coding {the} architecture

Sharing Architectures

Kevin Seal

www.codingthearchitecture.com

TheBerglasEffect anti-pattern

www.codingthearchitecture.com

There are several anti-patterns associated with architecture and management that relate to the sharing of architecture:
 * ArchitectureByImplication: the lack of an architecture for whatever reason
 * ThrowItOverTheWall: the disconnect between the information producer and consumer
 * ProjectMismanagement: covering a multitude of sins, of which communication is one

Let's start with a made-up anti-pattern to describe the symptoms and causes associated with failure to share the architecture.
This gives us the basis for ensuring our approach to sharing our architectures is suited to the problem as a whole and the particular causes or symptoms we're experiencing.

Incidentally, David Berglas was a famous mentalist - think mid-20th century Derren Brown.

Indicators

- "That's not what I had in mind"
  - "You've got the wrong end of the stick"
  - "It's not how I would have done it..."
  - "Hmm, interesting..."
- "It was in the architecture document"
  - "It was on the wiki"
  - "I'm sure I told everybody this already"
- "I'll do it myself"
  - "I'll pair with you"
  - "... and you ..."

Let's look at the symptoms of not sharing our architecture.
How do we know we're approaching an "I'm not a mind reader" moment between the architect and the development team?

Build In 1 : "That's not what I had in mind"
The clue's in the name here - if you find yourself saying or thinking these things during review then you might even be failing to define a suitable architectural vision or precedent, let alone share it appropriately.
In essence, you're surprised by the implementation - and not in a good way.

Build In 2 : "It was in the architecture document"
It's no use pointing people to the architecture after they've started design or implementation. While the architecture may well have been defined, and possibly even distributed, it may not have been shared effectively.
Here you're frustrated by the questions you're being asked because the answers are part of the defined architecture.

Build In 3 : "I'll do it myself"
Sometimes you can end up doing too much, or spreading yourself too thin. The question is, why do you have to be involved in the development of a particular feature? Where has this implementation risk come from?
Here you might be worried that you're the only person who knows enough about the architectural principles, designs or vision to implement some features.

Consequences

- The software has not been architected
  - High risk
  - Non-strategic
- The team doesn't have any guidance
  - Low consistency
  - Islands of knowledge emerge
  - High friction, low morale
- Code quality drops
  - Build fails
  - Productivity falls

What happens when the architecture's not being shared effectively?

Build In 1
In simple terms, if the architecture's not being shared then it's not being followed and if it's not being followed then it's not guiding the software.
So if you know why you need an architecture you can probably guess the possible consequences of not having one.
Any effort that's gone into the architecture that doesn't make it into the code is little more than documentation.

Build In 2
However, there's more to software implementation than software.
An architecture provides guidance to the team and having no apparent architecture can lead to developers making decisions in isolation. This is not to say that developers can't make these decisions, but they need to be the right ones, with the right information.
Inconsistency in the implementation can lead to islands of knowledge and friction between team members.
An ineffectively shared architecture can also leave the development team feeling disenfranchised or not trusted.

Build In 3
Without architecture and architectural review, code quality will drop (or remain low). Builds fail, noone takes responsibility and suddenly development isn't fun anymore.
Consider that an architecture can include development methodology and various non-functional requirements relating to productivity - not meeting those requirements will obviously impact delivery.

In short, if you're not sharing your architecture then every risk you think you're mitigating requires you implementing it.

Causes

- Poor architect

So what causes the architecture to stay stuck firmly in the elaboration phase?

Who's responsible for the architecture?

Build In 1
Architects must be more than technically skilled
They must engage with the team, be convincing and open.

I've worked on many projects that had good individuals but didn't seem to result in code that reflected the abilities of that team.
In most cases there is a weak architecture evident in the code.
Looking at the archaeology of the code you find how it decayed due to reviews going out the window or the architect not being strong enough to maintain consistency or at least a consistent vision.

Granted this is often the norm because the architect isn't given sufficient authority to push back on the requirements and timings. They are therefore poor by virtue of the environment but nonetheless their effectiveness is severely reduced.

Corrective actions

- Replace/reinforce the architect
  - Demolish the ivory tower
  - Share it amongst the team
  - Give them authority or help
  - Stand up for quality
- Start sharing the architecture!
  - Ensure the architecture's defined
  - Show the architecture to the team
  - Delegate
  - Review

www.codingthearchitecture.com

Build In 1
If the architect is being ineffective then you need to do something about it.
Ineffectual architectures are often hurled down to development teams like a cauldron of boiling oil.

If possible, entice the architect out of their ivory tower - give them help, demonstrate the troubles as an honest concern rather than a whine.
If not, try to go somewhere else for your architecture, such as back to your desk.
For architects, beware being replaced either by coup or by simply becoming non-executive.

Developers and architects alike must stand up for the quality of the code. It's a sign that the architecture isn't defined, isn't sufficient or isn't known - and most teams recognise something's wrong.

Build In 2
Architects must ensure they share the architecture!
Firstly they must have defined the architecture and be sufficiently confident in it to show it to the team with the expectation it will be implemented.

Delegation of the architectural decisions will help get them made in a timely fashion and propagate the history of that decision throughout the team. Hopefully someone in the team will at least stand up for that decision having been involved in it.

Again, back to quality, you must review! Review not only allows you to share the architecture, albeit somewhat after the fact, but also indicates that the team should find out the architecture as part of their implementation process.

Sharing Architectures

www.codingthearchitecture.com

Let’s take a look at some of the methods for sharing our architecture with the development team.

Sharing Architectures

Word Document

www.codingthearchitecture.com

Word seems to be a prevalent form of capturing architectures.
There's certainly nothing wrong with using it for the Software Architecture Document.

It also seems to be a fairly common way of attempting to share the architecture with the rest of the team.
In my experience, as both producer and consumer, of architecture documents the process of trying to propagate the information goes something like this:
 * the architecture document is written
 * it is placed in a project repository
 * a email is sent to the project team
 * someone might look at it

If you're lucky there might be feedback on the document and some of it might be understood.
A process that relies on documentation as its primary means of passing on information, as opposed to capturing it, needs to be quite rigorous to ensure the information is actually absorbed.

Sharing Architectures

PowerPoint Presentations

www.codingthearchitecture.com

Another common approach is to present certain aspects of the software architecture to the relevant stakeholders.
This typically takes the form of the the architect standing next to a big picture of it and talking.

In my experience this tends to be more successful than chucking a document into the development team enclave.
At the very least it shows that the architect exists and is prepared to stand up for, as well as next to, their architecture.

This approach often seems to stop soon after the architecture is complete.
However, the development team need access to this information throughout the implementation so unless ad-hoc sessions can be called it can be successful in only sharing the more obvious design decisions.

Sharing Architectures

CASE Tool

www.codingthearchitecture.com

Fortunately, CASE tools in the vein of Rational Rose seem to have fallen out of fashion.

I have had the dubious privilege of helping define an architecture using Rose and the project proceeded along the following lines:
 * requirements and design meetings were held between the architect and team lead
 * the design was captured in the CASE tool
 * the project was passed to the development team
 * the code was returned for review

At each stage of the process a person fell out of the loop - first the architect, then the team lead, then the developers. And so understanding of the requirements and design experience left the project.
Needless to say, the resulting code was a bit of a disappointment, to put it mildly. It focused on trivial features and shied away from the higher risk items.
Paddling the barrel back up the waterfall was nigh-on impossible.

While this can't all be attributed to the tools, it demonstrates the risks of trying a divide-and-rule approach to design.

Sharing Architectures

Project Wiki

www.codingthearchitecture.com

Good old wikis; they're everywhere.

Wikis are a good, lightweight means of capturing information, preferring on content above style.

However, it's quite difficult to ask a question on some projects without being told, "it's on the wiki" or "when you find out, put it on the wiki." Wiki seems to be a member of some project teams.

Although perhaps not intentional, this "find out for yourself" approach can be very productive, encouraging the developers to find, capture or create whatever designs they require.

Be careful that the wiki doesn't become a surrogate for the architect - for starters, it's not very good at changing with the requirements.

Sharing Architectures

Code Review

www.codingthearchitecture.com

Code review should form part of any development process, even when you trust all the developers.

Many of the projects I've worked on have used code review to ensure the architecture has been followed.
When they haven't, the developer is informed of how they should have implemented things - it's a post-hoc sharing of the architecture.
It also requires the architect to find the design in the code and root around for the things that don't adhere to the expected design.

Obviously this isn't ideal but in many cases I suspect this is the main mechanism by which the architect is actively involved in promoting their architectural vision.

More frequent code reviews can mitigate this, especially if they're brought forward in the development process or combined with...

Sharing Architectures

Pair Programming

www.codingthearchitecture.com

...pair programming.
This provides an almost inline code review mechanism.
However, it does require that one of the developers in the pair understands the architecture sufficiently to implement the prescribed design so it's not a replacement for evangelising the architectural vision.

The architect often takes the role of one of the pair to propagate their design.
I've found this approach very effective in small teams but it's naturally restricted by the architect's time and particularly by geographical constraints.

I've also found this can be quite disenfranchising for the development team if done wrongly. It can appear that the team isn't trusted to implement anything without the architect looking over their shoulder.

Again, the key is in how the architect treats the development team.

Sharing Architectures

Delegate

www.codingthearchitecture.com

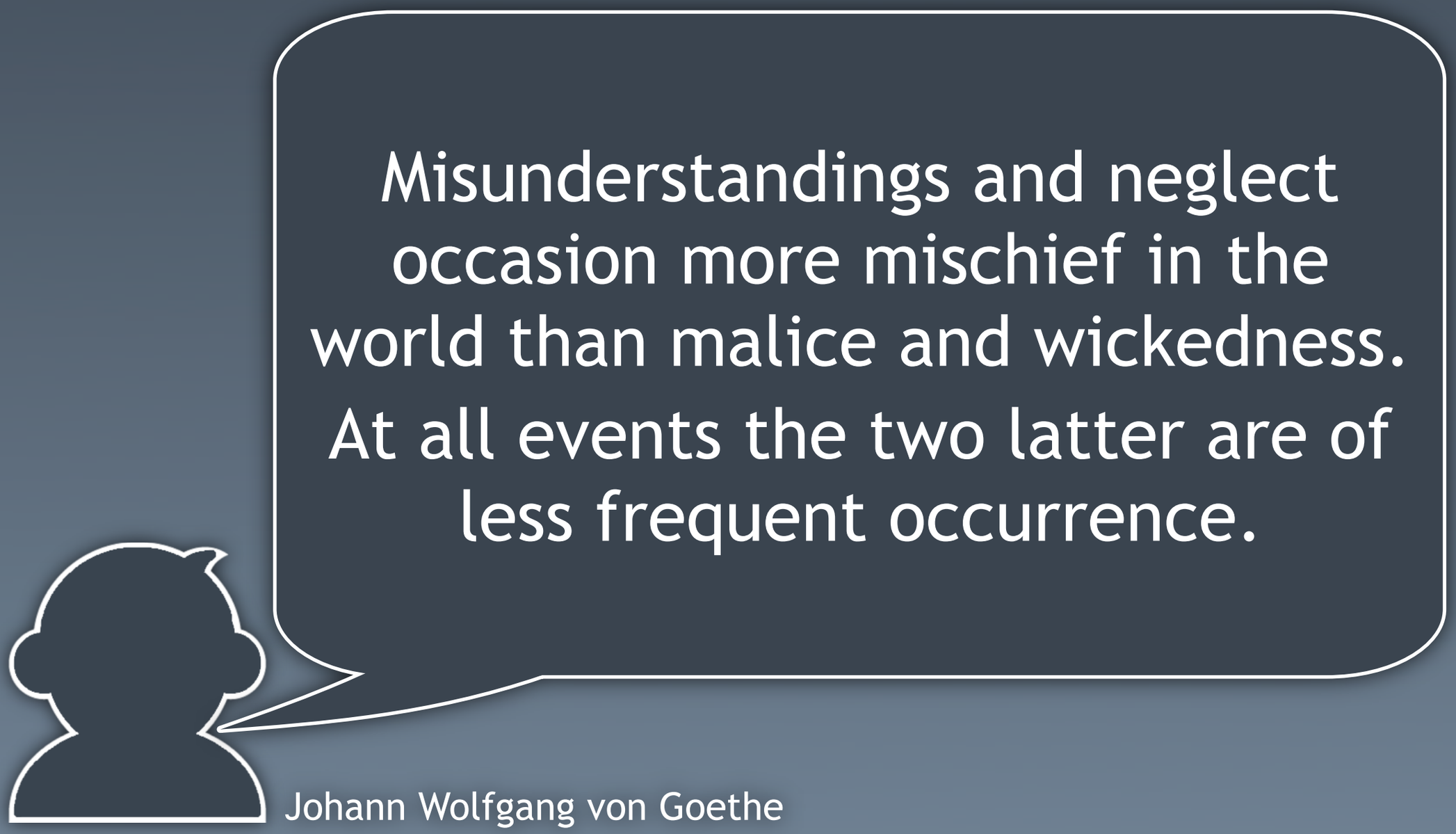
If architectural decisions are delegated to members of the team the knowledge is automatically shared, at least with the person it has been delegated to.
Combined with pair programming this can get the architecture running through the code quite easily.
However, the architect must be careful not to abdicate! Decisions still need to be made and they still need to be reviewed and validated.

In my recent experience delegation requires an adept team - essentially one that you can trust to make the architectural decisions and present it for review.
If you find such a team then by providing strong architectural principles it's easy to get the team to generate a consistent, appropriate design that meets the architectural vision, even if it doesn't wholly align with how you would have done it.

For example, I believe one of the most successful architectures I've defined was captured almost entirely in the architectural principles section of the software architecture document.
The architectural principles were defined using terms like "middleware", "data agnostic".
As long as the development team (which included me) adhered to those principles I wasn't too concerned about much else.

The architectural principles are an extremely good aid to delegation:
 * they are short and so suited to sharing by various means
   - soundbites, wiki, mail
 * they are qualified with a rationale
 * they tend to form a checklist
 * they are easily added to or changed

Summary



www.codingthearchitecture.com

While not an expert in the software development lifecycle, Goethe sums things up fairly well.

If the software doesn't meet your expectations then it's most likely down to neglect or misunderstanding. As the owner of software architecture and the one charged with its implementation the architect needs to question what they've done to minimise these.

coding {the} architecture

Website

http://www.codingthearchitecture.com

London User Group

Monthly mix of presentation and discussion

Google Group

http://groups.google.com/codingthearchitecture