coding {the} architecture

Software architecture for developers

What is software architecture?

What is the role of a software architect?

How do you define software architecture?

How do you share software architecture?

How do you deliver software architecture?

Good code isn't enough

simon.brown@codingthearchitecture.com

@simonbrown on Twitter

C5 alliance

coding {the} architecture

“Software Architecture for Developers”

Enterprise Architecture

Technology and business strategy across organisations and organisational units.

System Architecture

Software and infrastructure architecture for an end-to-end system.

Application Architecture

Software architecture for an application, sub-system or component.

```
/// <summary>
/// Represents the behaviour behind the ...
/// </summary>
public class SomeWizard : AbstractWizard
{
    private DomainObject _object;
    private WizardPage _page;
    private WizardController _controller;

    public SomeWizard()
    {
    }

    ...
}
```

We call this

software

architecture

and it's the scope of the training course

“Enterprise Software Developer”

A four-day practical training course about building software within an enterprise environment in a structured, lightweight and pragmatic way.

Does **good code** guarantee
a successful software project?

Thgt1me

Good code

- maintainable, flexible
- meets acceptance criteria + fit for purpose
- traceability
- readable
- documented
- testable
- not complex
- loosely coupled / high cohesion
- robust
- non-functional
- well designed - good OO + logical flow
- usable
- the right tools
- reusable components / patterns

cost
timescale
resources

project methodology — team
— customers
— mgt

short cycles

change in requirements — robust requirements capture process — stories
— use cases
— templates — SRS/catalog

conflicts — contractuals — no push-back!
— communication

deployment — delays
— downtime

dependencies

non-functional — scalability
— performance

human — education + experience

usability — comms — inter-team
— location
— mgt

configuration

testing

team

continuous build

scope + creep

documentation — organised
— up to date

?

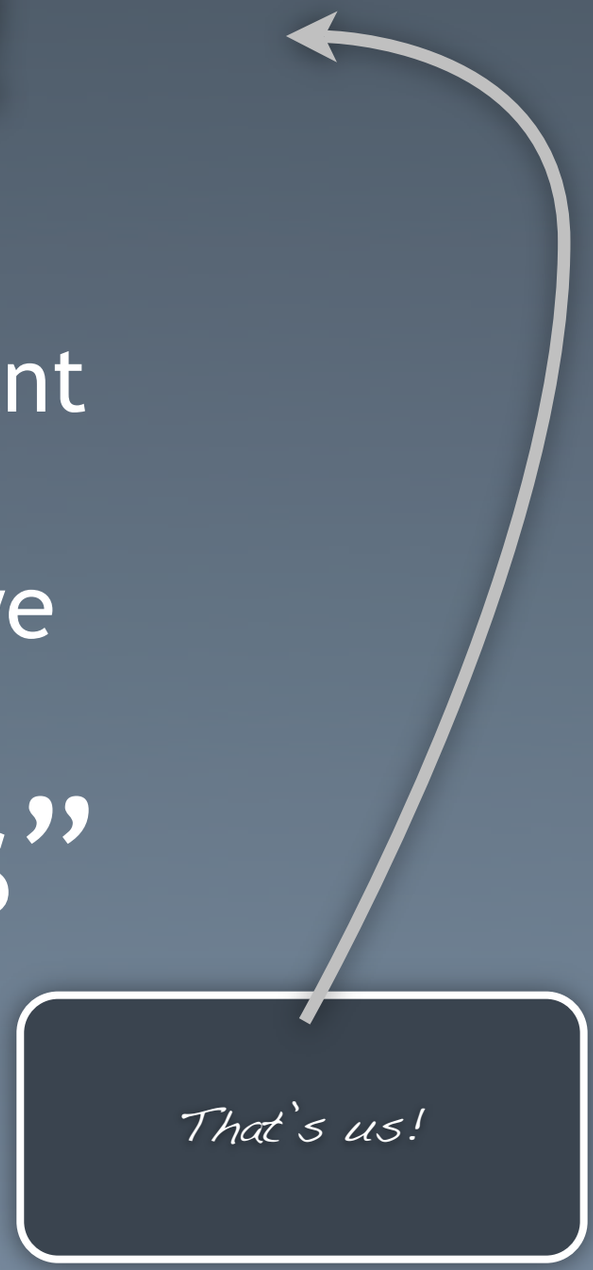
A successful software project is about much

more than good code

The reputation of the development team is at stake!

It's important that we know **what** we're releasing

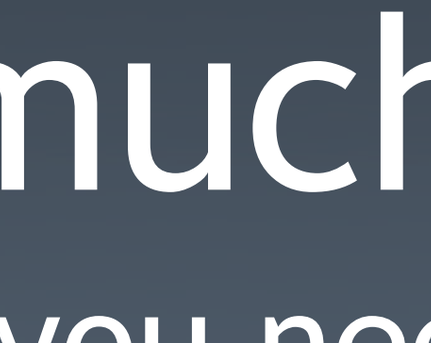
It's important that the software we release **"works"**



How much up-front work do you need to do?

This much?

Serial Development



Iterative & Incremental Development

This much?

Most projects simply do

what they've always done

And this is usually a variation of the "Waterfall" model! :-)

“what

are we building?”

User stories
are high-level requirements statements

(003) As a customer, I want to
login so that I c
accounts online

Priority: Must

(009) Customers can download statements for the last three months.

Priority: Must

Software architecture is the

big picture

Structure = components/services and their interactions

Guidelines = patterns, templates and examples

Software architecture introduces

structure and guidelines

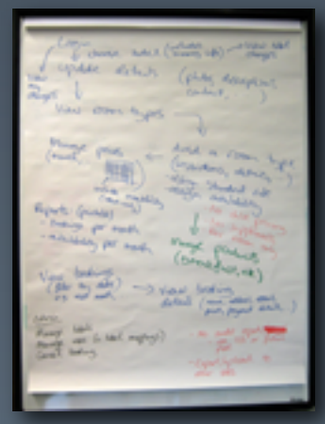
into a software system, leading to

consistency and clarity

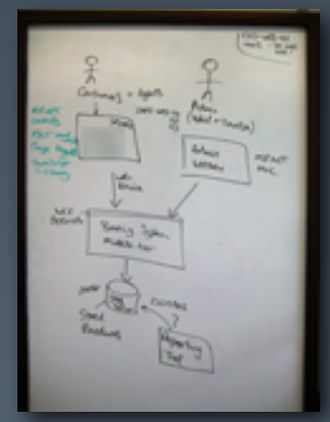
Consistency = a standard approach to solving common and recurring problems

Clarity = a thought out design with a clear architectural vision

Requirements

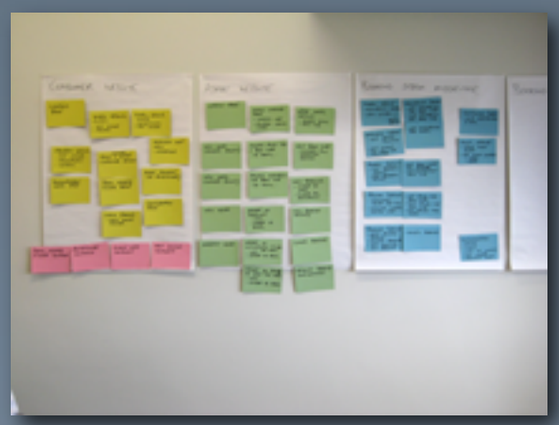


Systems & Containers



1-2 days

for an initial design



Components & Estimates

Doing this collaboratively allows people's separate ideas to meet

Can you deliver everything?

Probably not!

Refine and challenge the scope

Prioritise

Kanban boards

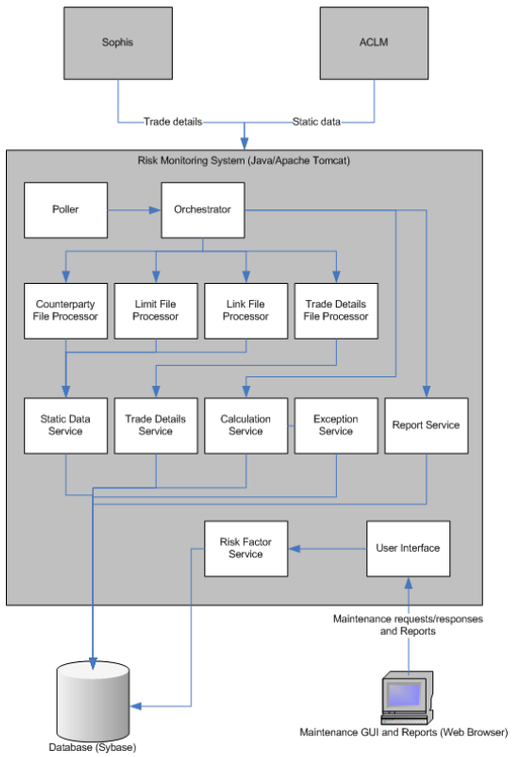
are an excellent way to visually track progress

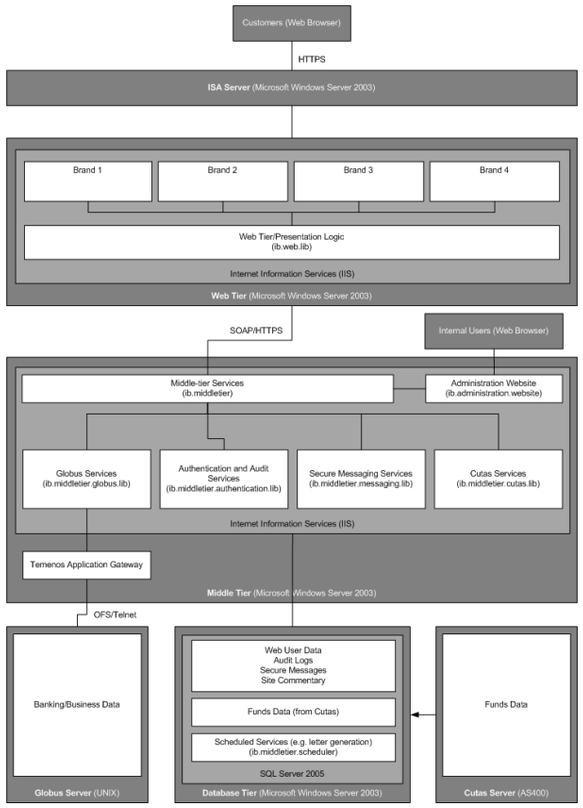
Does your architecture work?

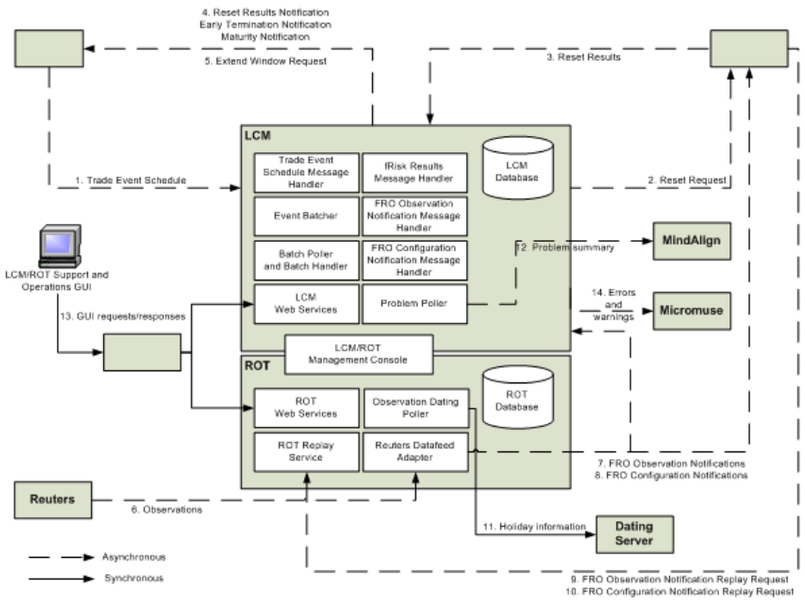
Satisfies the architectural drivers

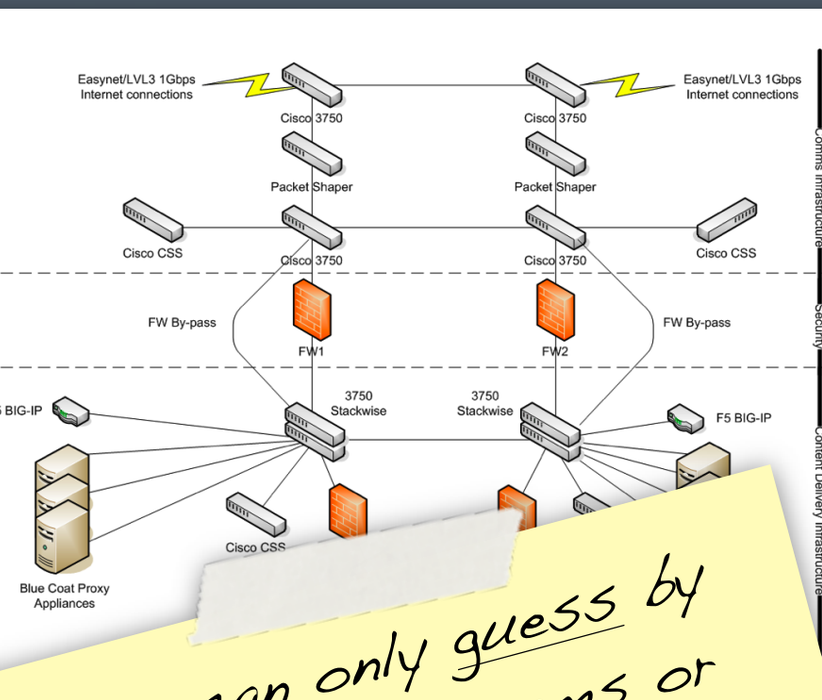
Foundations for the code

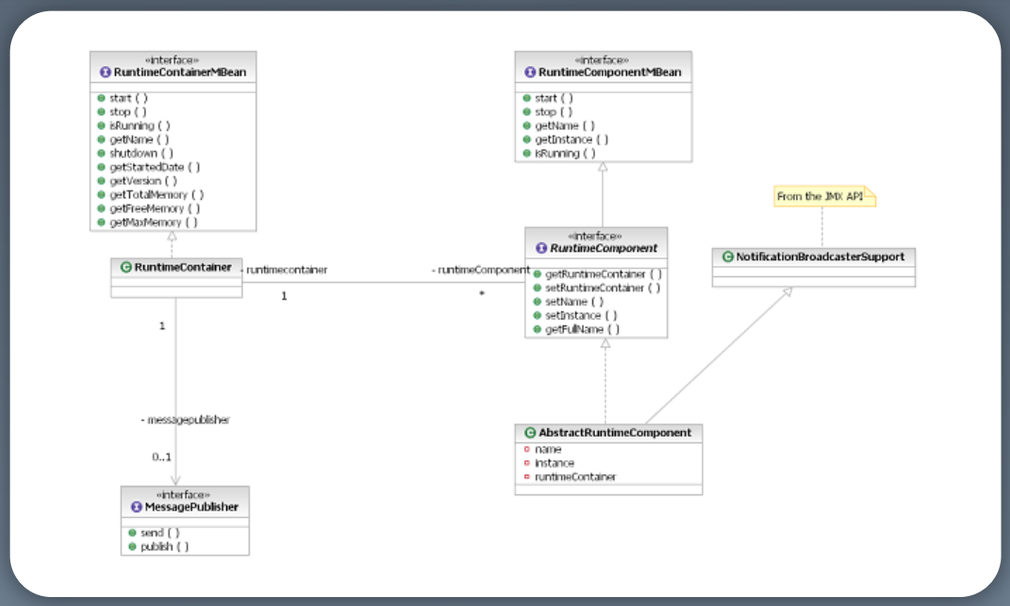
Platform for solving the business problem











Will these software systems perform and scale acceptably?

You can only guess by looking at diagrams or source code

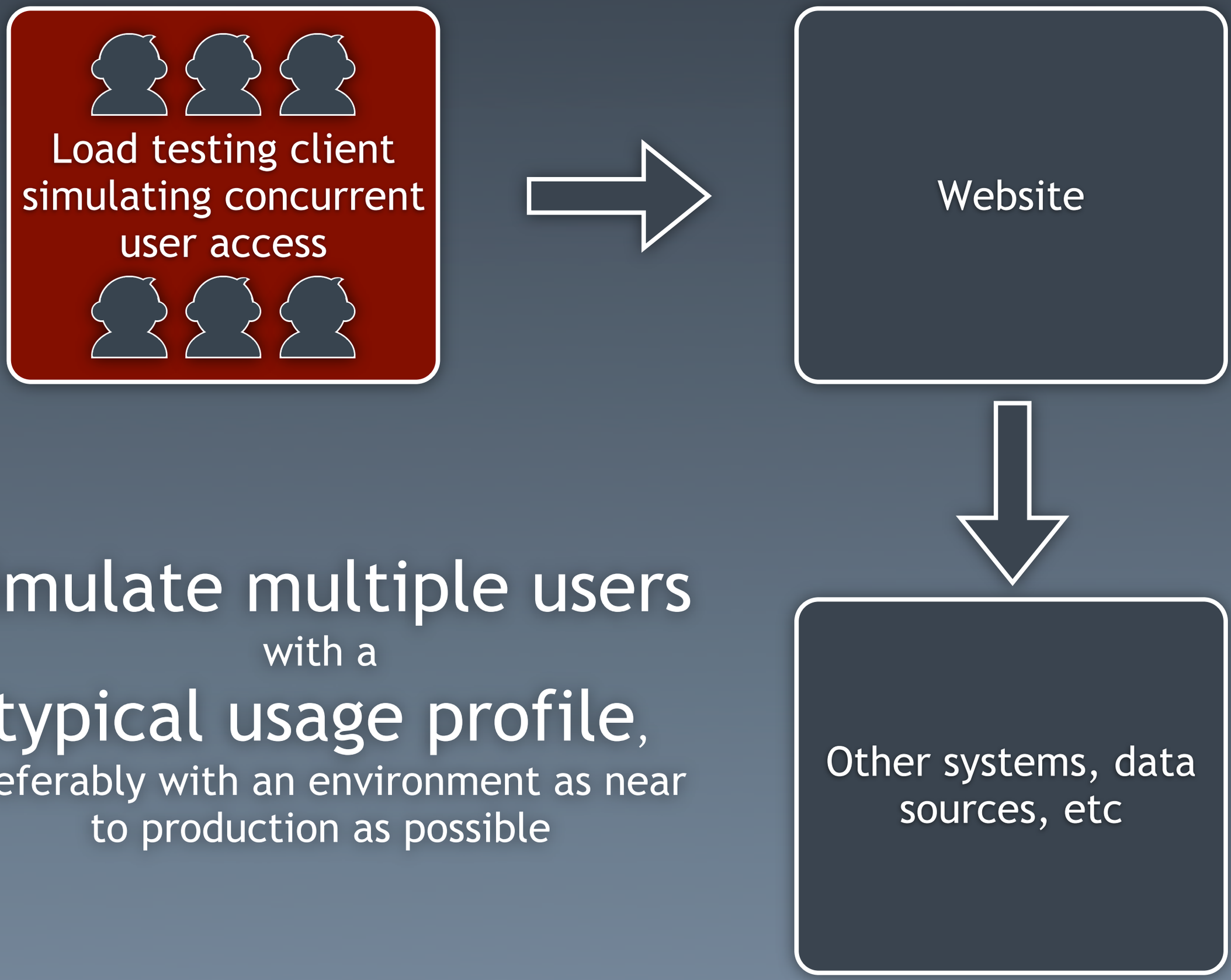
Concrete experiments

Requirements

A system

Foundations

A throwaway prototype/ proof of concept or production code



Simulate multiple users

with a

typical usage profile,

preferably with an environment as near to production as possible

Source code control

introduces a number of things, including:

Backed-up source code

A log of the changes made

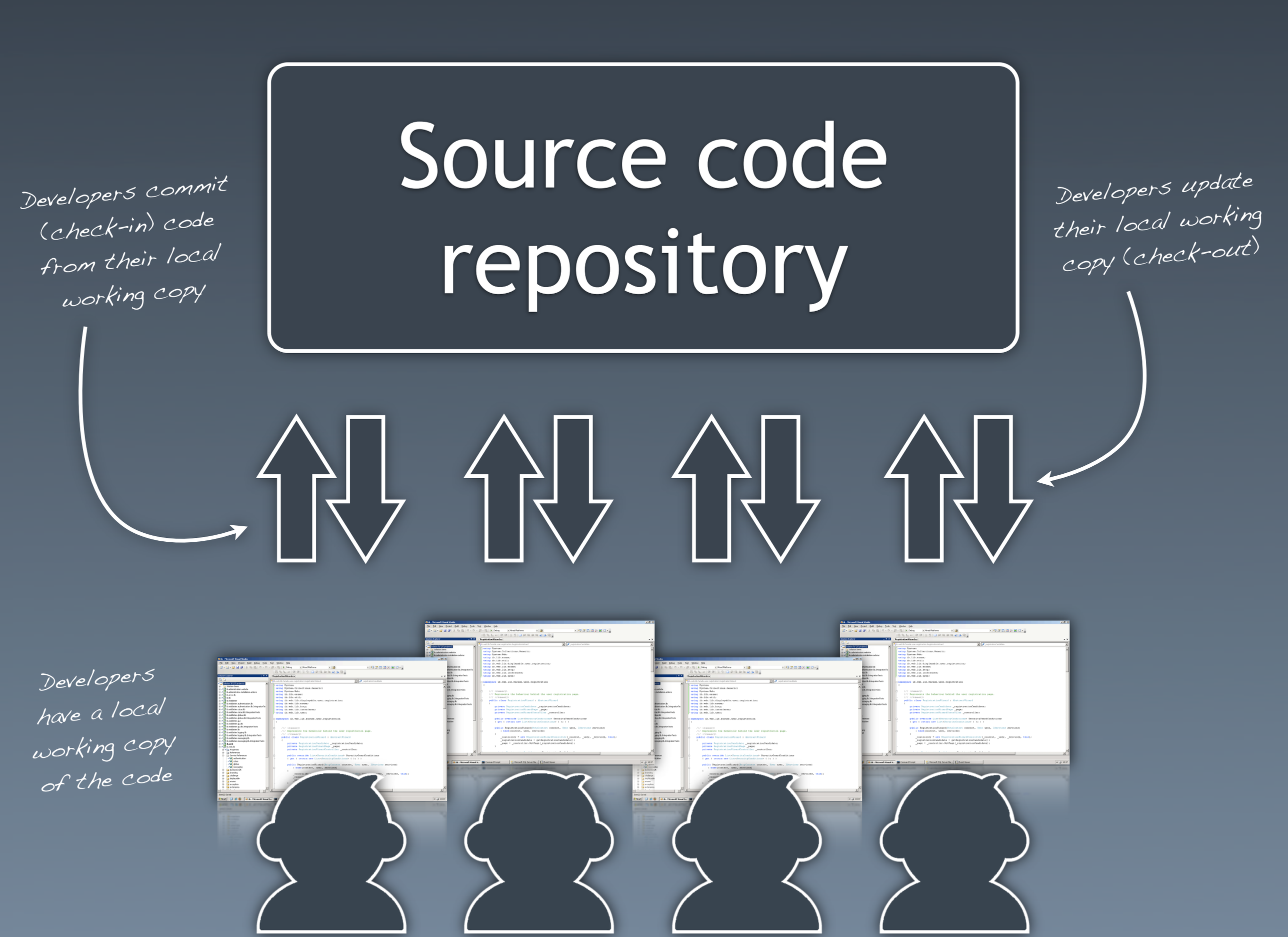
A simple way to revert to a previous working copy

Simplified parallel development

Source code repository

Developers commit (check-in) code from their local working copy

Developers update their local working copy (check-out)



Developers have a local working copy of the code

2. Work on the branch instead of the trunk

branch

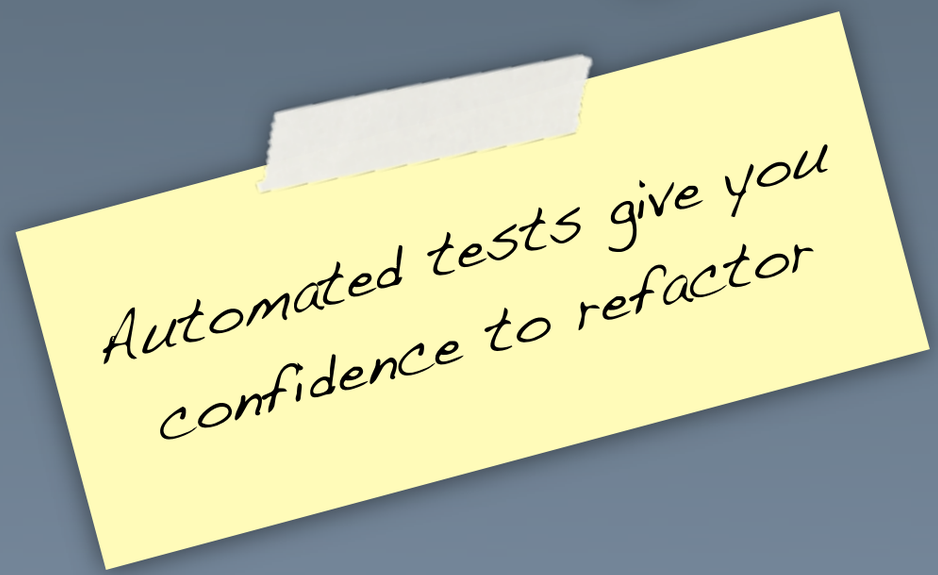
1. Create a branch from the trunk

3. Merge from the trunk to keep the branch up to date

4. Merge the branch back to the trunk when finished

trunk

How do you know that you've not broken something?



Low-level tests for classes and components

Unit &

integration tests

"Functional" tests across container boundaries

How much do you test?

This much?

0% ---------- 100%

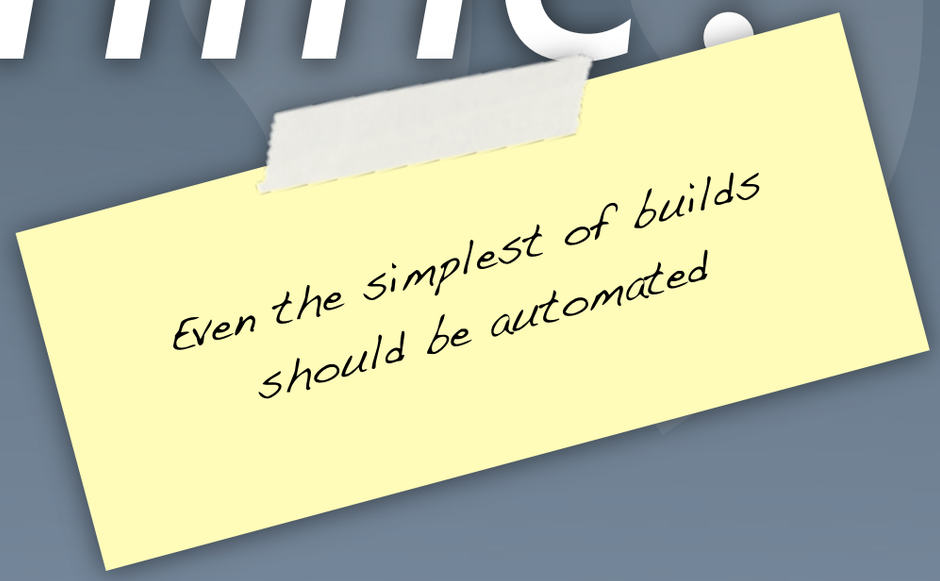
This

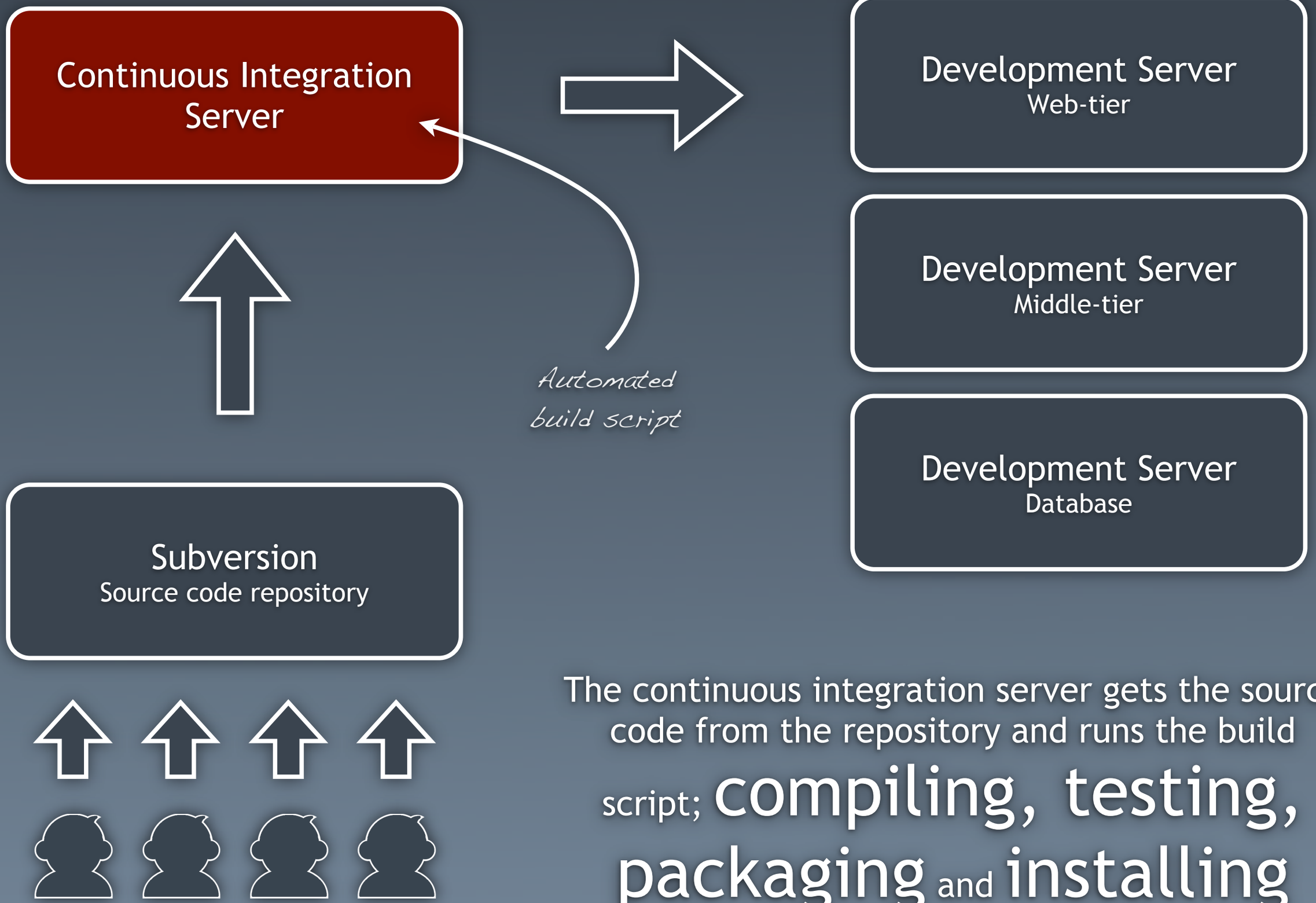
Small steps, test the "important" code (70%+ is a good goal)

Things to do when building the software

1. Compile the code
2. Publish files to your web server
3. Maybe check it works

It works on my machine!





The continuous integration server gets the source code from the repository and runs the build script; compiling, testing, packaging and installing the software

Build-Test-Run when we commit code to the repository

Continuous &

nightly builds

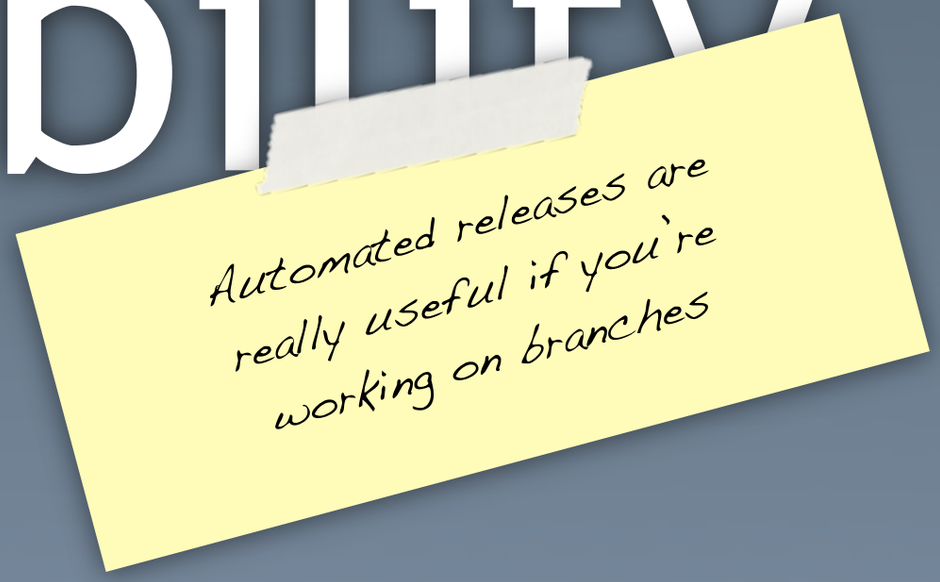
Continuous build + longer running integration tests

Automation introduces

consistency

and

repeatability



Database connection strings

Web service URLs

Server IP addresses

E-mail addresses

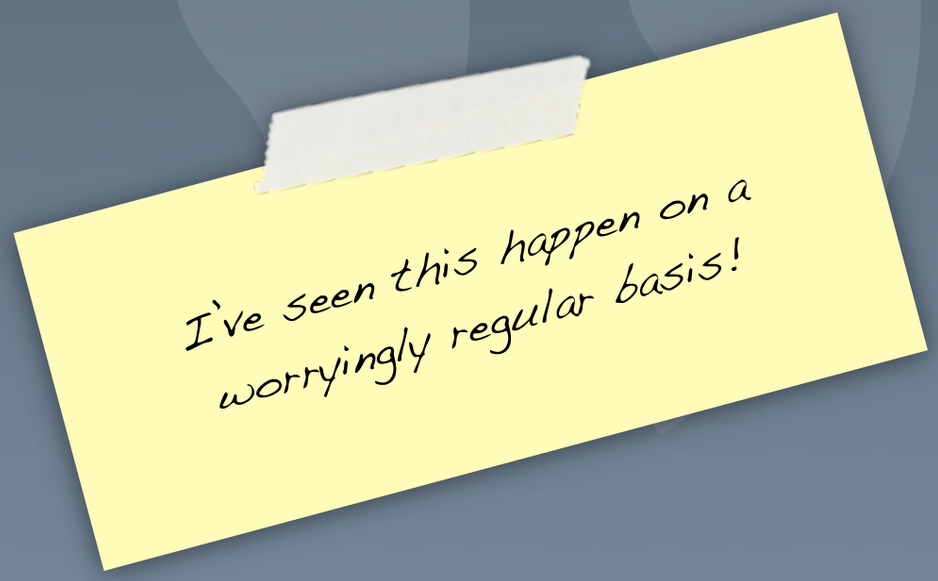
File locations

Service credentials

Many configuration items are environment specific; externalise them from the source code

Anybody know which version we're running?

It doesn't look like version 1.2



Software architectures don’t live in isolation

Current Development Team

Business Sponsors

Future Development Team

Database Administrators

Your system

Operations/Support Staff

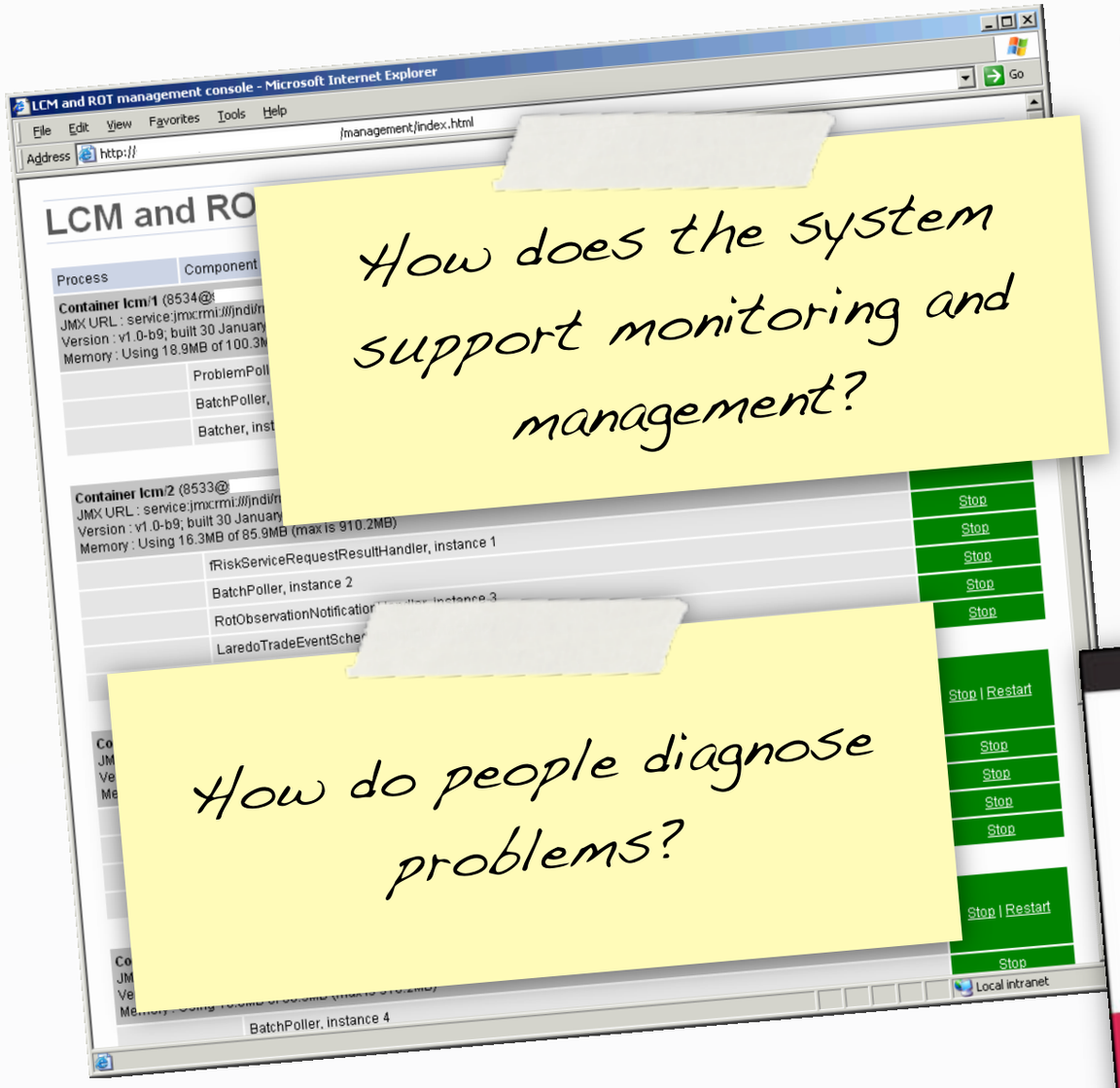
Other Teams

Security Team

Compliance and Audit

Software architecture is a platform for conversation ... be social!

Operational View



Let’s wrap up…

“Enterprise Software Developer”

A four-day practical training course about building software within an enterprise environment in a structured, lightweight and pragmatic way.

Enterprise software developer

Source Code Control

Release

Software Development Processes

Automated Unit and Integration Testing

Configuration Management

Requirements

Automated Builds

Load Testing

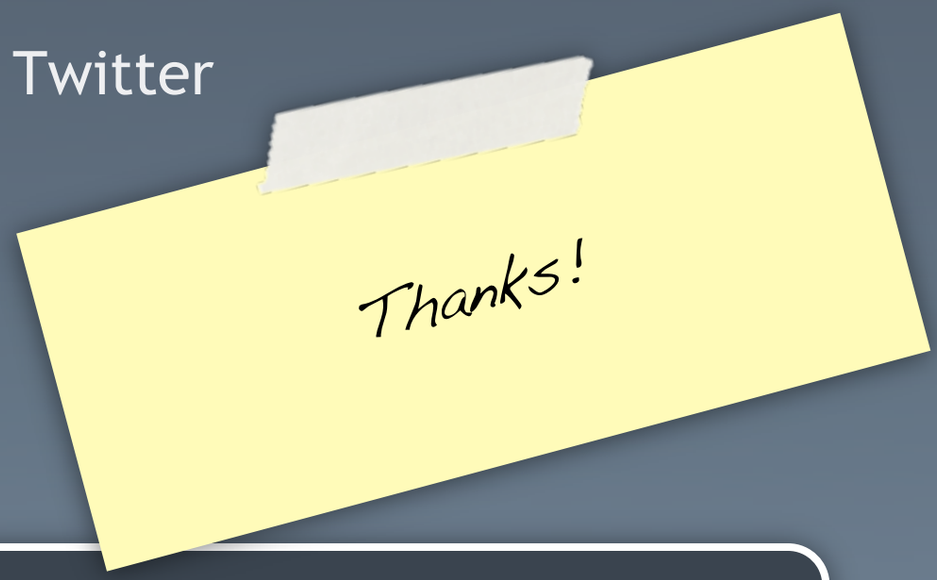
Architecture and Design

Continuous Integration

Operational Hand-over

simon.brown@codingthearchitecture.com

@simonbrown on Twitter



coding {the} architecture

http://www.codingthearchitecture.com

Software architecture for developers