Department *of* Computing
COVENTRY UNIVERSITY

302CEM

Agile Planning

Mark Tyers 2016

Coventry University

Requirements Gathering

Coventry University

Requirements

The features your application will provide

Gathered from your customers

Used to guide the development process

Are you heading in the right direction?

Used to verify the finished product does as it should

Features of Good Requirements

Clear

Unambiguous

Consistent

Prioritised

MoSCoW Method

User stories fit into four different priority groups:

- Must Have: the top items will be essential
- Should Have: next there will be the important but lower priority tasks
- Could Have: next are the low priority features that will be completed eventually
- Would Be Nice: the tasks at the bottom may never be implemented

THEORY

User Requirements

Unit of requirements gathering is the "user story"

User-visible functionality that can be developed within one iteration or less.

Planning

Business people decide:

- scope
- priority of features
- composition of releases
- date of releases

Development team decides

- estimates
- technical consequences
- process
- detailed scheduling

Planning Phases

Planning has three phases:

- Exploration—learn what the system will eventually do: use cases (stories), estimates
- Commitment—choose the scope and date of next release: business sorts by priority, development sorts by how well they can estimate, business sets the scope
- Steering—update the plan: iterations last one to three weeks, new uses cases as needed, re-estimate

XP Planning Works

The result is what some would consider to be only a rough plan.

It works because:

- the customers were involved
- the release intervals are short, so mistakes become apparent
- the customers continue to work with the development team
- the plan is revised as needed

Simple Design

"The simplest thing that could possibly work.“

Start with a test, so you will know when you are done.

Design and implement enough to make the test run.

Simplify the design if the opportunity arises.

Product Owner

The development team must be aligned to the business

Entire team must focus on a single problem domain

One member of the team must be dedicated to the product

They must have a clear understanding of the requirements of the problem domain

They are responsible for prioritising work (user stories)

They should discuss these with the team but they have to take the final decision

Key Steps

Requirements gathering

High-level design

Low-level design

Development

Testing

Deployment

Maintenance

Planning Documentation

Coventry University

Specification Documents

Functional Specification

- Describe functionality from the user's perspective

Design Specification

- Describes how the software meets the functional specification

Form the basis for your product development

Common Content

Purpose and scope
Intended audience
Assumptions
Dependencies
Risks
Constraints
Goals
Guidelines
References
Glossary
Revision history

Functional Specification

Product architecture

Development methods

Features (ID, status, owner)

User groups and roles

Design Specification

Interfaces

Policies

Dependencies

System Design

- Type (class, file, function, etc.)
- Description
- Function (behaviour)
- Composition (describe sub-components)
- Limitations (related constraints)
- Interface

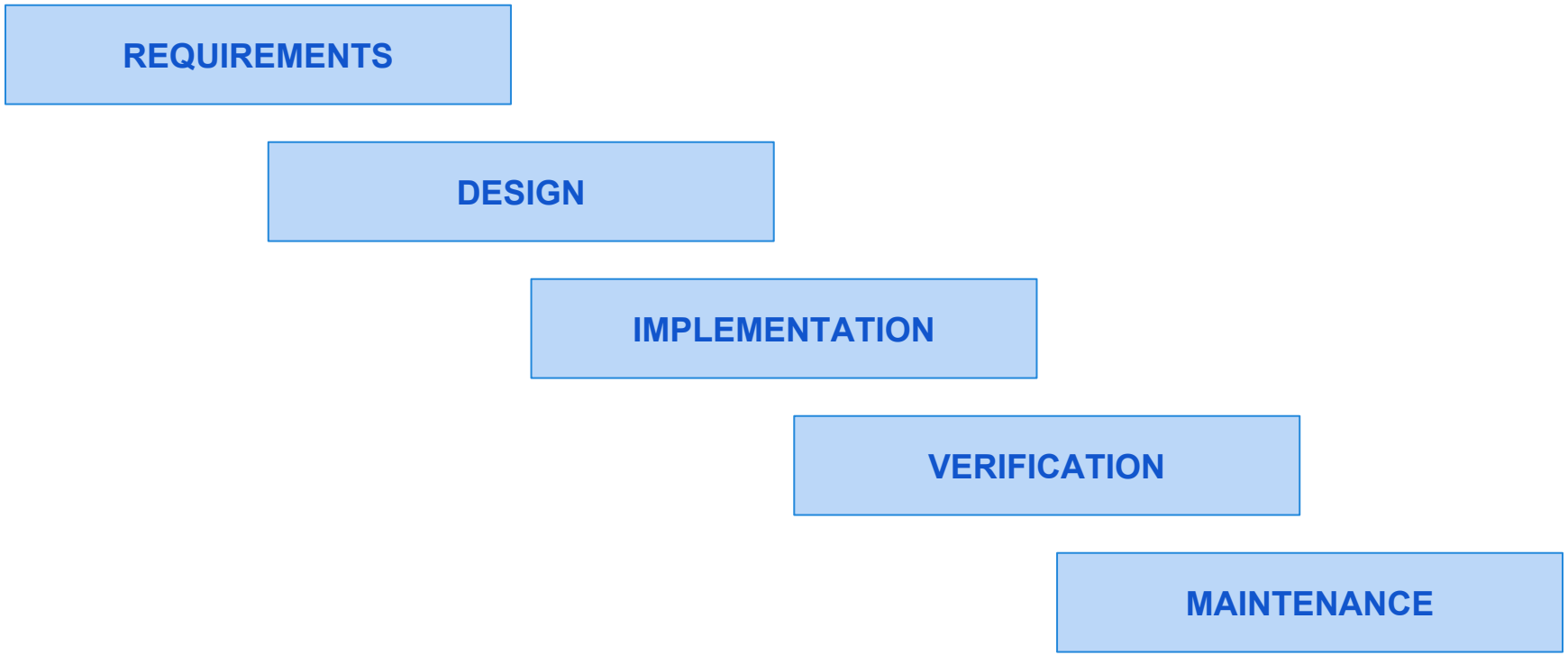
Capturing Customer Requirements

Together these capture the customer requirements

By capturing these detailed requirements the product can be developed in an efficient and timely manner

BRUF (Big Requirements Up Front)

Develop a Detailed Plan



Except it's complete rubbish...

Challenges

Coventry University

Detailed Specifications

Customers rarely get what they want

Teams rarely build what is needed

Huge amounts of time is wasted

What's wrong with detailed specifications?...

They Can't Handle Change

Agile Samurai

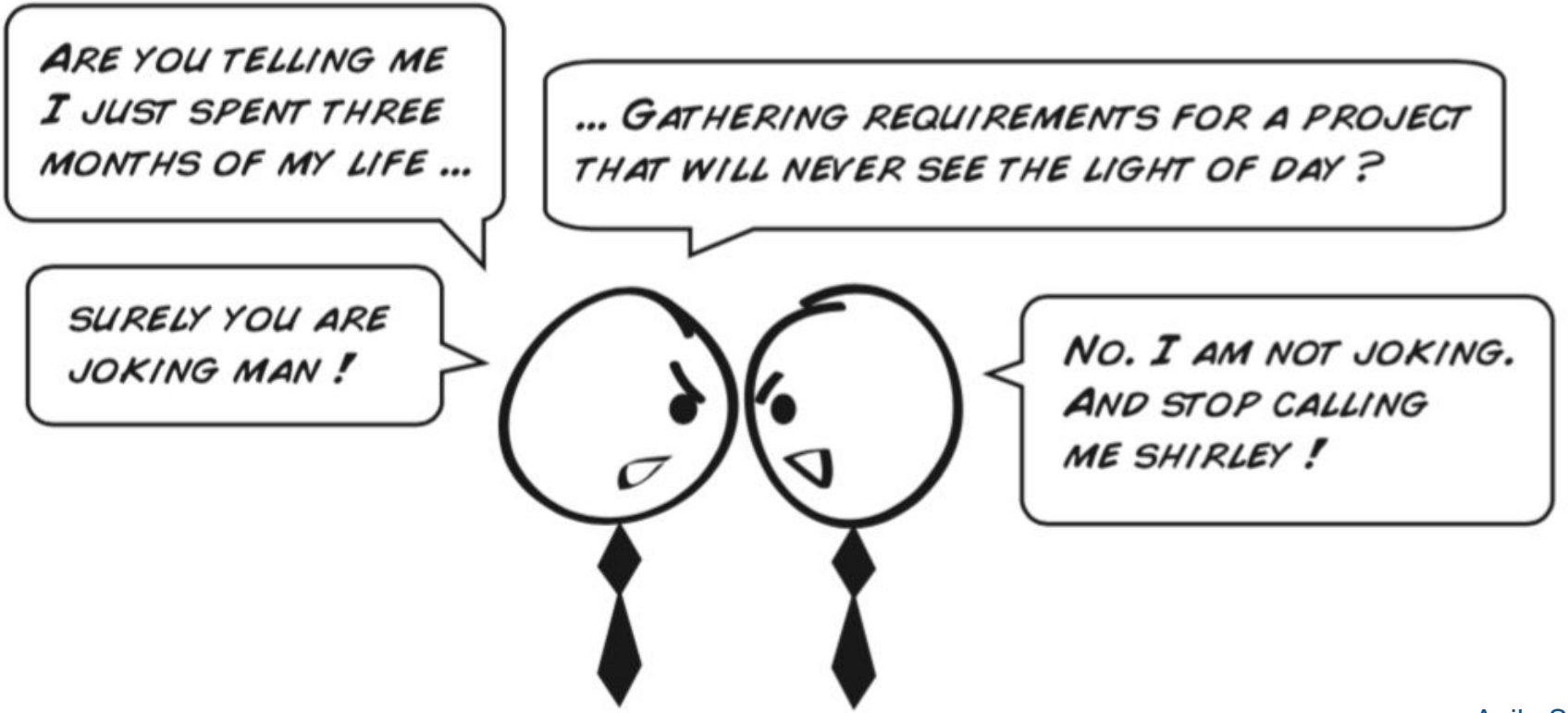
They Build to Spec, Not Wants

Agile Samurai

Bad Guesses and False Assumptions

Agile Samurai

They Waste a Lot of Time



Agile Samurai

So No Documentation...

Still a need for documentation:

- Shared understanding of the project
- Knowledge-base

But:

- Live documents
- Focus on building the software (don't get distracted)

Document Management

Documents are important in an agile project

Many of these are 'living documents' (including emails)

How will your team manage these?

- Cloud-based office suite?
- DVCS?

How will you manage changes?

Who is the final arbiter?

Agile Development

Coventry University

Benefits of Agile

Deliver on time and budget

Deliver a high quality product

Deliver maintainable code

Make the client happy

Work happy

THEORY

What is Agile?

Set of methods and methodologies

Made up of a number of good practices

Help a team think and work more effectively

A mindset to improve team communication (effectiveness)

Only effective if the team's mindset shifts

Agile

The ability to create and respond to change in order to profit in a turbulent business environment.
Companies need to

- innovate better and faster
- respond quickly to
  - competitive initiatives
  - new technology
  - customer's requirements

Agile Practices

MacCormack identified four practices that lead to success:

An early release of the evolving product to the customer.

Getting rapid feedback from the customer and incorporating that feedback into new design experiments.

A team structure that will allow the right decisions to be made on the fly.

Choosing a product architecture that allows for change rather than attempting to get optimal performance

THEORY

Agile Manifesto

Individuals and interactions over processes and tools

Working software over comprehensive documentation

Customer collaboration over contract negotiation

Responding to change over following a plan

THEORY

Principles

The highest priority is to satisfy the customer through early and continuous delivery of valuable software.

Changing requirements are welcome, even late in development. Agile processes harness change for the customer's competitive advantage.

Working software is delivered frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.

Business people and developers must work together daily throughout the project.

Projects should be built around motivated individuals. If the right environment and support is provided, the developers can be trusted to get the job done.

THEORY

Principles

The most efficient and effective method of conveying information to and within a development team is face-to-face conversation

Working software is the primary measure of progress.

Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.

Continuous attention to technical excellence and good design enhances agility. Simplicity, the art of maximizing the amount of work not done, is essential.

The best architectures, requirements, and designs emerge from self-organizing teams.

At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

THEORY

Managing Risk

Schedule slips - Short release cycles

Project cancelled - Smallest release that makes sense

System goes sour - Maintain a suite of tests

Defect rate - Testing by programmers and customers

Business misunderstood - The customer is part of the team

Business changes - Short release cycles

False feature rich - Address only the highest priority tasks

THEORY

Agile Methodologies

Scrum

Extreme Programming (XP)

Dynamic Systems Development Method (DSDM)

Crystal Methods

Feature-Driven Development (FDD)

Lean Development (LD)

Adaptive Software Development (ASD)

THEORY

User Stories

Coventry University

User Stories

The unit of requirements gathering is the "user story", which is user-visible functionality that can be developed within one iteration or less.

Customers write the stories.

Customers and programmers negotiate what will get done in the next iteration. This is known as the "planning game".

User Stories

Short descriptions of features the customer would like

Written on index cards

Features of User Stories

Captures the spirit

Ignores details

Make sense to customer

Delivers value to customer

End to end (full stack)

Independent

Testable

Small (1-5 days) so easy to estimate

THEORY

Format of User Stories

As a <user type>, I want to <function> so that <benefit>

As a consumer, I want shopping cart functionality to easily purchase items online.

As an executive, I want to generate a report to understand which departments need to improve their productivity.

The Five 'Whys'

Need to ensure the user stories fit the business outcomes

Address the root cause of the business.

Apply the 5 whys

- Process developed by Sakichi Toyoda
- Avoids assumptions and logic traps
- Traces the chain of causality

Carrying Out The 5 Whys

For each story ask why it is important

Take this answer and ask why the answer is important

Repeat with this answer until you have asked the question 5 times

Now you have the root purpose...

Use Frequency

How frequently will the feature be used?

Important information to include in the user stories.

FREQUENCY

High, Medium, Low or

Hourly, Daily, Weekly, Monthly, Quarterly.

Value to User

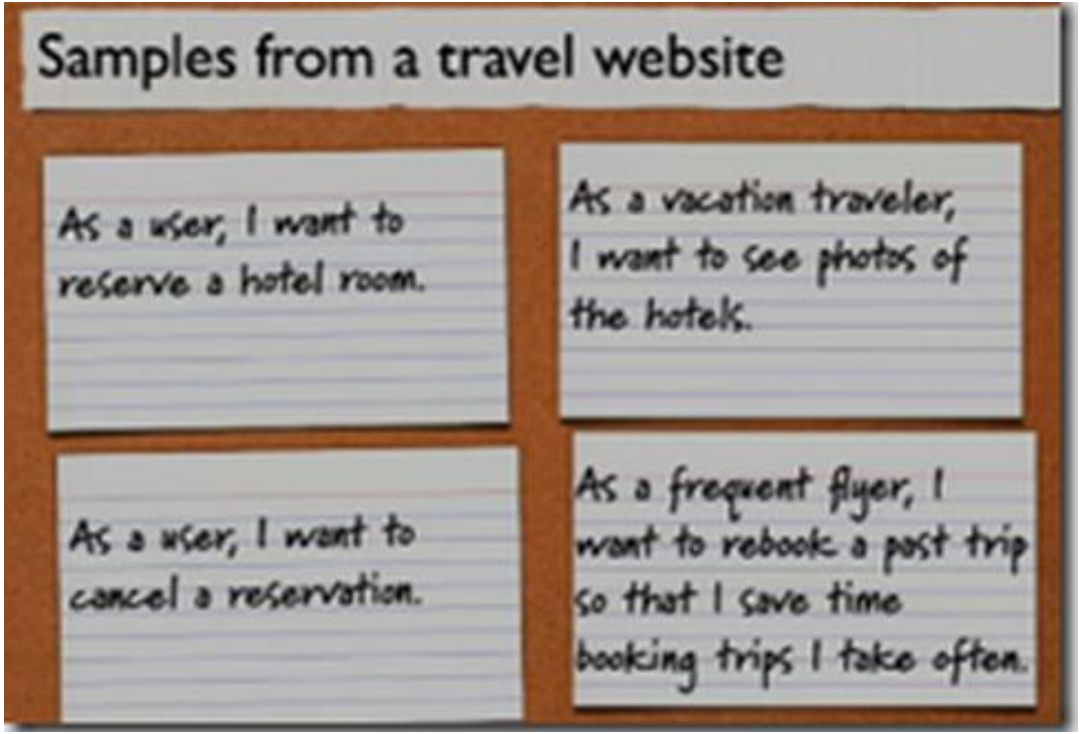
How valuable is this story to the user?

Also needs to be added to the user stories.

VALUE:

High, Medium Low

Examples


Samples from a travel website
As a user, I want to reserve a hotel room.
As a vacation traveler, I want to see photos of the hotels.
As a user, I want to cancel a reservation.
As a frequent flyer, I want to rebook a past trip so that I save time booking trips I take often.

INVEST

Independent

Negotiable

Valuable

Estimatable

Small

Testable

THEORY

Master Story List

The stories that define the end product

Should define the feature-complete system

Gives the development its sense of purpose and direction

THEORY

Gherkin

Most popular business-readable DSL for BDD

Over 60 (human) languages supported

Written in plain text

Located in a `features/` directory in the project root

Saved with the extension `.feature`

THEORY

Documenting using Gherkin

You should document your user stories using the Gherkin syntax (more on this in later weeks).

- Create a `features/` directory at the root of your repository
- Create a file for each user story with a `.feature` file extension
- Write up the user story
- Use an automated tool to generate documentation for the client

Example Feature File

```
# features/search.feature

Feature: Search

 As a University student

 I want to search for when academics are free

 So that I know when I can get help
```

Rendering the Scenarios

Tools available to render the `.feature` files

Most popular format is to `html`

Output is a nicely formatted static website

Can be shared with customers

Recommendation is to save html docs in subdir of docs/

Means it can be displayed in github as html page.

Popular one is a NodeJS module called `gherkindoc`

NodeJS Documentation Builder

```
// testRunner.js
const gherkindoc = require('gherkindoc')
// use the features/ directory as the input
// save pages in the /docs/features/ directory
gherkindoc.generate('features', 'docs/features/')
```

Detail of Use Cases

What level of goal are we trying to describe?

Alastair Cockburn in Writing Effective Use Cases

Easy way to visualise the goal level by thinking in terms of the sea

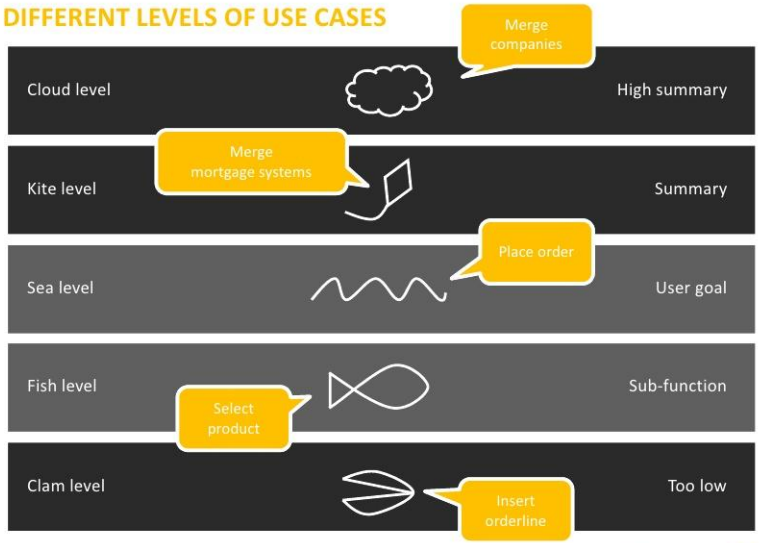
- Sea level represents user goal level
- Sky is high level
- Below the surface is lots of detail

THEORY

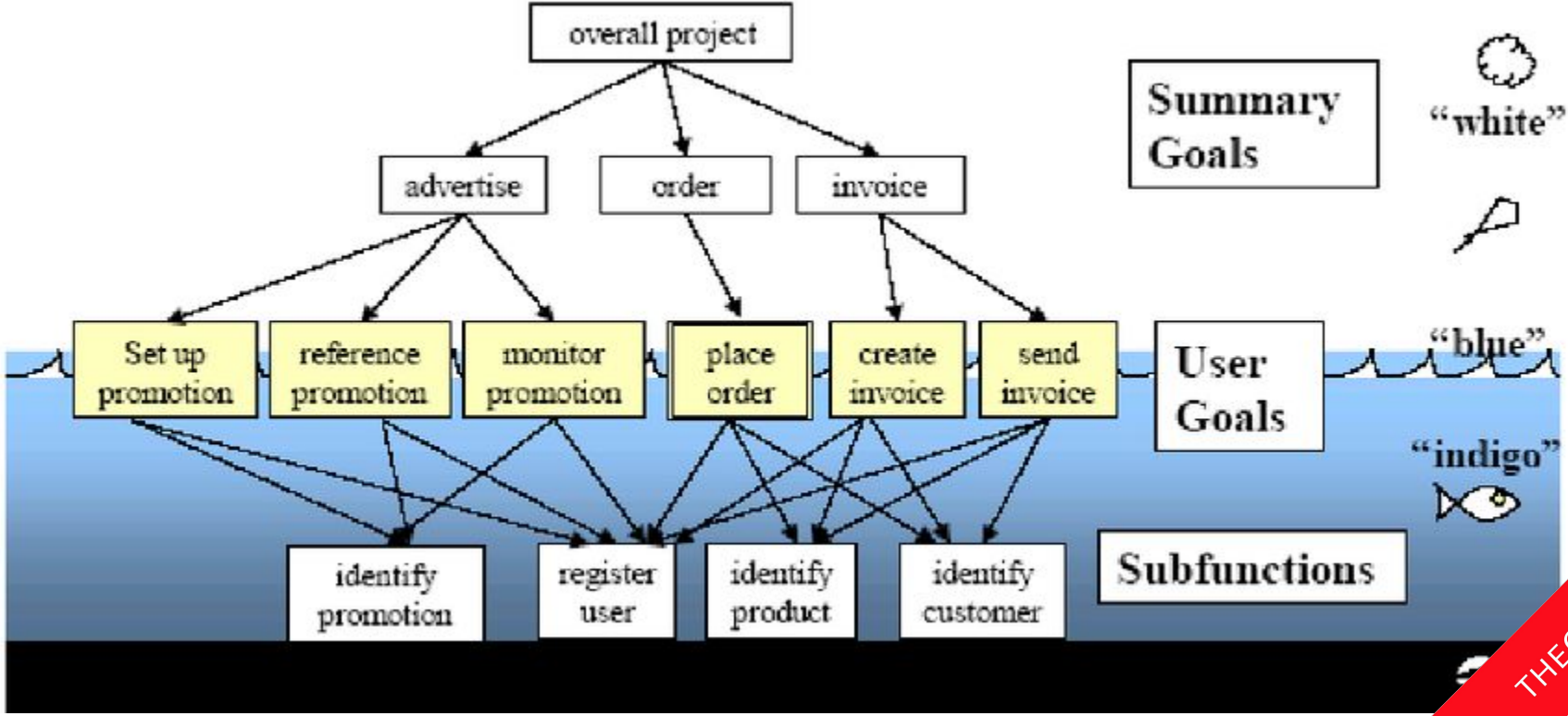
Smart Use Cases

How much detail is needed in user stories?

Alistair Cockburn has a metaphor to help us decide



Goal Levels



THEORY

Releases

A logical grouping of user stories

Make sense to the customer

Make sense to deploy as a group

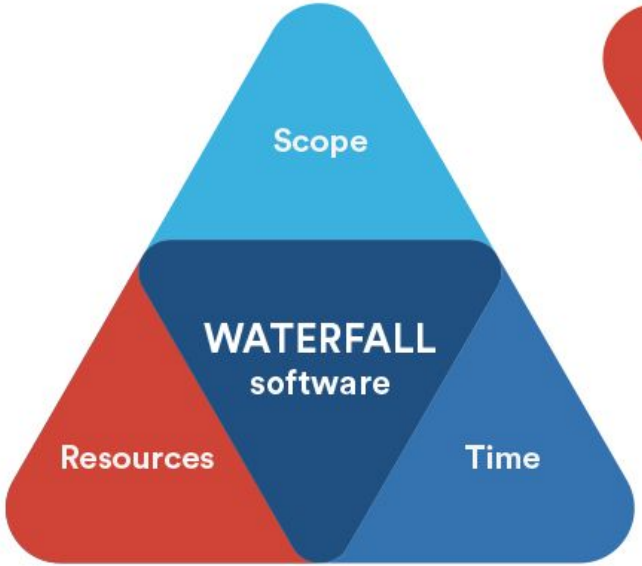
The Iron Triangle of Planning

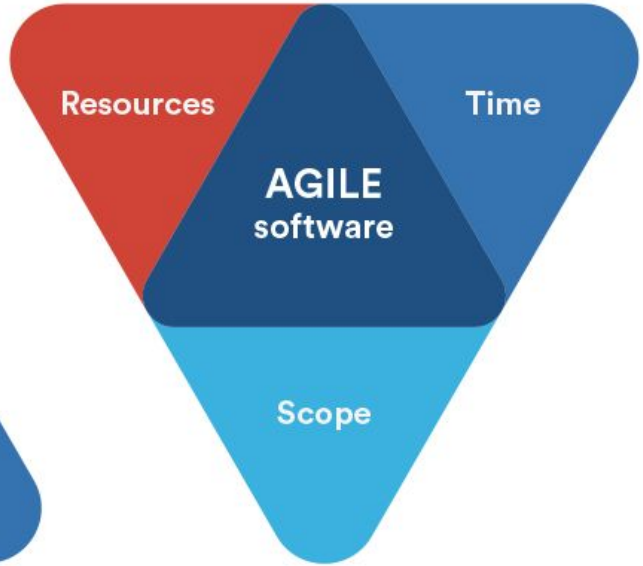
Three constraints in project management

- Scope (on spec)
  - The work to be done to deliver a working product.
- Resources (on budget)
  - The budget and team members working to deliver and execute.
- Time (on time)
  - When teams will deliver (releases and milestones).

Traditional v Agile Planning

Fixed





Estimated

Roadmaps

Most projects have a fixed budget, timescale and scope

How can you get away without a fixed scope?

Create a roadmap:

- A prioritised list of features
  - The most important features are at the top
  - Guarantee a minimum level of capability

Minimum Viable Product

Also Minimum Marketable Feature Set (MMF)

Need to deliver value quickly

Need to get early user feedback

Fewest number of features that deliver value to the customer

Comparison vs Specification

Just in time

Encourage face to face comms

Simplified planning

Cheap and fast

Never out of date

Based on latest data

Real-time feedback

Allow for innovation

Heavy, inaccurate and out of date

Encourage false assumptions

Complex planning

No real time feedback

Discourage collaboration

User Story Mapping (Agile Roadmaps)

Coventry University

Problems with User Stories

We can place the user stories in order of importance

Difficult to see the overall project plan

How will the user interact with the system

What will the process flow be?

Epics

Simple approach is to group stories together

These are normally based on software releases

Agile term is epics

User Story Mapping

User Story Mapping aims to solve these issues

Maps the user's journey through your product

builds a simple model that tells your user's story as you do

Benefits of User Story Mapping

Helps:

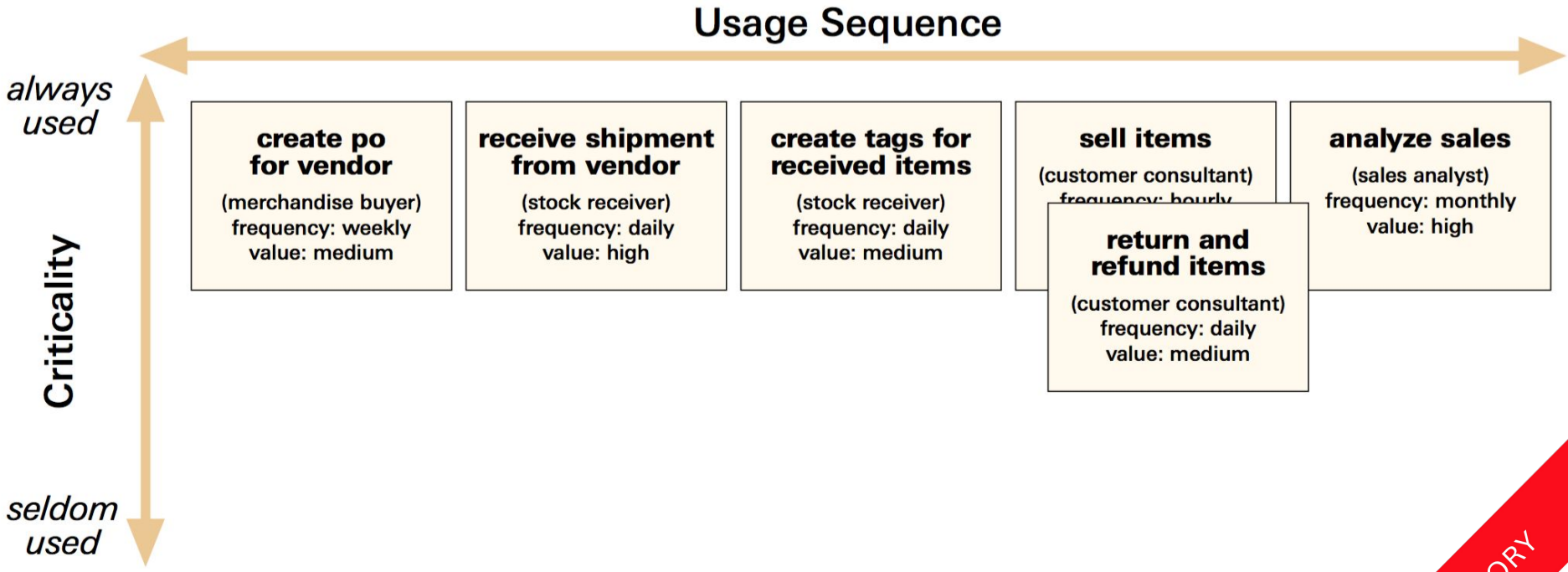
- Focus on the user requirements
- Avoid feature arguments

Identifies:

- Minimum viable product
- Releases
- Alternative stories

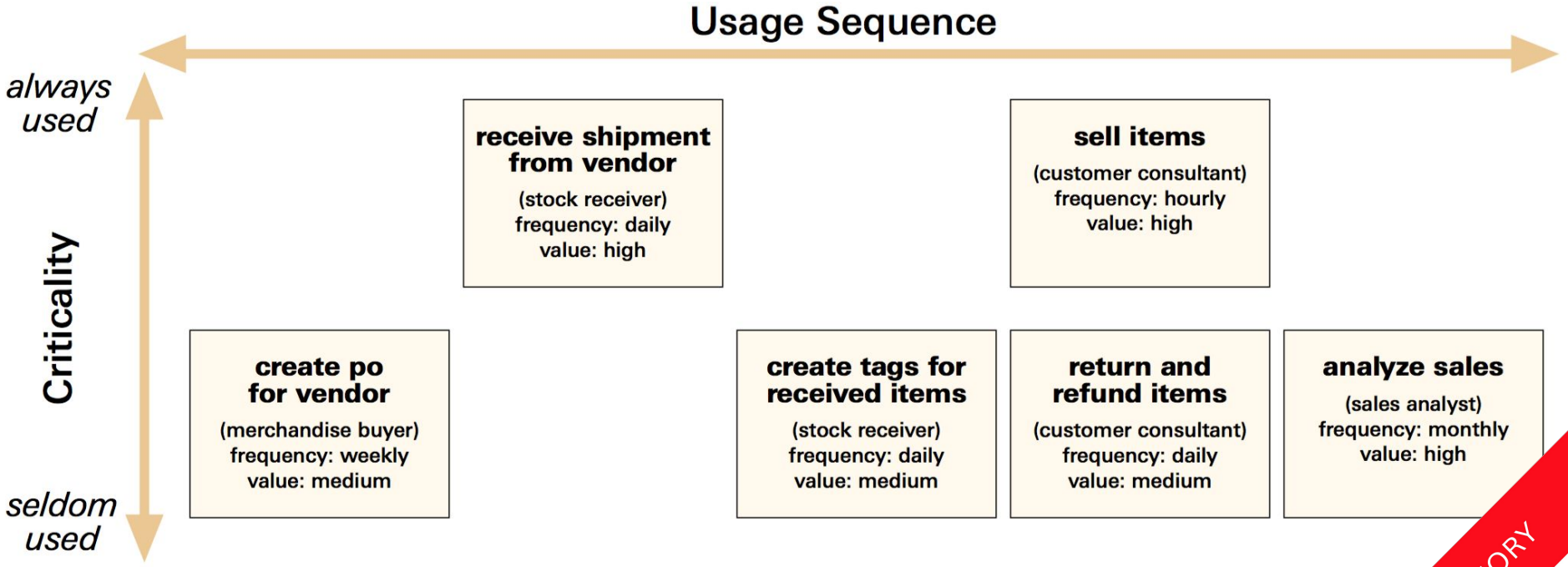
THEORY

Cards in Sequence



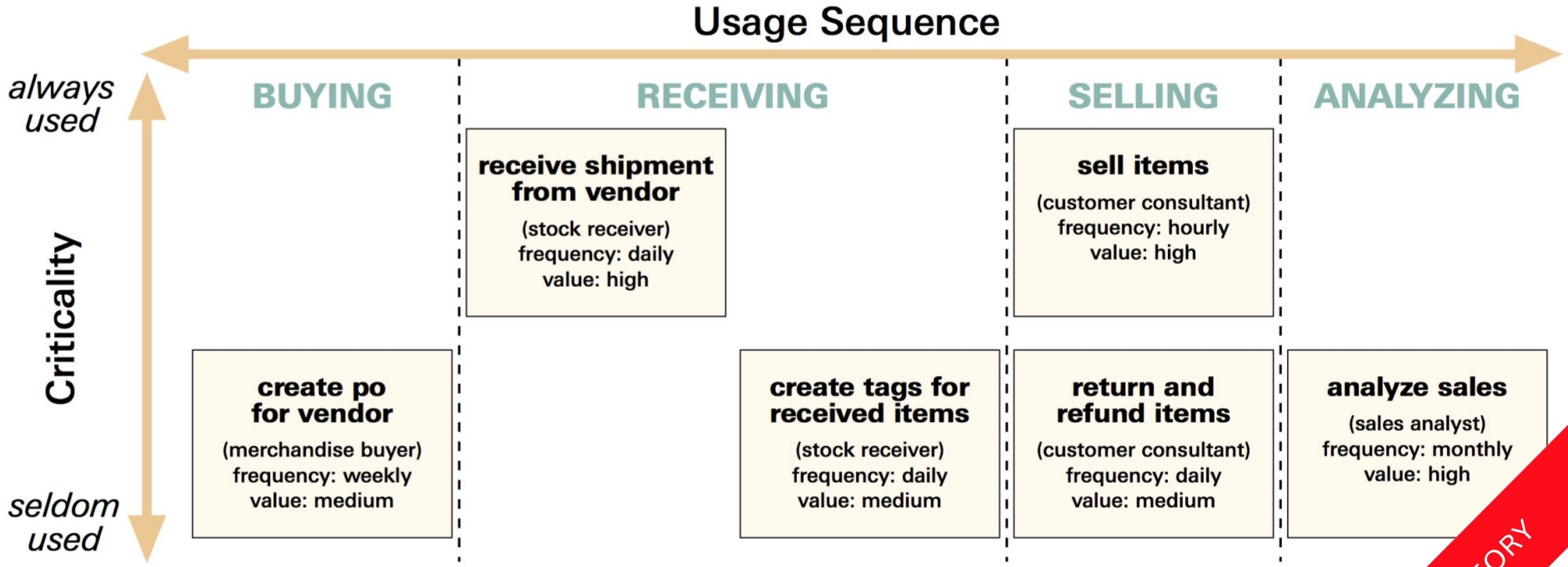
THEORY

Filtered by Criticality



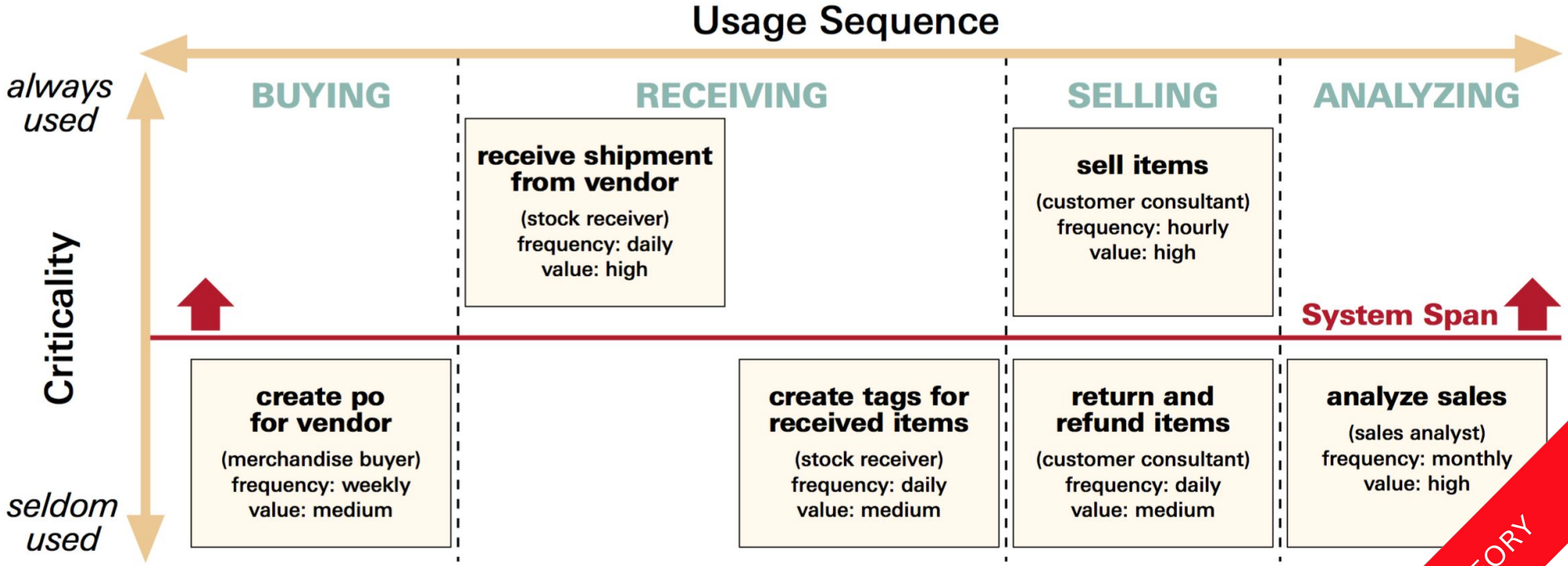
THEORY

Divided by Business Process



THEORY

Identifying Minimum Viable Product



Alternative User Stories

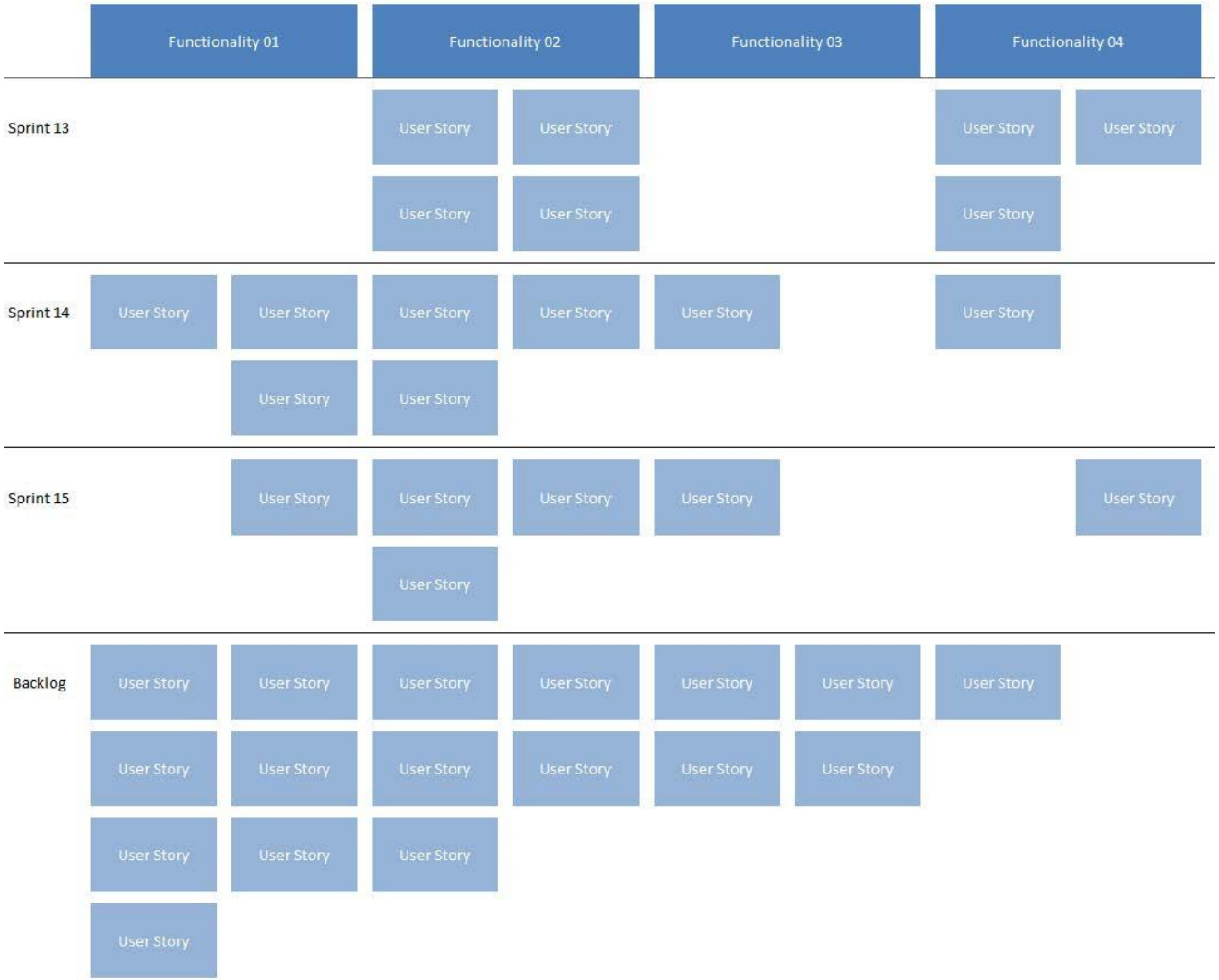
Most users will follow a similar flow through the product

- These should be implemented first

What about the alternative paths/edge cases?

- These need to be considered
- Implementation is a lower priority

THEORY


Functionality 01
Functionality 02
Functionality 03
Functionality 04
Sprint 13
Sprint 14
Sprint 15
Backlog
User Story

Alternative Stories

At this stage you have a narrative flow left to right

These form the routes taken by most users

What about other routes:

- Less Common Actions
- Errors

These need to be added to your User Story Map

THEORY

Information Radiator

Generic term for handwritten, drawn, printed or electronic displays

Placed in a highly visible location

Everyone can see the latest information at a glance

THEORY

Information Icebox

Information kept out of sight

Can be paper or electronic

No-one can see it so no-one updates it

THEORY

Recap

Coventry University

Recap

Requirements gathering

Planning documentation

Challenges

Agile development

User stories

Agile roadmaps

And finally… radiators and iceboxes

Meeting Reminder

All BIT stage 3 students

Important meeting Friday 3rd December

15:40 in ASG31

Essential you attend

Friday lab session will finished 10min early to allow for this

Week 2 Lab Activities

'Client' will need to attend

- Documentation
- Requirements gathering
- Agile roadmap
- Technical planning

Updated Team Reminder

Team membership needs to be updated on Moodle

Need to know which lab sessions will be attended

Each team email me today:

- Team name
- Names and email addresses of team members
- Dates and times of the labs to attend

Lab Schedule

	Mon	Tue	Wed	Thu	Fri
9		EC1-01	~~JL1-04~~		
10					
11	Lecture				
12		ASG-31			
13				EC1-01	
14	EC1-01				ASG-31
15					
16		ECG-14			

List and explain four problems with detailed specifications.

How does an agile methodology reduce risk?

Explain, using a diagram, the process of User story mapping

What are its benefits

Explain the following terms:

- Goal levels (Alistair Cockburn)
- Alternative stories

Explain the following terms used in agile development

1. Incremental development
2. Proof of Concept
3. Minimum viable product
4. Timeboxing

MoSCoW is an acronym often used in agile planning

1. Expand the acronym and explain it
2. Explain the concept of timeboxing
3. How does the MoSCoW method relate to timeboxing?

List the four points in the agile manifesto

For each, explain how it leads to better code

Tools

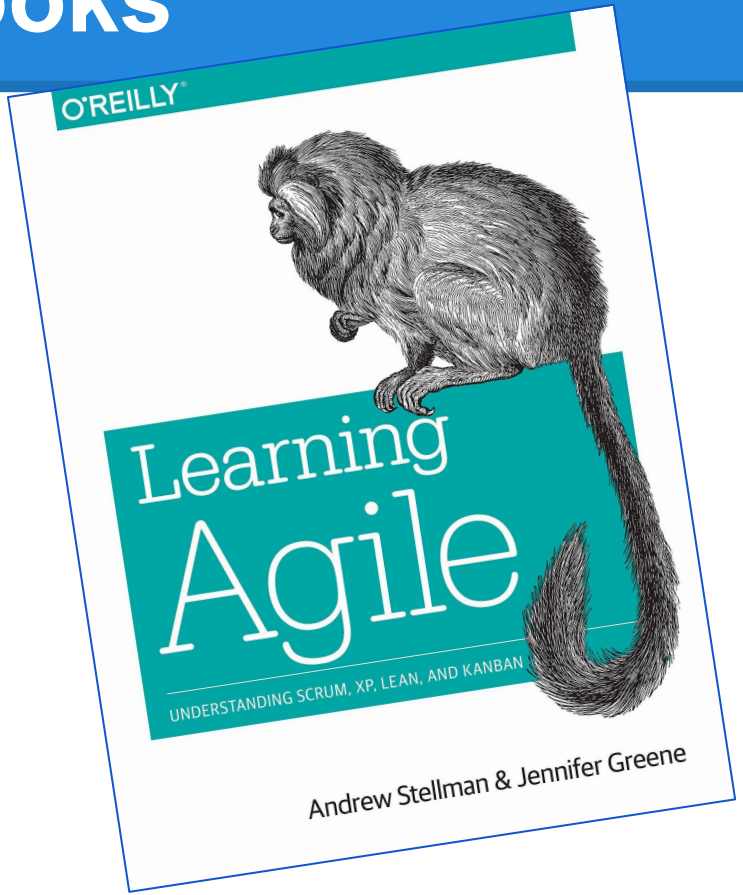
Stories on Board

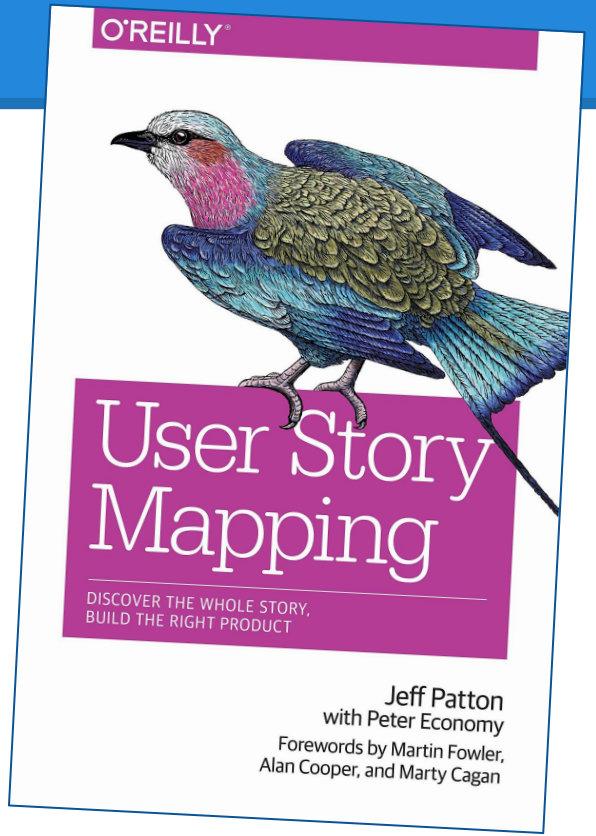
http://storiesonboard.com

Real Time Board

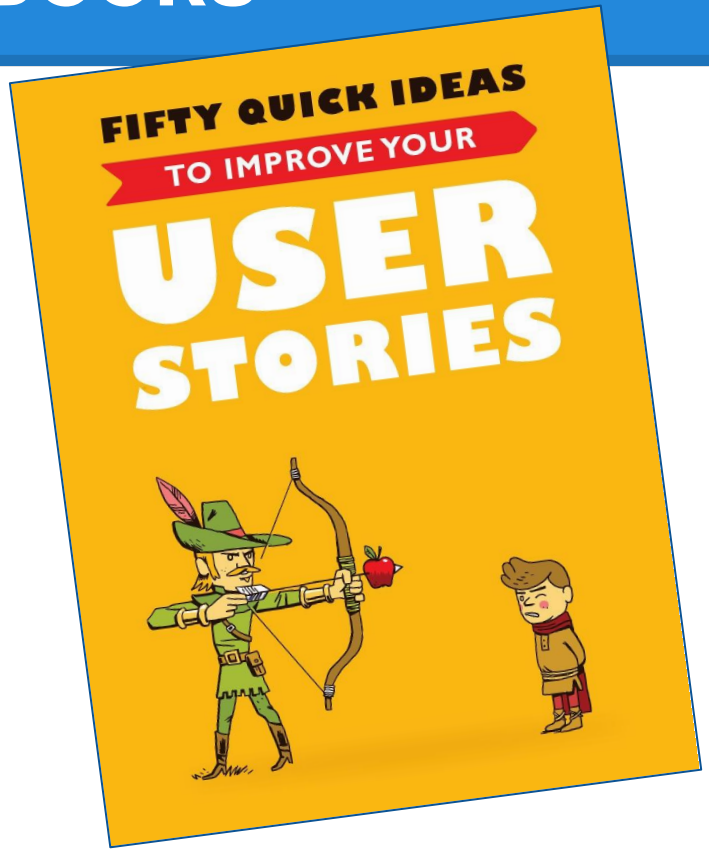
https://realtimeboard.com

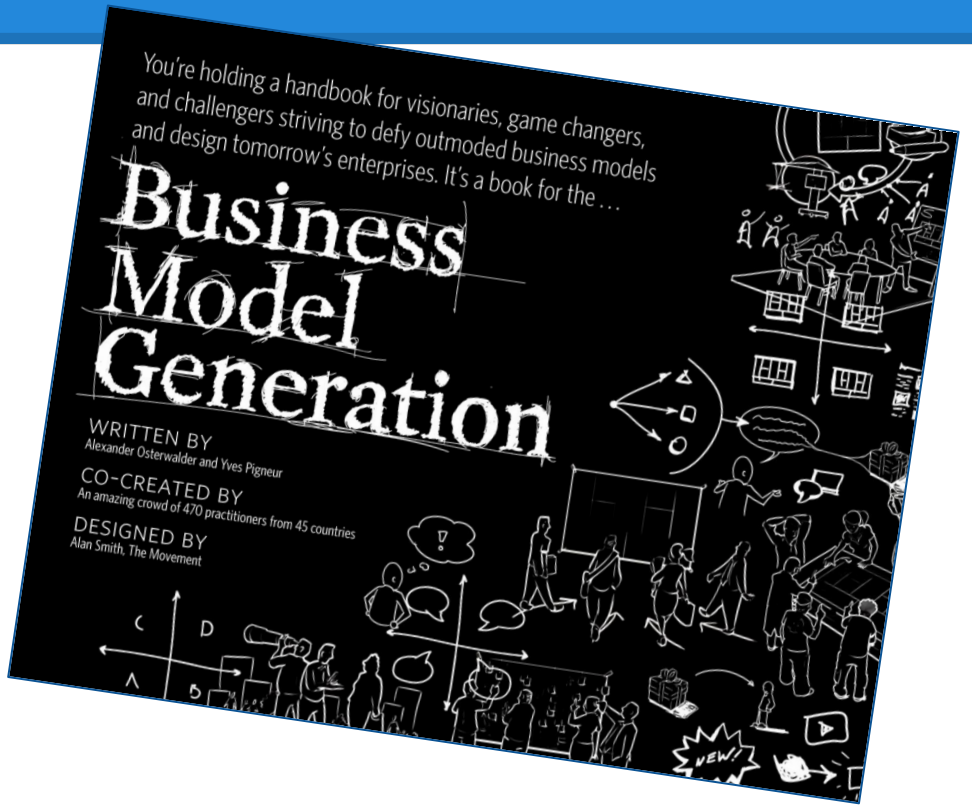
Books

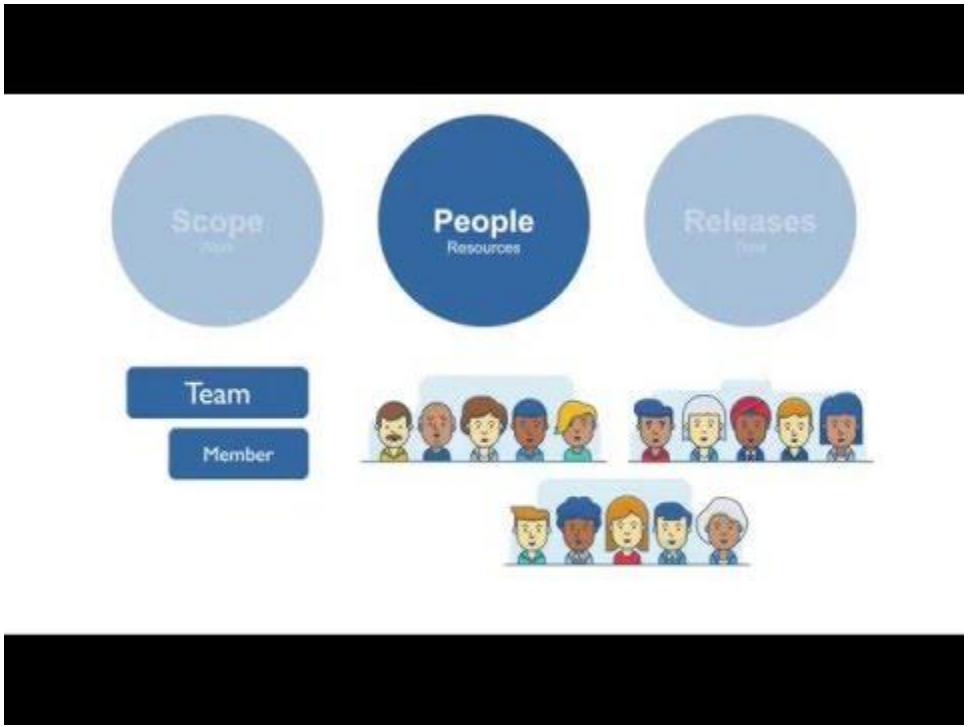

O'REILLY
Learning Agile
UNDERSTANDING SCRUM, XP, LEAN, AND KANBAN
Andrew Stellman & Jennifer Greene


O'REILLY
User Story Mapping
DISCOVER THE WHOLE STORY, BUILD THE RIGHT PRODUCT
Jeff Patton
with Peter Economy
Forewords by Martin Fowler, Alan Cooper, and Marty Cagan

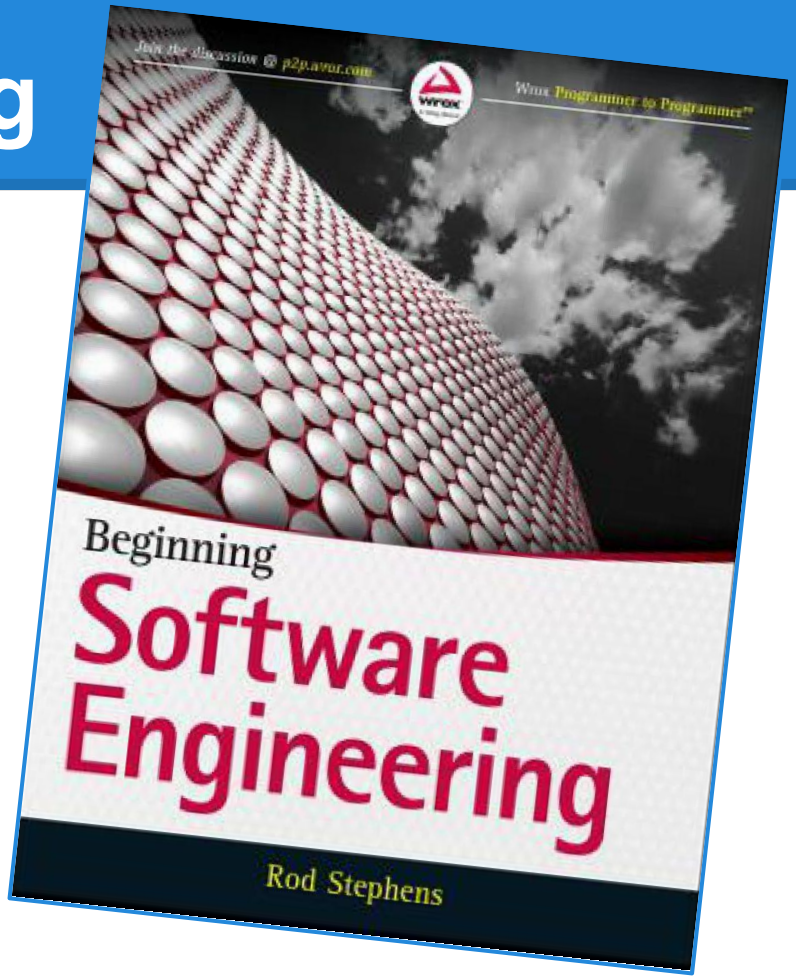
Books


FIFTY QUICK IDEAS
TO IMPROVE YOUR
USER
STORIES


You're holding a handbook for visionaries, game changers, and challengers striving to defy outmoded business models and design tomorrow's enterprises. It's a book for the …
Business Model Generation
WRITTEN BY
Alexander Osterwalder and Yves Pigneur
CO-CREATED BY
An amazing crowd of 470 practitioners from 45 countries
DESIGNED BY
Alan Smith, The Movement
NEW!


Scope
People
Resources
Releases
Team
Member

Software Engineering



Opportunity Canvas

Title:

Date:

Iteration:

Users & Customers

What types of users and customers have the challenges your solution addresses?

Look for differences in user's goals or uses that would affect their use of the product. Separate users and customers into different types based on those differences that make a difference. It's a bad idea to target "everyone" with your product.

2

Problems

What problems do prospective users and customers have today that your solution addresses?

1

Solution ideas

List product, feature, or enhancement ideas that solve problems for your target audience.

1

User Value

If your target audience has your solution, how can they do things differently as a consequence? And, how will that benefit them?

4

User Metrics

What user behaviors can you measure that will indicate they adopt, use, and place value in your solution?

5

Solutions Today

How do users address their problems today?

List competitive products or work-around approaches your users have for meeting their needs.

3

Leap-of-faith assumptions

What about the user, problem or solution would cause you unrecoverable failure, if this assumption turns out to be false?

0

Adoption Strategy

How will customers and users discover and adopt your solution?

6

Business Problems

What problem for your business does building this product, feature, or enhancement solve for your business?

7

Budget

What's it worth to you?

How much money and/or development would you budget to discover, build, and refine this solution?

9

Business Metrics

What business performance metrics will be affected by the success of this solution?

These usually change as a consequence of behavior metrics changing.

8

The Business Model Canvas

Designed for:

Designed by:

On:

Iteration:

Key Partners

Who are our Key Partners?
Who are our key suppliers?
Which Key Resources are we acquiring from partners?
Which Key Activities do partners perform?

Key Activities

What Key Activities do our Value Propositions require?
Our Distribution Channels?
Customer Relationships?
Revenue streams?

Key Resources

What Key Resources do our Value Propositions require?
Our Distribution Channels? Customer Relationships?
Revenue Streams?

Value Propositions

What value do we deliver to the customer?
Which one of our customer's problems are we helping to solve?
What bundles of products and services are we offering to each Customer Segment?
Which customer needs are we satisfying?

Customer Relationships

What type of relationship does each of our Customer Segments expect us to establish and maintain with them?
Which ones have we established?
How are they integrated with the rest of our business model?
How costly are they?

Channels

Through which Channels do our Customer Segments want to be reached?
How are we reaching them now?
How are our Channels integrated?
Which ones work best?
Which ones are most cost-efficient?
How are we integrating them with customer routines?

Customer Segments

For whom are we creating value?
Who are our most important customers?

Cost Structure

What are the most important costs inherent in our business model?
Which Key Resources are most expensive?
Which Key Activities are most expensive?

Revenue Streams

For what value are our customers really willing to pay?
For what do they currently pay?
How are they currently paying?
How would they prefer to pay?
How much does each Revenue Stream contribute to overall revenues?

www.businessmodelgeneration.com

References

http://www.techrepublic.com/article/capture-the-right-user-requirements-with-these-best-practices-for-writing-software-specifications/

http://agileproductdesign.com/writing/how_you_slice_it.pdf

https://blog.assembla.com/AssemblaBlog/tabid/12618/bid/9767/Using-Agile-methods-to-deliver-on-a-fixed-budget-fixed-time-commitment.aspx