# Requirements, Specifications and Documentation - oh my!

Joshua Lewis

School of Electrical and Information Engineering
University of the Witwatersrand

2015

# Today's topic

What is the best way to communicate software requirements?

# Let's get started

Think about what you already know about 'communicating and documenting requirements'.
Write 3 of these things down and be ready state them when asked.
Also write down either 1 question you want answered or one thing want to be able to do as a result of this lecture.

# Class Exercise

## Conway's Game of Life

Choose one person to be the Product Owner, who will communicate the requirements of our implementation of Conway's Game of Life to the rest of the team, the Developers.

# Motivation

## Why do we need to communicate requirements?

# Why do we need to communicate requirements?

We're not experts in the domain

- Need to know enough to code the domain
- Transfer of knowledge

# Why do we need to communicate requirements?

We're not experts in the business

- How will users use the system?
- Customer value
- Market

# Why do we need to communicate requirements?

Customers don't know what they want
(despite what they think or tell you)

- Help them make up their mind
- Make it cheap, easy and quick to change

# Why do we need to communicate requirements?

Software is often long lived

- Help yourself in the future
- Help others in the future

# Motivation

*We want to create a shared understanding of what's required, so that we can build the right system, now and in the future.*
*We need to ensure that when 'the right thing' changes, that we can change the system so it still does the right thing, and that we don't break anything that shouldn't change.*

# Common problems

What problems have you had with requirements documents?

# Common problems

- Ambiguous, open to interpretation
- Lack of shared understanding
- Edge cases
- People don't know what they want
- Defined in isolation and 'given'
- Include implementation or design
- Difficult to change, so fall out of sync

# What we really need

- Effective communication - shared understanding
- Shortcut to idea refinement
- Destination beacon - what does the right system look like?
- Documentation - maintain understanding over time
- Recognise different needs over time, for different people

# Good requirements

What are characteristics of good requirements?

# Good requirements I

- Convey intent and goals of system
- Unambiguous
- Comprehensive
- Understood by all stakeholders
- Consistent

# Good requirements II

- Modifiable
- Verifiable
- Traceable
- Up-to-date
- Don't include implementation or design

# Documenting requirements

The way in which requirements are documented must suit the *purpose*

- Different audiences
- Different levels
- Different times

# Techniques for documenting requirements

There are many different formats, techniques and methods for documenting requirements

- Lightweight to heavyweight
- Traditional to modern
- Formal to casual

# Techniques for documenting requirements

We will discuss three:

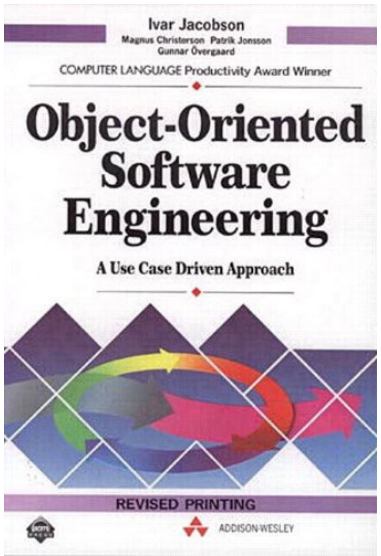
- Use Cases
- User Stories
- Specification by Example

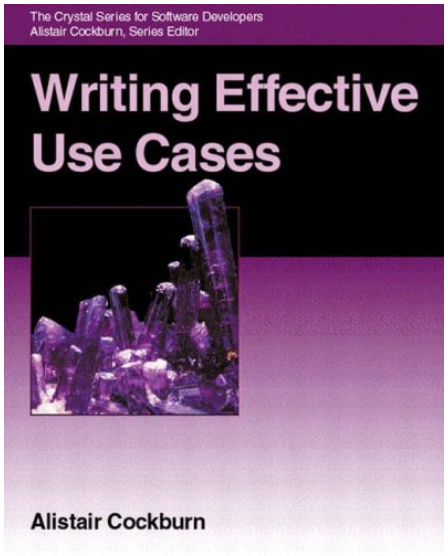
# Use Cases

## Use Cases

# Some context

- First described by Ivar Jacobson in mid 80s
- Important contributions from Alistair Cockburn



# What is a Use Case?

- A contract between the stakeholders of a system about its behaviour
- Describes the systems behaviour as it responds to a request from one of the stakeholders, called the *primary actor*
- Use cases are best presented as prose (simple writing)

# Use Case Example I

**Primary Actor:**
**Scope:** Finance package ('package')
**Level:** User goal
**Stakeholders and Interests:** Purchaser - wants to buy stocks, get them added to the package portfolio automatically.
Stock agency - wants full purchase information.
**Precondition:** User already has package open.
**Minimal guarantee:** sufficient logging information that package can detect that something went wrong and can ask the user to provide details.
**Success guarantee:** remote web site has acknowledged the purchase, the logs and the user's portfolio are updated.

# Use Case Example II

**Main success scenario:**

1. User selects to buy stocks over the web.
2. Package gets name of web site to use (E*Trade, Schwabb, etc.) from user.
3. Package opens web connection to the site, retaining control.

etc. . .

# Use Case Example III

**Extensions:**

2a. User wants a web site package does not support:

2a1. System gets new suggestion from user, with option to cancel use case.

etc. . .

# Classification

How would you classify Use Cases?

- Lightweight or heavyweight?
- Traditional or modern?
- Formal or casual

# Evaluation

## What do you think of Use Cases?

- Convey intent and goals of system
- Unambiguous
- Comprehensive
- Understood by all stakeholders
- Consistent
- Modifiable
- Verifiable
- Traceable
- Up-to-date
- Don't include implementation or design

# Evaluation

## What do you think of Use Cases?

- Convey intent and goals of system
- Unambiguous
- Comprehensive
- Understood by all stakeholders
- Consistent
- Modifiable
- Verifiable
- Traceable
- Up-to-date
- Don't include implementation or design

# Pros

## Pros of Use Cases

- Good for understanding intent and goals
- Possible flows (extensions)
- Finding stakeholders
- Other actors and systems

# Pros

## Pros of Use Cases

- Good for understanding intent and goals
- Possible flows (extensions)
- Finding stakeholders
- Other actors and systems

# Cons

## Cons of Use Cases

- How do you test? (Verifiability)
- Planning - breaking down work, estimating, allocating
- Prose - ambiguity
- Longer - harder to write and maintain

# Cons

## Cons of Use Cases

- How do you test? (Verifiability)
- Planning - breaking down work, estimating, allocating
- Prose - ambiguity
- Longer - harder to write and maintain

# Use Cases and Game of Life

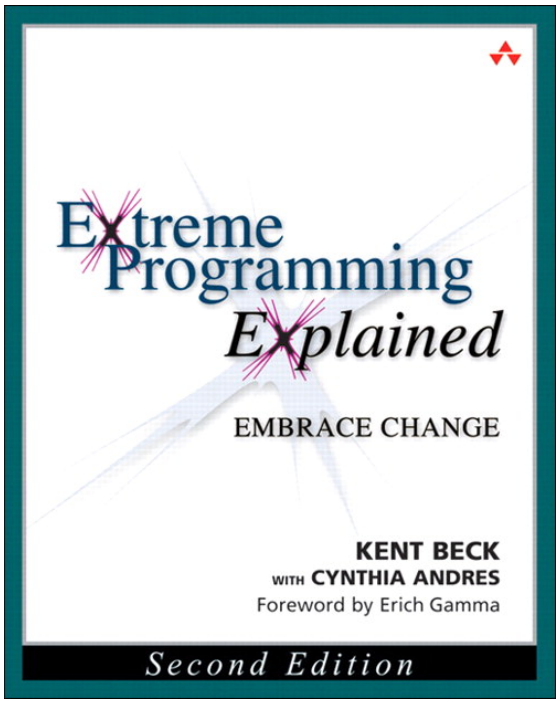
## Use Cases and Game of Life?

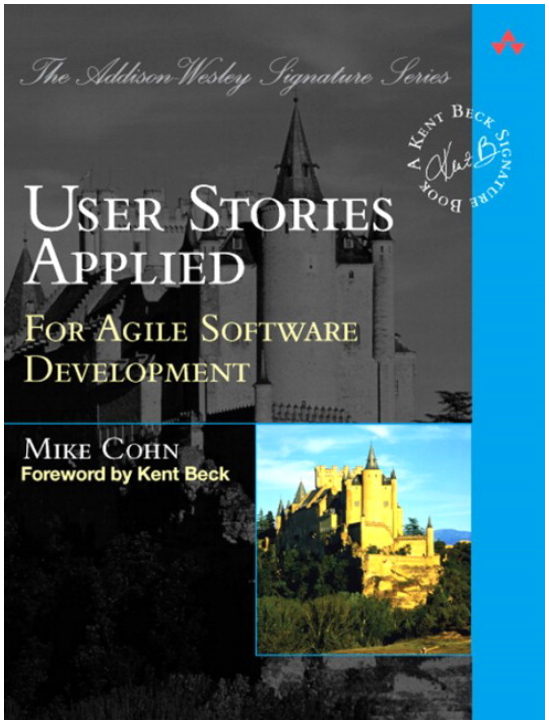
# User Stories

## User Stories

# Some context

- First described by Kent Beck in late 90s as part of XP
- Important contributions by others over time



# What is a User Story?

## Definition

Brief, textual description in business language that captures what a user needs to do as part of his job

- Not intended to be complete - a promise for later conversation
- Used for planning - sizing, estimation, allocation etc
- Must be accompanied by success and failure criteria

# User Story Example

*A company can pay for a job posting with a credit card*

— User Stories Applied, Mike Cohn

# Classification

How would you classify User Stories?

- Lightweight or heavyweight?
- Traditional or modern?
- Formal or casual

# Evaluation

## What do you think of User Stories?

- Convey intent and goals of system
- Unambiguous
- Comprehensive
- Understood by all stakeholders
- Consistent
- Modifiable
- Verifiable
- Traceable
- Up-to-date
- Don't include implementation or design

# Evaluation

## What do you think of User Stories?

- Convey intent and goals of system
- Unambiguous
- Comprehensive
- Understood by all stakeholders
- Consistent
- Modifiable
- Verifiable
- Traceable
- Up-to-date
- Don't include implementation or design

# Pros

## Pros of User Stories

- Short, lightweight - easy to maintain
- Conveys user goals well - derive value
- Includes verifiability if done right
- Promotes conversation

# Pros

## Pros of User Stories

- Short, lightweight - easy to maintain
- Conveys user goals well - derive value
- Includes verifiability if done right
- Promotes conversation

# Cons

## Cons of User Stories

- Easy to leave incomplete
- Potential lack of context
- May not scale well for large projects

# Cons

## Cons of User Stories

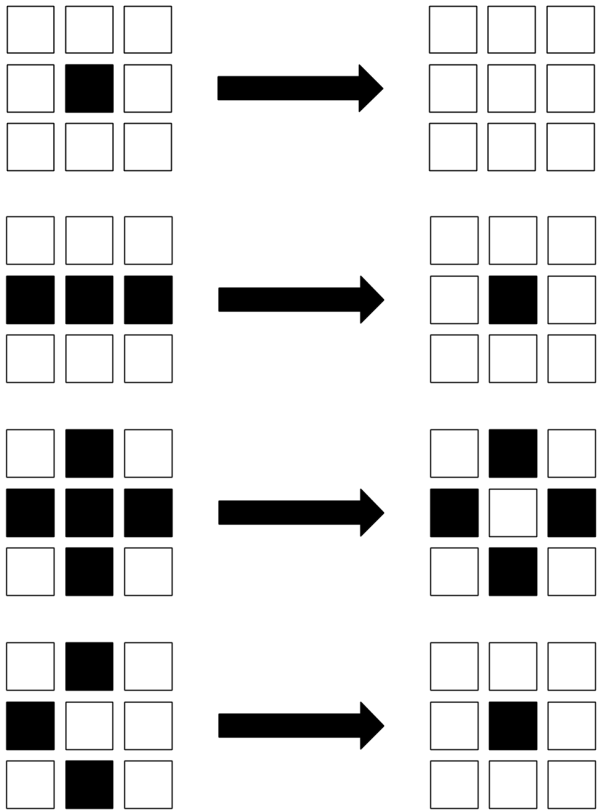
- Easy to leave incomplete
- Potential lack of context
- May not scale well for large projects

# User Stories and Game of Life

Usee Stories and Game of Life?

# User Stories and Game of Life

## Success Criteria?

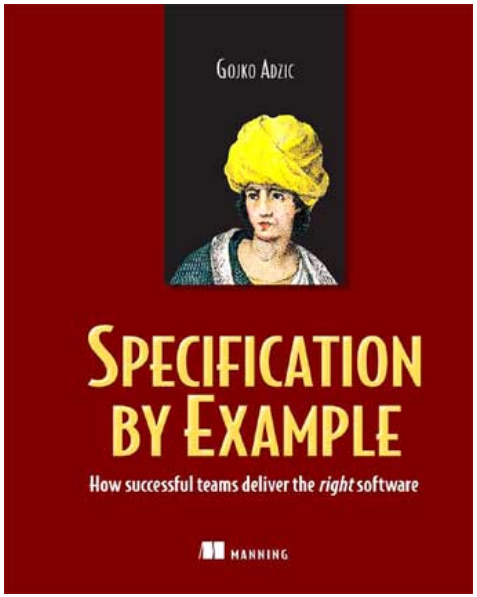


# Specification by Example

## Specification by Example

# Some context

- Evolved out of several testing practices, especially from XP
- Term coined in early 2000s
- Gojko Adzic's book written in mid 2011



# What is Specification by Example?

> *Specification by example is a* **collaborative** *approach to defining requirements and business-oriented* **functional tests** *for software products based on capturing and illustrating requirements using* **realistic examples** *instead of abstract statements.*
>
> — Wikipedia

Similar concepts and alternative names:

- Executable specification
- Acceptance-test Driven Development

# Specification by Example - Example

**I want to** *pay for my purchase by credit card* **as a** *customer of Amazon.com* **so that** *I don't have to go in to my bank* **and** *I can buy on credit*

**Given** I have items in my shopping cart
**And** I have completed my shipping details
**When** I complete the purchase
**Then** The order confirmation is shown
**And** Amazon attempts to debit my card
**And** An email confirming the order is sent to me

# Classification

How would you classify Specification by Example?

- Lightweight or heavyweight?
- Traditional or modern?
- Formal or casual

# Evaluation

What do you think of Specification by Example?

- Convey intent and goals of system
- Unambiguous
- Comprehensive
- Understood by all stakeholders
- Consistent
- Modifiable
- Verifiable
- Traceable
- Up-to-date
- Don't include implementation or design

# Evaluation

What do you think of Specification by Example?

- Convey intent and goals of system
- Unambiguous
- Comprehensive
- Understood by all stakeholders
- Consistent
- Modifiable
- Verifiable
- Traceable
- Up-to-date
- Don't include implementation or design

# Pros

## Pros of Specification by Example

- Promotes conversation and collaboration
- Applicable at various levels
- Living documentation
- Verifiable
- Unambiguous
- Maintainability
- Automatable

# Pros

## Pros of Specification by Example

- Promotes conversation and collaboration
- Applicable at various levels
- Living documentation
- Verifiable
- Unambiguous
- Maintainability
- Automatable

# Cons

## Cons of Specification by Example

- Needs a lot of collaboration - may be hard
- ?

# Cons

## Cons of Specification by Example

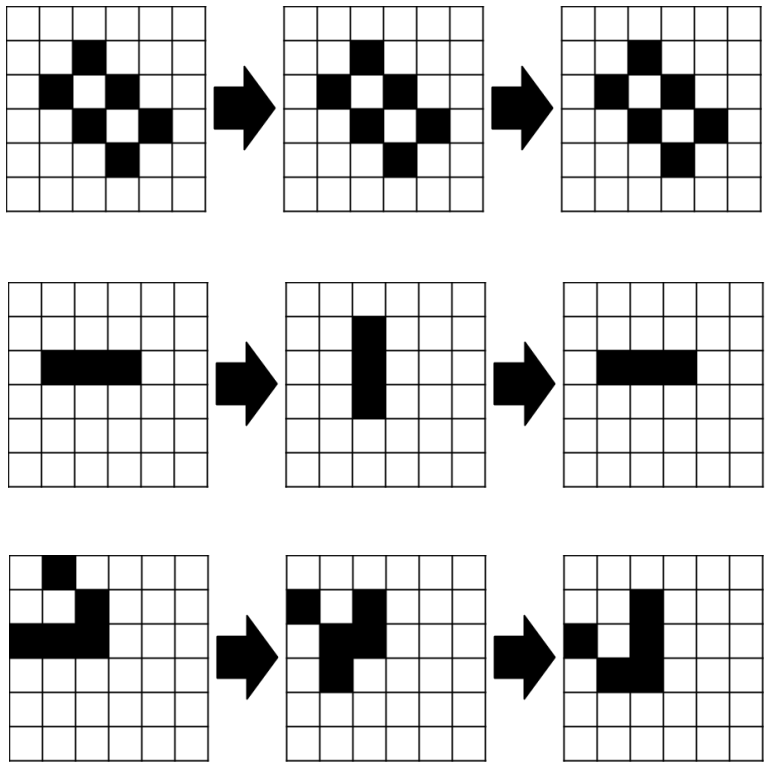
- Needs a lot of collaboration - may be hard
- ?

## Specification by Example and Game of Life

# Specification by Example and Game of Life?

# Specification by Example and Game of Life

Specification by Example and Game of Life



# Let's wrap up

Choose 2 of the questions below to answer:

- What are the most important concepts we discussed today?
- How will you use this information?
- What behaviour changes will you make as a result of what we discussed?
- What other steps can you take to learn more about what we discussed?
- Who can you share what you learned with and what would you tell him or her?
- What is one question you still have about this topic? How will you find the answer?

Form standing groups of 3-5 with people from other groups and read your written answers out loud

# Conclusion

- Good understanding of requirements is critical
- Many methods and approaches for specifying and documenting requirements - choose what's applicable
- Don't take anything as gospel, use what works for you - experiment

# Conclusion

- Good understanding of requirements is critical
- Many methods and approaches for specifying and documenting requirements - choose what's applicable
- Don't take anything as gospel, use what works for you - experiment