# Software Development in the Large

Fethi Rabhi

(Thanks to Chris Mendes who designed most of these slides)

# Process vs practice

- Industry does not like prescriptive processes
- They like
  - Rules of thumb
  - Practices, *not processes*
- Many projects fail
- Successful projects have these features
  - Measures:
    - Happy customers
    - Happy teams
    - Software that is being used

# Process vs Practice

Processes prevent the mediocre from making mistakes.
Practices make the professional exceptional.

# Why are you here today?

Are there any concerns you have about how you are going to get your project done?

# What is covered

- Planning
- Typical methodologies
- What does a successful software project look like?
- Roles
- Practices
- Bringing it together

# What is covered

- **Planning**
- Typical methodologies
- What does a successful software project look like?
- Roles
- Practices
- Bringing it together

# Planning

- It helps to know where you are going...
- How do I plan and estimate?
  - Understand the goals
  - Plan work to make the goals happen
  - Think about risk and how to minimise it
  - Write it down and share it
  - Estimation...a problem
- Is it just an exercise in futility?
  - Start with a broad brush and paint in detail as you go
  - Invest the right amount in planning

# What is covered

- Planning
- **Typical methodologies**
- What does a successful software project look like?
- Roles
- Practices
- Bringing it together

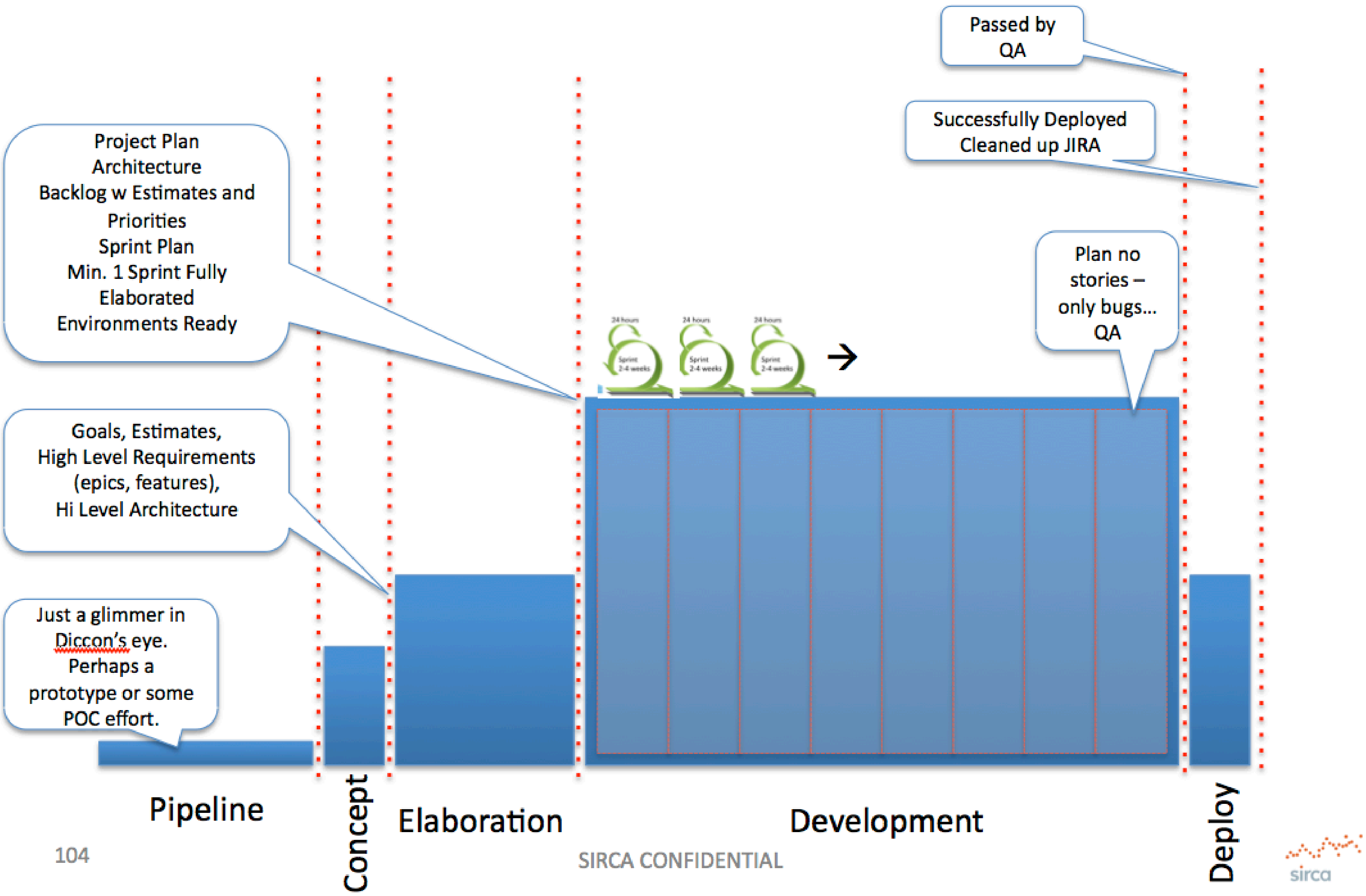
# Typical Methodologies

- List:
  - Waterfall, Iterative, Spiral, RAD
- Agile is increasingly used and here's why...
  - It's a natural process
  - It brings risk forward, rather than pushing it to the end
  - It puts quality, value and end-user priority first

# What is covered

- Planning
- Typical methodologies
- **What does a successful software project look like?**
- Roles
- Practices
- Bringing it together

# Project Shape



Passed by QA

Successfully Deployed
Cleaned up JIRA

Project Plan
Architecture
Backlog w Estimates and Priorities
Sprint Plan
Min. 1 Sprint Fully Elaborated
Environments Ready

Plan no stories – only bugs…
QA

Goals, Estimates,
High Level Requirements (epics, features),
Hi Level Architecture

Just a glimmer in Diccon's eye.
Perhaps a prototype or some POC effort.

104

SIRCA CONFIDENTIAL

# Project Shape - About Scrums

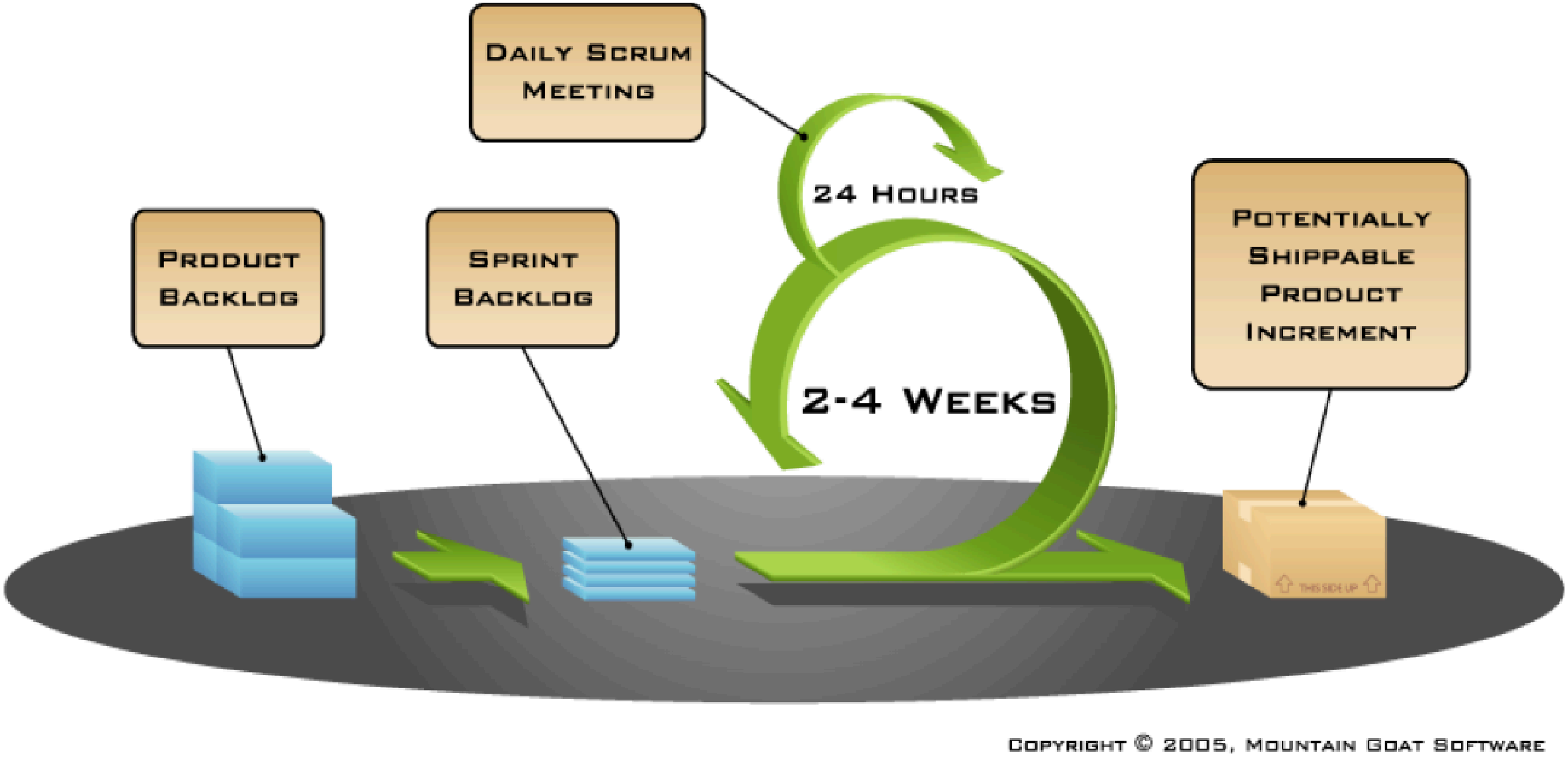
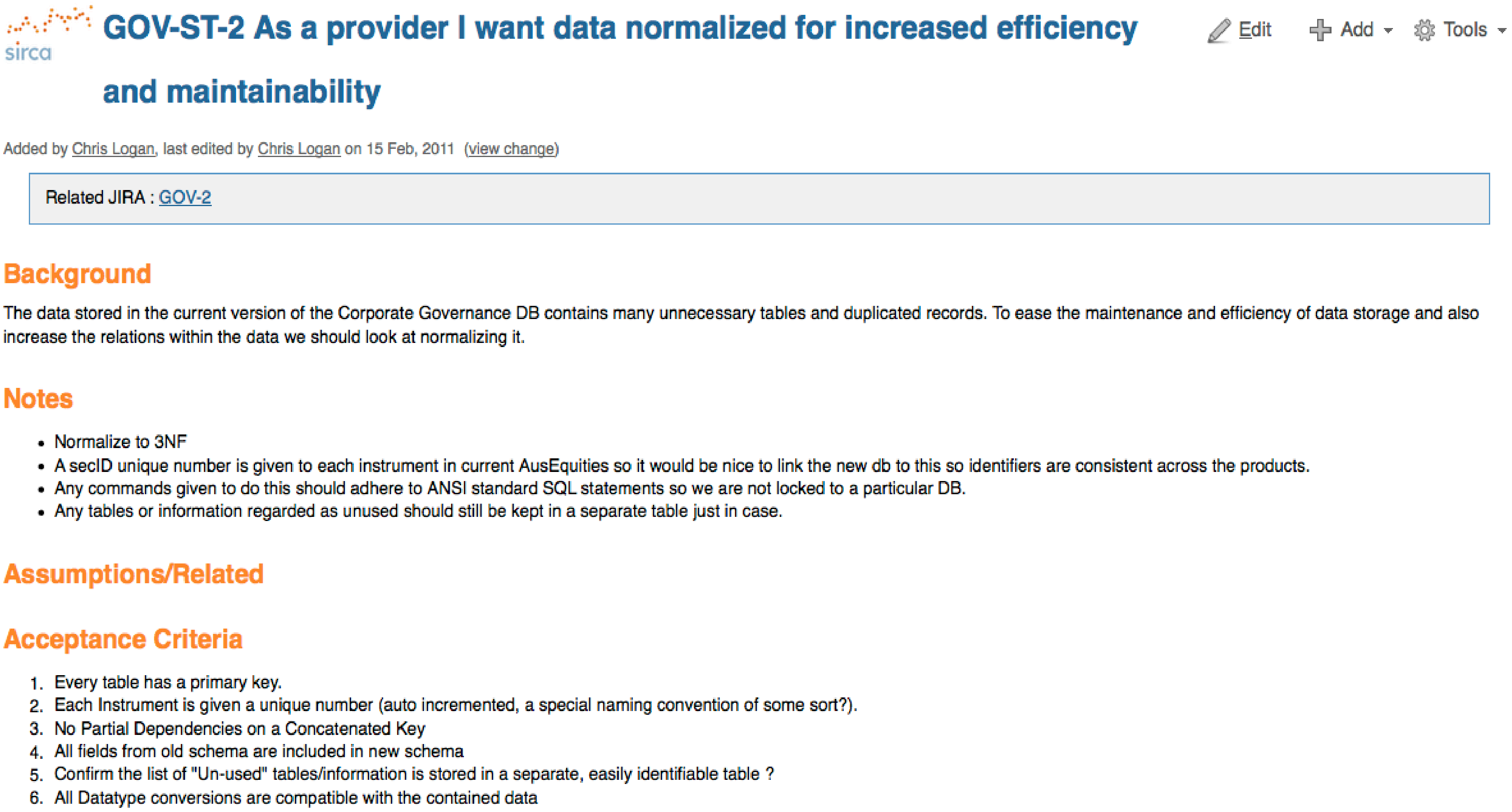

Daily Scrum Meeting
24 Hours
Product Backlog
Sprint Backlog
2-4 Weeks
Potentially Shippable Product Increment
This side up
Copyright © 2005, Mountain Goat Software

Image available at
www.mountaingoatsoftware.com/scrum

# Epics, Stories, Backlogs...

- Your assignment is currently set up as *Epics*
- You need to break them into stories, prioritise them and elaborate them
- Start by elaborating just the story titles
  - Estimate them...
- Before your first sprint, fully elaborate the stories for that sprint (at least)

# User Story Example



## GOV-ST-2 As a provider I want data normalized for increased efficiency and maintainability

Added by Chris Logan, last edited by Chris Logan on 15 Feb, 2011 (view change)

Related JIRA : GOV-2

### Background

The data stored in the current version of the Corporate Governance DB contains many unnecessary tables and duplicated records. To ease the maintenance and efficiency of data storage and also increase the relations within the data we should look at normalizing it.

### Notes

- Normalize to 3NF
- A secID unique number is given to each instrument in current AusEquities so it would be nice to link the new db to this so identifiers are consistent across the products.
- Any commands given to do this should adhere to ANSI standard SQL statements so we are not locked to a particular DB.
- Any tables or information regarded as unused should still be kept in a separate table just in case.

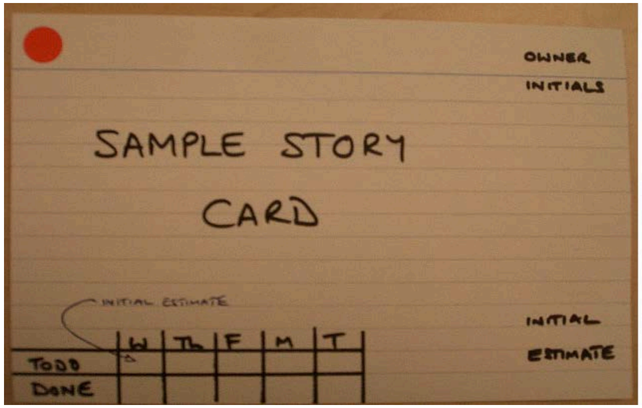
### Assumptions/Related

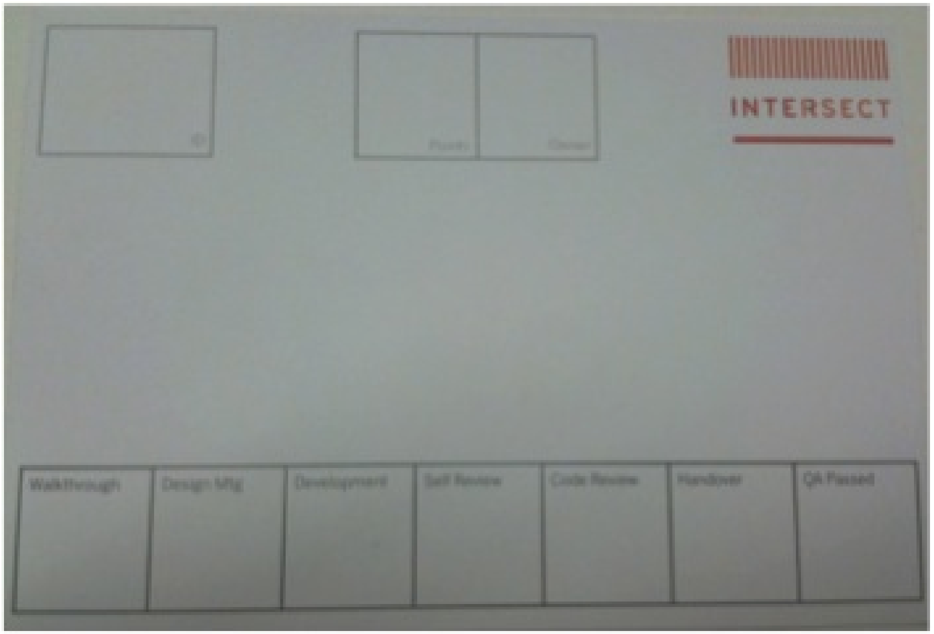
### Acceptance Criteria

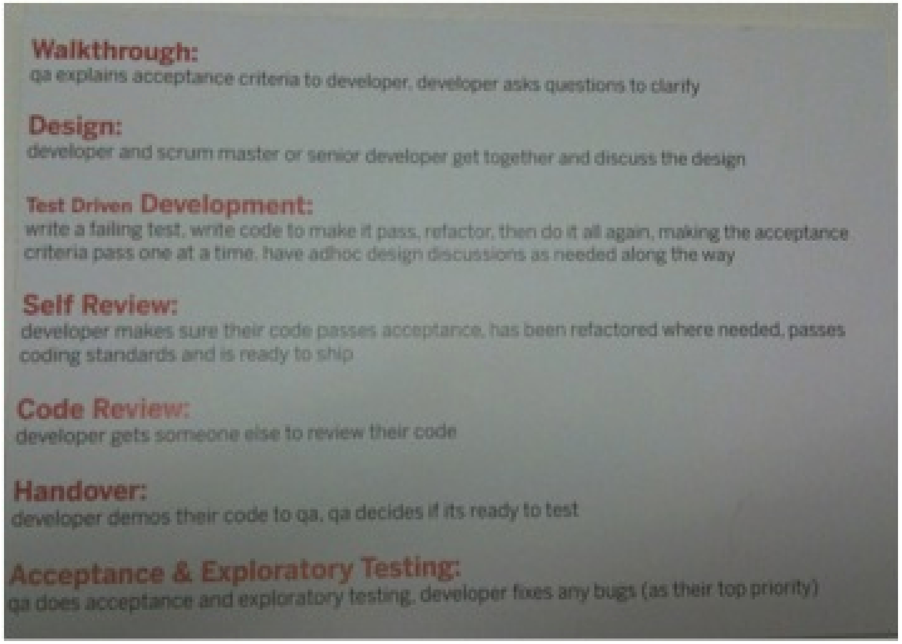
1. Every table has a primary key.
2. Each Instrument is given a unique number (auto incremented, a special naming convention of some sort?).
3. No Partial Dependencies on a Concatenated Key
4. All fields from old schema are included in new schema
5. Confirm the list of "Un-used" tables/information is stored in a separate, easily identifiable table ?
6. All Datatype conversions are compatible with the contained data

# Agile Tools

- Story Cards
- Simple Tools - Spreadsheets, Whiteboards






**Walkthrough:**
qa explains acceptance criteria to developer, developer asks questions to clarify

**Design:**
developer and scrum master or senior developer get together and discuss the design

**Test Driven Development:**
write a failing test, write code to make it pass, refactor, then do it all again, making the acceptance criteria pass one at a time. have adhoc design discussions as needed along the way

**Self Review:**
developer makes sure their code passes acceptance, has been refactored where needed, passes coding standards and is ready to ship

**Code Review:**
developer gets someone else to review their code

**Handover:**
developer demos their code to qa. qa decides if its ready to test

**Acceptance & Exploratory Testing:**
qa does acceptance and exploratory testing, developer fixes any bugs (as their top priority)

# What might your project look like?

# What is covered

- Planning
- Typical methodologies
- What does a successful software project look like?
- **Roles**
- Practices
- Bringing it together

# Roles

- Product Owner
- Project Manager
- Scrum Master
- Dev Tester
- Developer

# Roles

- Assign them, assign them wisely
- Of course, there are many more roles but you don't need them in this case
- Roles:People is not 1:1. One person can have many roles, you can swap roles.
- You can not be the Dev-Tester for your own code - you can be a developer and a tester but only testing other people's code
- You can't review your own work

# Product Owner

- Is a business analyst
- Consults with stakeholders and documents their input
- Creates stories that can be implemented
  - Does enough analysis up front
  - Sufficiently described
  - Are not designs
- Works with QA and stakeholders to ensure acceptance criteria are right
- Owns the product backlog
- Prioritises all stories into backlog in consultation with the project sponsor

# Scrum Master

- Ensures engineers have no roadblocks and are doing what they should be doing
- Keeps the PM updated re progress, risks, issues
- Has daily meetings with her team
- Provides technical leadership, design guidance, design review and code review
- Ensures quality is maintained at an appropriate level
- Helps his team to plan and estimate work
- Updates Sprint Board Daily
- Maintains backlog for the project, Updates backlog each Sprint
- Facilitates Showcase and Retrospectives each Sprint

# Developer

- Understands the story and acceptance criteria
- Analyses problems
- Designs solutions
  - in collaboration with architect, infrastructure etc
  - let's us know if major re-design is needed on existing systems
- Plans and Estimates their own work
- Codes and Unit Tests
- Participates in design and code reviews
- Asks for review and assistance
- Tests their own work against acceptance criteria

# Developmental Tester

- Understands requirements provided by PO and writes acceptance criteria
- Develops & maintains test scripts
- Updates API test harnesses
- Ensures unit testing happens
- Ensures Dev Testing happens
- Reports on progress, raises bugs, retests
- Regression tests
- *Breaks stuff*
- Monitors automated build and test
- Monitors code coverage

# Project Manager

- Is a Communicator, Facilitator and Trouble Shooter
- Understands project goals and benefits and shares vision with the team
- Ensures work is prioritised and scheduled
- Ensures work is planned by the engineers and is carried out as scheduled
- Manages project risks and ensures mitigation is executed
- Helps make sure project stakeholders are communicated with and that goals are met
- Maintains schedules and other project documentation

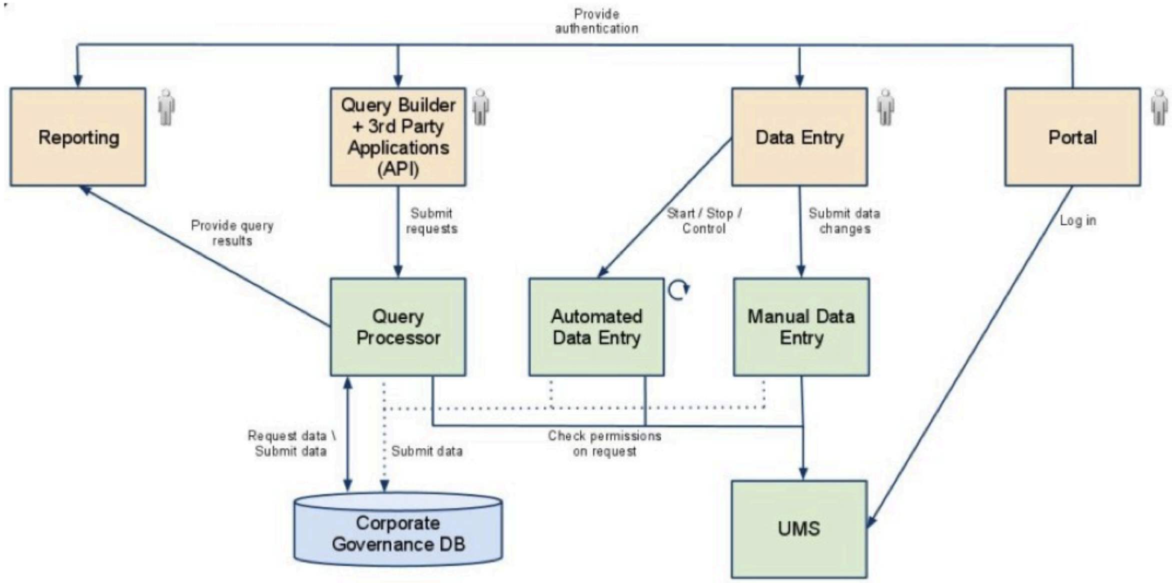
# What do I want to cover?

- Planning
- Typical methodologies
- What does a successful software project look like?
- Roles
- **Practices**
- Bringing it together

# Practices

- Review (documents, code, design, architecture, test scripts, everything)
- Architecture - have one
- Design - everything
- Unit Testing - no excuses
- Regression Testing - automated
- Configuration Management (software & environments)
- Daily Check-in

# Architecture - logical



ⓘ Click for document

### Reporting

UI responsible for presenting the results of queries to the user. Supports displaying and downloading results. This will communicate with the **Query Processor** inside the **Corporate Governance Application Server**. The **Query Processor** will publish results to the **Reporting** module.

### Query Builder

UI to allow the user to build queries for the data they require. Phase 1 development will include a basic interface allowing end users to select 10 or so basic queries to run. They will not have the functionality to build their own queries in Phase 1. This component will pass the queries to the **Query Processor**.

### 3rd Party Applications

The Query Processor supports API access to it's querying functions allowing 3rd Parties to access the Corporate Governance data. The UMS must also support this interaction for security.

### Data Entry

UI to allow data entry operators to enter Annual Report data into the Database. Supports tagging of PDF documents and presenting the appropriate sections of the PDF to the data entry operator.

### Automated Data Entry

Processes annual report PDF documents to provide some level of automation. Can be controlled (start/stop) via the **Data Entry** module.

### Query Processor

Processes queries and requests from the front end systems. Checks for security and malformed or dangerous queries. Processes API calls and transforms them into SQL queries for the **Corporate Governance Database**. Returns results to the Reporting component. This interface will also provide usage statistics to provide further information to base full UI development on during Phase 2.

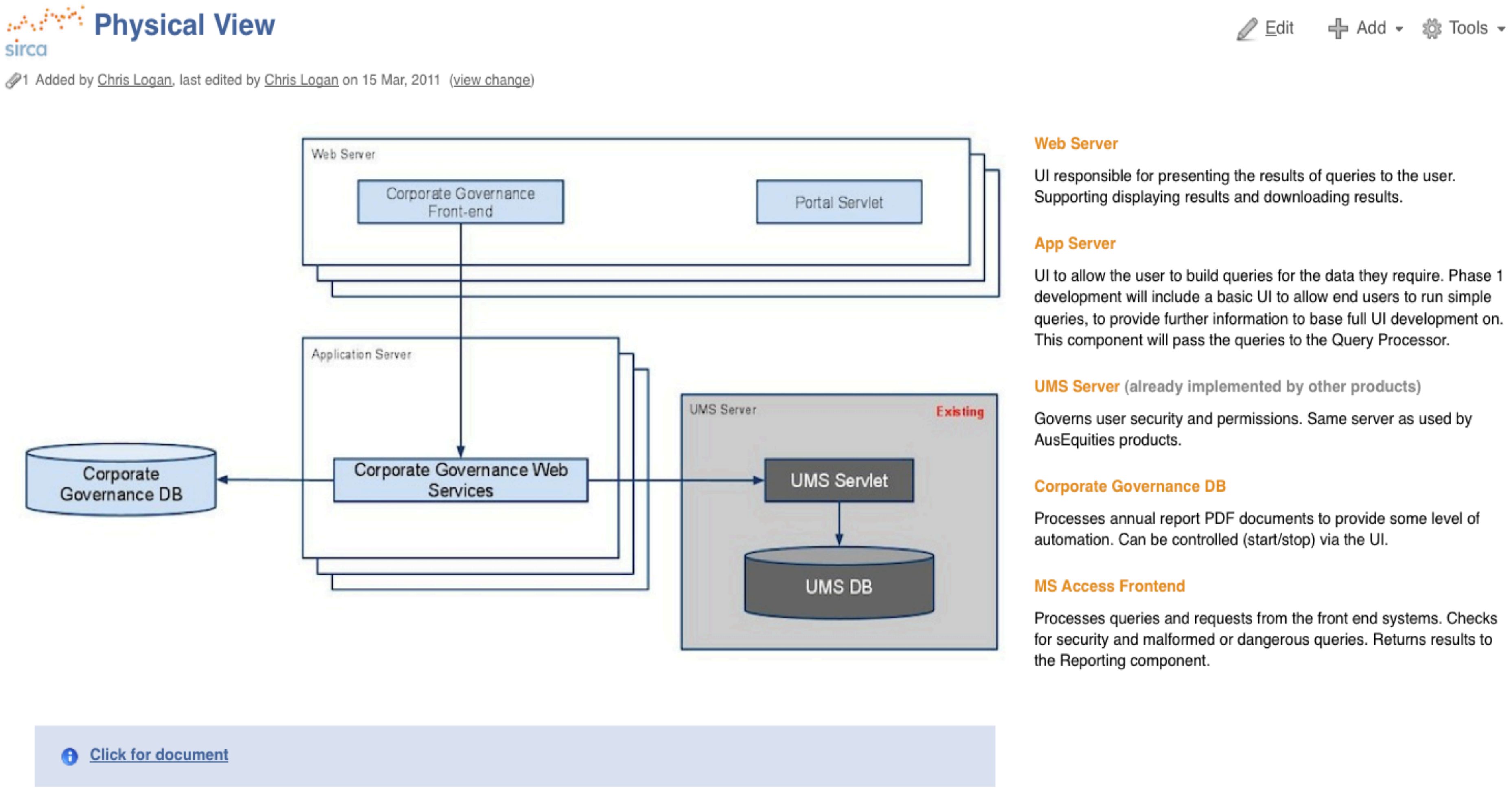
### Portal

The Portal enables client login and authentication with the UMS. It is a common feature for each AusEquities product.

### UMS

Governs user security and permissions. Same server as used by AusEquities products.

# Architecture - Physical



## Physical View

Edit Add Tools

1 Added by Chris Logan, last edited by Chris Logan on 15 Mar, 2011 (view change)

**Web Server**

UI responsible for presenting the results of queries to the user. Supporting displaying results and downloading results.

**App Server**

UI to allow the user to build queries for the data they require. Phase 1 development will include a basic UI to allow end users to run simple queries, to provide further information to base full UI development on. This component will pass the queries to the Query Processor.

**UMS Server** (already implemented by other products)

Governs user security and permissions. Same server as used by AusEquities products.

**Corporate Governance DB**

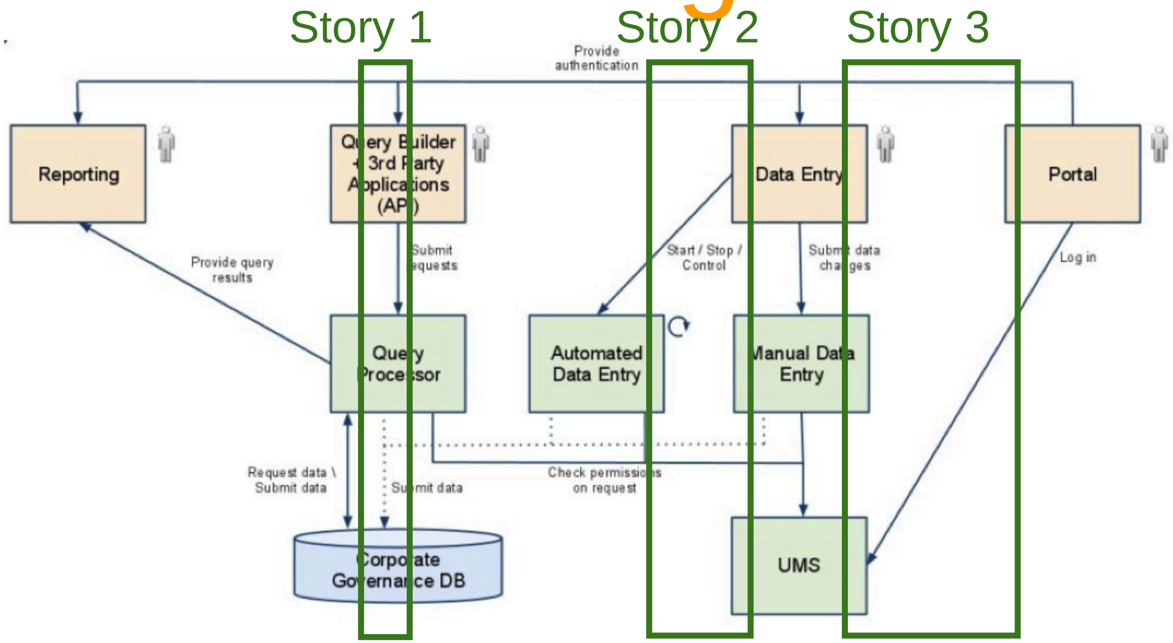
Processes annual report PDF documents to provide some level of automation. Can be controlled (start/stop) via the UI.

**MS Access Frontend**

Processes queries and requests from the front end systems. Checks for security and malformed or dangerous queries. Returns results to the Reporting component.

Click for document

# A trick with Agile...



Click for document

**Reporting**

UI responsible for presenting the results of queries to the user. Supports displaying and downloading results. This will communicate with the **Query Processor** inside the **Corporate Governance Application Server**. The **Query Processor** will publish results to the **Reporting** module.

**Query Builder**

UI to allow the user to build queries for the data they require. Phase 1 development will include a basic interface allowing end users to select 10 or so basic queries to run. They will not have the functionality to build their own queries in Phase 1. This component will pass the queries to the **Query Processor**.

**3rd Party Applications**

The Query Processor supports API access to it's querying functions allowing 3rd Parties to access the Corporate Governance data. The UMS must also support this interaction for security.

**Data Entry**

UI to allow data entry operators to enter Annual Report data into the Database. Supports tagging of PDF documents and presenting the appropriate sections of the PDF to the data entry operator.

**Automated Data Entry**

Processes annual report PDF documents to provide some level of automation. Can be controlled (start/stop) via the **Data Entry** module.

**Query Processor**

Processes queries and requests from the front end systems. Checks for security and malformed or dangerous queries. Processes API calls and transforms them into SQL queries for the **Corporate Governance Database**. Returns results to the Reporting component. This interface will also provide usage statistics to provide further information to base full UI development on during Phase 2.

**Portal**

The Portal enables client login and authentication with the UMS. It is a common feature for each AusEquities product.

**UMS**

Governs user security and permissions. Same server as used by AusEquities products.

# Practices

- APIs
- Automated Testing
- Automated Build
- Continuous Integration - at least daily
- Daily meetings...?
- Sprints (Momentum, Planning, Showcases, Retrospectives)
- Co-locate during development

# Useful Links

- SCRUM Stuff:
  - http://www.mountaingoatsoftware.com/presentations
  - http://www.scrumalliance.org/
- Practices:
  - http://www.martinfowler.com/articles/continuousIntegration.html
  - http://martinfowler.com/design.html
  - Just read martinfowler.com !