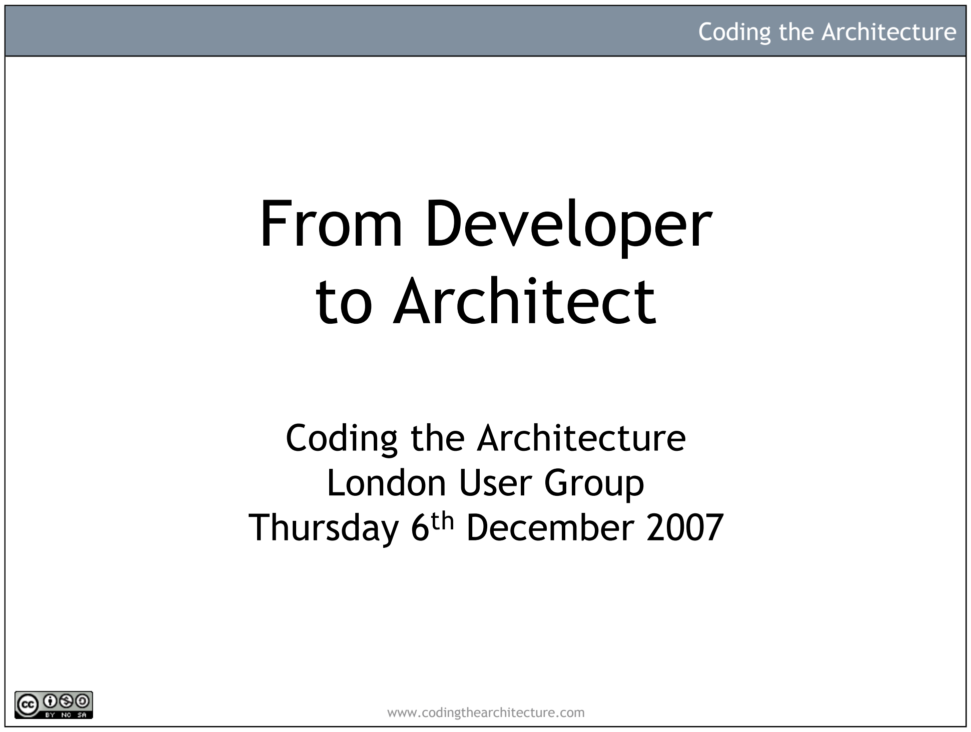


From Developer to Architect

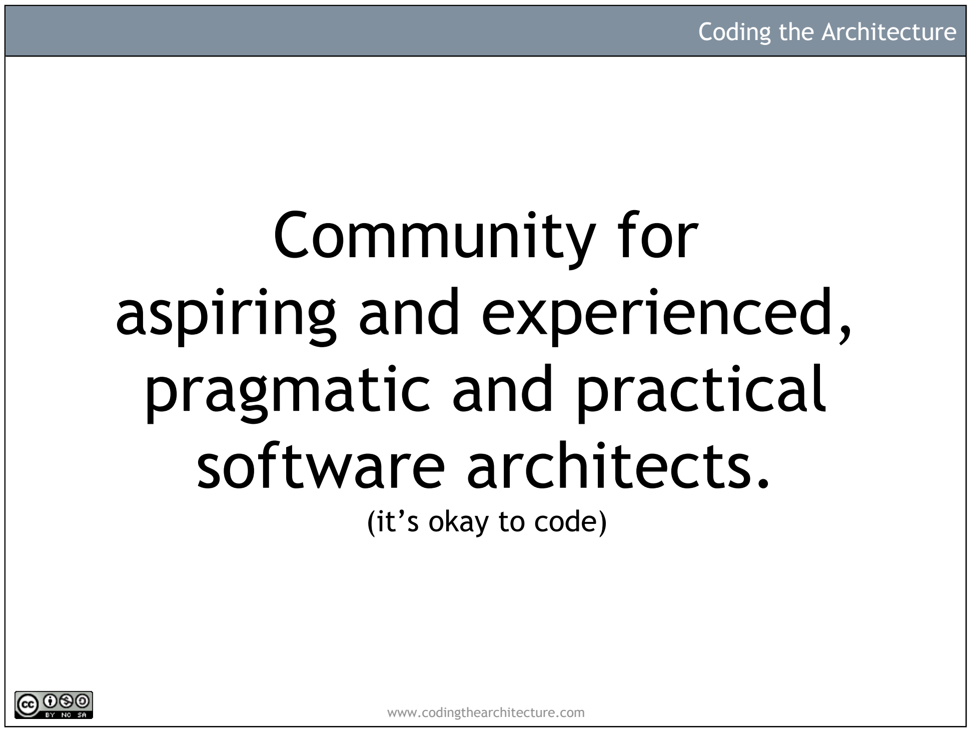
Coding the Architecture
London User Group
Thursday 6th December 2007

Tonight's topic is "from developer to architect" where we'll be exploring some of the differences between the roles and discussing how you might move into your first architect role.

We have 5 speakers lined up this evening, who are all going to talk for ~5 minutes about their own experiences of transitioning into an architecture role.

We'll have the usual 5 minute break, then come back together for discussion.

We'll end with the draw for the QCon London 2008 conference pass and the four books.



Community for aspiring and experienced, pragmatic and practical software architects.

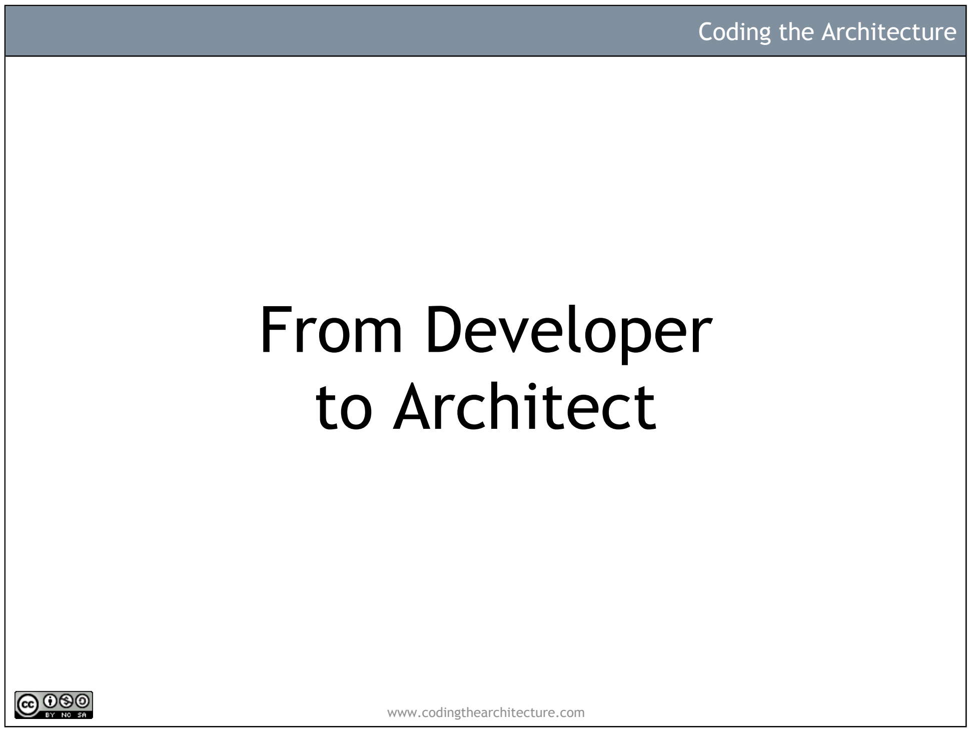
(it’s okay to code)

The purpose of Coding the Architecture is to build a community and share information that is relevant to aspiring and experience, pragmatic and practical software architects.

Architects that sit in ivory towers and mandate solutions through the exchange of PowerPoint decks are giving us a bad name … and we want to change this. It *is* okay to code!

Website : majority of the content on the site revolves around the role of the software architect and reflects our own real-world experiences, good and bad.

LUG : software architecture isn’t necessarily hard, but it can seem a little mysterious at times, particularly if you’re an aspiring architect. We want to help you and build a physical community. Monthly meetings with a good mix of presentation and discussion.



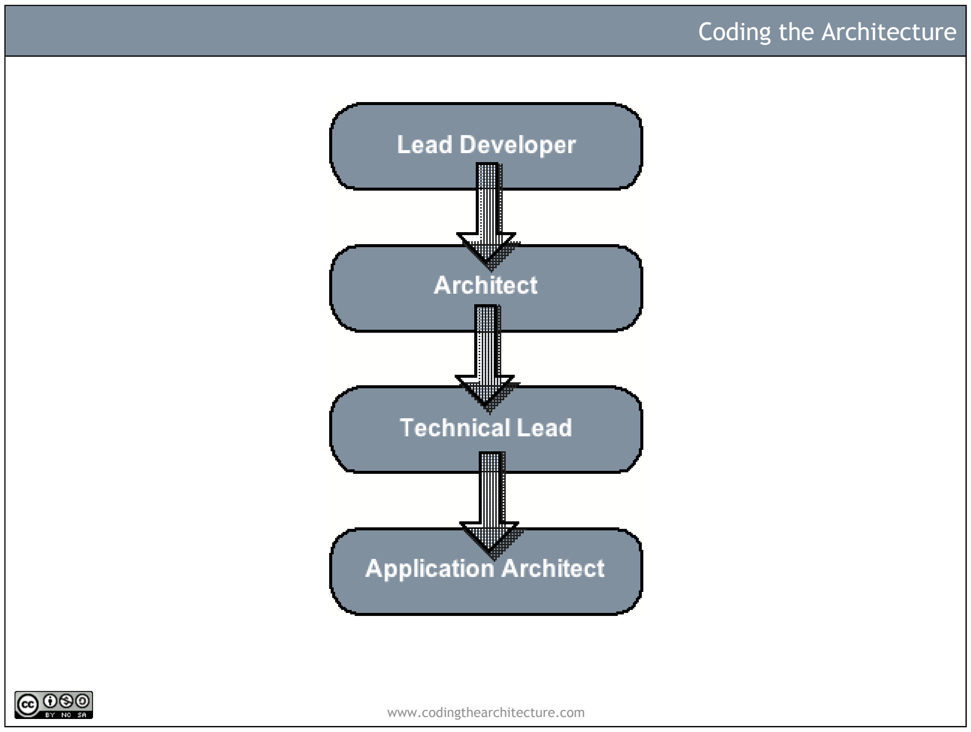
Coding the Architecture

From Developer to Architect

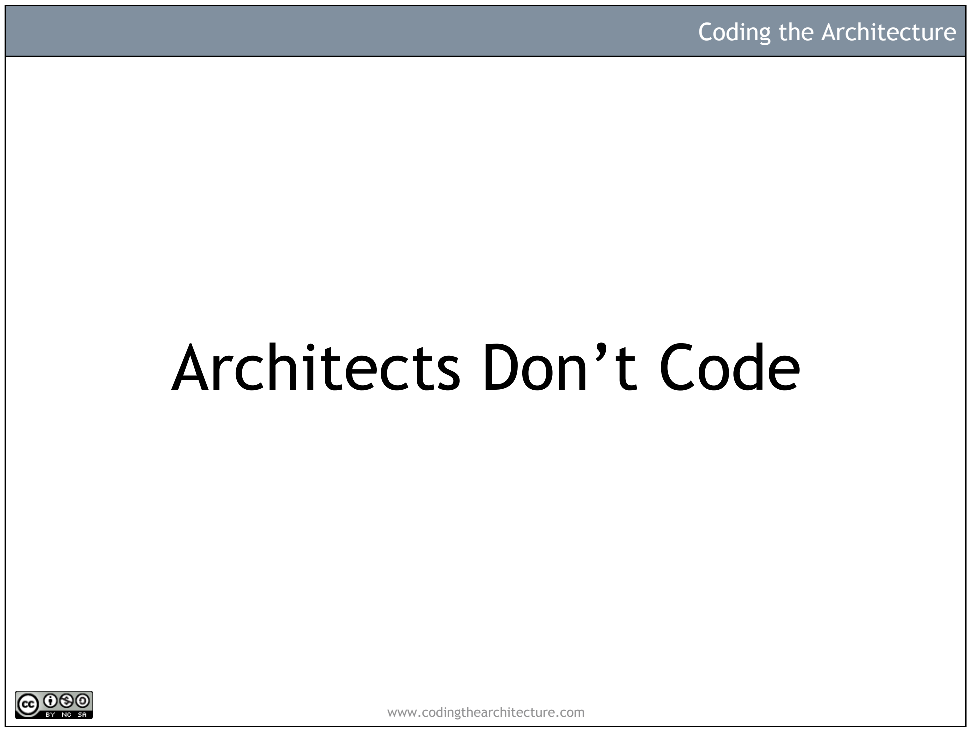
www.codingthearchitecture.com

So let’s move on to the presentation with the first of our speakers.

Kevin Seal



Labelled an architect well before I would (now) consider myself an architect.

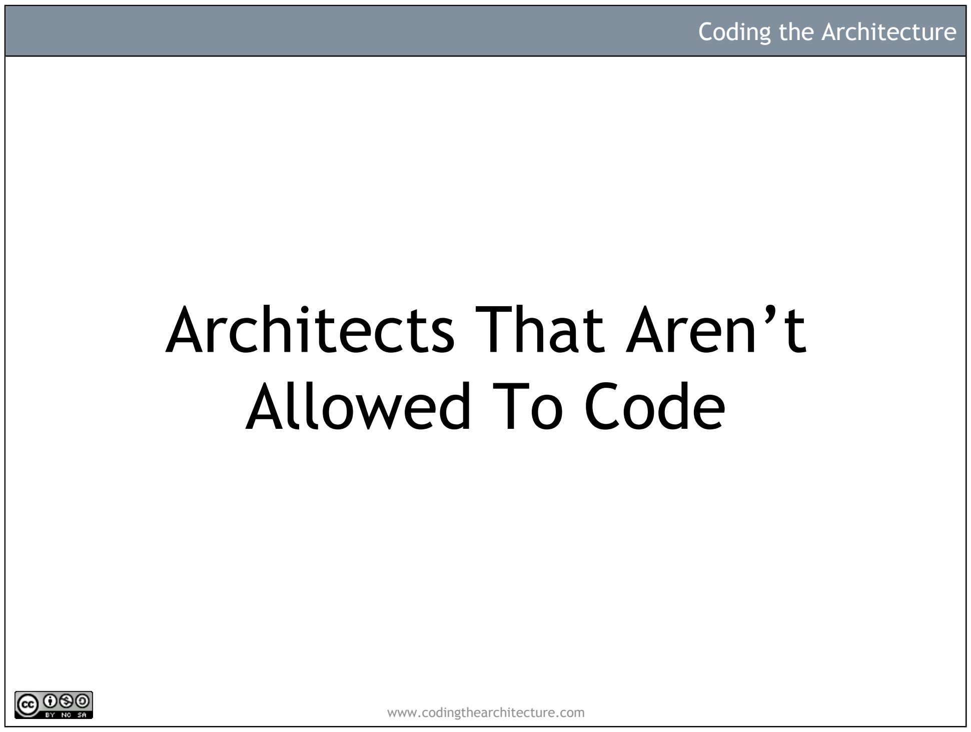


Architects Don’t Code

The “ArchitectsDontCode” anti-pattern rails against architects that don’t, can’t or won’t code.

Based on the various horror stories on forums this is probably how a lot of developers view architects.

What I encountered was more like…



Architects That Aren’t Allowed To Code

Another anti-pattern in my opinion, although possibly not as serious as the previous.

This is probably my abiding memory of transitioning from developer to architect.

Of course I was expected to design the software and discuss this with the team.

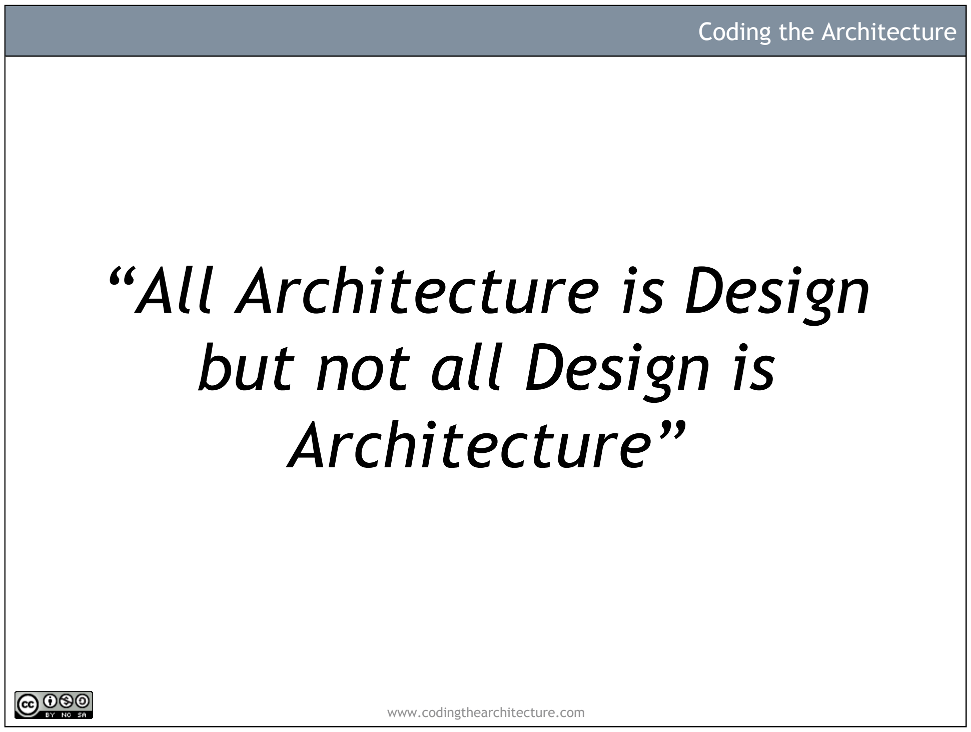
I was naturally expected to review the implementation against the design and standards.

What I wasn’t able to do was to get involved in the implementation. (geography, contract, time, cost)

But for me this constraint really helped distinguish between being a lead developer and being an architect. I actually had to consider my decisions and to determine which ones were worth fighting for.

In the past I’d have simply corrected something I didn’t like myself. Now I either had to convince someone to change it or simply let it go.

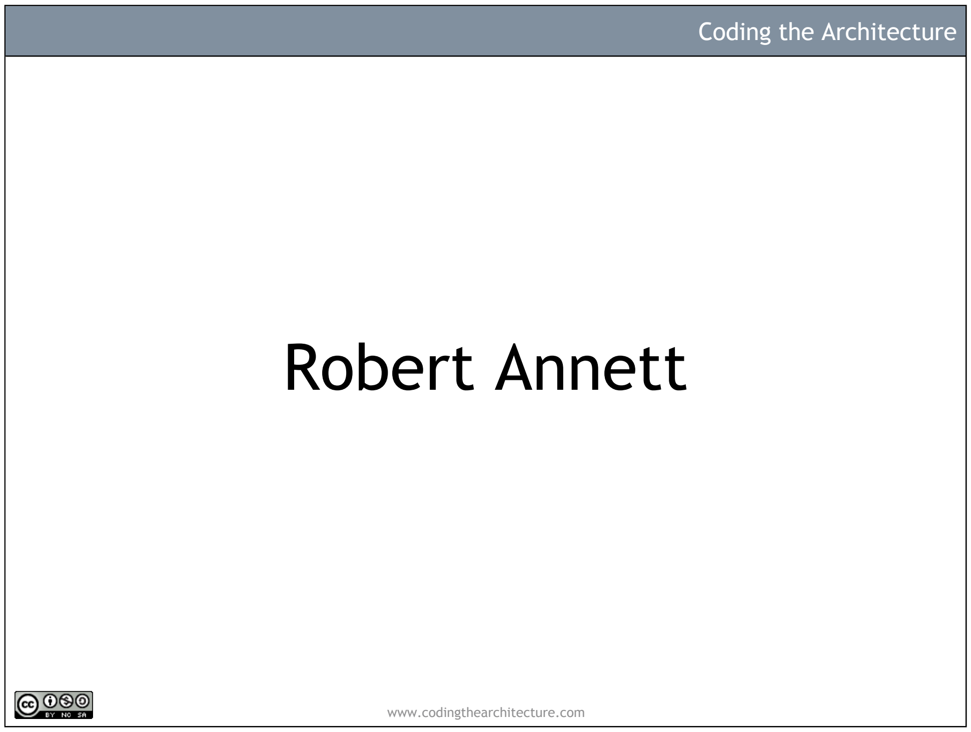
This constraint also highlighted the role I was expected to perform. My employers didn’t want a lead developer – they wanted someone to ensure a project’s success – and that entailed more than code. Arguably, the code and its design was one of the more minor parts of the system.



"All Architecture is Design but not all Design is Architecture"

My advice for transitioning from developer to architect is probably contained in this Grady Booch quote.

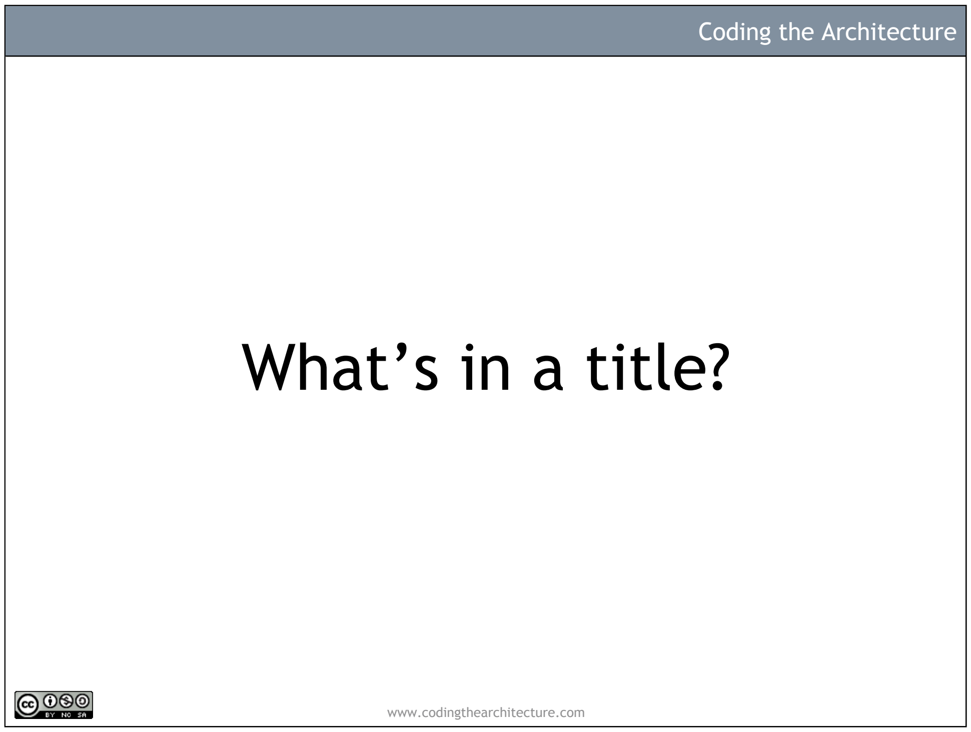
At some point the design decisions you're making become significant insofar as their cost of change is prohibitive; that's when you're defining the architecture.



Coding the Architecture

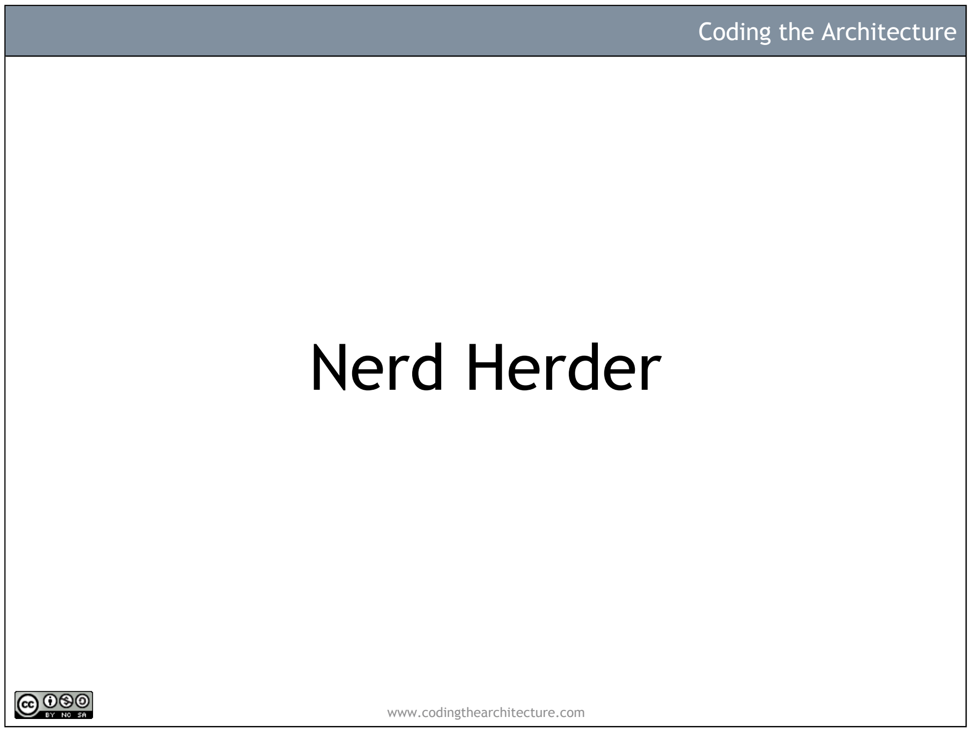
Robert Annett

www.codingthearchitecture.com



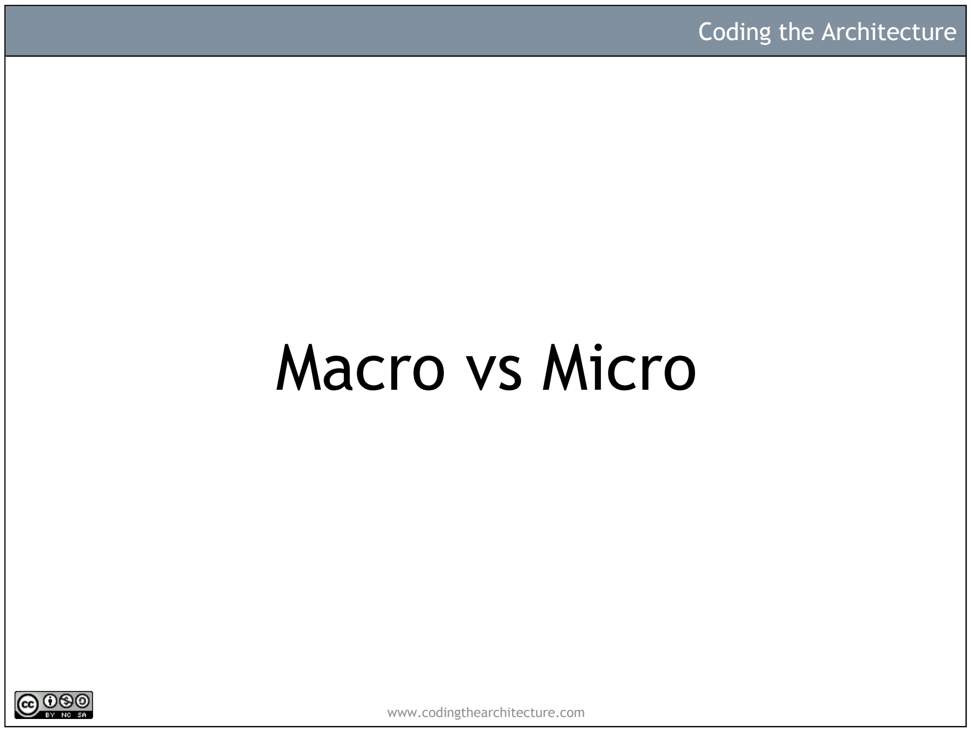
What’s in a title?

One of the posters on CTA recently said that he was tired of the discussion about titles as they weren’t important. He was right that titles aren’t important but the roles we perform are.



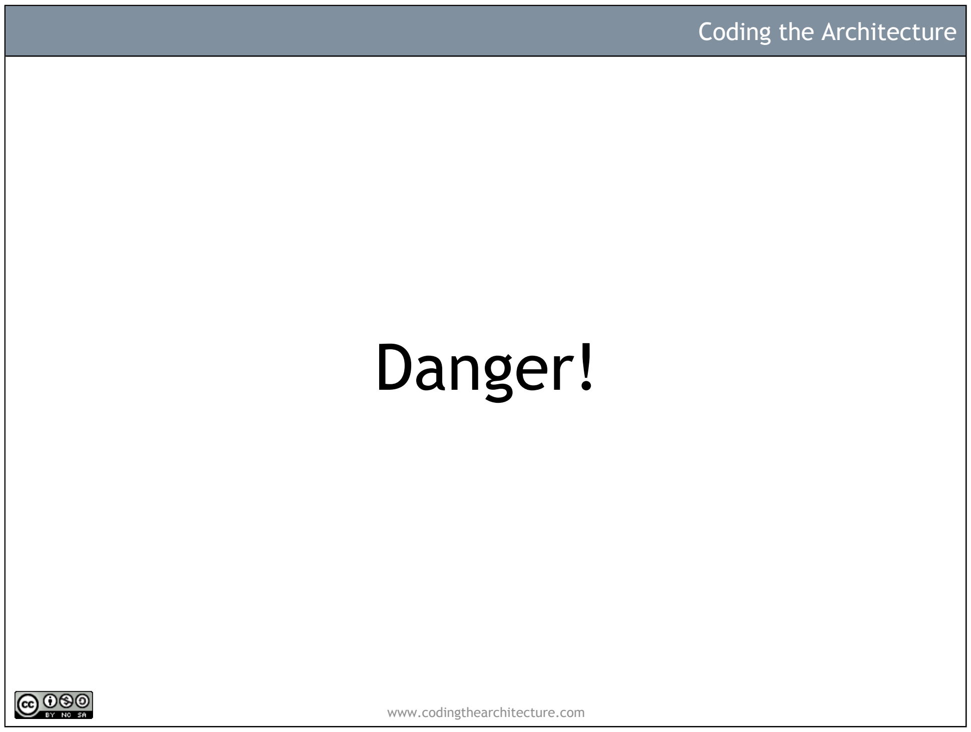
Nerd Herder

Whatever title you're given (including this one) there is a very good chance that you'll perform a number of roles. Architect is a role more than a title. Importantly…



Macro vs Micro

As a developer with management and design responsibilities (multiple roles are certainly true or SME) you will be pulled in many directions. You may have development management, project management and architectural responsibilities. The difference is that the architectural role is macro. The architect needs to consider the whole of the system and how it interacts with the outside world.



Coding the Architecture

Danger!

www.codingthearchitecture.com

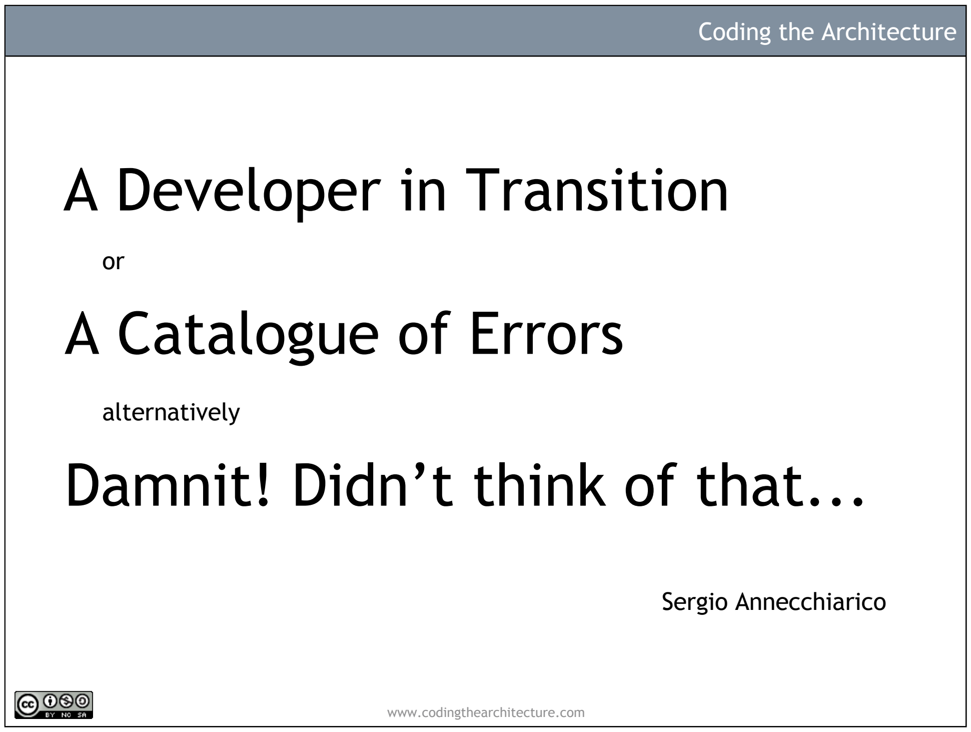
There is a problem with combining these roles - especially as they are very different. The difficulty I faced is that the tasks you need to perform as development manager tend to take over and push out the architecture tasks. Architecture tasks can be vague and hard to define - where to start and when you have finished whereas many of the development manager tasks aren't. They also come thick and fast - there is aways a code review, QA report to read or test coverage to examine. It becomes very easy to ignore your architecture role and responsibility.

Solution?

You have to be aware of the (potential) problem and make sure you give yourself the time for architecture. The simplest way I found to do this was to work from home one day a week. My day at home (largely in front of visio) was my architecture day. I was not constantly interrupted by questions or demands on my time. Simple but effective.

I'm sure that one of the main reason that architectures suffer is the other demands made on the person who should be performing the architecture role - they always complete one more 'micro' task.

Sergio Annecchiarico



A Developer in Transition

or

A Catalogue of Errors

alternatively

Damnit! Didn’t think of that...

Sergio Annecchiarico

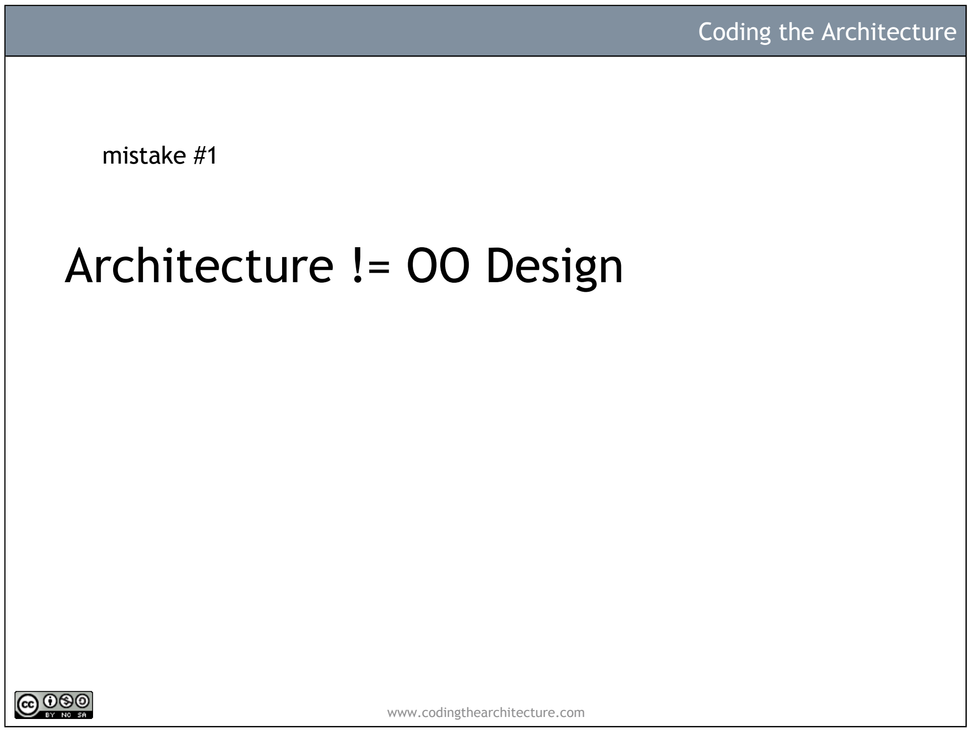
Introduction

Background: developer, consultant, lead developer, general typical think-I-know-it-all consultant

New role: Technical Architect – mixed feelings about taking the role. Enjoy writing code. Technical Architect is a tainted term. Actually, was slightly persuaded by forums like this one to give it a go.

What could possibly go wrong.

Well, here is my list of what has gone wrong in the last few months.



mistake #1

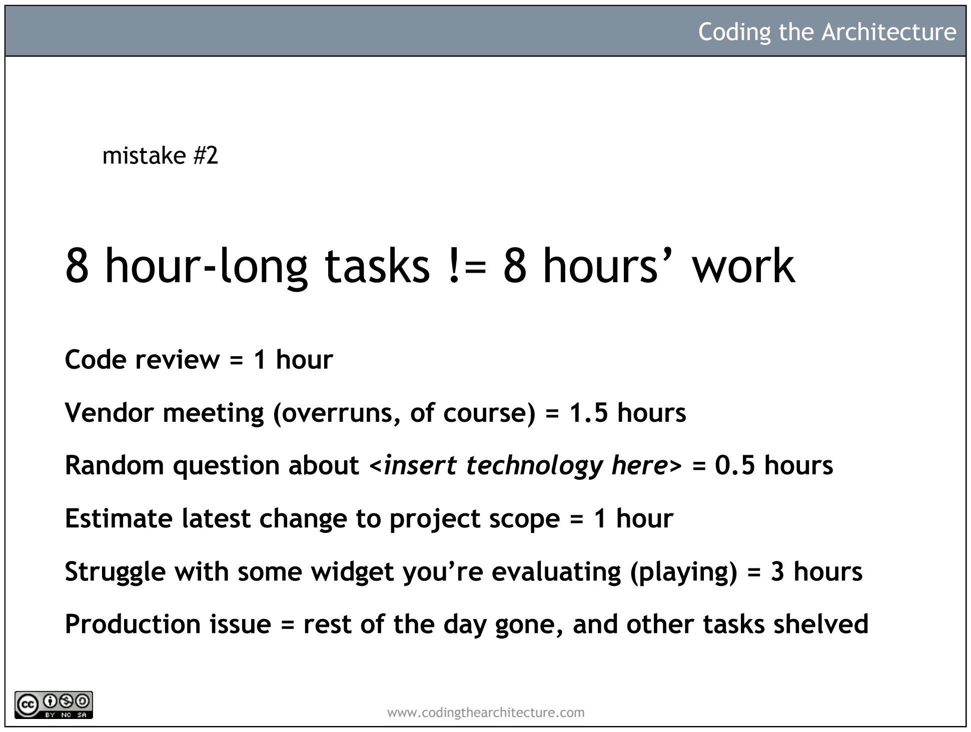
Architecture != OO Design

I naively expected to be churning out a lot of OO designs

UML– class diagrams, statecharts, activity diagrams, and so on - lots of these in fact.

In reality this is a very small part of the package as you are with the dev team most closely, and they probably need this the least.

There is some OO design, but on my client, I have actually spent more time just knocking up a prototype, or doing an example as a reference for the dev team



mistake #2

8 hour-long tasks != 8 hours' work

Code review = 1 hour

Vendor meeting (overruns, of course) = 1.5 hours

Random question about <*insert technology here*> = 0.5 hours

Estimate latest change to project scope = 1 hour

Struggle with some widget you're evaluating (playing) = 3 hours

Production issue = rest of the day gone, and other tasks shelved

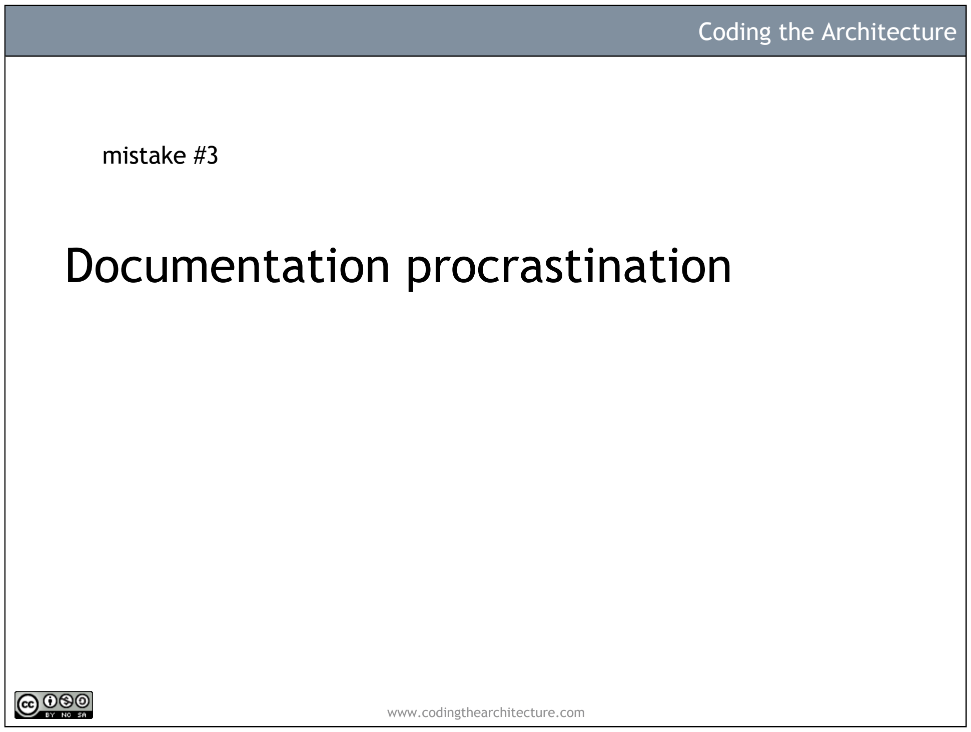
Sounds kind of obvious. This has been the single hardest learning curve.

Time management for an architect is different from a developer. Tasks are shorter, less well defined. Much more isolated.

Stuff comes at you from all angles.

Deep technical problems require my brain to go into some kind of deeper concentration mode. Architects are very visible. Always introduced to new starters. Always asking other people questions. That means you are a sitting target for being interrupted and losing your train of thought.

You really learn quickly what is important that it gets done, and what can be left.



mistake #3

Documentation procrastination

Learned this lesson pretty quick.

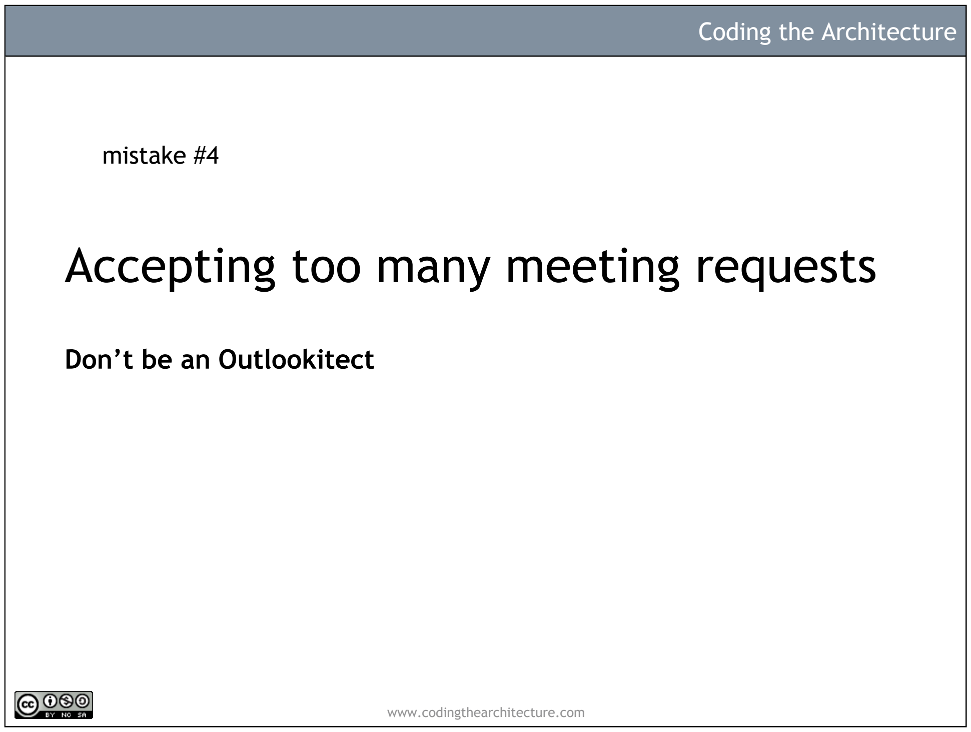
Agile development project = just in time documentation (= no need for documentation…)

For the architecture document, the latest possible moment is probably much sooner than you think at first.

Architecture – need documentation. You really only need one document, but it needs to be factually accurate, complete, and designed to be viewed by lots of different groups:

Dev team (although, not as much as I thought)
Testers
Project Management
Infrastructure
Support
Suppliers
Customers

Next lesson is not finishing the document. Need to get it updated regularly. Living document. Some things may just not be known (prod box IP addresses, whether you're using sendmail or postfix, even which version of libraries are going to be used)



mistake #4

Accepting too many meeting requests

Don't be an Outlookitect

People love to have meetings.

Some people take pride in how packed their calendar is.

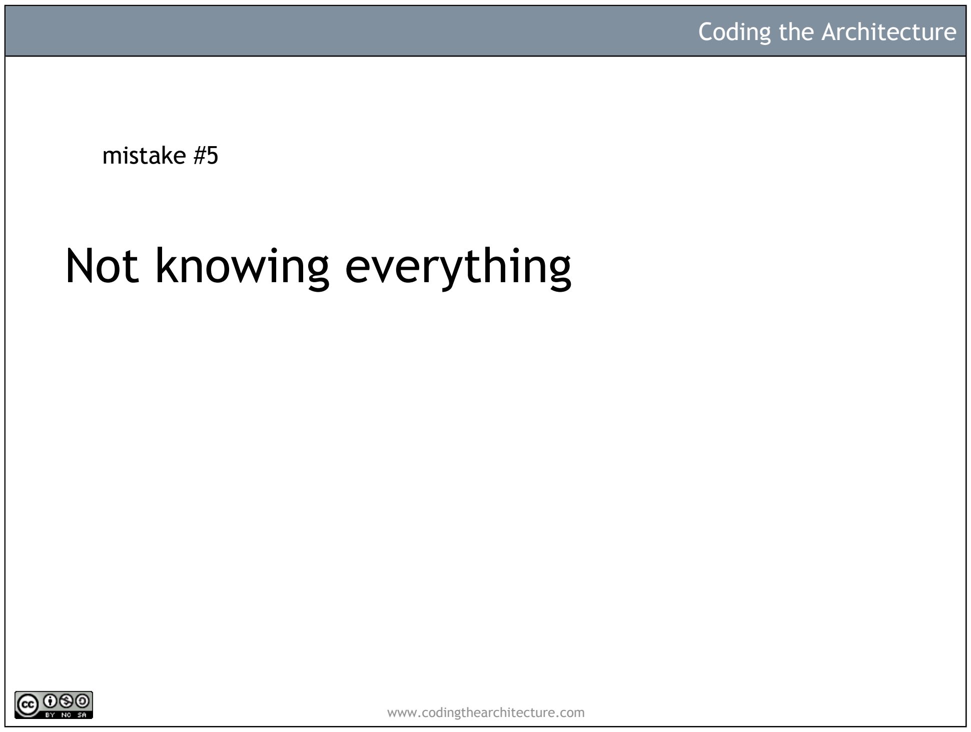
Architect job is to see that a project is done.

To get a project done, you need to be available to the development team at the coal face more than any other group in the organisation.

Don't block out your whole day in meetings. Make sure the first meeting of the day is with the dev team – doesn't have to be a formal scrum thing, just a chat is fine.

Make sure you aren't just going to meetings because that is what top brass does. For each and every one ensure that you really are needed.

Beware meetings with more than 5 people (or to be honest, more than 2 people) and are scheduled for more than an hour. Take your iPod.



mistake #5

Not knowing everything

The customer expects me as Technical Architect to be expert in everything.

Everything.

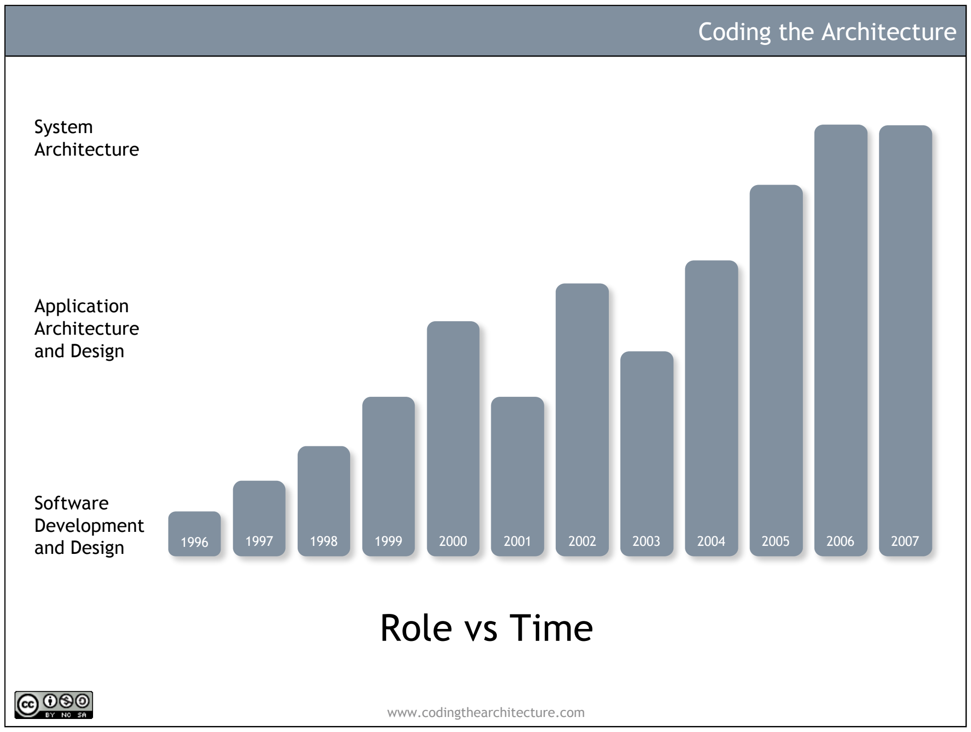
They do not want to hear
“sorry, it has been a while since I looked at EJB clustering on JBoss”, or
“I don’t know how many CPUs an Oracle Se1 licence permits” or
“nope, I’ve never actually tried load testing with WebLoad”, or
“is it ftps or sftp that is more secure?”

They want an authorative answer. You’re a trusted resource and all round technical guru or eveyrthing. EVERYTHING. So I have had to learn how to answer this sort of question with an authorative “I will let you know by lunchtime”.

Sam Dalton

The truths and mistruths of being an architect.

Simon Brown



Since I started my IT career, I've had a very evolutionary path to architecture.

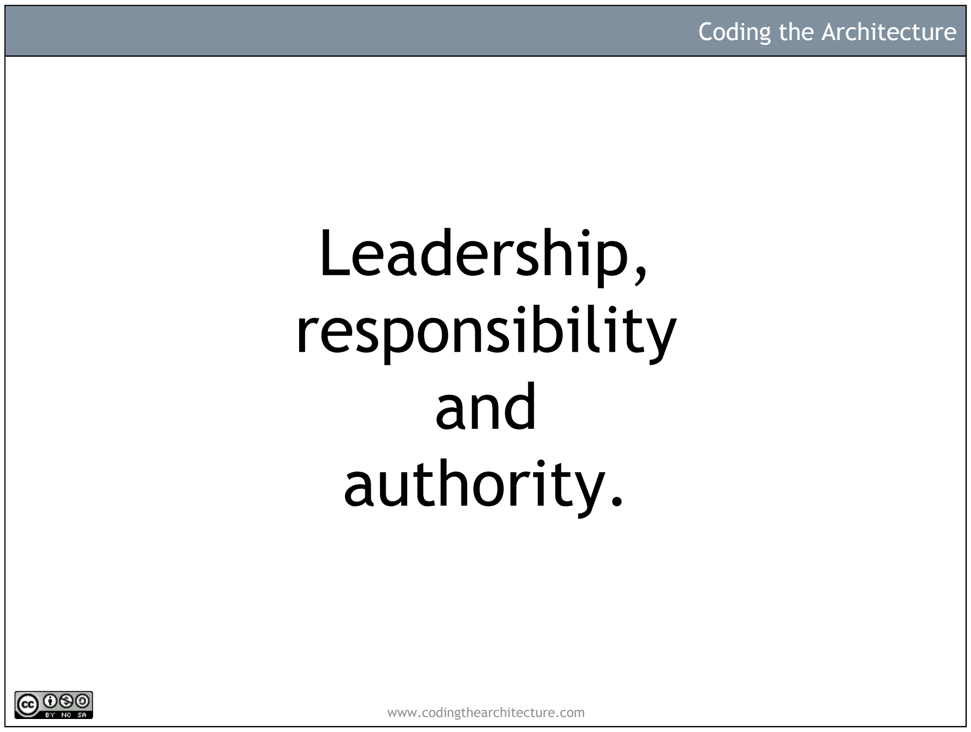
This diagram shows that I didn't suddenly wake up one morning and become an architect.

It wasn't always an upwards path though – depending on available roles, etc.

Development : 1996 onwards

Application architecture : 2000 onwards

System architecture : 2005 onwards



Leadership, responsibility and authority.

For me, the key differences between 1996 and 2007 : leadership, responsibility and authority.

5 minute break.

Discussion

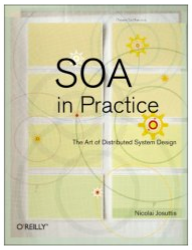
What are other people's experiences of the transition?
What do *you* think the key differences between the developer and architect roles are?
Do you want to become an "architect"?
How do you think you'll move into an architect role?

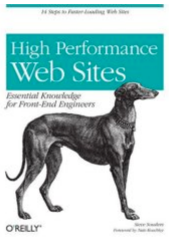
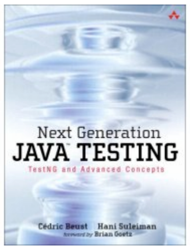
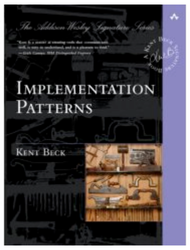
Draw

A 3 day conference pass to QCon London, March 2008

QCon

and some books




High Performance Web Sites
Essential Knowledge for Front-End Engineers
O'REILLY

Next Generation JAVA TESTING
TestNG and Advanced Concepts
Cédric Beust Hani Suleiman


Thanks to our sponsors and see you next year.

SKILLS MATTER

Member of the O'REILLY® User Group Program

PEARSON Education

Website

http://www.codingthearchitecture.com

Google Group

http://groups.google.com/group/codingthearchitecture

Please feel free to continue the discussion on the Google Group.