coding {the} architecture

Are you a software architect?

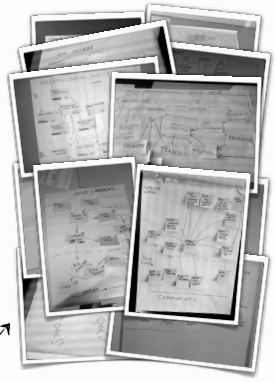
Simon Brown

Jersey, Channel Islands

simon.brown@codingthearchitecture.com

@simonbrown on Twitter

Writing



Training and consulting

Published incrementally

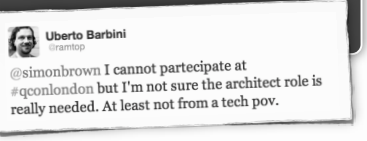
Variable pricing

Buy now and get free updates

Leanpub

leanpub.com

Perceptions



A conflict in team structure

Dedicated software architect

Single point of responsibility for the technical aspects of the software project

vs

Everybody is a software architect

Joint responsibility for the technical aspects of the software project

Big up front design
and analysis paralysis

Waterfall

UML

I'm a
software
architect

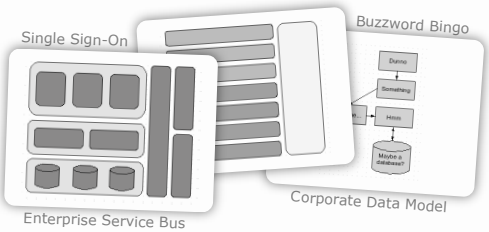
Ivory Tower

PowerPoint Architect

Architecture Astronaut

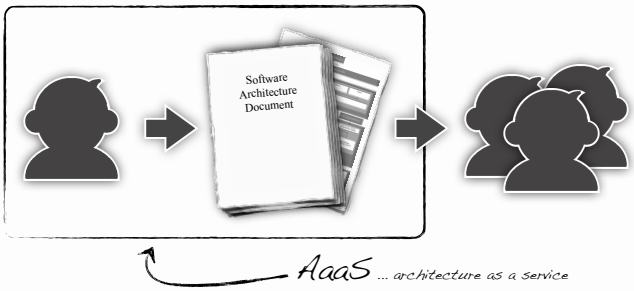
PowerPoint

Architecture

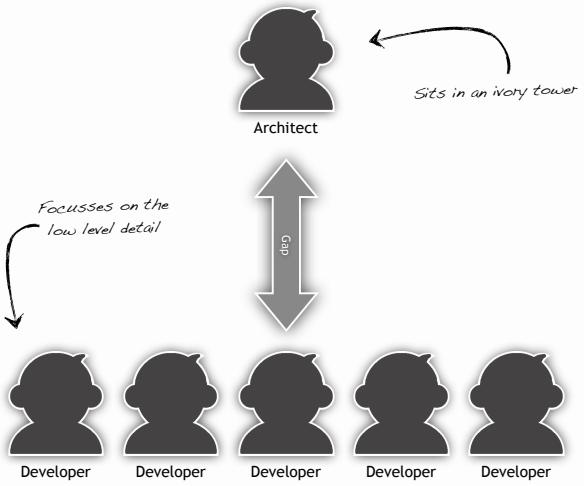


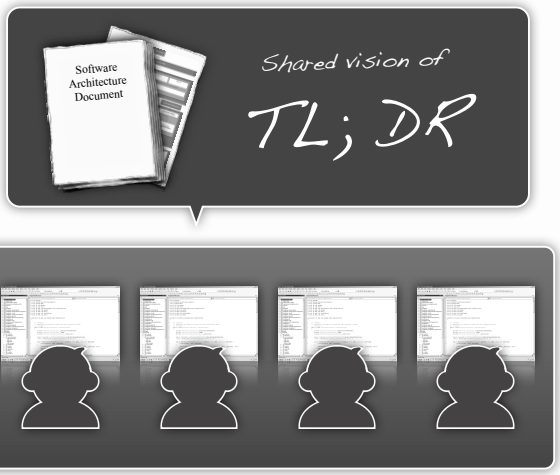
Software development is not a

relay sport



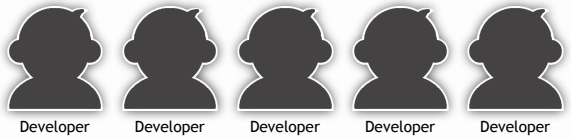
AaaS ... architecture as a service





Architects?

We don't need no
stinkin' architects!



Small teams of generalising specialists, everybody does everything

With agile, there is often a perception that you must have self-organising teams

"we don't need an architect, we have smart developers"
(we only hire the best)

"you must follow what our central architecture team dictates"
(the ones in the ivory tower)

#fail

"let's get an architect in for the first few weeks"
(they are too expensive and they don't code)

Buzzword bullsh*t

"Architects are business technology strategists."

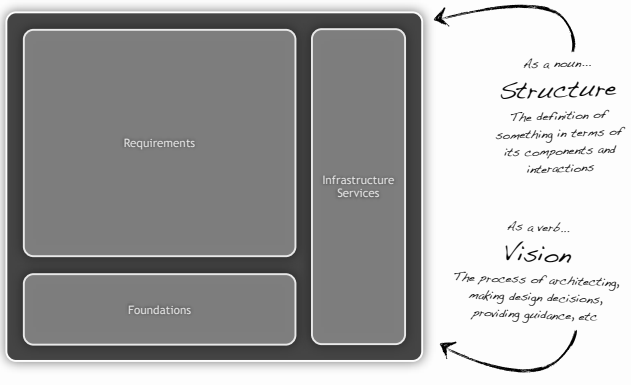
"Microsoft SharePoint has a solid ROI and TCO."

"What's the value proposition of MongoDB? Show me your business case."

True or not, this stuff doesn't make much sense to your typical software developer
#IJustWriteSoftware

What is software architecture?

What is architecture?



As a noun...
Structure
The definition of something in terms of its components and interactions

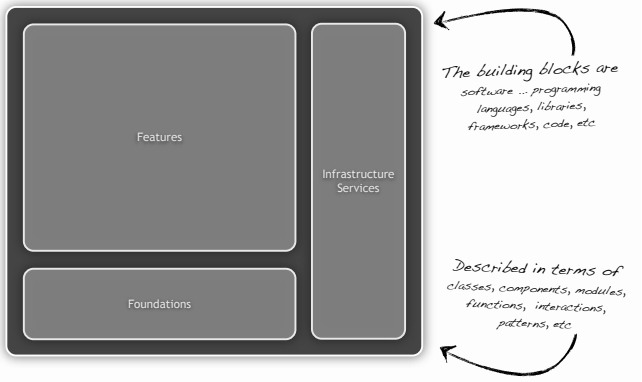
As a verb...
Vision
The process of architecting, making design decisions, providing guidance, etc

Types of architecture

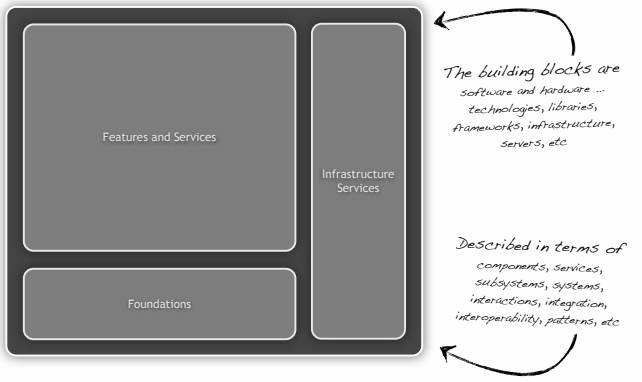
Application
Business
Data
Enterprise
Hardware
Information
Infrastructure
Network
Platform
Performance
Security
Software
Solution
System
Technical
Technology
Web
...

There are lots!

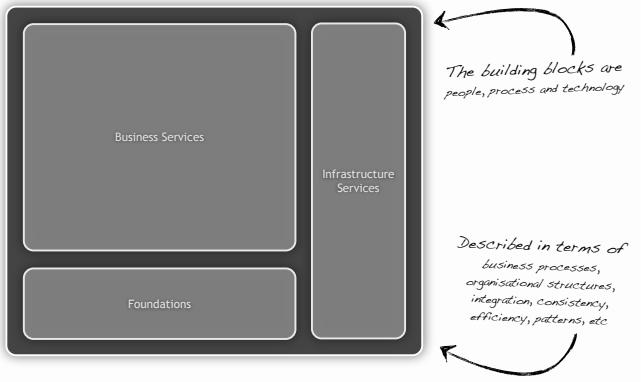
Application architecture

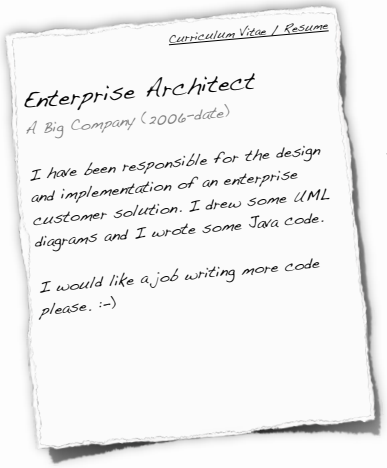


System architecture

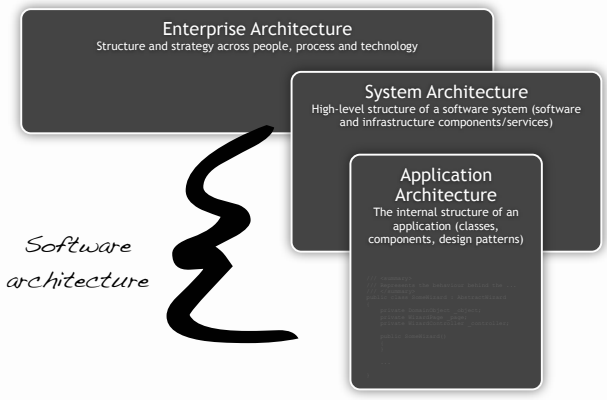


Enterprise architecture

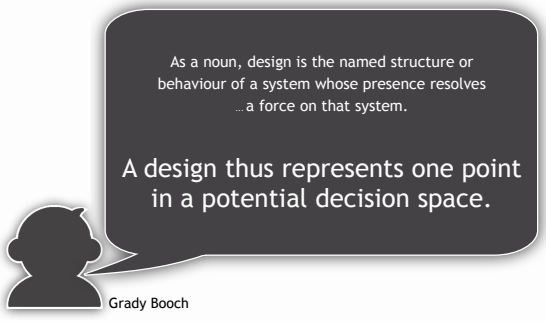




Err, no; you're a software architect who just happens to work in a large organisation



What is design?



Grady Booch

What is design?

All architecture is design, but not all design is architecture.

Grady Booch

What is design?

Architecture represents the **significant decisions**, where significance is measured by **cost of change**.

Grady Booch

Can you **refactor** it in an afternoon?

Architecture is the deliberate and considered resolution of

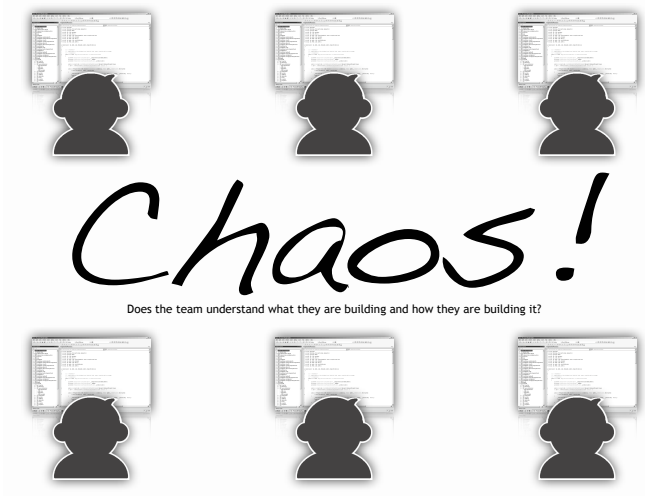
significant

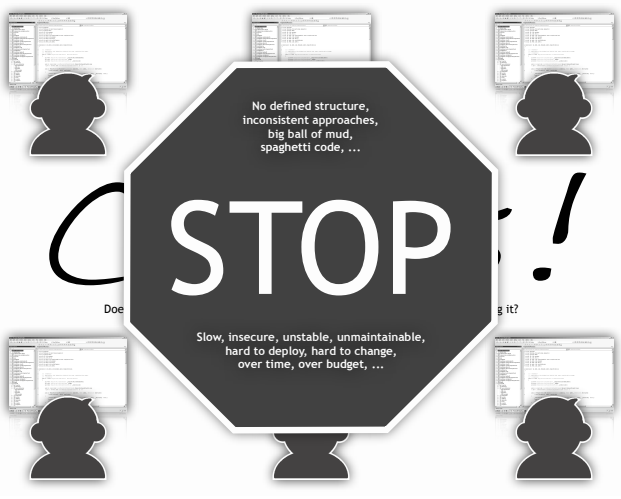
problems

Is software architecture

important?

Why? Benefits? Applicable to all software projects?



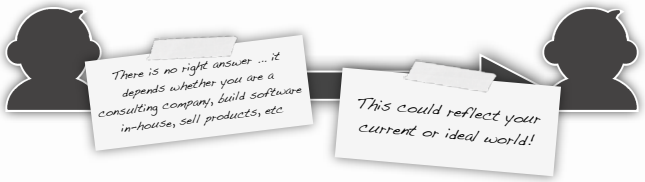


Software architecture introduces
structure and vision
into a software system, leading to
consistency and clarity

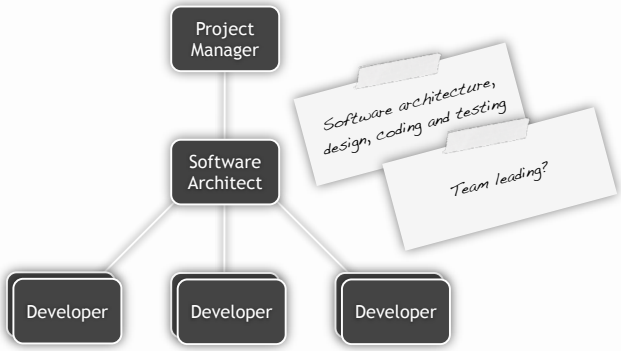
The role of a software architect

How you would organise a software team comprising of 8-10 people?

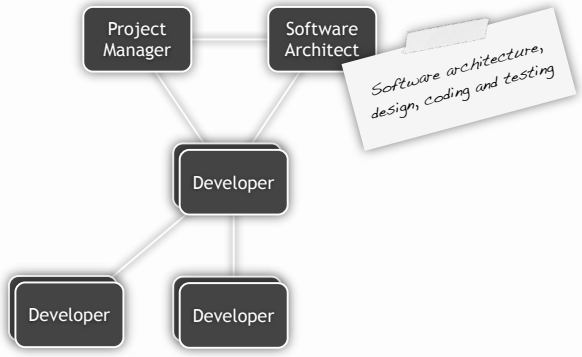
What do they all do?



Architect as team leader



Architect and manager as peers



Flat, self-organising structure

The software architect role is

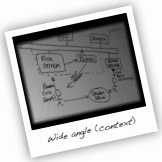
different

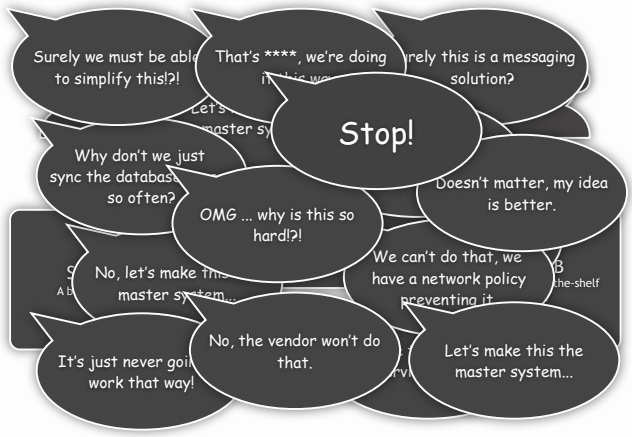
to that of a lead developer

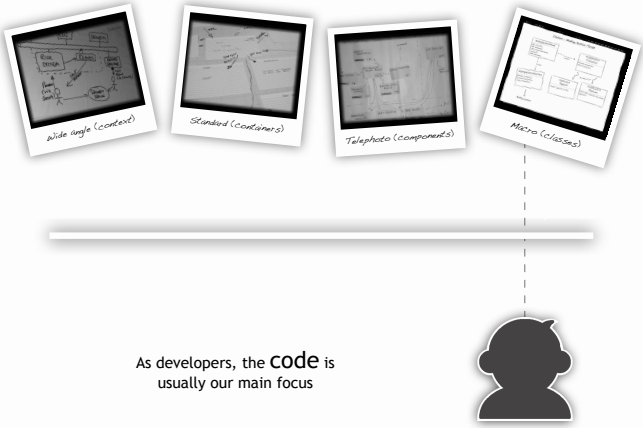
What information does the architect need to deliver the right solution?

It's about the

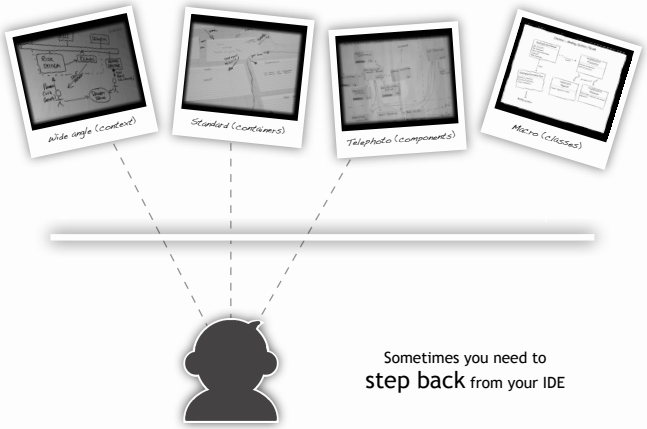
big picture



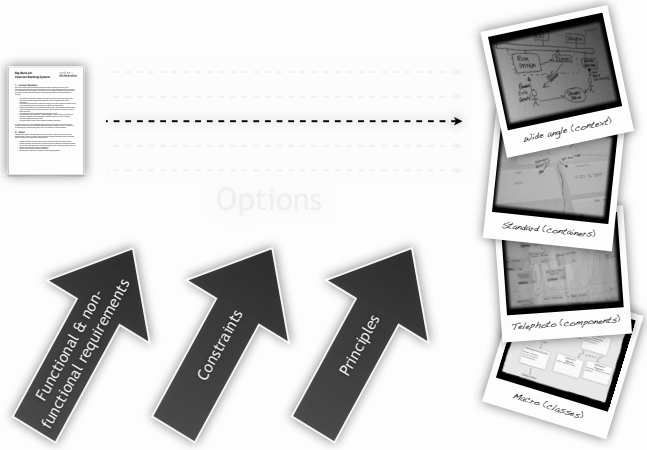




As developers, the **code** is usually our main focus



Sometimes you need to **step back** from your IDE

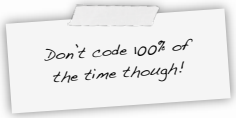


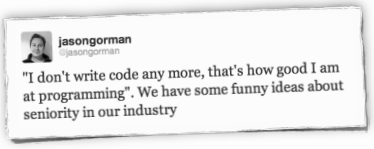
Learn about and understand

the **big** picture

The **low-level detail**

is equally important





Would you **hire** a

software architect

that **wouldn't code**?

Should a software architect

code?

Would you code it that way?

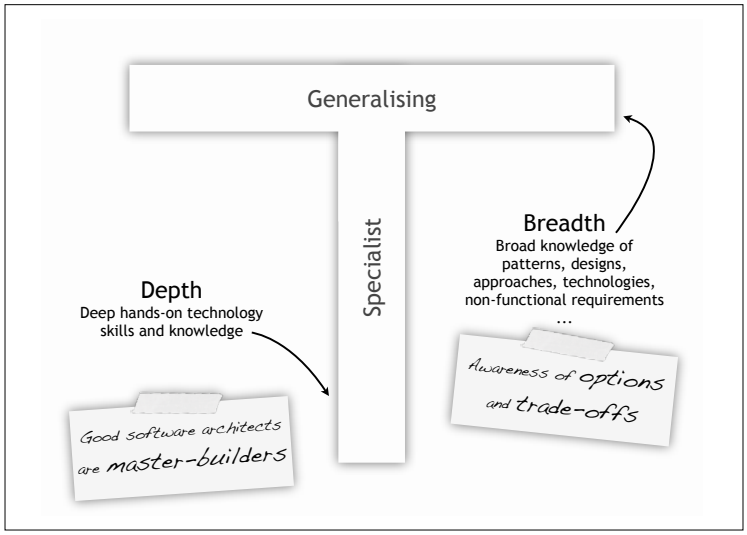
Will it work?

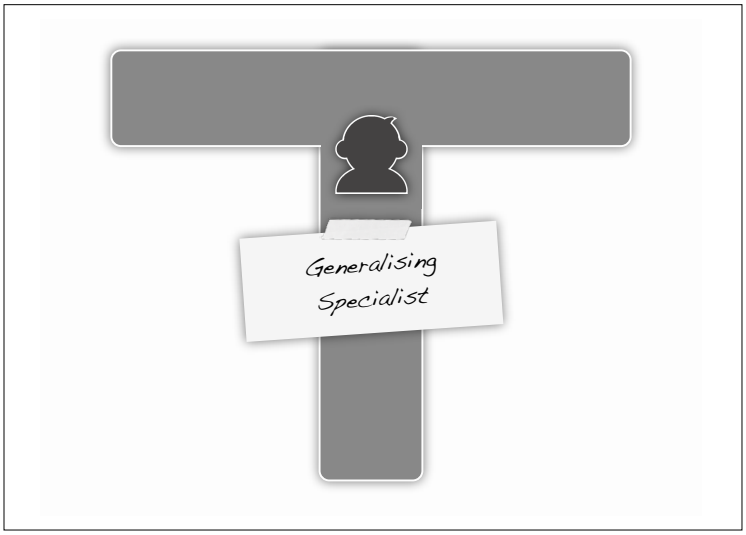
Top 10 traits of a

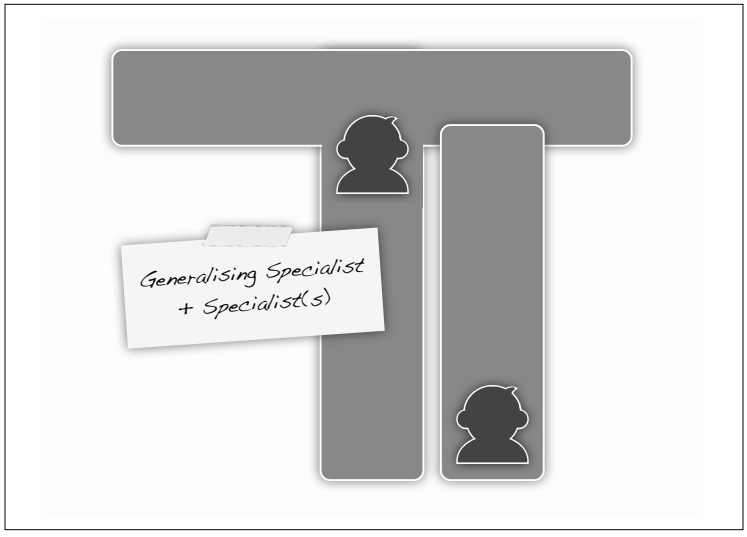
rockstar software engineer

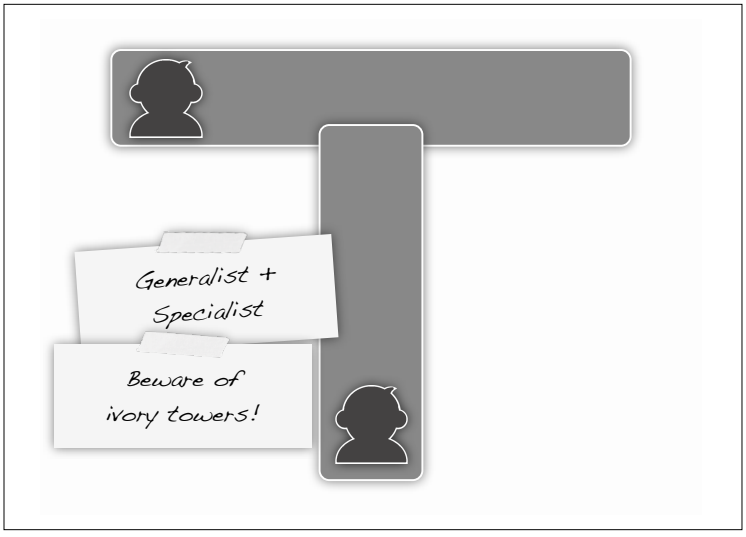
http://www.readwriteweb.com/archives/top_10_software_engineer_traits.php

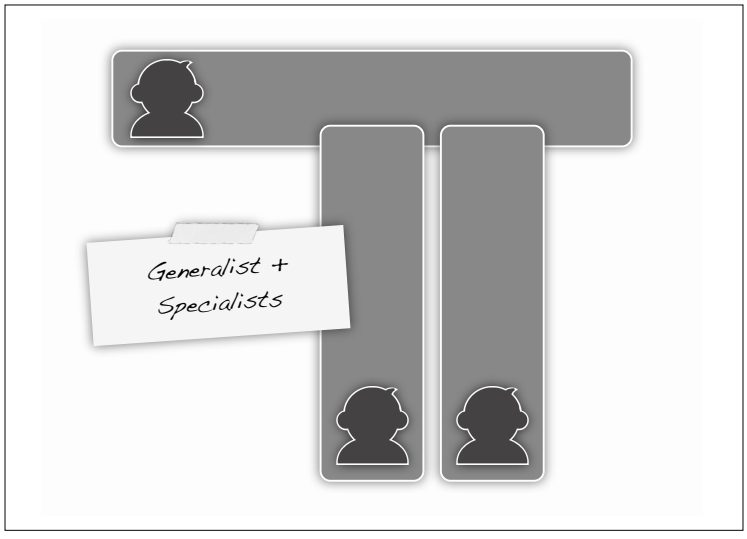
1. Loves to code
2. Gets things done
3. Continuously refactors code
4. Uses design patterns
5. Writes unit tests
6. Leverages existing code
7. Focuses on usability
8. Writes maintainable code
9. Can code in any language
10. Knows basic computer science
11. Understands why


Generalising
Specialist
Breadth
Broad knowledge of patterns, designs, approaches, technologies, non-functional requirements
...
Depth
Deep hands-on technology skills and knowledge
Awareness of options and trade-offs
Good software architects are master-builders

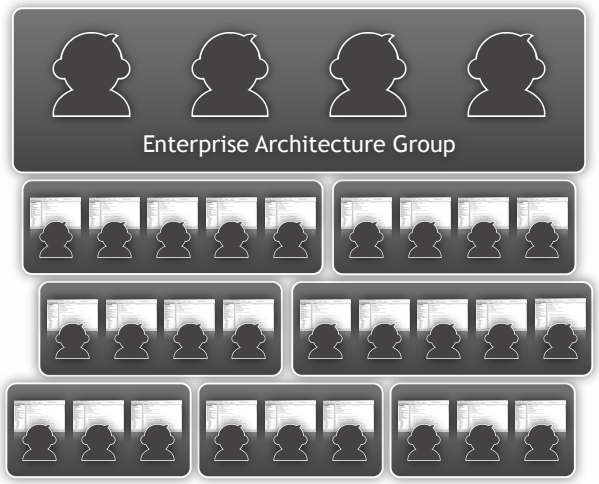

Generalising Specialist


Generalising Specialist + Specialist(s)


Generalist + Specialist
Beware of ivory towers!


Generalist + Specialists


How can I *possibly* be hands-on when I have
seven teams to look after?!
Software Architect
You can't ... but why are you looking after seven teams?



Every software development team needs a master builder

Software architecture is about

technical

and

soft skills

What sort of **soft skills** does a software architect need?

Software Architect

Leadership

Communication

Influencing

Negotiation

Collaboration

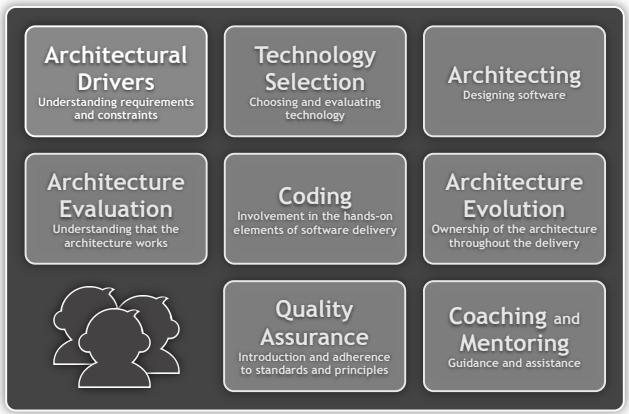
Coaching and Mentoring

Motivation

Facilitation

Political

The software architecture role



“Software Architect”

is **not** an

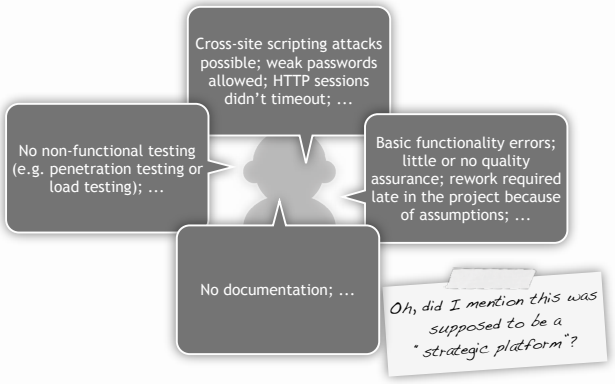
organisational rank

It's a role that you ***evolve*** *into*

Are you a software architect?

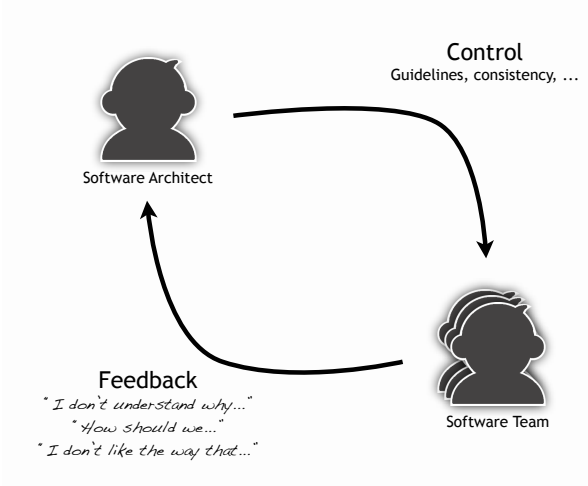
Experience is a good gauge,
but you need to look deeper

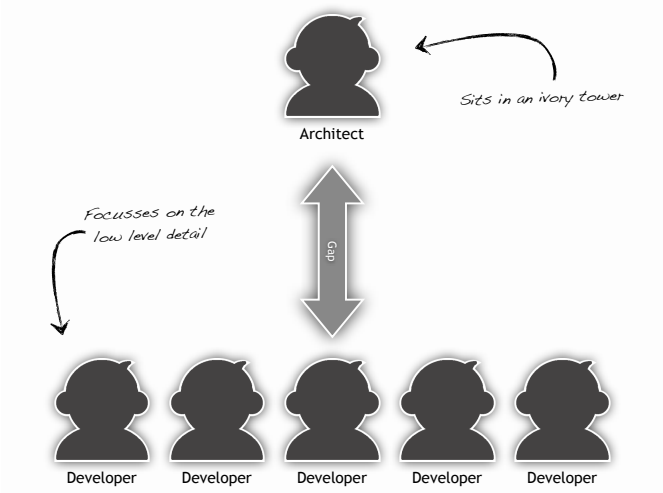
The irresponsible architect

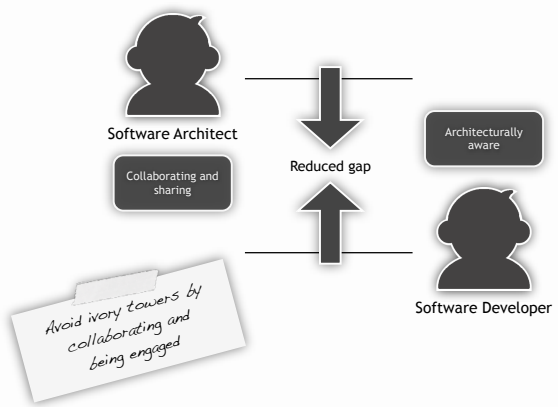


Software architecture introduces

control?







Most importantly,
rockstar engineers believe in
simplicity and
common sense

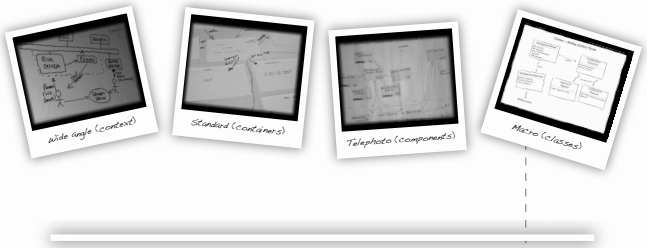
The software architecture role is different to that of a lead developer; requiring a
breadth and **depth** of technical skills
along with good **soft skills**

What **pitfalls**
should new software architects watch out for?

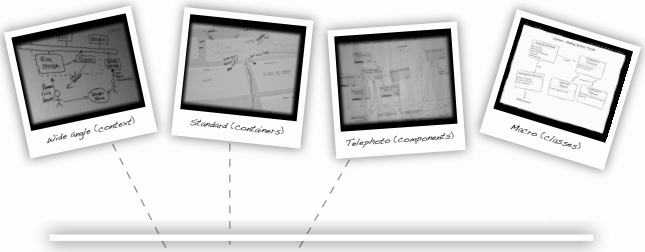
How would you
define
the software architecture role in your team?

In summary

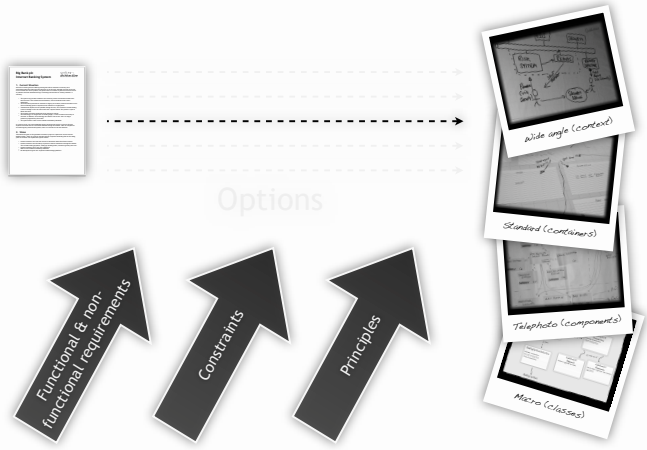
When should a software architect be involved in a software project and **what** should they do?



As developers, the **code** is usually our main focus



Sometimes you need to **step back** from your IDE



The software architecture **role** and the **process** of software architecting are

different

The software architecture role

Dedicated software architect

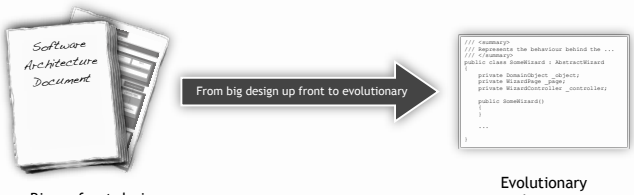
Single point of responsibility for the technical aspects of the software project

Elastic Leadership (Roy Osherove)
Chaos (command and control),
learning (coaching),
self-organising (facilitation)

Everybody is a software architect

Joint responsibility for the technical aspects of the software project

The process of software architecting



Big up front design

Requirements capture, analysis and design complete before coding starts

Evolutionary architecture

The architecture evolves secondary to the value created by early regular releases of working software

Define

the software architecture role and

collaborate

Be pragmatic

rather than dogmatic

Do you want to write

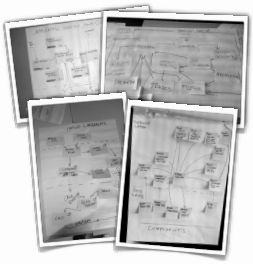
code

as a software architect?

Apply *craftsmanship*

to

software architecture



Effective yet lightweight Sketches and documentation

Software systems that actually work

Drawing diagrams

doesn't make you a software architect!

Would you

code

it that way?

This is why software architects must be able to code!

Be **proactive**

and take the **lead**

Take ownership and lead by example

We need to **grow** the

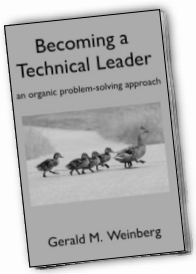
software architects

of **tomorrow**

Becoming a Technical Leader

An organic problem-solving approach

Gerald M. Weinberg



Some excellent advice on leading teams without being a dictator (i.e. ivory tower architect!)

A software architecture manifesto :-)

simon.brown@codingthearchitecture.com
@simonbrown on Twitter

1. Some up-front architecture over doing the whole lot or none at all
2. Designing for non-functional requirements over trying to tack them on at the end of the project
3. Evaluating technology over just choosing it because it looks good on your CV/resume
4. Experimentation over hoping your design will work
5. Code over endless discussions, meetings and diagrams
6. Collaboration over ivory tower dictation
7. Involvement over running away and letting the team deal with the "implementation" problems
8. Coaching team members over stroking your own ego
9. Pragmatism over perfection
10. Real problems over intellectual (self) stimulation
11. Simplicity and common sense over complex and convoluted ideas
12. Short and useful architecture documentation that reflects reality over encyclopaedias that nobody reads, understands or cares about