coding
{the}
architecture

Why Software Projects Fail
Why software projects fail...
...architects are here to help, not to hinder
Software projects fail for a number of reasons.
Iterative and agile techniques solve some problems...
...but not all of them
Software architects have a bad reputation ("thinkers")
“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)

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

Project failure
Time for change
1. Management of the non-functional requirements
“it should be fast”
“the main flow of the search use case should respond within 10 seconds, for all concurrent users”
- Performance
- Scalability
- Availability
- Security
- Disaster Recovery
- Accessibility
- Monitoring
- Management
- Auditability
- ...

Runtime

- Flexibility
- Extensibility
- Maintainability
- Interoperability
- Legal
- Regulatory
- Compliance
- i18n
- L10n
- ...

Non-runtime
Somebody needs to take ownership
It should be the architect
2. Architecture definition
Every system has an architecture, but not every architecture is defined.
Just enough
(pragmatic rather than big design up front)
Architecture definition introduces structure, guidelines and leadership.
Somebody needs to take ownership
It should be the architect
(to be fair, defining the architecture *is* our job)
3. Technology selection
Do you believe vendor hype?
“does it actually work? show me.”
Technology selection is about managing risk.
Somebody needs to take ownership
It should be the architect
(if we recommend some technology, we better make sure it works!)
4. Architecture evaluation
Does your architecture work? (and what does “work” mean?)
Do you trust yourself?  
(I don’t)
Testing provides confidence
I want to prove my architecture will work
(management generally give me less grief when I do this)
Performance and scalability tests
(automated if possible)
Somebody needs to take ownership
It should be the architect
(we should taste our architecture)
5. Architecture collaboration
The team needs to understand the architecture.
Architectures don’t live in isolation
“Sharing Architectures”
(a quick plug)
Somebody needs to take ownership
It should be the architect
(our architecture needs to integrate)
This is what we write about on our website...

www.codingthearchitecture.com
6. Ownership of the bigger picture
Software development is not a relay sport
“how do you know that your solution will work?”
“that’s an implementation detail”
“we have a spreadsheet from past projects”
Look after your architecture
Somebody needs to take ownership
It should be the architect

(why hand-off *our* architecture?)
7. Leadership
Mind the gap

There's an essay on our website about this...
Experience and focus
Developer

Focusses only on the low-level detail

Developer

Focusses only on the low-level detail

Developer

Focusses only on the low-level detail

Developer

Focusses only on the low-level detail

Developer

Focusses only on the low-level detail
Gap

Architect

Sits in an ivory tower and dictates

Gap

Developer

Focusses only on the low-level detail
Reduced gap

Architect

Increased collaboration

Developer

Increased architectural awareness

Reduced gap
Somebody needs to undertake technical leadership (continuously)
It should be the architect

(sounds obvious, but it doesn’t always happen - who’s going to steer the boat?)
8. Coaching and mentoring
Project teams need leadership and support.
Somebody needs to coach and mentor
It could be the architect

(why not share some of our vast experience?)
9. Quality assurance
Quality needs standards and practices
1. Continuous integration
2. Automated unit testing
3. Code coverage
Source code analyzers
Most projects don’t do enough quality assurance
What’s important?
Do
or
delegate
Somebody needs to assure quality
It should be the architect

(agile would say differently, but that’s a bit harsh on junior team members)
10. Design, development and testing
Hands-on doesn’t necessarily mean coding
Hands-on means engaged, but...
...why shouldn’t the architect code?
I enjoy coding
{the} architecture
;-)
“Our architects are too valuable to code”
“Does it matter if your projects fail?”
the architect can code
(it’s not mandatory; project size or politics may prevent this)
This is what we write about on our website...

Ownership of the bigger picture

Leadership

Coaching and mentoring

Quality assurance

Design, development and testing

Delivery

This is what we write about on our website...

www.codingthearchitecture.com
Summary
Software projects fail for many reasons
(some of which agile doesn’t have the answers for)
Software architects have a bad reputation because of the limited role we often undertake.
Software architecture is more than choosing technology and defining architecture.
Project success is more than choosing technology and defining architecture.
Who should do all of this?
It should be the architect

(somebody has to do it and that’s why we get paid the big bucks)
The role of a software architect

Definition
- Management of non-functional requirements
- Technology selection
- Architecture collaboration
- Architecture definition
- Architecture evaluation

Delivery
- Ownership of the bigger picture
- Leadership
- Coaching and mentoring
- Quality assurance
- Design, development and testing

This is what we write about on our website...

www.codingthearchitecture.com
A hands-on software architect can be invaluable for preventing project failure.
A hands-on software architect can be invaluable for driving project success.
Website
http://www.codingthearchitecture.com

London User Group
Monthly mix of presentation and discussion

Google Group
http://groups.google.com/codingthearchitecture