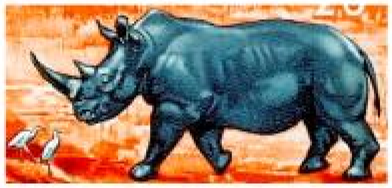
What is Software Architecture?



George Fairbanks

28 December2012

Rhino Research
Software Architecture Consulting and Training
http://RhinoResearch.com

What is

Software Architecture

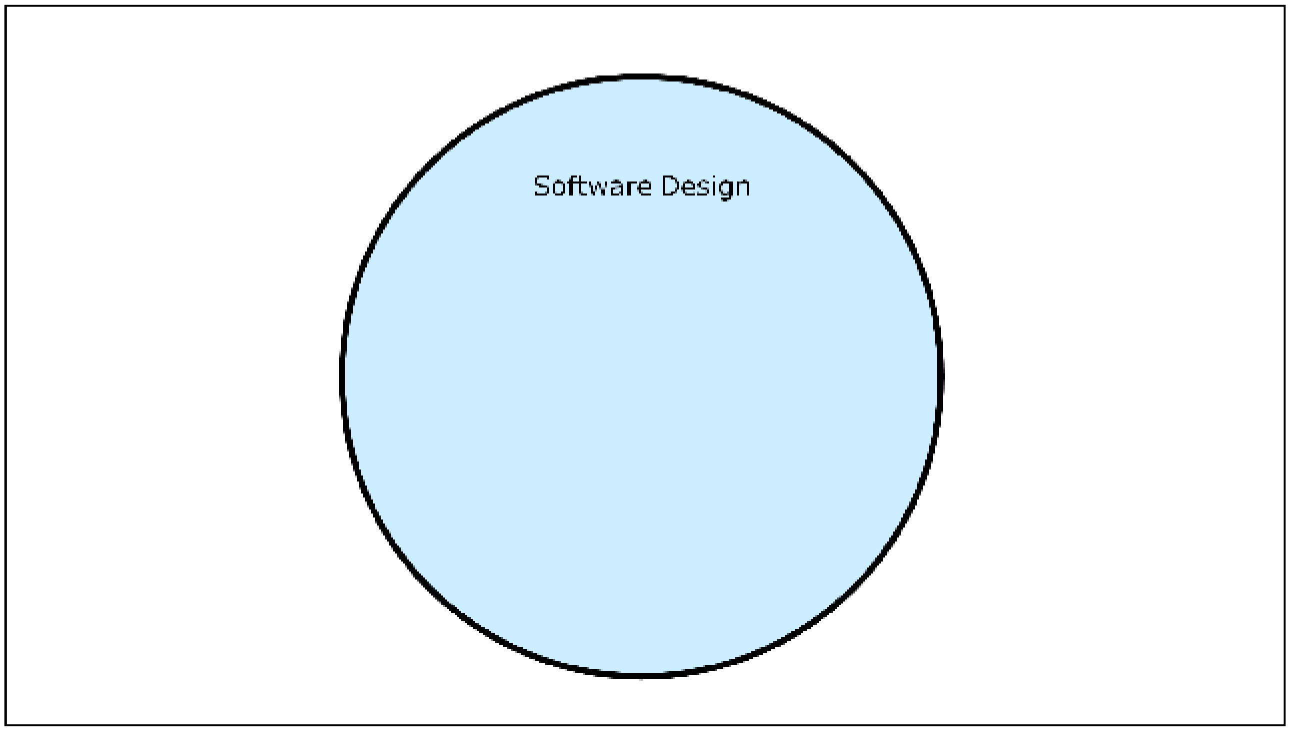
and how does it relate to

Software Design

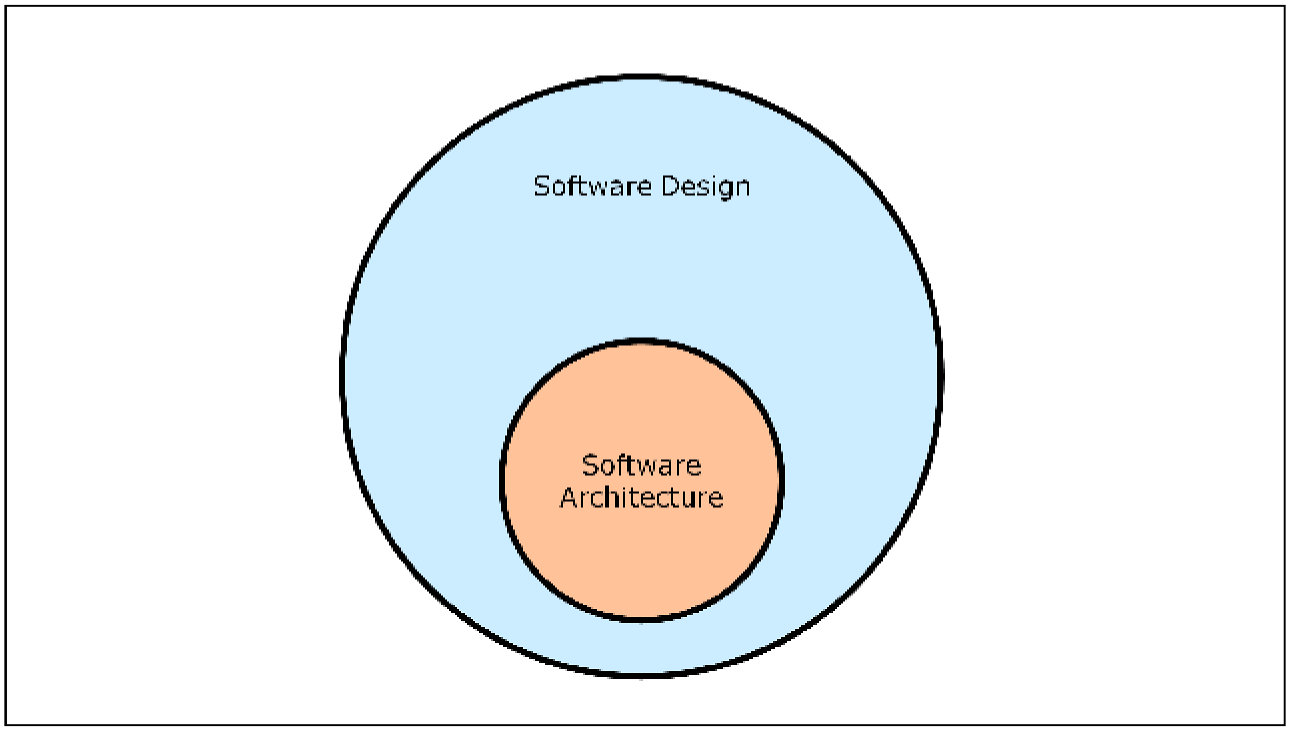
?

What is software architecture and how does it relate to software design?

This is a surprisingly difficult question and you'll get different answers from different people.



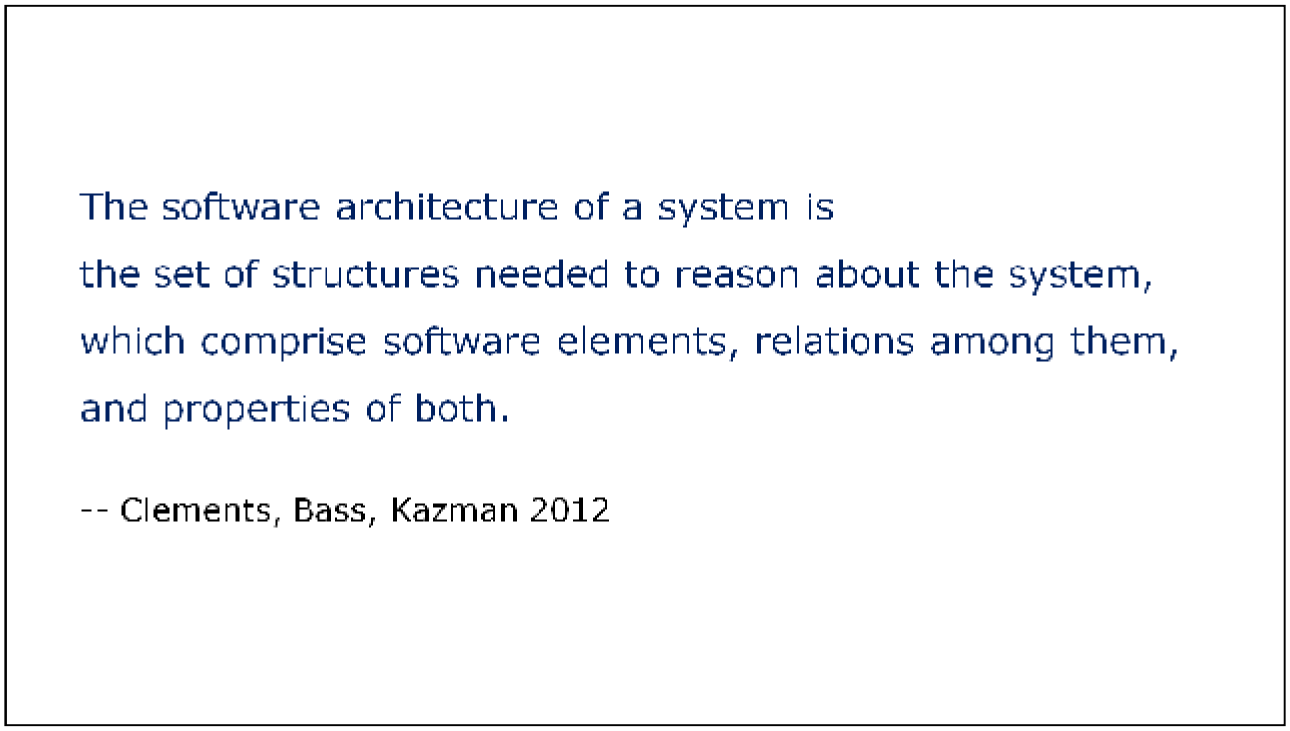
Everybody agrees on this part:



Software architecture is a kind of software design.

Architecture deals with the macroscopic or “big picture” aspects of a system.

What people don't agree about is which details are architectural.



The software architecture of a system is
the set of structures needed to reason about the system,
which comprise software elements, relations among them,
and properties of both.

-- Clements, Bass, Kazman 2012

Here's a definition from the Software Engineering Institute.

This is a great definition, but it won't really make sense until we provide some background.

To get that background, let’s look at some parallels between software architecture and other kinds of engineering.

My brother is an engineer who builds schools and skyscrapers in Chicago.

He tells me that sometimes the windows on the building are details specified by the architect.

Other times they are not.

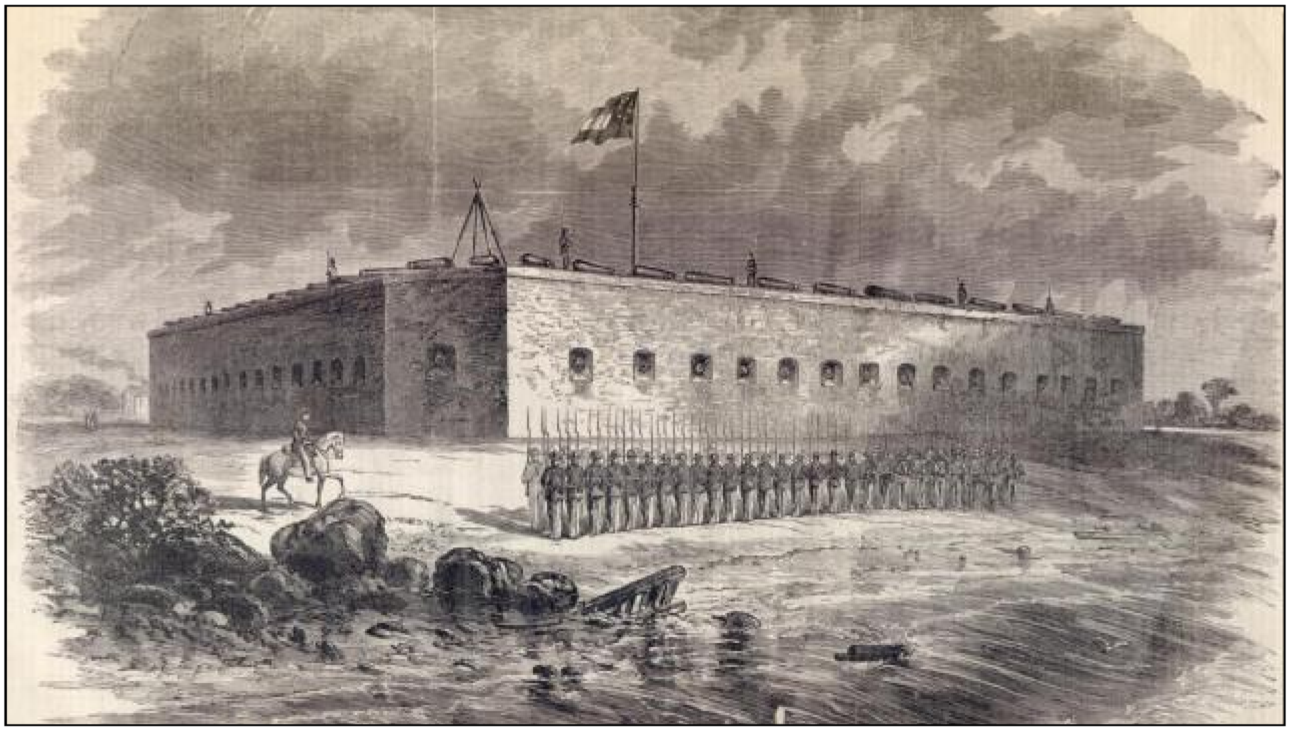
So, how do they decide which details are architectural?

Architects choose a few key details of the building to pay attention to.

On one building, perhaps an important detail is resistance to heavy rain.

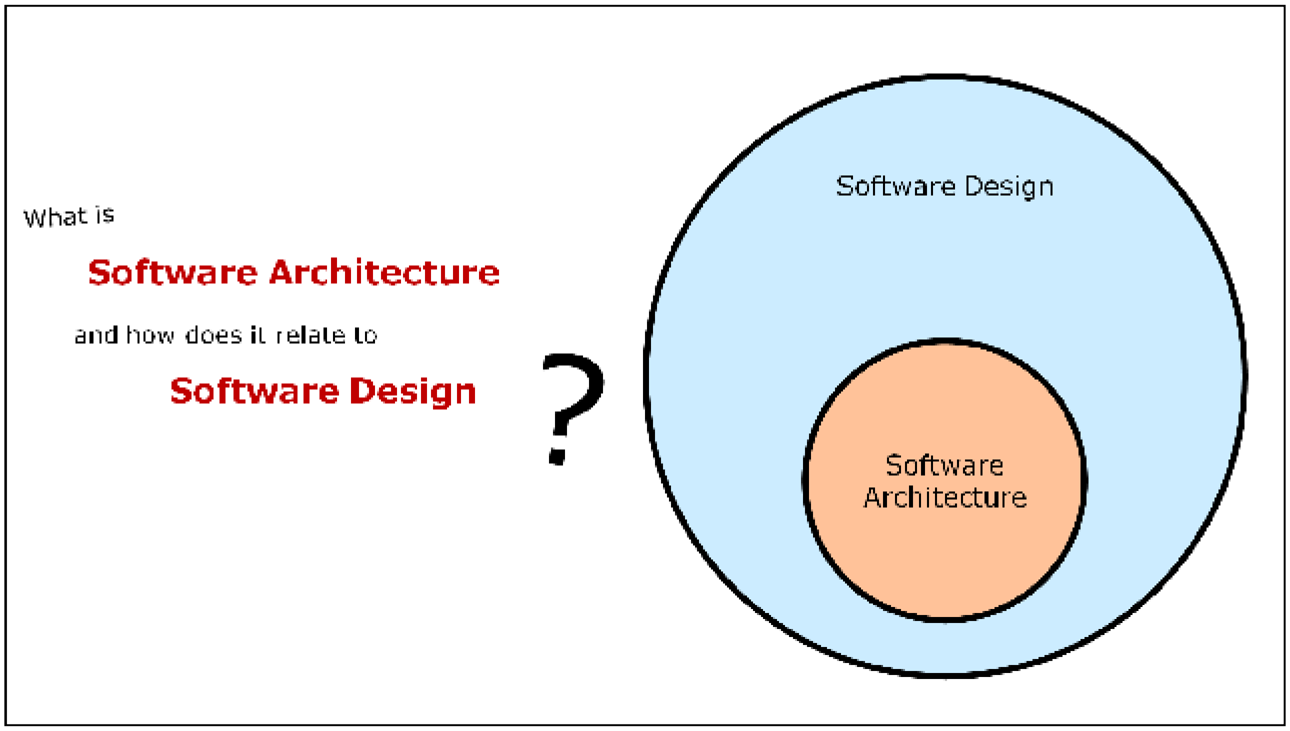
On another, it is the aesthetics of how the building looks from the street.

In both of these cases, the windows would be an important architectural detail.



In other cases, the architect pays attention to other details.

For this fort, the architect was paying attention to offense and defense, not windows and aesthetics.


What is

Software Architecture

and how does it relate to

Software Design

?

Going back to our diagram of architecture and design, does this help us choose which details are architectural?

Not yet.

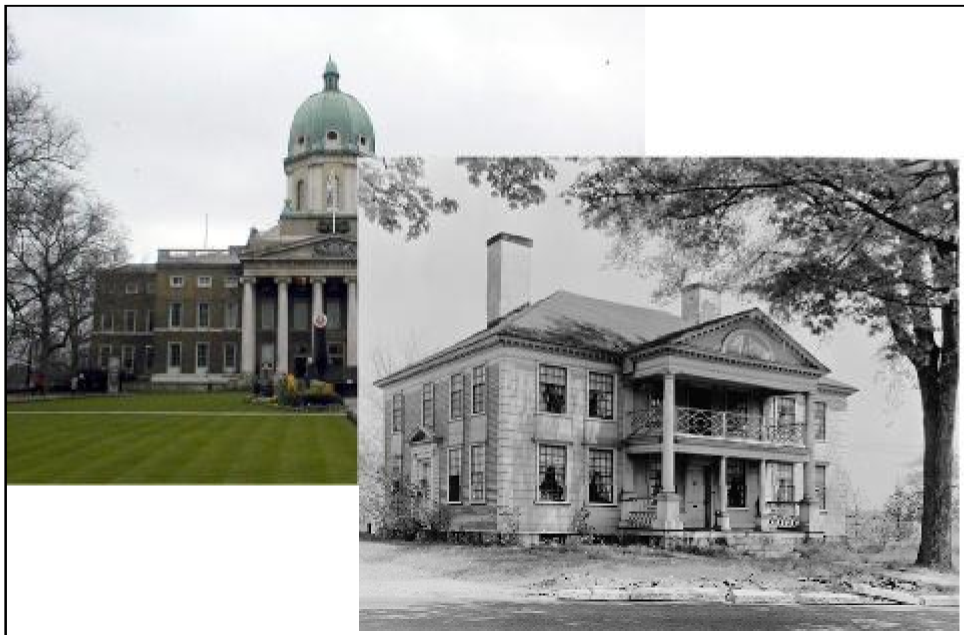
We've learned that architecture details are those that the architect decides are important.

But that sounds pretty subjective.

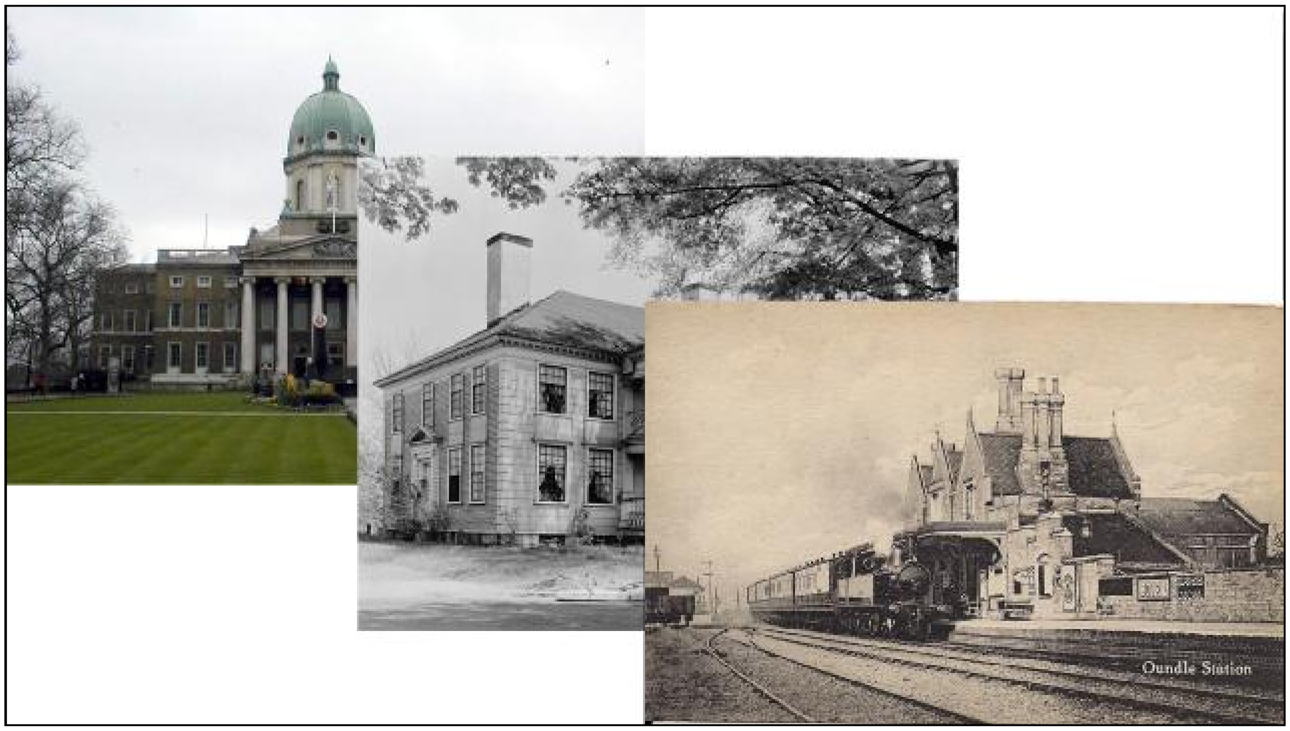
So let's dig a bit deeper and ask how an architect decides that something architecturally important.

Every building must perform its **function**.

A museum must display art;



a house must shelter a family,

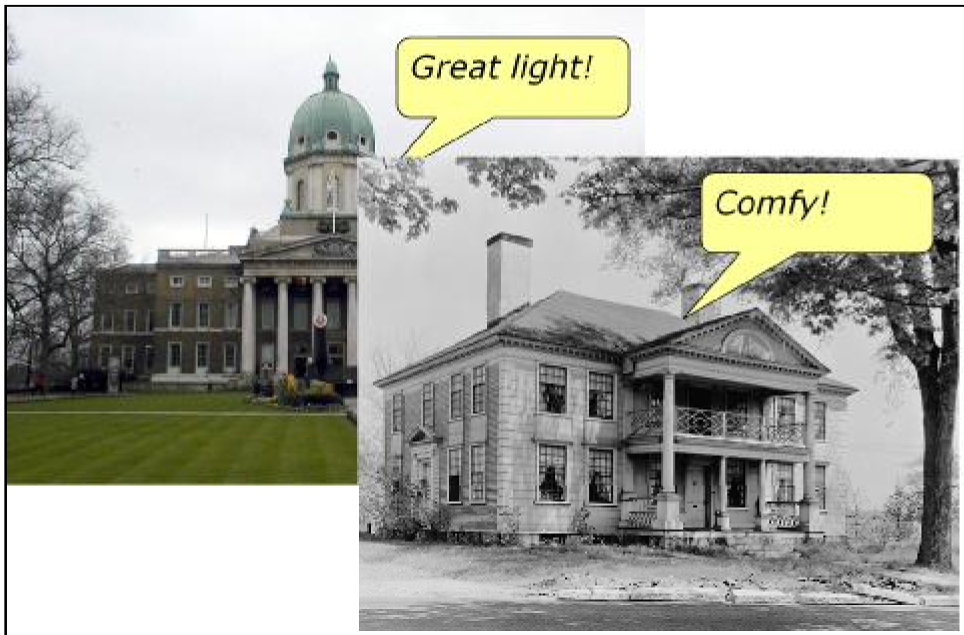


and a train station must transit passengers.

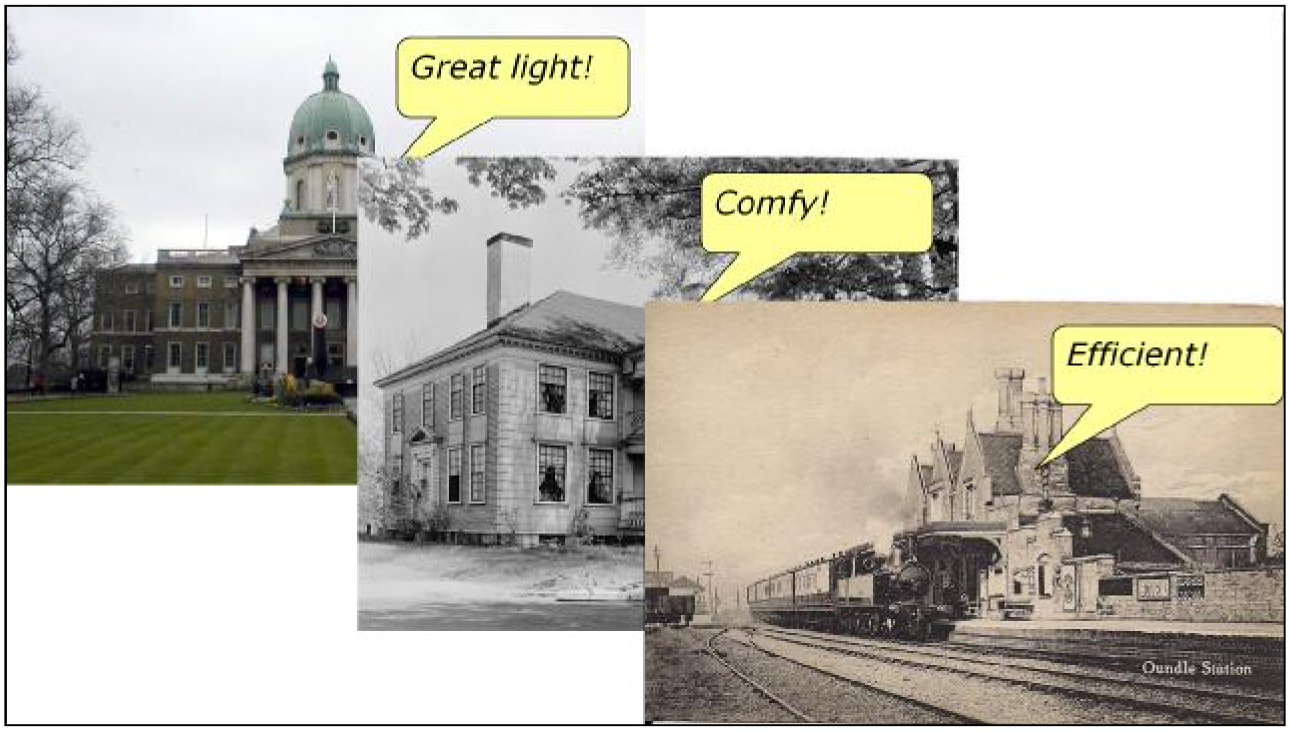
If it does not, it is an engineering failure.

Beyond functionality, architects evaluate other qualities of buildings.

One museum may use natural light better than another;



one house may be more comfortable than another;



and one train station may be more efficient than another.

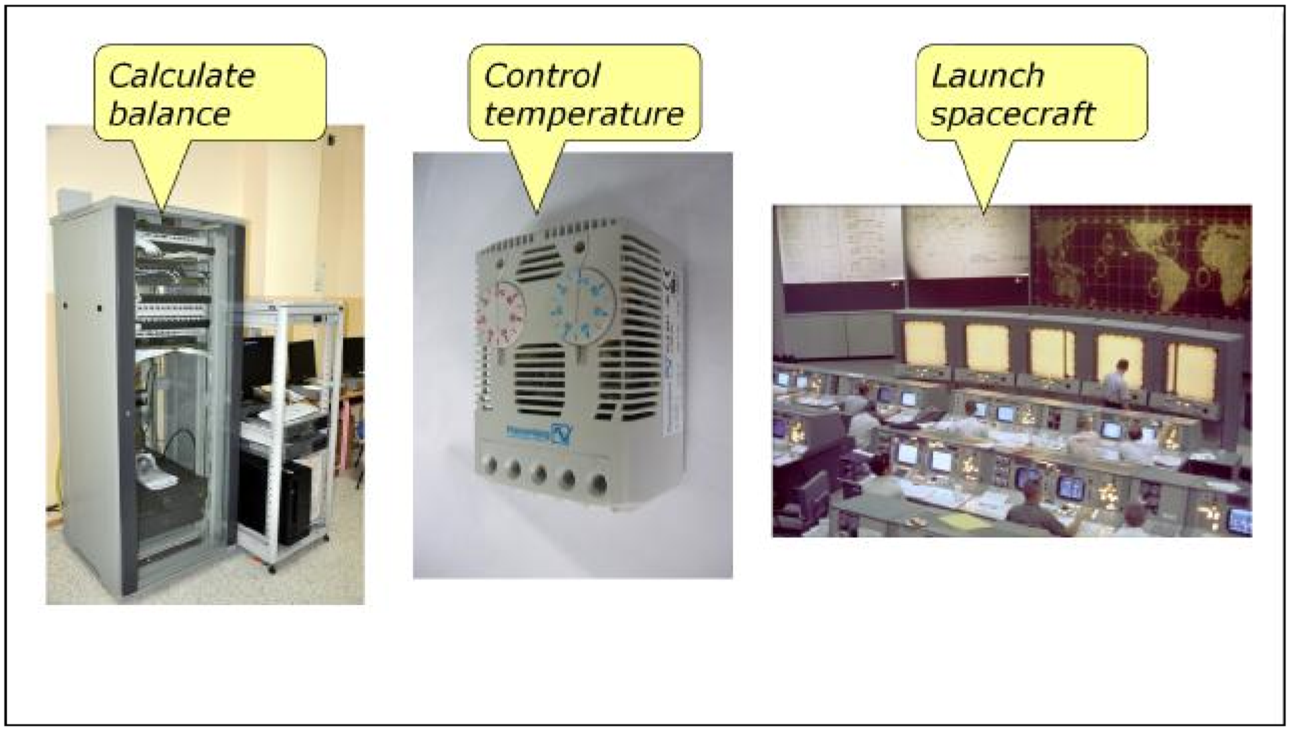
All of these are qualities of the building.

Similarly, software has functionality and qualities.

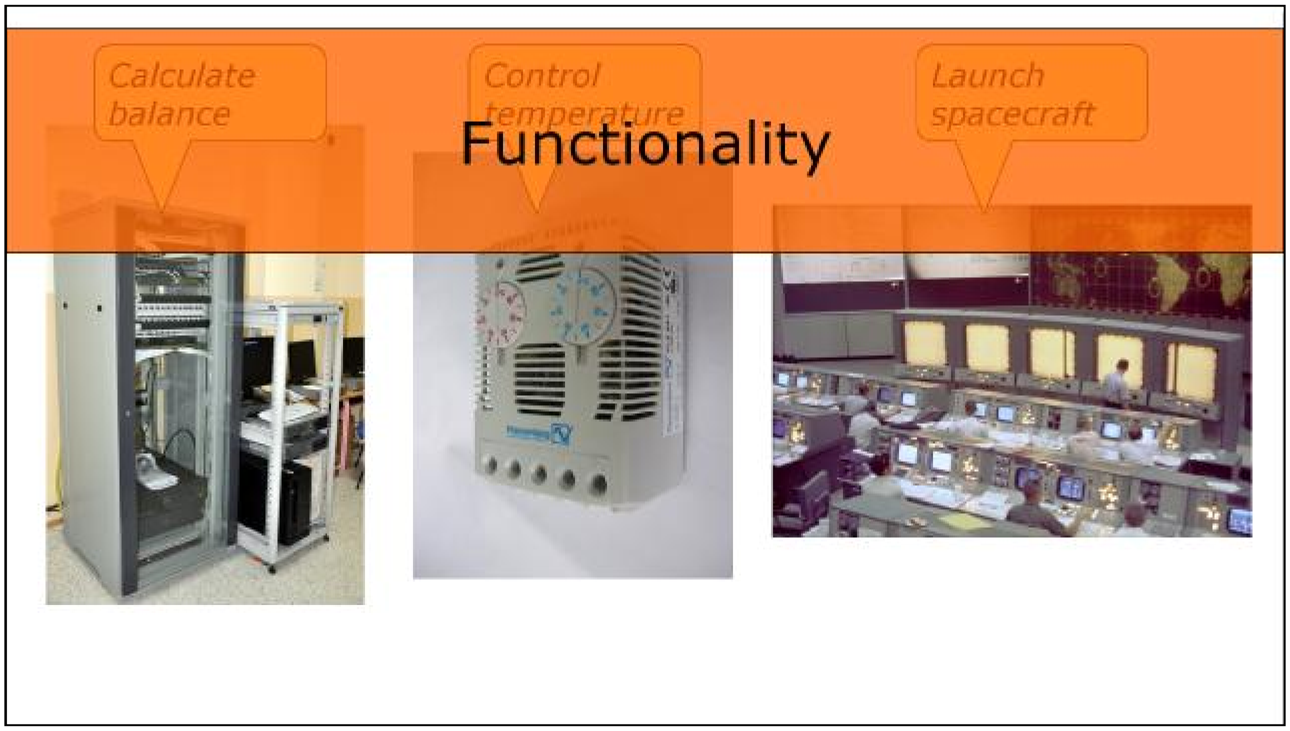
We think about what software does --

calculate a bank balance

control temperature



launch spacecraft

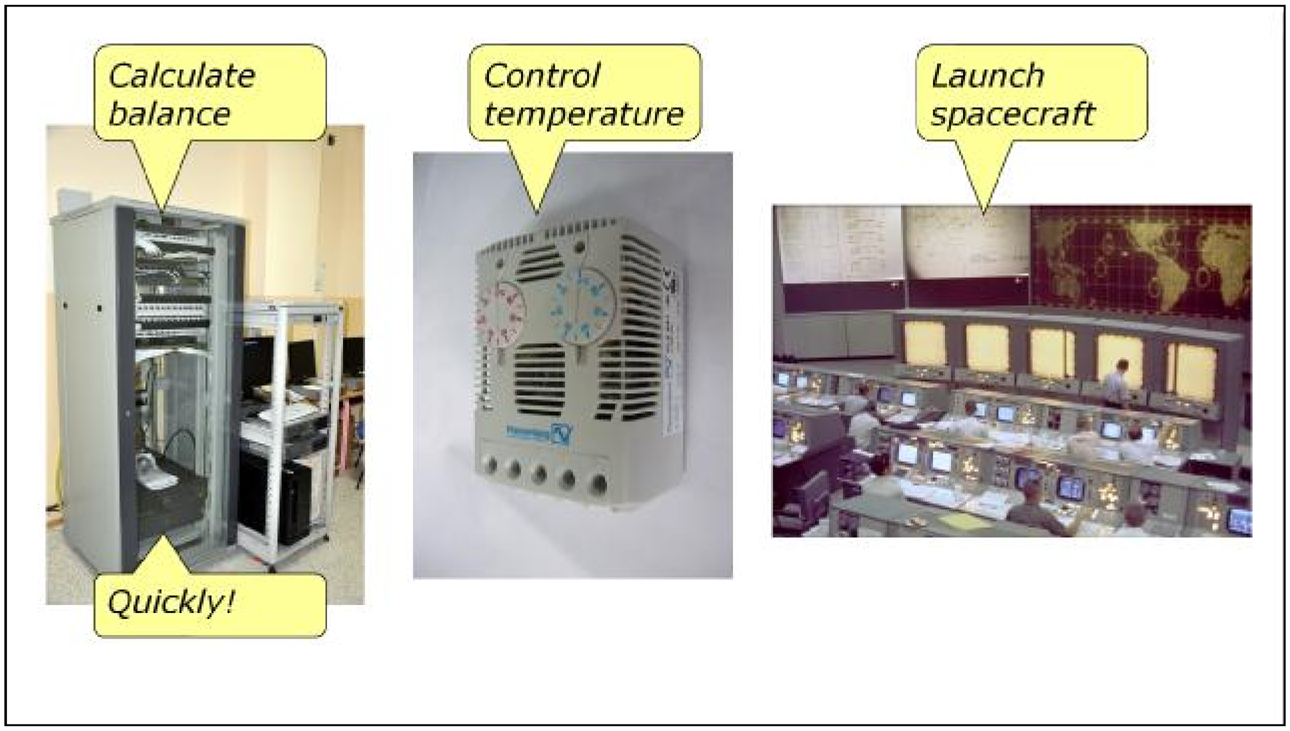


Functionality

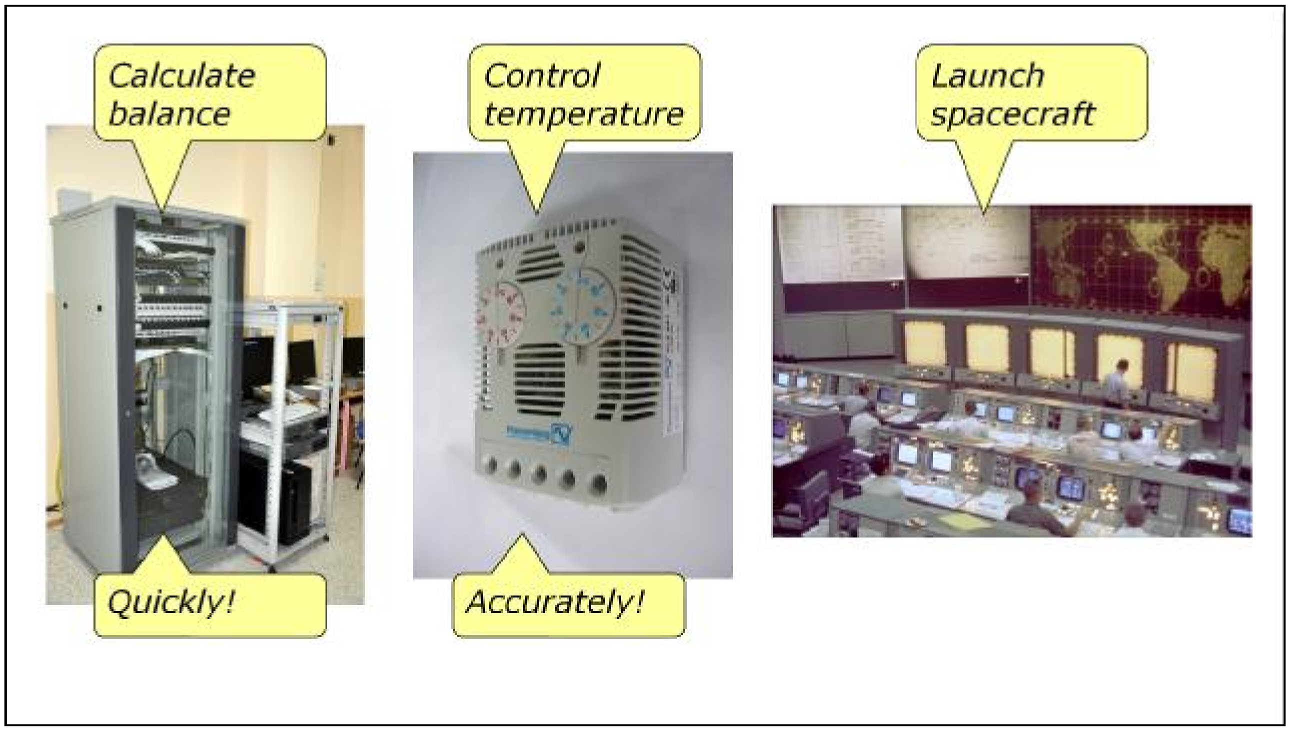
Calculate balance
Control temperature
Launch spacecraft

and call that functionality.

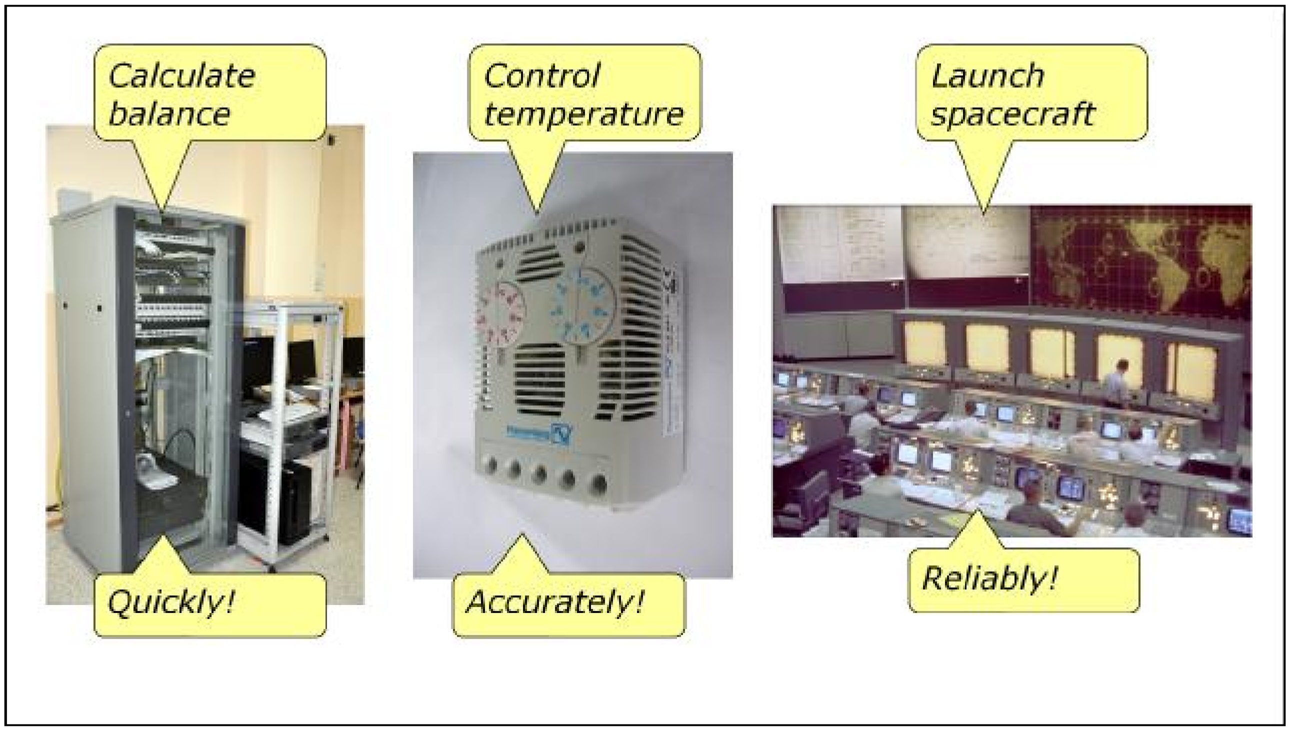
We also think about how the software does its function …


Calculate balance
Quickly!
Control temperature
Launch spacecraft

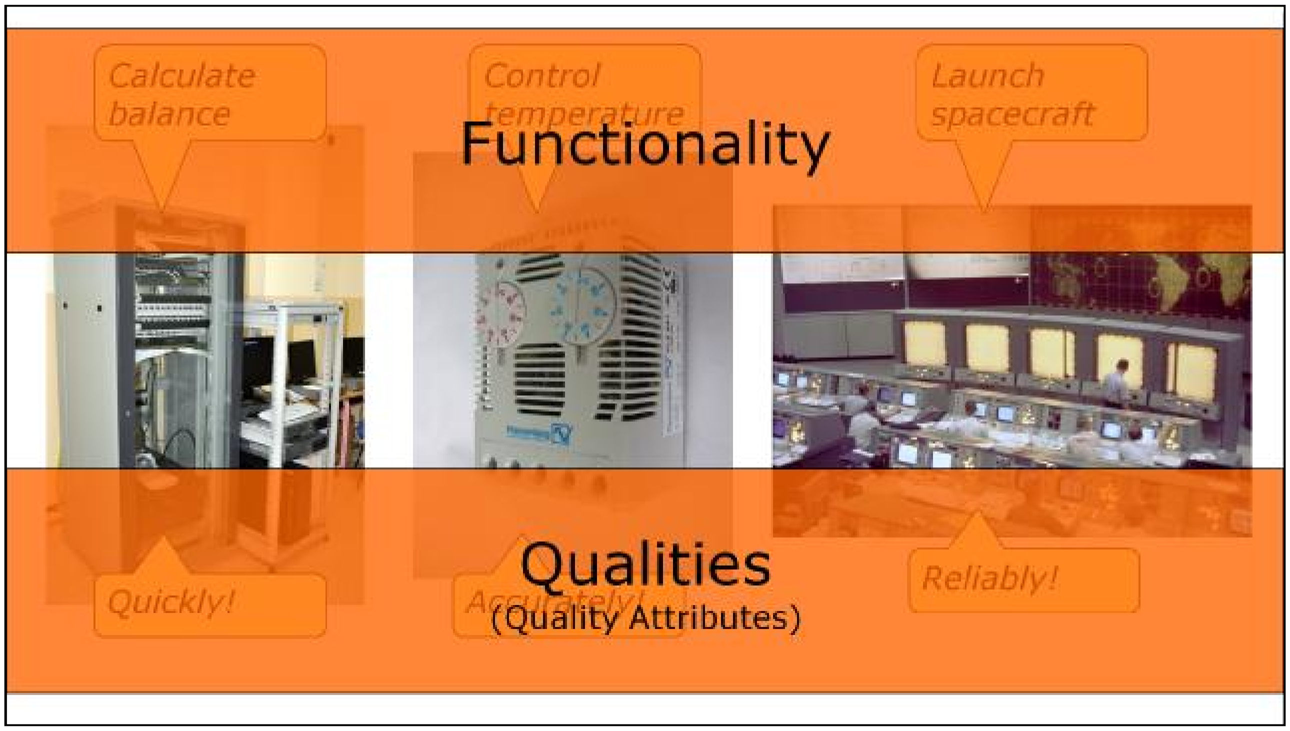
quickly


Calculate balance
Quickly!
Control temperature
Accurately!
Launch spacecraft

accurately


Calculate balance
Quickly!
Control temperature
Accurately!
Launch spacecraft
Reliably!

reliably



Functionality

Qualities

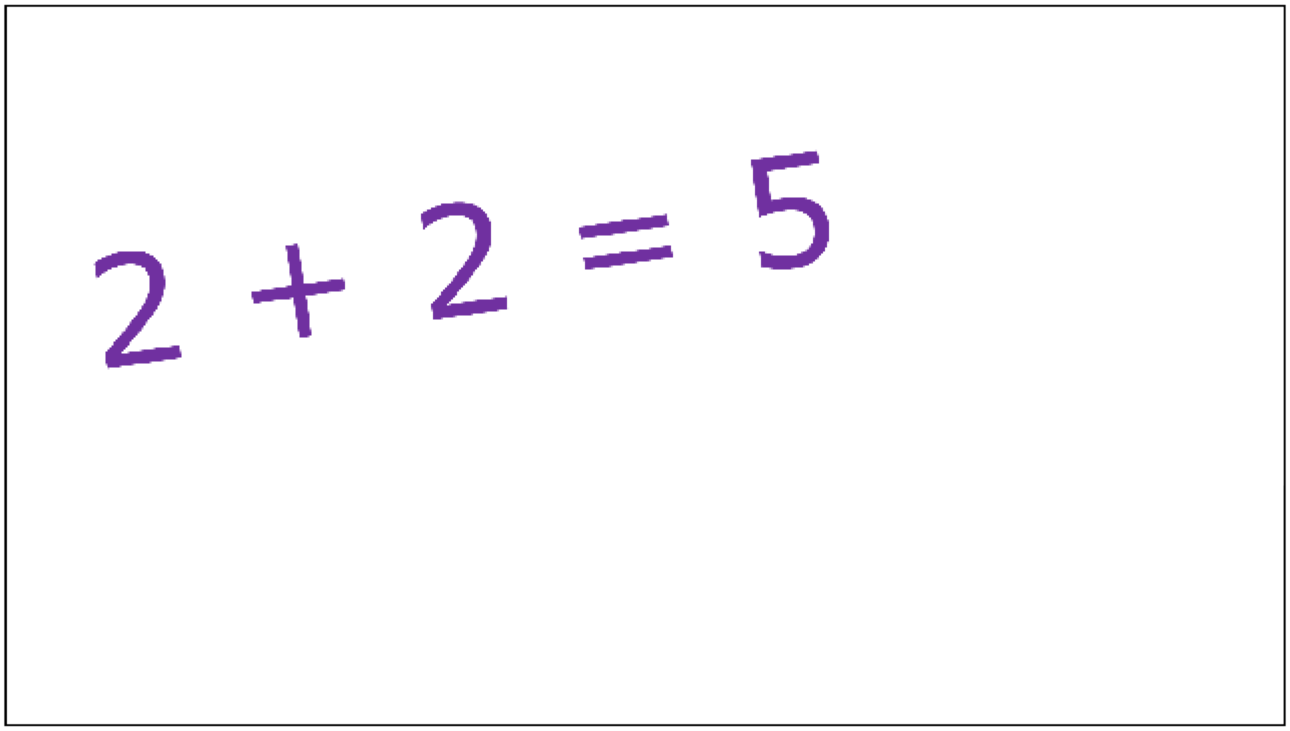
(Quality Attributes)

Calculate balance
Control temperature
Launch spacecraft
Quickly!
Accurately!
Reliably!

and we call these Qualities.

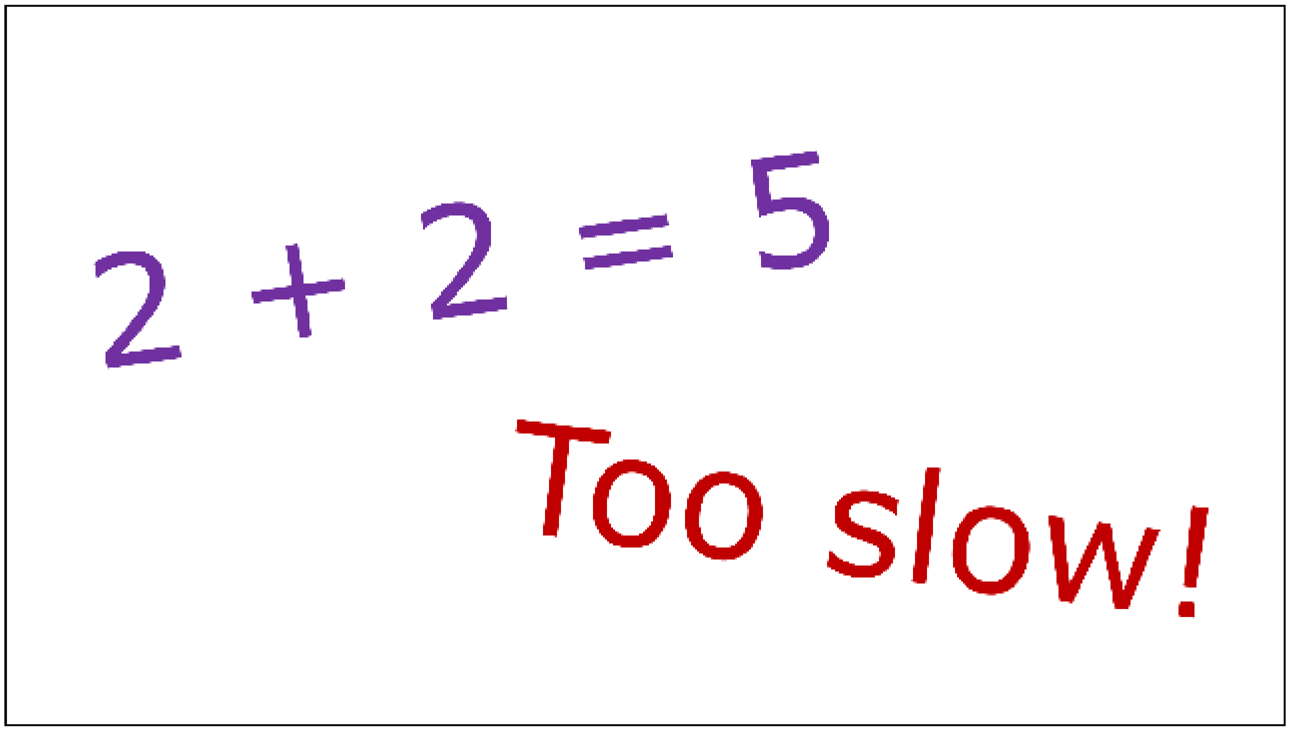
You may have heard Quality Attributes called “non-functional requirements”, but between you and me, let's avoid calling them that.

Would you drink from this fountain?
When most people hear “non-functional” they think “broken”.



$2 + 2 = 5$

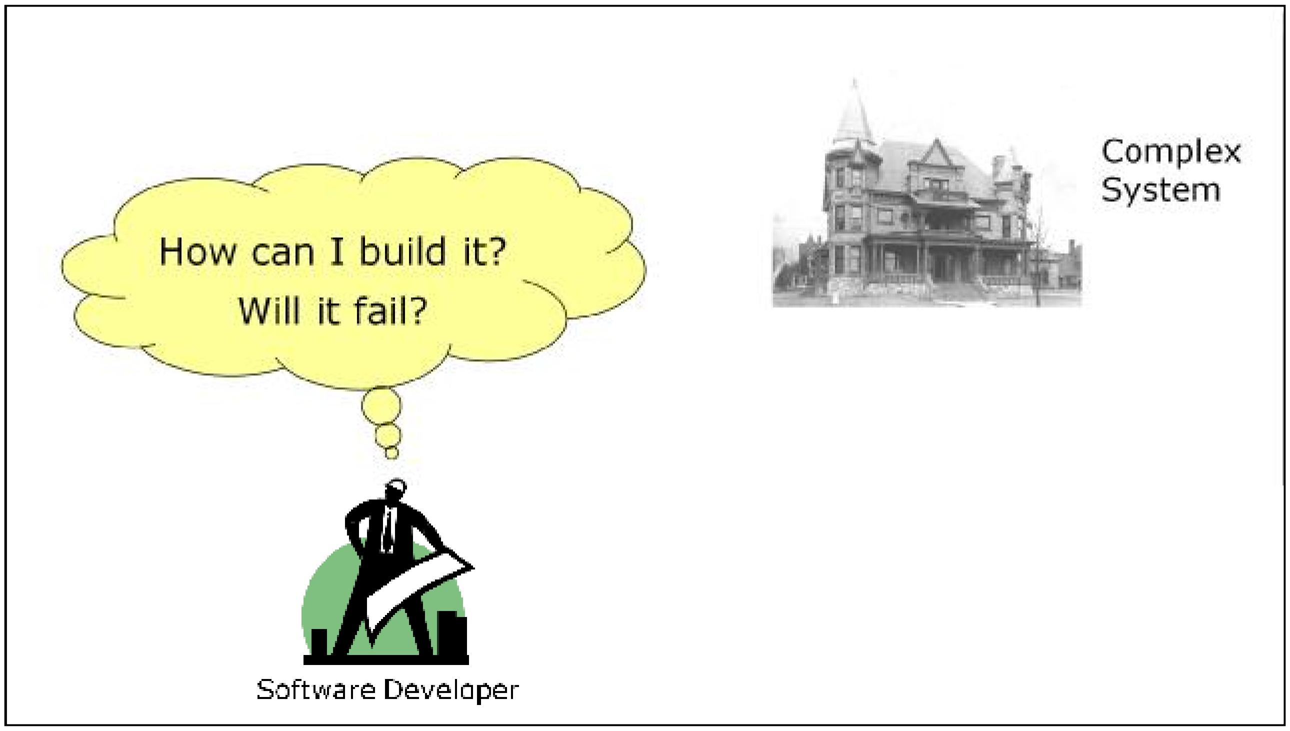
Software can **fail** either because it doesn’t do what it should – its functionality



… or because it does it the wrong way – its qualities.

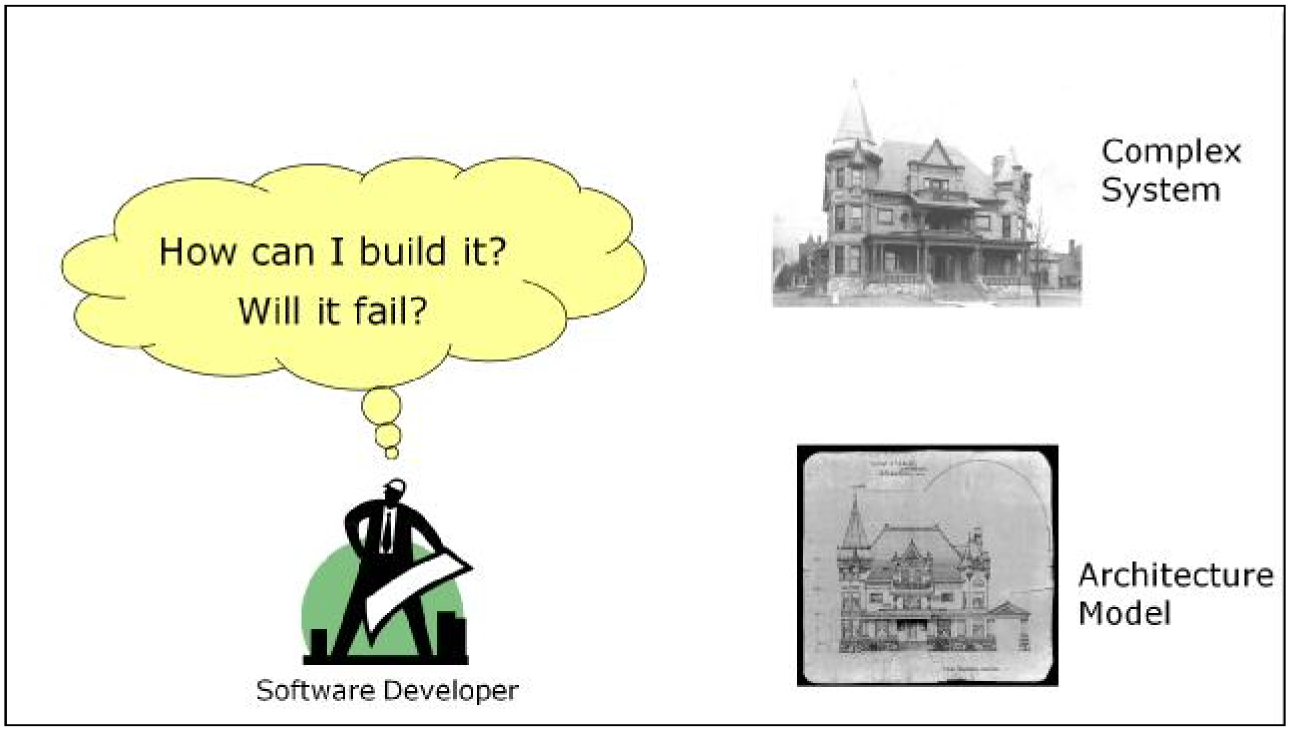
So you must reason about failure of both functionality and quality attributes.

But how do you reason about complex systems?



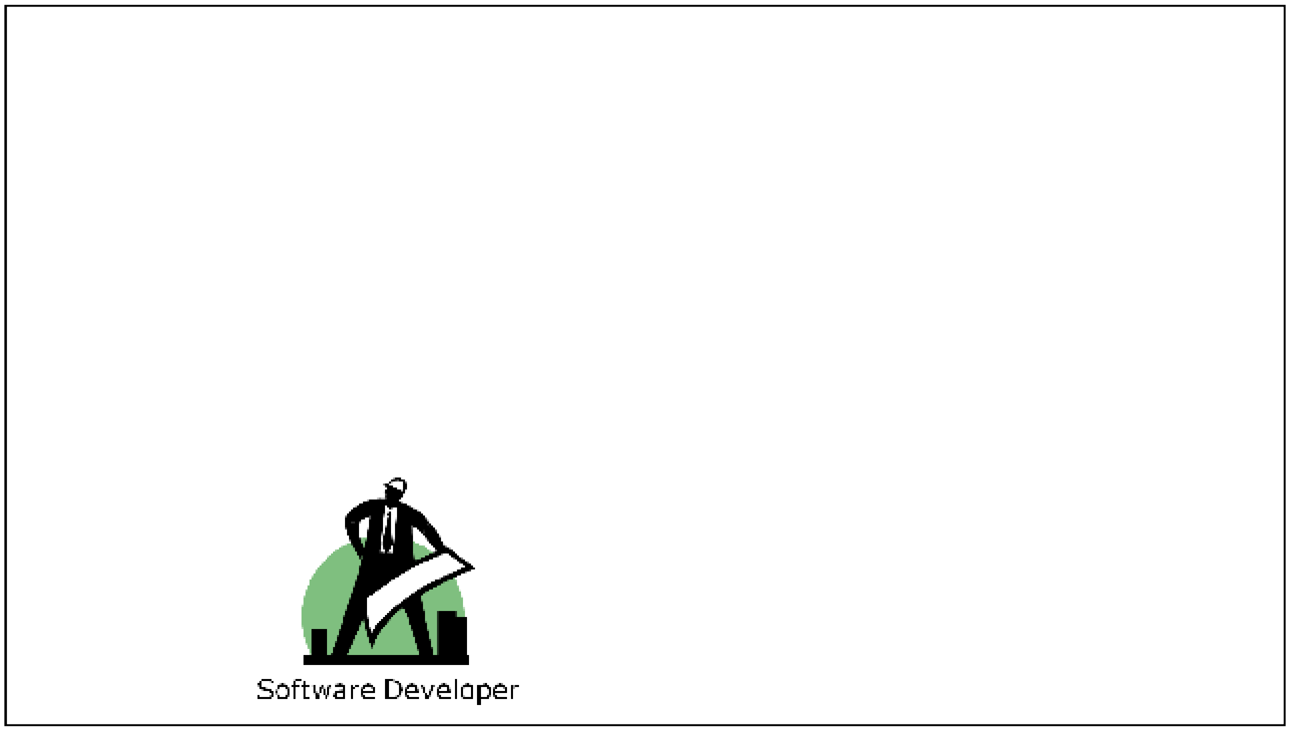
Real systems are too complex to fit everything in your head.

So when reasoning through “will this work” and “what might it fail”, engineers and architects work with **models**.



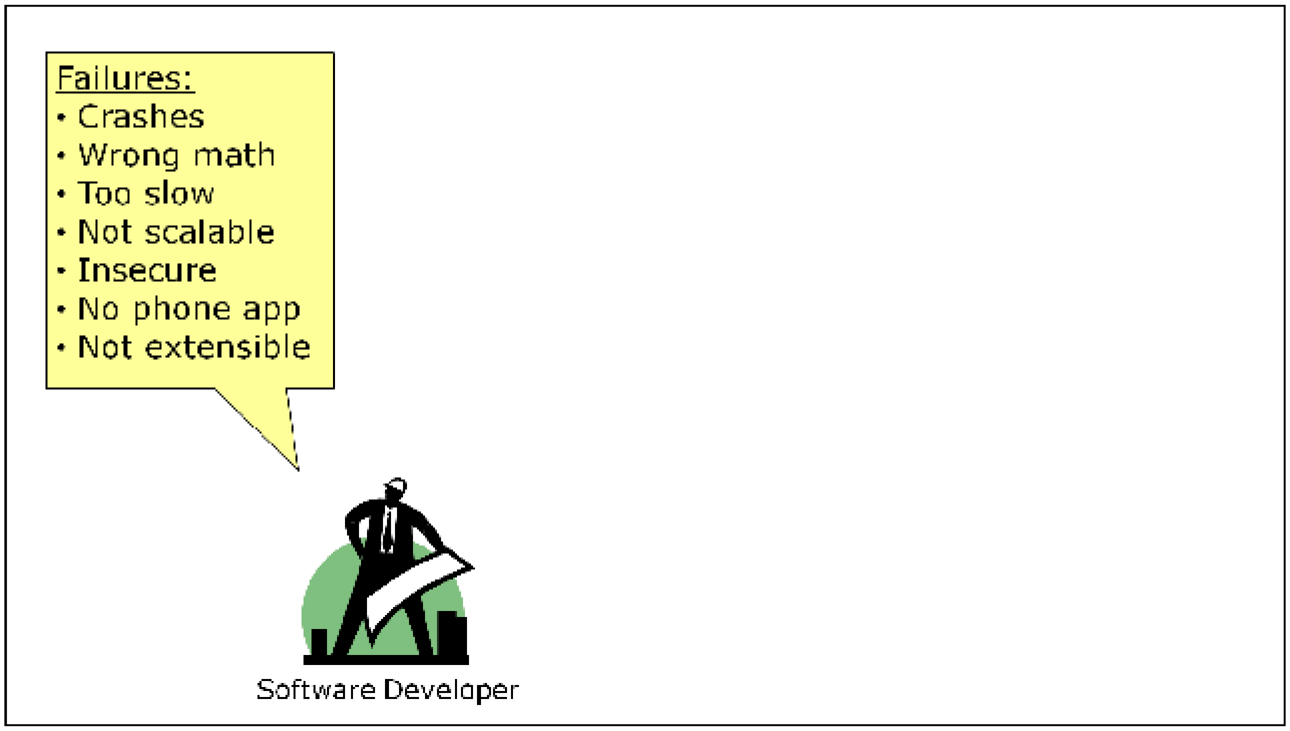
Models are simpler than the real system so you can reason about different designs and failures.

But because they are simpler, they don’t have all details.

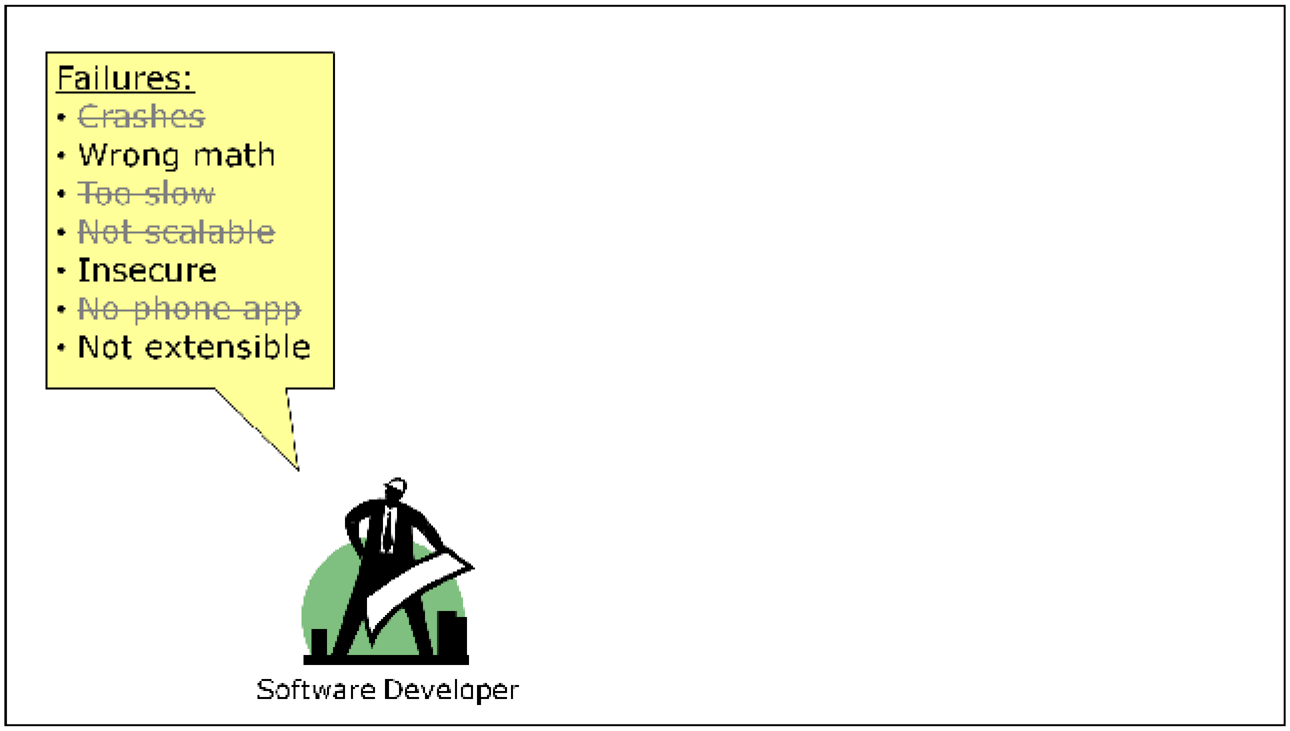


Now we are ready to answer our question about architecture and design.

So here is what differentiates them.

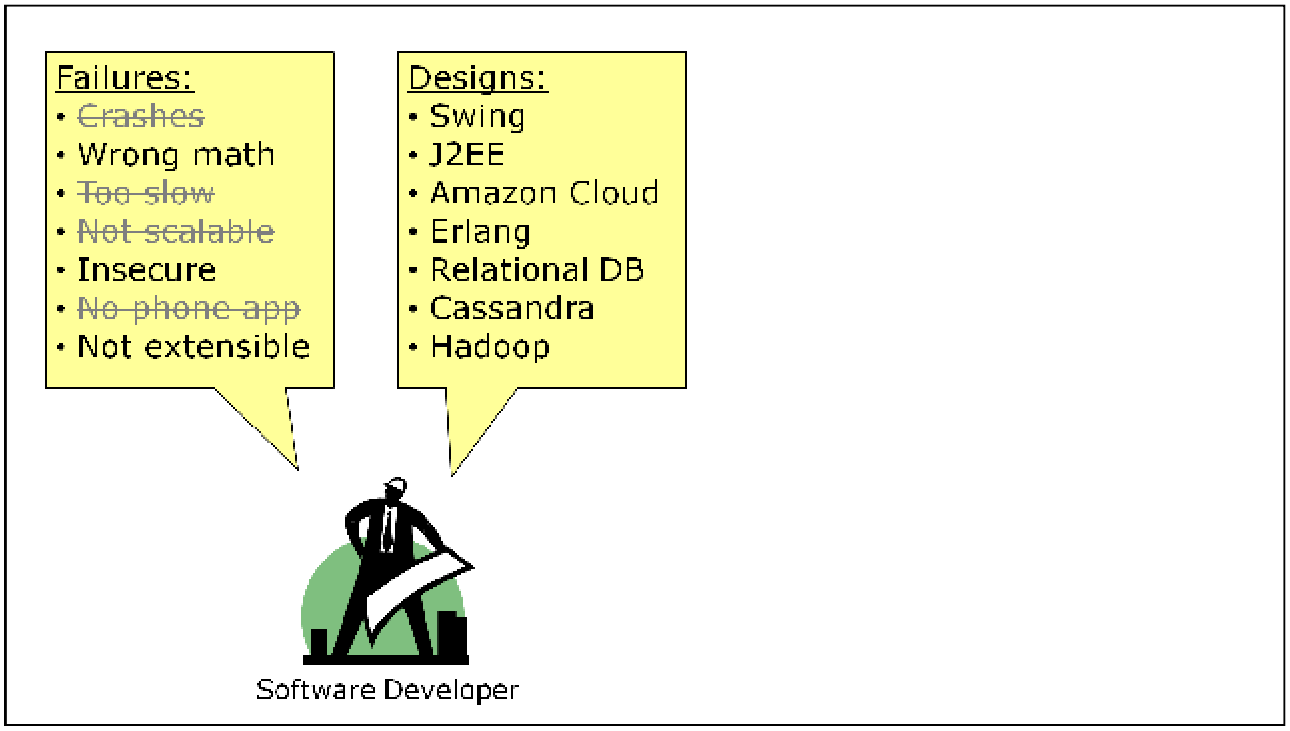


As a software developer, you think about possible failures.

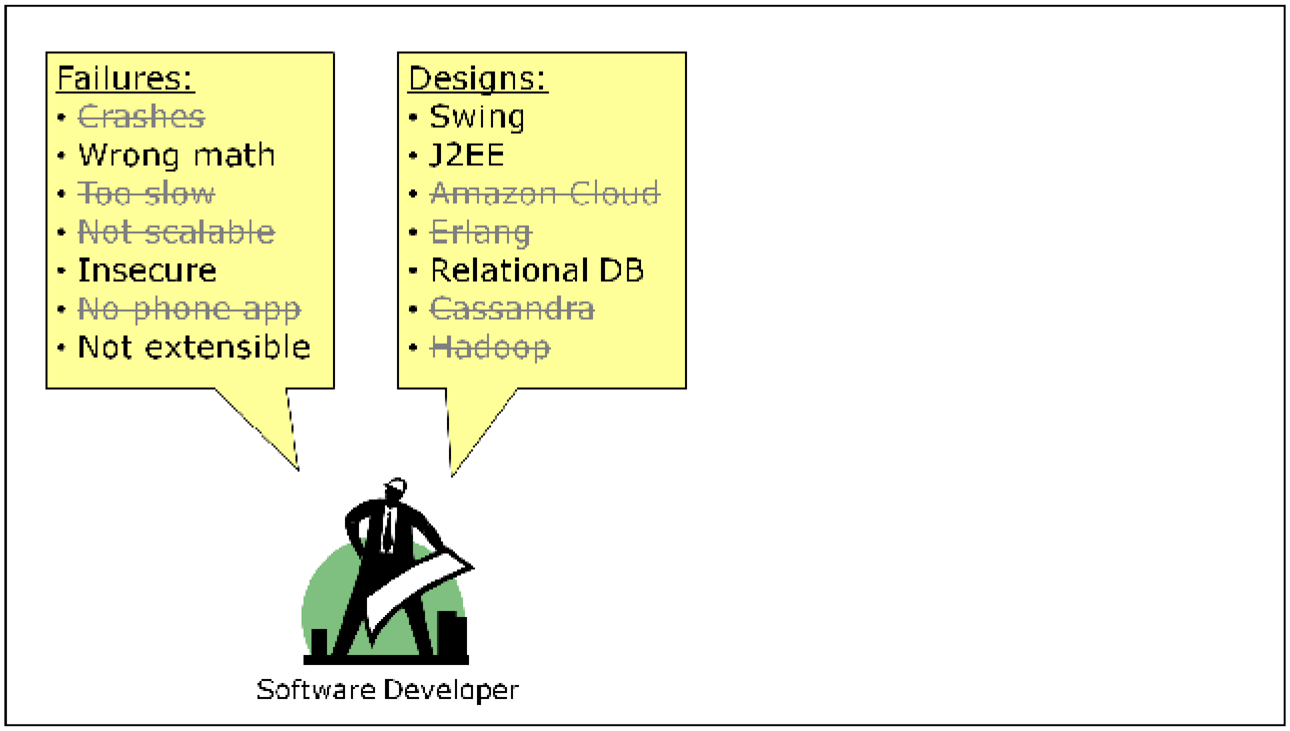


You decide which kinds failures you will try to avoid.

Remember to think about failures related to quality attributes, not just functionality.



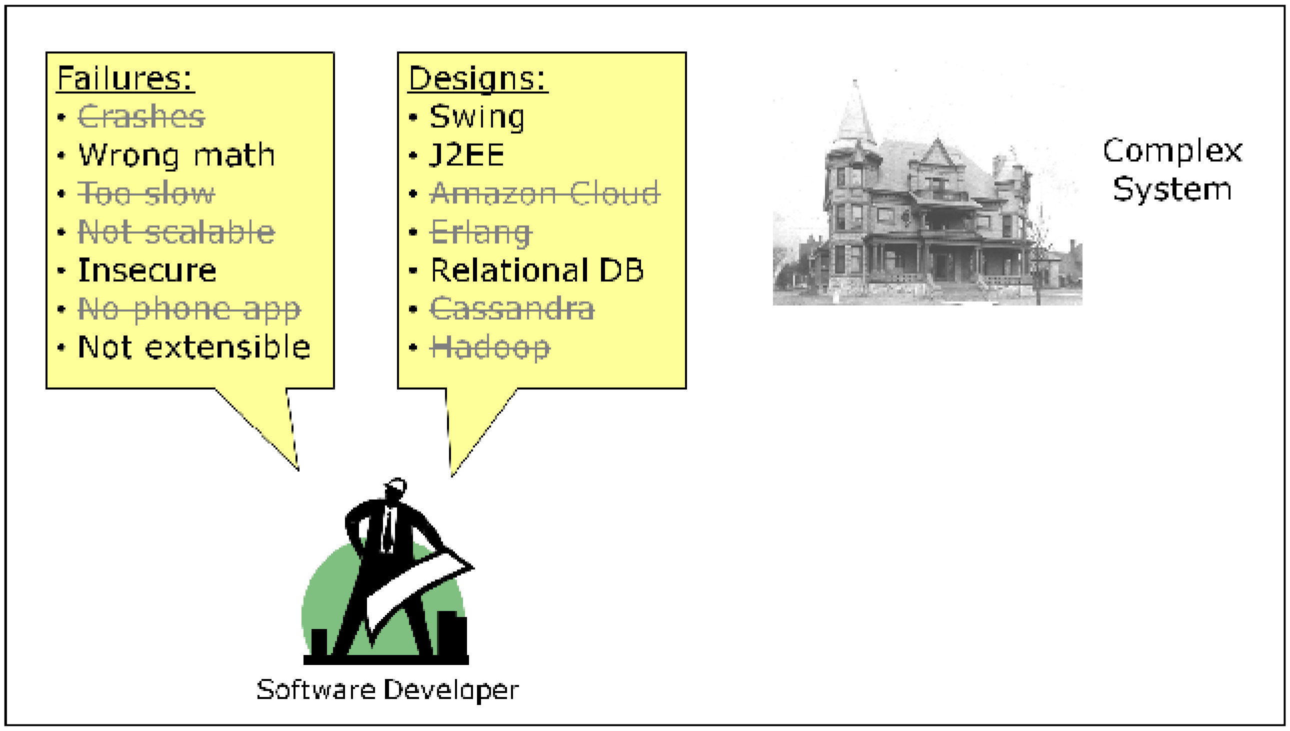
You think about lots of possible designs too ...



Failures:
- ~~Crashes~~
- Wrong math
- ~~Too slow~~
- ~~Not scalable~~
- Insecure
- ~~No phone app~~
- Not extensible

Designs:
- Swing
- J2EE
- ~~Amazon Cloud~~
- ~~Erlang~~
- Relational DB
- ~~Cassandra~~
- ~~Hadoop~~

and exclude the ones that are susceptible to the failures you are worried about.



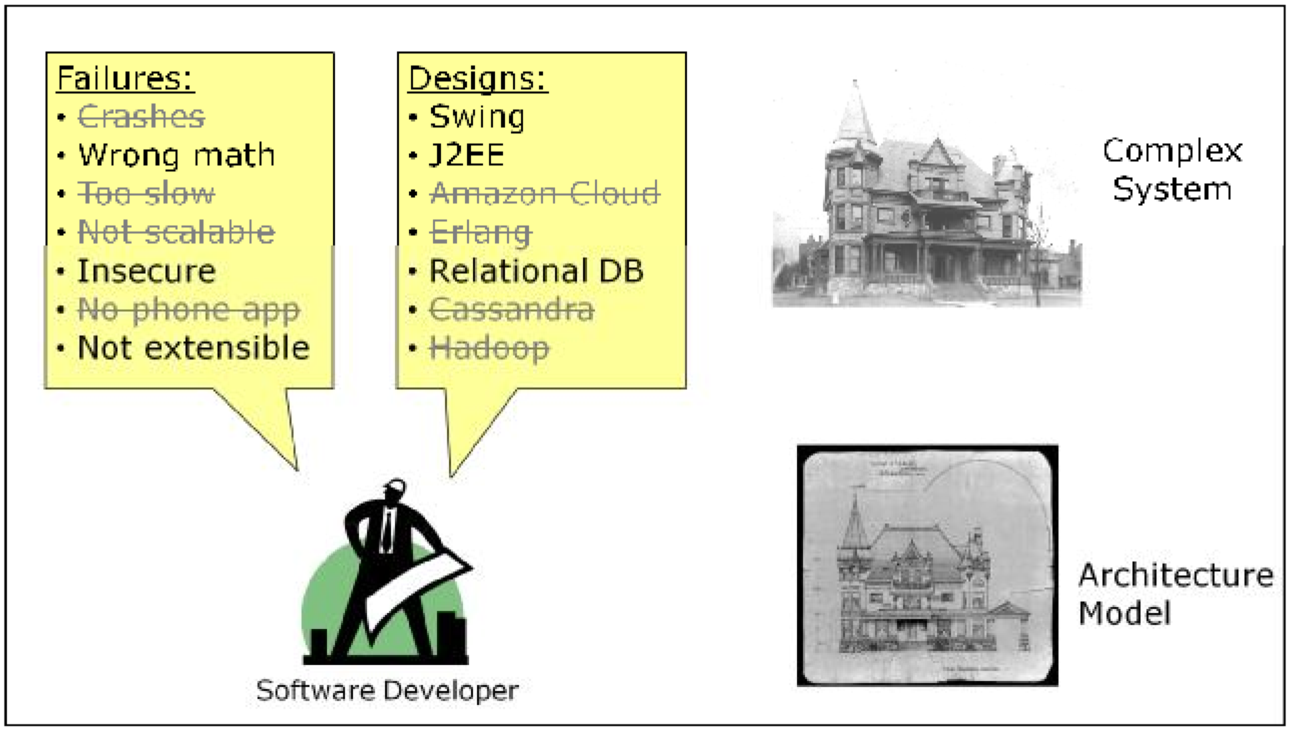
Failures:
- ~~Crashes~~
- Wrong math
- ~~Too slow~~
- ~~Not scalable~~
- Insecure
- ~~No phone app~~
- Not extensible

Designs:
- Swing
- J2EE
- ~~Amazon Cloud~~
- ~~Erlang~~
- Relational DB
- ~~Cassandra~~
- ~~Hadoop~~

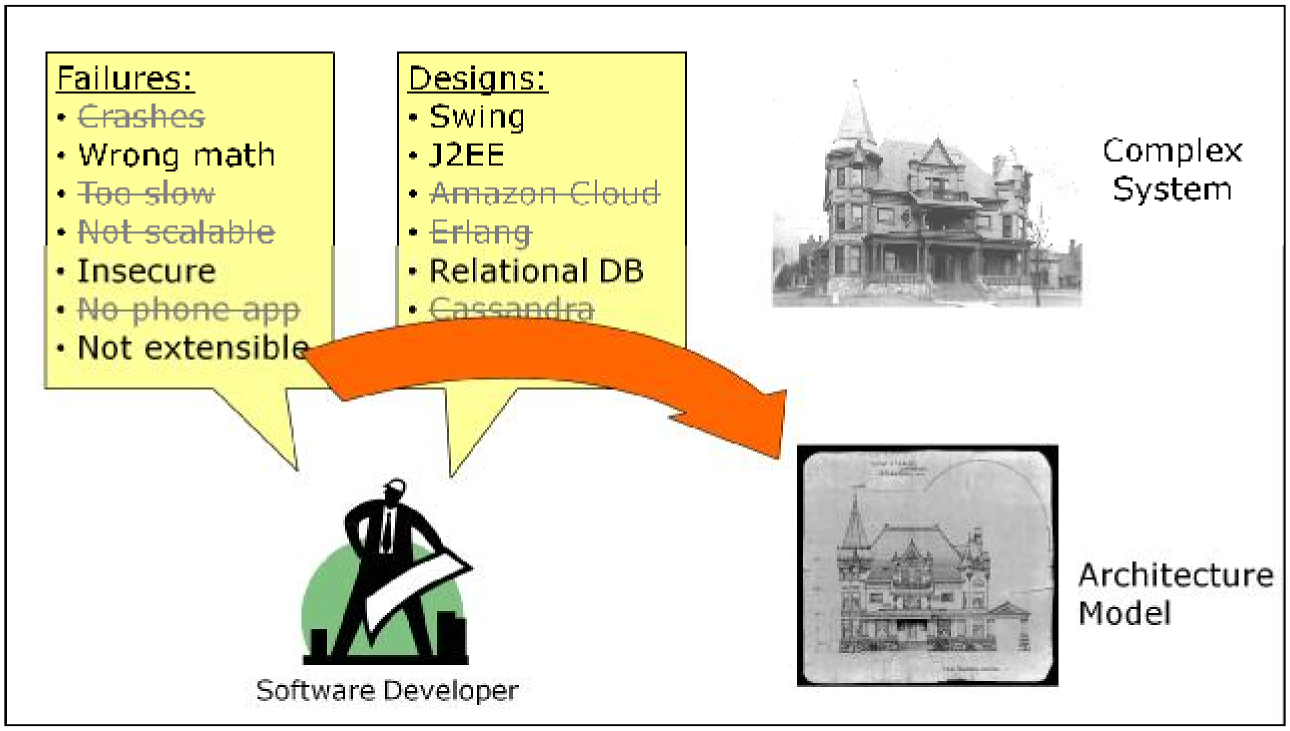
Complex System

Software Developer

Since the system you will build is complex ...



you do your reasoning with a model of the system.

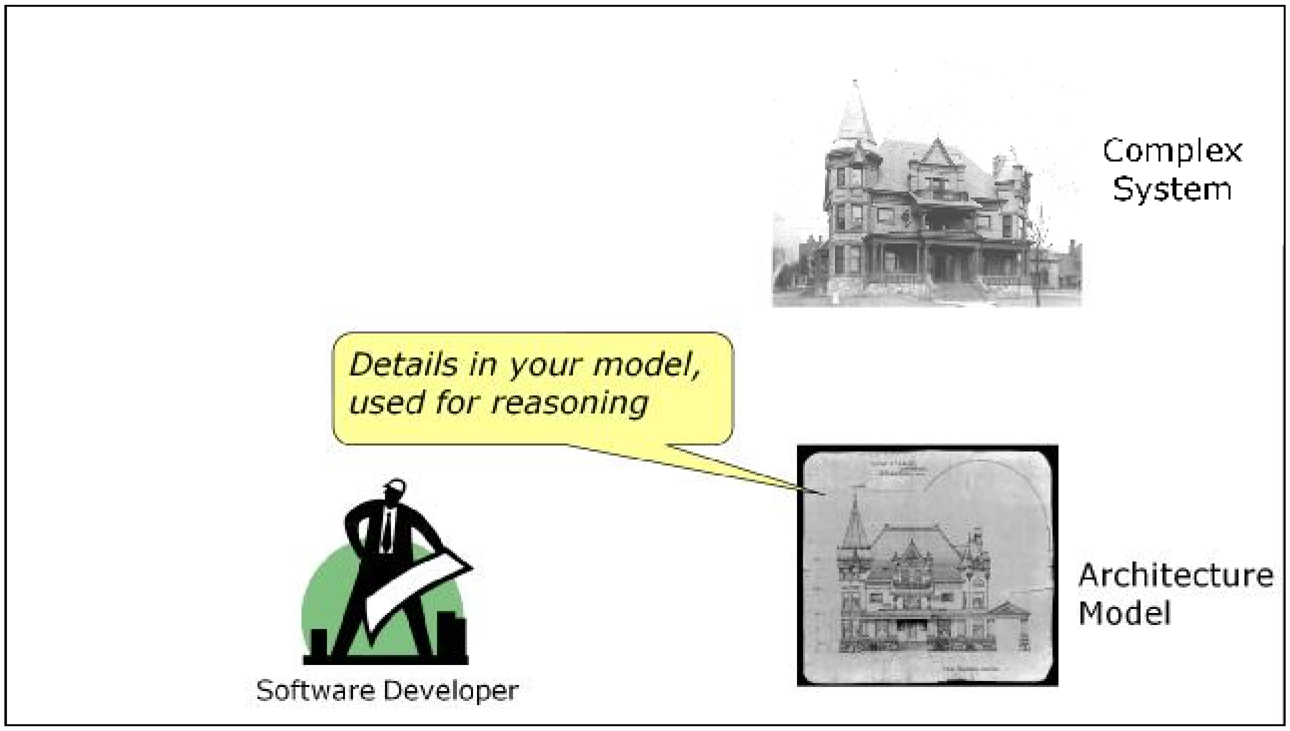


You include details in your model only if they help you reason about the failures you identified.

You omit other details.

This is of course a simplification of the process.

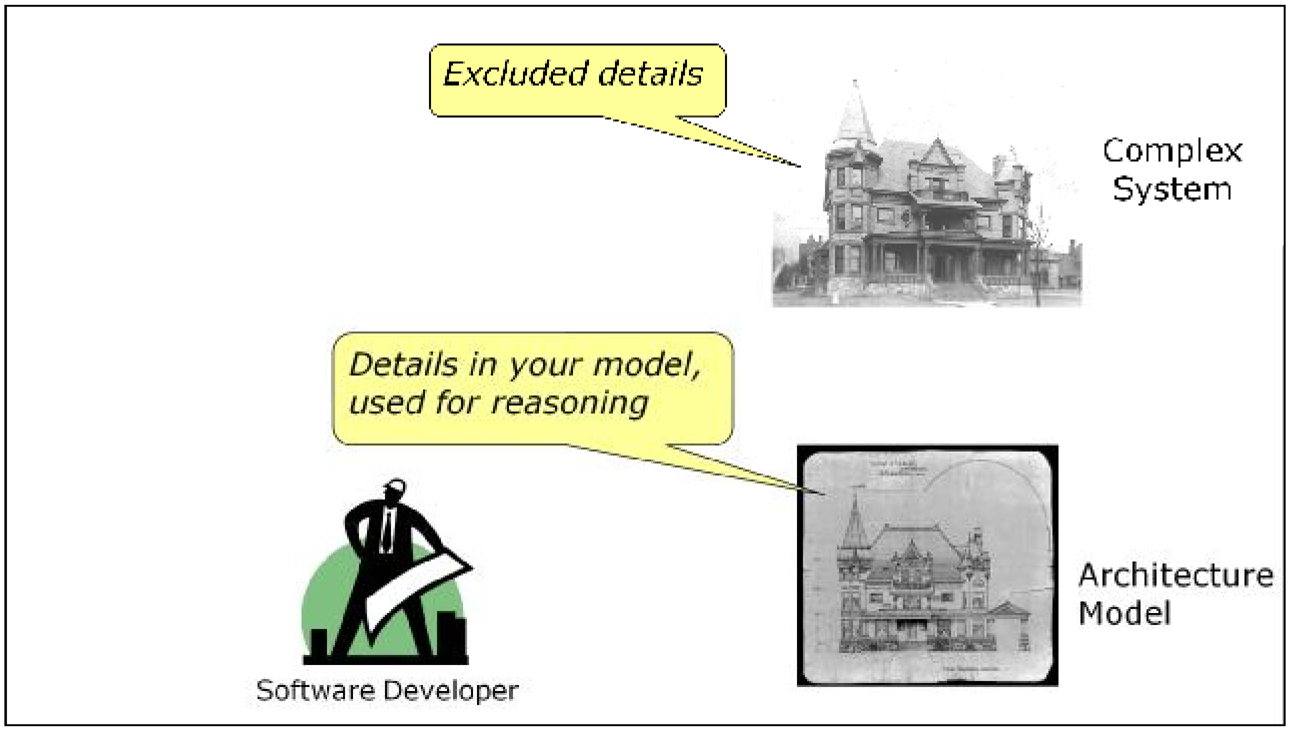
On real projects, things are not as linear as shown here.



So, as an architect you build models of the system you are designing.

The model helps you reason about and avoid the failures you are worried about.

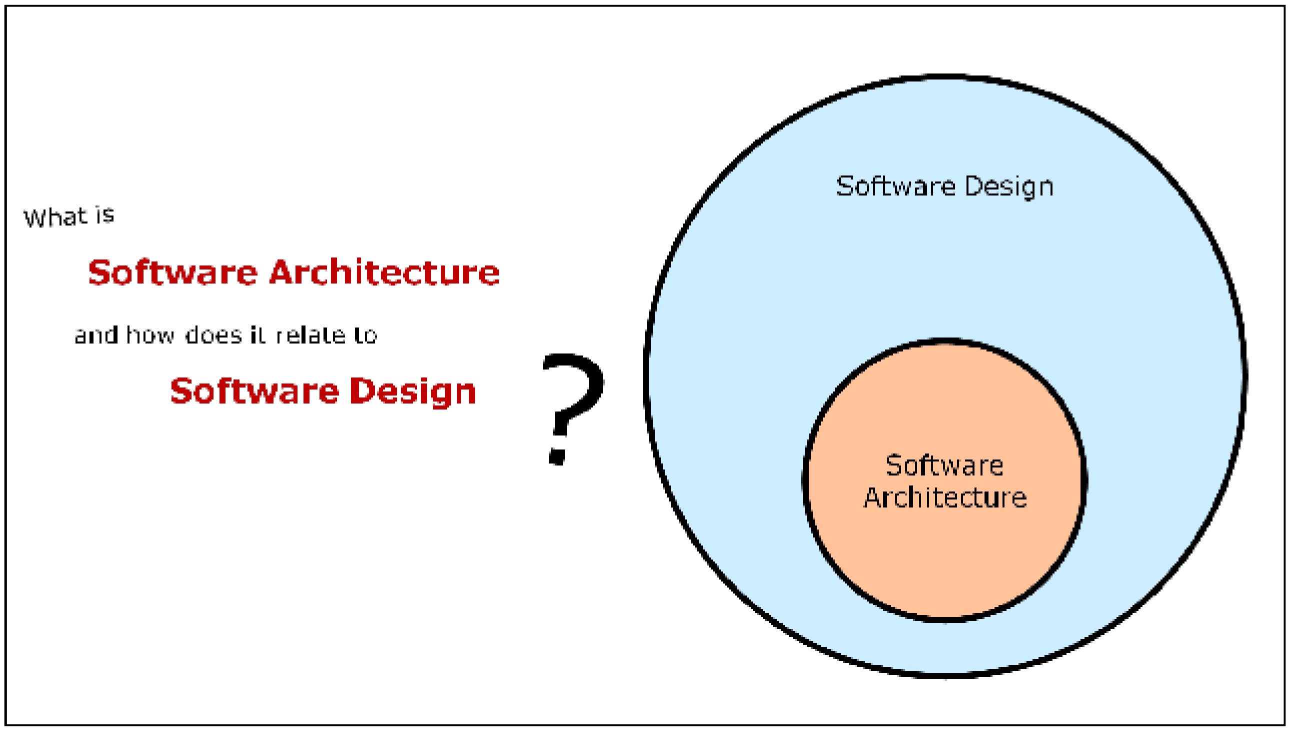
The details you include in your model are **architectural**.



The full design of the system contains other details,

details needed to build the system but not needed to reason about failures.

Those details are not architectural.


What is

Software Architecture

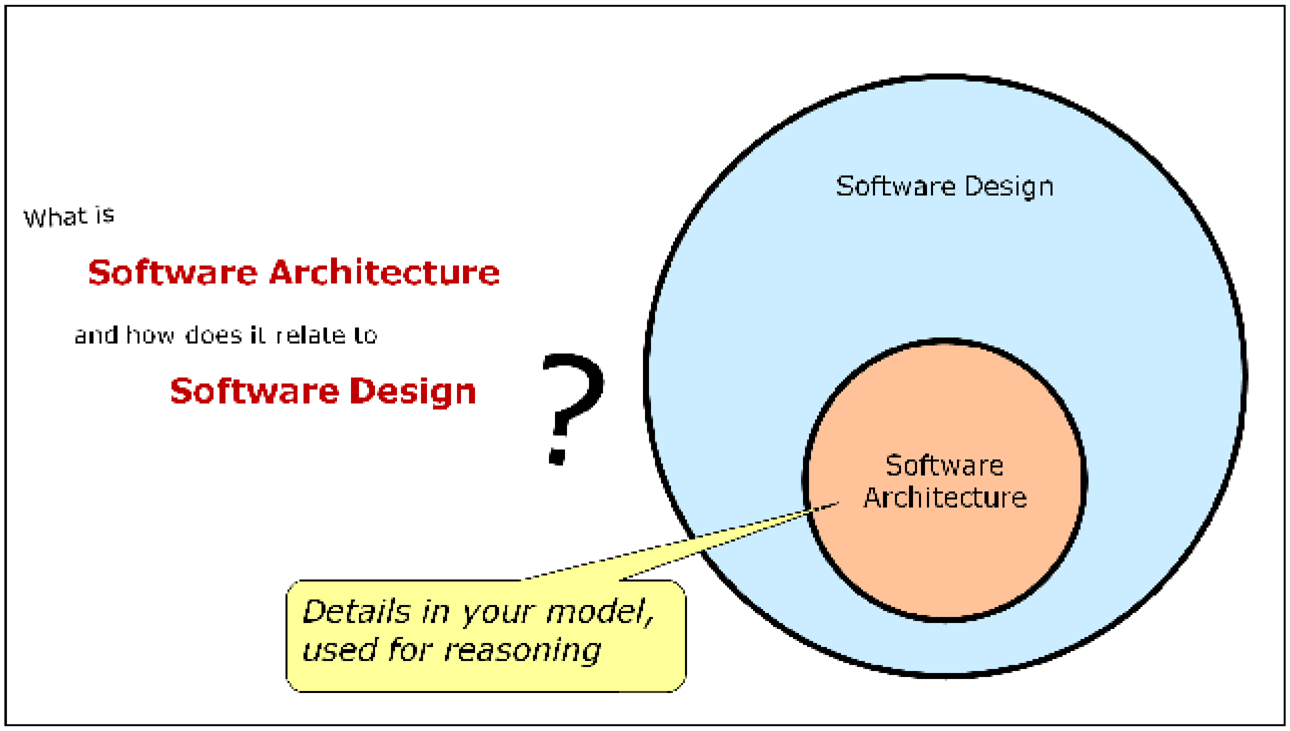
and how does it relate to

Software Design

?

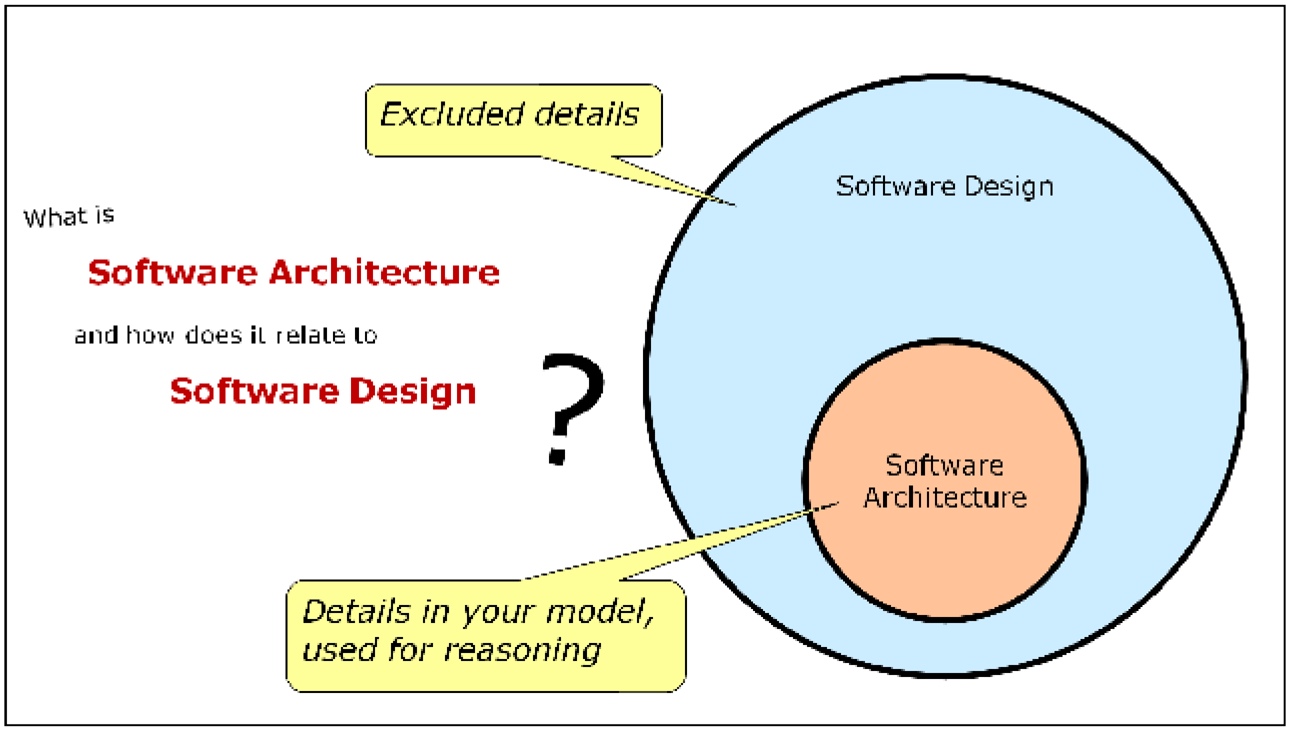
Now let’s return to the question:

What is software architecture and how does it relate to design?



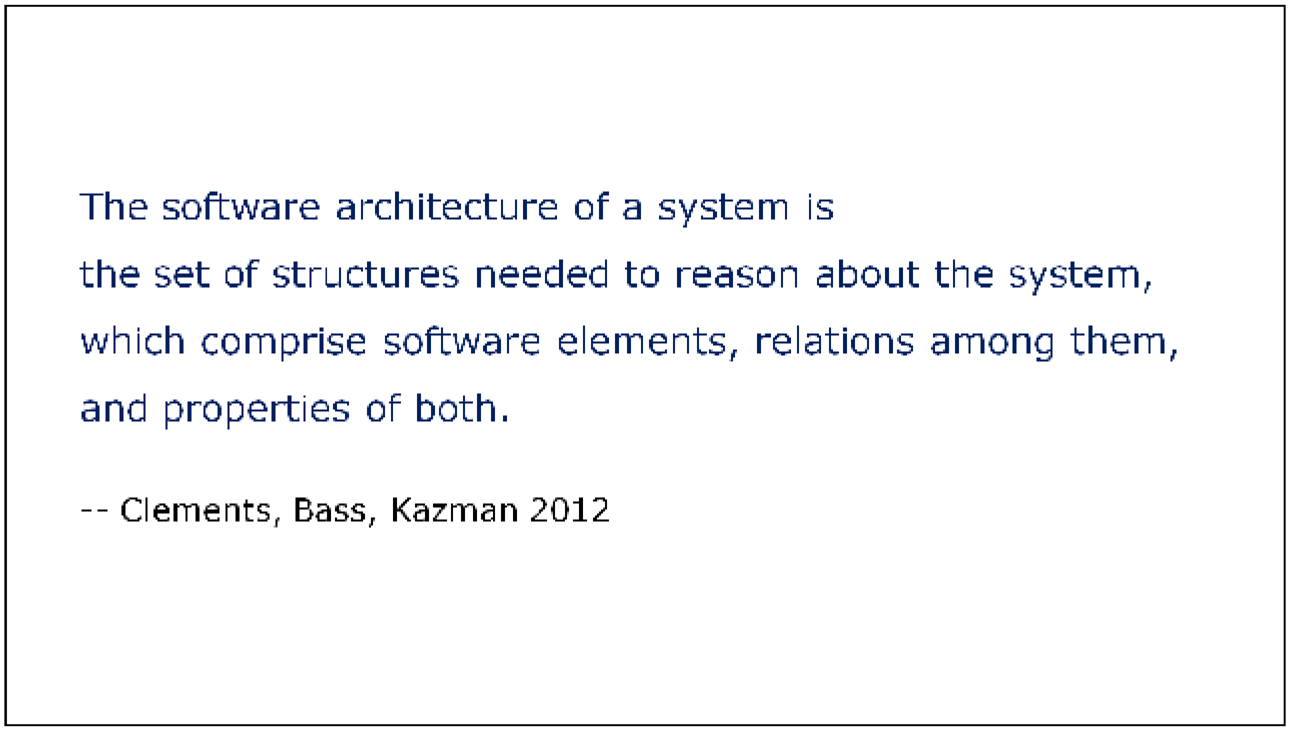
Architects build models to help them reason about failures.

The details in those models are the software architecture.



The other details needed to build the software are the rest of the software design.

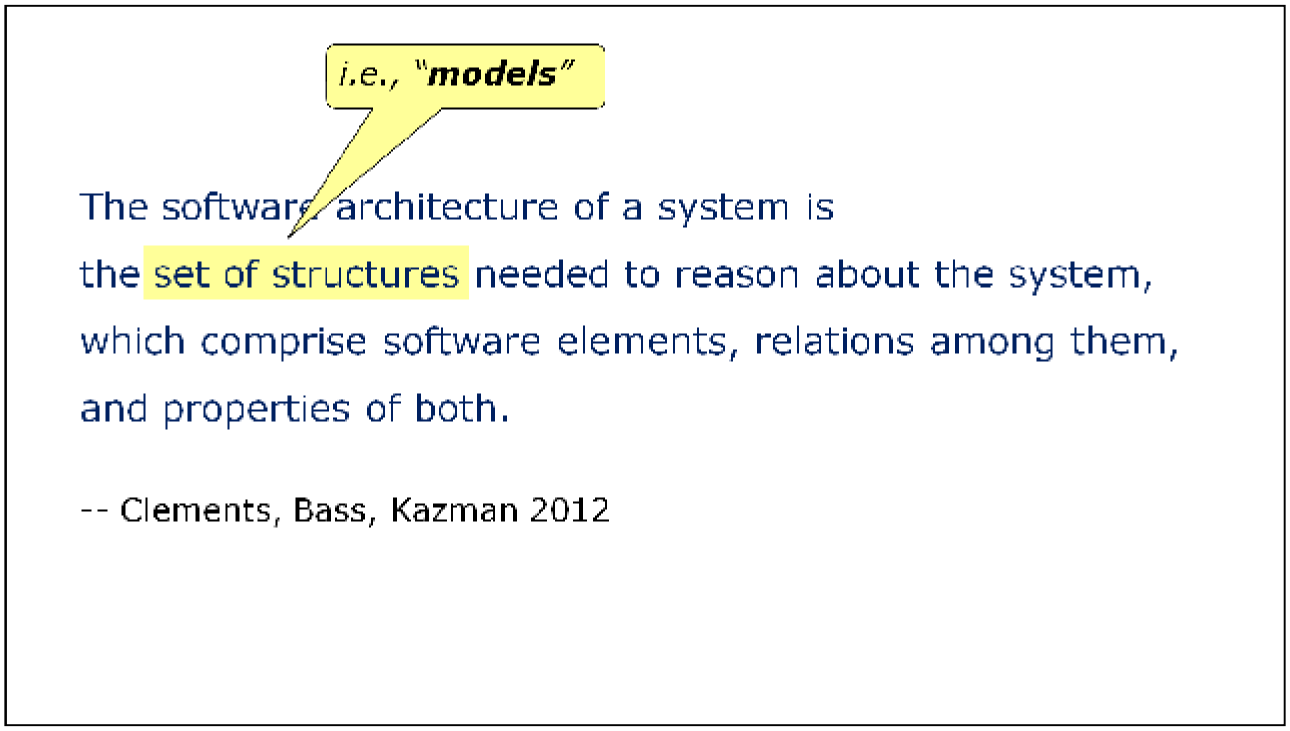
These details are not architectural because they do not help you reason about possible failures.


The software architecture of a system is
the set of structures needed to reason about the system,
which comprise software elements, relations among them,
and properties of both.

-- Clements, Bass, Kazman 2012

Here's the definition of software architecture again and this time it should make a lot more sense.

"The software architecture of a system is the set of structures needed to reason about the system, which comprise software elements, relations among them, and properties of both."



*i.e., "**models**"*

The software architecture of a system is the set of structures needed to reason about the system, which comprise software elements, relations among them, and properties of both.

-- Clements, Bass, Kazman 2012

"set of structures" just means **models**, models you use to reason about the system

What is

Software Architecture

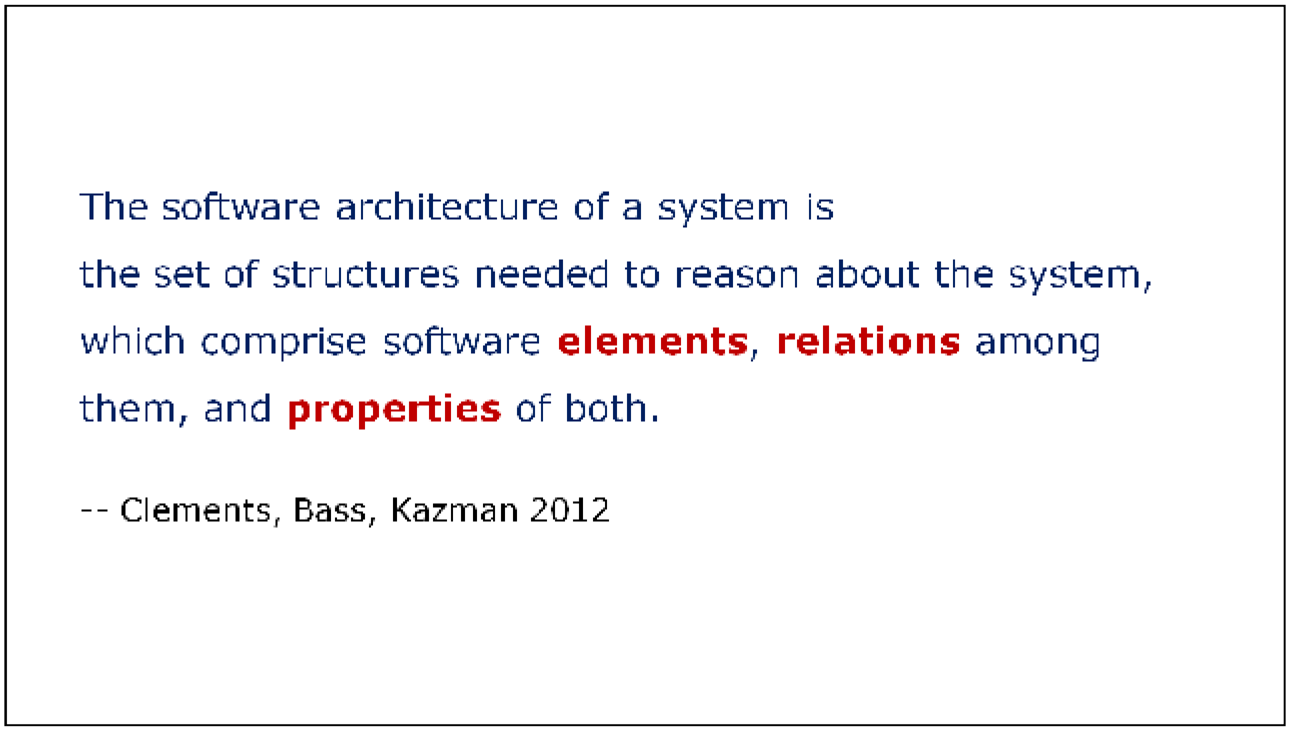
and how does it relate to

Software Design

?

We started out with a simple question -- what is software architecture and how does it relate to design?

To answer it, we have taken a tour through **functionality** and **qualities**, avoiding engineering **failures**, and reasoning using **models**.

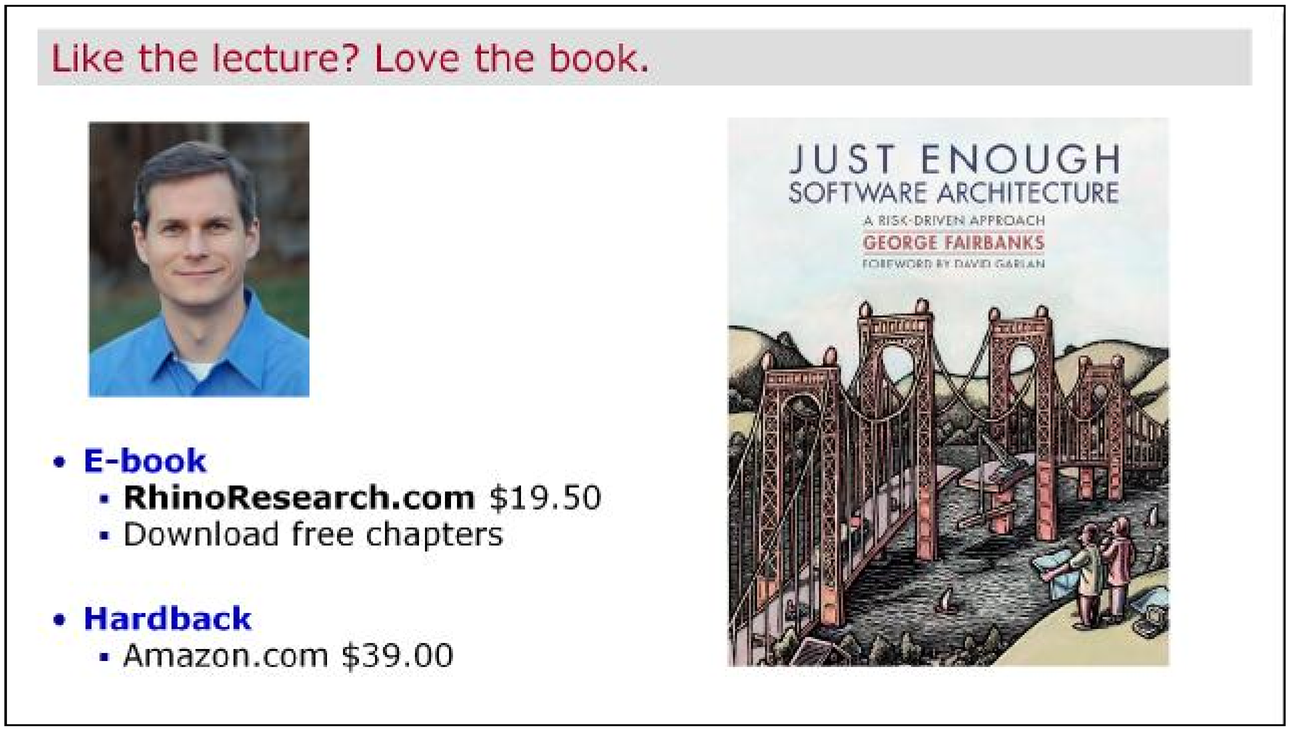

The software architecture of a system is
the set of structures needed to reason about the system,
which comprise software **elements**, **relations** among
them, and **properties** of both.

-- Clements, Bass, Kazman 2012

Still, there's much more to learn about software architecture.

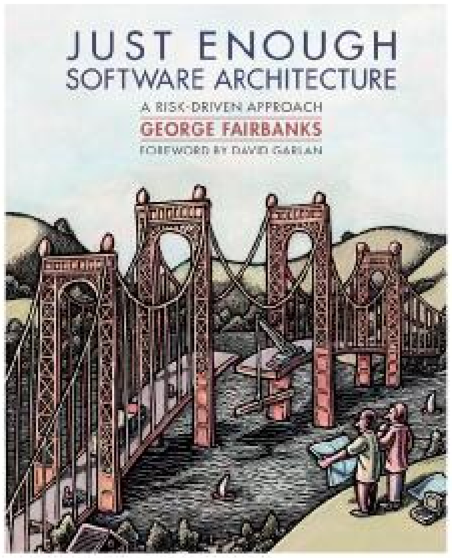
We have not yet discussed parts of the definition: software **elements**, **relations**, and **properties**.

So check out our other videos to learn more.



Like the lecture? Love the book.

- **E-book**
  - **RhinoResearch.com** $19.50
  - Download free chapters
- **Hardback**
  - Amazon.com $39.00



If you found this lecture interesting, take a look at the book. Several chapters are available for free download.