal.htm