DEPARTMENT OF COMPUTER & INFORMATION SCIENCES

University of Strathclyde Science

# Software Engineering

## Lecture 4: Requirements

Billy Wallace
w.wallace@strath.ac.uk

CS993

# Lecture Outline

- What are requirements?
- Why do we need requirements?
- Who produces requirements?
- How do we create requirements?

# What are requirements?

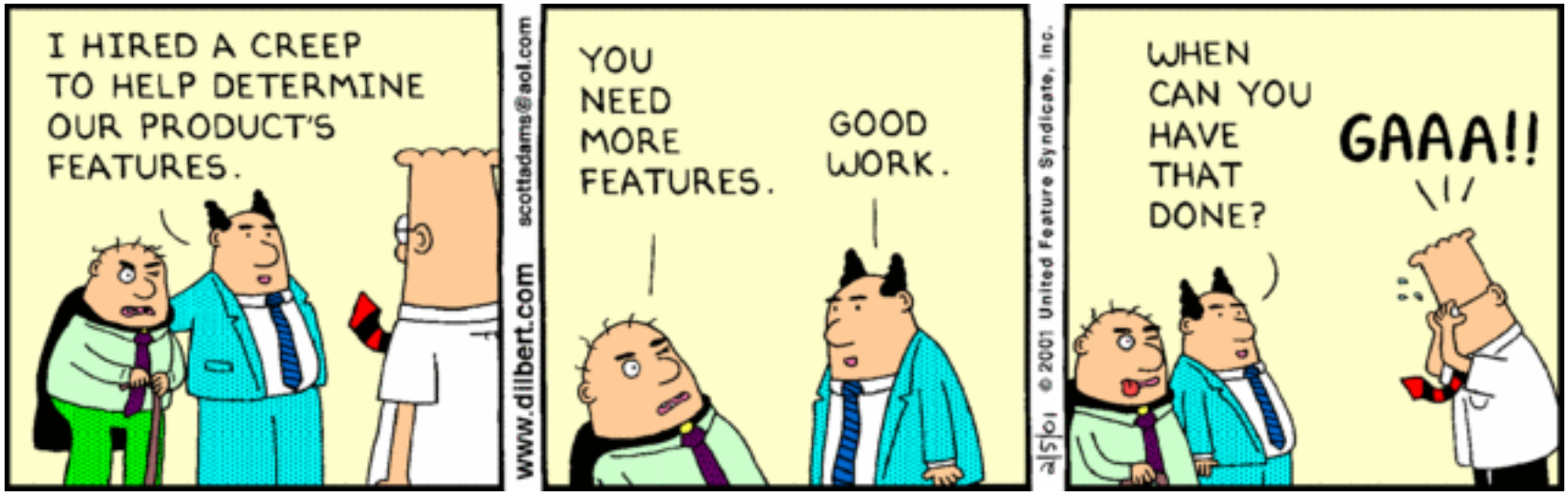
- Simply put, a description of what the customer needs from a piece of software.
- Documented, hopefully clearly, so that the developers know what to build.
- A potential source of conflict between customers and developers!
- Possibly part of a legal contract between customers and developers.

# Police Scotland/Accenture

University of Strathclyde Science

There was “no single reason” why the Police Scotland i6 computer system project failed, a report from Audit Scotland has concluded.
The report finds that good practice was followed in the planning and procurement stages of the ambitious project, which was intended to replace 130 electronic and paper-based systems covering 80 per cent of police processes for recording crime and missing persons.
Multinational professional services company Accenture was awarded a ten-year, £46.11m contract to provide the i6 computer system to Police Scotland by the Scottish Police Authority (SPA) in 2013, with projected savings of £200m to the authority and Police Scotland.
However, the “complex and ambitious programme” suffered early on as Police Scotland and the contractor disagreed over the terms of the contract, leading to “damaged relationships and trust between the two organisations from a very early stage.”
There was also an issue with the waterfall approach used to produce the software – common at the time the project was commissioned – which meant that each stage was dependent on the previous one and the system could not be tested by police until it was almost complete. -- ***www.holyrood.com***

# Beware the Feature Creep!



# Communication is Key

- This is not a scientific process with one right answer, there is also an aspect of negotiation.
- The Dilbert cartoon is from the viewpoint of the software engineer, the other stakeholders have their own views.

# Why do we need requirements?

- To remove ambiguity as much as possible and to communicate to everyone involved what will be built.
- Good requirements should minimise the risk of conflict.
- Requirements will change, so the way they are documented should allow that to happen.

# Who produces requirements?

- Initially the customer should do this. This might be in the form of a Request for Quote (RFQ).
- Remember though, this is all about communication, so both parties have a say in what goes in here.
- The developers will know what's possible and will have ideas for useful features.

University of Strathclyde Science

# Pick any two:

- Good
- Fast
- Cheap

University of Strathclyde Science

# Who's your customer?

- Many stakeholders will not be technical.
- How do you communicate the requirements to them?
- Write these in natural language so that they can be understood, but then they may be ambiguous.
- Have a look at systems like Cucumber to see an alternative.

# Scope

- It is often easier to say what isn't required, so you might want to rule-out certain things at the start.
- We can defer the things that are "out of scope" for this release.

# What are we building?

- Proof of Concept (PoC) - it doesn't need to be saleable, it needs to be demonstrable in order to attract funding.
- Minimum Viable Product (MVP) - someone must be willing to pay for it, but it doesn't need bells and whistles.
- Full product.

# Prioritising - Moscow Method

- Must have
- Should have
- Could have
- Won’t have (this time)

- This works very well with agile methods like Scrum

# Non-functional requirements

- We also need to specify some things about how the system should work, not only what it should do.
- e.g. performance, scalability, security, availability, capacity, etc.
- Think of these as qualities of the system.

# Knowing when you're done

- It is important to make your requirements measurable so that you can say when they are complete.
- This allows acceptance tests to say that you are done.
- In the testing lecture we'll speak more about validation, i.e. testing that we've built the right thing.

# Knowing when everything's done

University of Strathclyde Science

- In a sprint we can say we're done when everything marked as "must have" is done.
- In a large software system, when we have hundreds of requirements, how do we know we did them all?
- Traceability of requirements is important, so we should have a consistent numbering scheme to show requirements match acceptance tests.
- Version control and issue tracking numbers can help with this.

# Old School Requirements

- Elicitation - interviews, etc with the customer to see what is needed.
- Analysis - usually as part of documentation, build a consistent and structured set of requirements.
- Classically, systems analysts have done this work.

# Requirements and Scrum

- The product owner comes up with requirements.
- The user stories are the requirements.
- The product owner is the customer, so we are minimising the potential sources of conflict by working closely with the customer.
- This is trickier if the customer is not in-house. Do we need the product owner to “own” the negotiation/contractual aspects?

# User Stories

- As a …
- I need …
- So that …

# … and all the other bits

- What you've read about Scrum doesn't say anything about requirements documents. Which is fine if the customer is part of the team.
- When a story is done and moves out of the sprint backlog, where does it go?
- You'll want to track it so that you can do acceptance tests, also for things like the release notes.

# Summary

- Requirements tell us what we need to build.
- We need them so that all parties are agreed on what is needed.
- The customer usually starts to produce these, but developers are also involved.
- We're focussing on developing requirements in an agile environment where the customer is part of the team.

# Moving on from here …

- John McGuire of Pulsion supplied the project description on the next slide. The lab this week will expand on the requirements. You can send me more questions for John after that.
- On Thursday our practical is about taking the requirements gathered, doing some analysis of the requirements and entering them into a project backlog.
- Remember you'll be writing a report about this, so your blog entry will be a good source of info for your report.
- BTW the Pulsion website has a blog article that highlights the importance of requirements.

# Group Project Outline

University of Strathclyde Science

We want to develop a time booking system that can allow logged in users to:

1. Register Projects and Tasks.
2. Have user roles (admin or time booker).
3. Admins can assign users to projects/tasks.
4. Any assigned user can book time periods against the project/task with notes (on what was done) plus the start time and end time of the time booked.
5. Users should have to login and only be allowed to book to project/tasks they have been assigned to.
6. The system should allow web and mobile access. A disconnected mobile app would be an interesting addition.

University of Strathclyde Glasgow