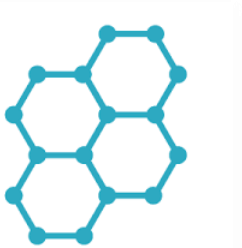
Australian National University

Planning with the Big Picture in Mind



Log into Socrative! http://www.socrative.com CALDWELL8573

COMP3120/8110
Tuesday 17 March 2020

Dr Sabrina Caldwell
The Australian National University
Research School of Computer Science
Acton, ACT, Australia

Course representatives
Zefan Wu, Runxiang Huang and Yixin Cheng …

… have asked me to let you know that there is a survey for students now open for your feedback on the course so far.

Please participate!

https://forms.office.com/Pages/ResponsePage.aspx?id=XHJ941yrJEaa5fBTPkhkNysfZwz4Kp5CkKDqV2d1rN1UOElBNFBCVDk0RDFaRjBCSjlQSlY1SDk2MC4u

Assignment #1 due next week

You (in your groups) have now created initial key message(s) and in workshops this week you are learning more about how to improve them.

Assignment #1 is due Friday Week 5 (26 March) at 5pm.
Submission link for Assignment #1 will open at the beginning of Week 5.
One submission per group.

Peer assessment will be done through Wattle.
I will discuss how this will work on Tuesday next week.
Peer assessment are due on Tuesday 9am Week 6 (30 March).
Submission link will open at 6pm Friday Week 5.
One submission per student.

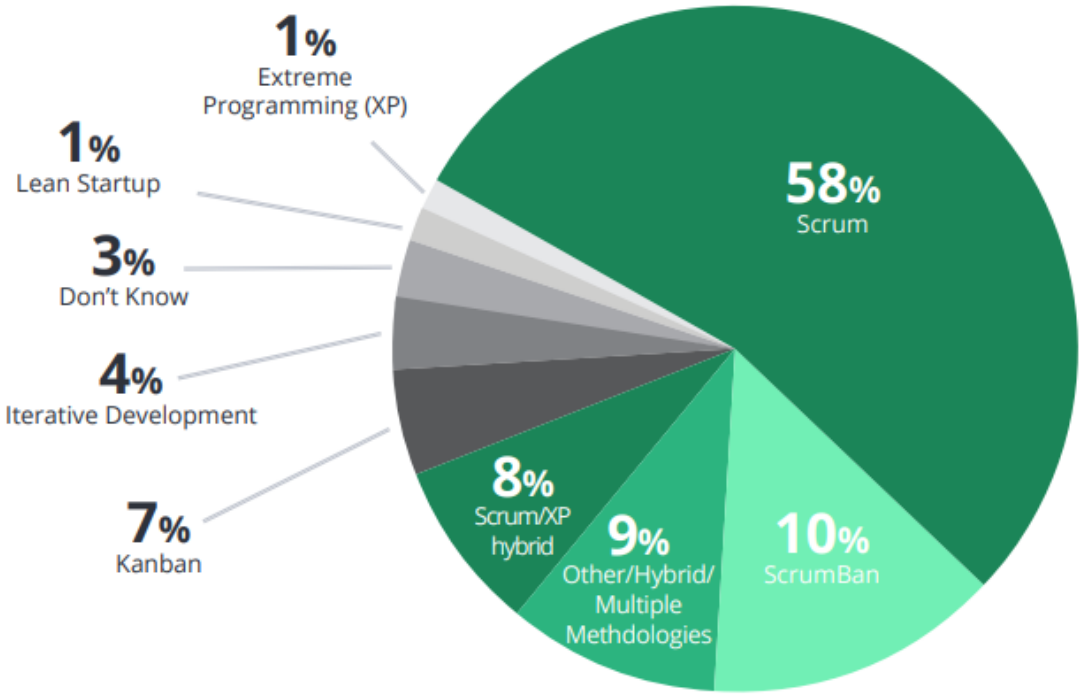
Popular Agile Methodologies

AGILE METHODS AND PRACTICES

2020

AGILE METHODOLOGIES USED

Scrum and related variants continue to be the most common Agile methodologies used by respondents' organizations.



Total exceeds 100% due to rounding.

76% of the Agile methods you will encounter will be Scrum or a variant of Scrum

Agile PM in Action - Scrum Methodology

Is a holistic approach for use by a cross-functional team collaborating to develop a new product. It defines product features as deliverables and prioritizes them by their perceived highest value to the customer. Priorities are re-evaluated after each iteration (sprint) to produce fully functional features.

Scrum consists of
four iterative phases:

- analysis
- design
- build
- test

Agile Project Management: Key Scrum Concepts

- **outcomes vs outputs**
- documenting the scope -- the **product backlog**
- agile requirements using **user stories**
- using a **visual mapping tool** to provide an overview of the total scope of the project

Why do we need to move the software development process from focusing on outputs to focusing on outcomes?

Outcomes vs Outputs

We consider concepts and practices that help shift project focus from outputs to outcomes.

These concepts and practices help move the process from one of requirements delivery to one of requirements discovery.

FALLACY - Requirements ***delivery*** process

- The customer knows what she/he wants
- The developers know how to build it
- Nothing will change along the way

REALITY - Requirements ***discovery*** process

- The customer discovers what he/she really wants (not what he/she first thinks he/she wants)
- The developers discover how best to build it
- There is acceptance there will be many changes along the way.

Output (Oxford dictionary definition)
The amount of something produced by a person, machine, or industry.

Outcome (Oxford dictionary definition)
The way a thing turns out; a consequence.

Myth or Fact:

Agile projects are not really concerned about scope?

Product Backlog

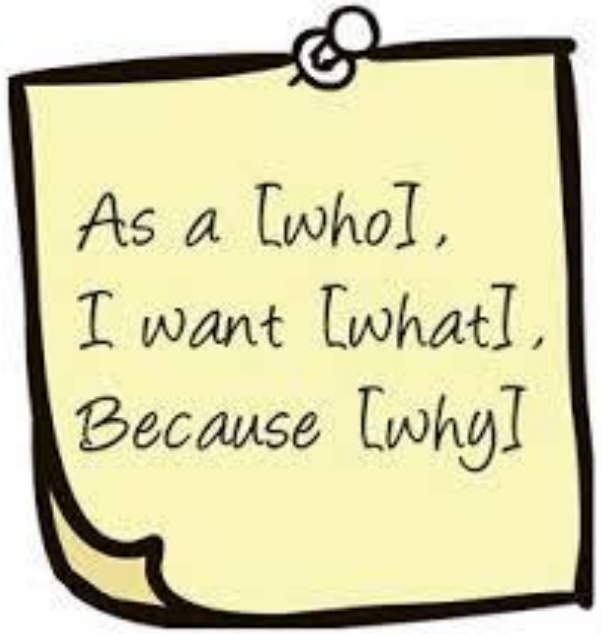
In agile project management, the scope of the project is captured in the "**Product Backlog**"

The agile product backlog is
a prioritized features list,
containing short descriptions of all functionality desired in the product.

Unlike many traditionally managed projects, agile projects don't start a project with a lengthy, upfront effort to document all requirements. Instead, an agile team and its product owner typically begin by writing down everything they can think of to include in the backlog.

A typical Scrum backlog comprises the following different types of items:

- Features
- Bugs
- Technical work
- Knowledge acquisition



The predominant way for a Scrum team to express features on the agile product backlog is in the form of user stories.

User stories are short, simple descriptions of the desired functionality told from perspective of the user.

An example: "**As a** shopper, **I can** review the items in my shopping cart before checking out **so that** I can see what I've already selected."

User stories = requirements
Well, sort of.

Product Backlog

User stories = requirements Well, sort of.

No difference between a bug and a new feature -- each describes something different that a user wants -- bugs are also on the put on the Scrum product backlog.

Technical work and knowledge acquisition activities also belong on the agile backlog.

Example of technical work could be:

"Upgrade all developers' workstations to Windows 10."

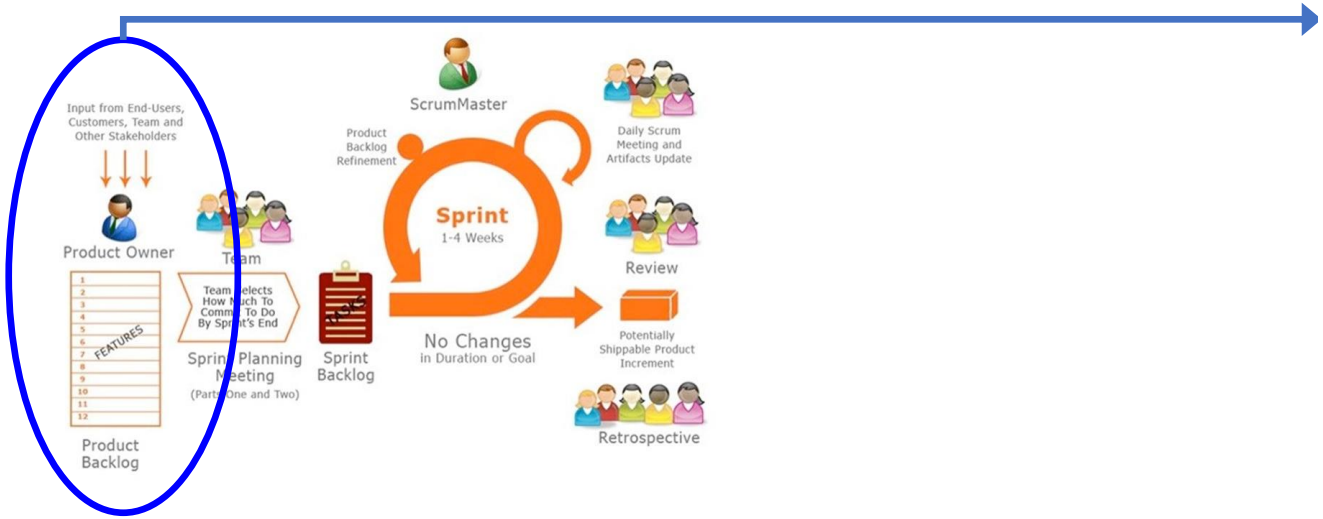
Example of knowledge acquisition could be:

Backlog item about researching various JavaScript libraries and making a selection.

Controlling work flow with Product Backlog

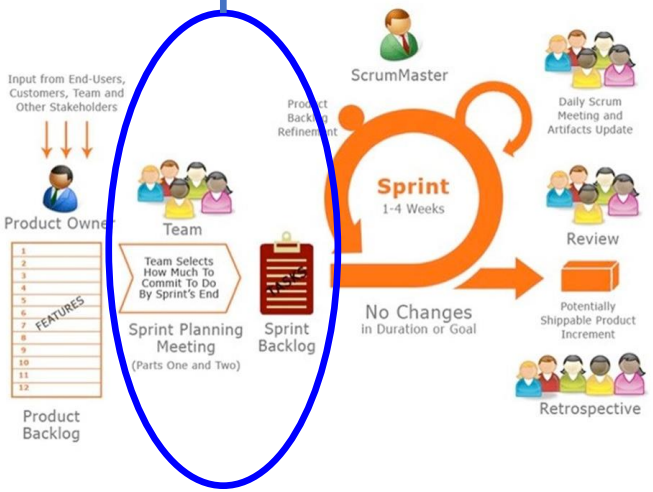
The **product owner** shows up at the **sprint planning meeting** with the **prioritized agile product backlog** and **describes the top items** to the team.

The **team then determines which items they can complete** during the coming sprint.



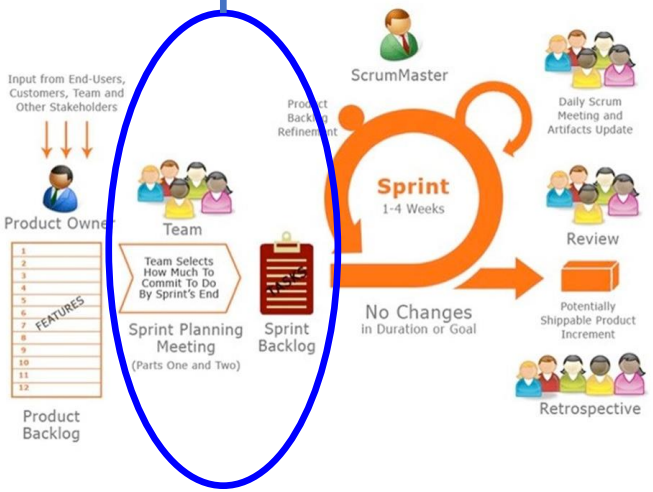
Controlling work flow with Product Backlog

The team then moves items from the product backlog to the sprint backlog. In doing so, they expand each Scrum product backlog item into one or more sprint backlog tasks so they can more effectively share work during the sprint.



Controlling work flow with Product Backlog

Conceptually, the **team starts at the top of the prioritized Scrum backlog** and draws a line after the lowest of the high-priority items they feel they can complete. In practice, it is not unusual to see a team select, for example, the top five items and then two items from lower on the list that are associated with the initial five.



Product Backlog Issues

Long lists of "stuff to do" - flat structure.

No better than poorly developed Work Breakdown Structure – that **focuses on delivering the items on the list rather than delivering value.**

A list of stories **without any connection to the value** that is to be delivered to users who are trying to solve problems.

Example of a typical Product Backlog

Backlog Description	Initial Estimate	Adjust-ment Factor	Adjusted Estimate	work remaining until completion						
				1	2	3	4	5	6	7
Title Import				256	209	193	140	140	140	140
Project selection or new	3	0.2	3.6	3.6	0	0	0	0	0	0
Template backlog for new projects	2	0.2	2.4	2.4	0	0	0	0	0	0
Create product backlog worksheet with formatting	3	0.2	3.6	3.6	0	0	0	0	0	0
Create sprint backlog worksheet with formatting	3	0.2	3.6	3.6	0	0	0	0	0	0
Display tree view of product backlog, releases, sprints	2	0.2	2.4	2.4	0	0	0	0	0	0
Sprint-1	13	0.2	15.6	16	0	0	0	0	0	0
Create a new window containing product backlog template	3	0.2	3.6	3.6	3.6	0	0	0	0	0
Create a new window containing sprint backlog template	2	0.2	2.4	2.4	2.4	0	0	0	0	0
Burndown window of product backlog	5	0.2	6	6	6	0	0	0	0	0
Burndown window of sprint backlog	1	0.2	1.2	1.2	1.2	0	0	0	0	0
Display tree view of product backlog, releases, prints	2	0.2	2.4	2.4	2.4	0	0	0	0	0
Display burndown for selected sprint or release	3	0.2	3.6	3.6	3.6	0	0	0	0	0
Sprint-2	16	0.2	19.2	19	19	1.2	0	0	0	0
Automatic recalculating of values and totals	3	0.2	3.6	3.6	3.6	3.6	0	0	0	0
As changes are made to backlog in secondary window, update burndown graph on main page	2	0.2	2.4	2.4	2.4	2.4	0	0	0	0
Hide/automatic redisplay of burndown window	3	0.2	3.6	3.6	3.6	3.6	0	0	0	0
Insert Sprint capability ... adds summing Sprint row	2	0.2	2.4	2.4	2.4	2.4	0	0	0	0
Insert Release capability ... adds summary row for backlog in Sprint	1	0.2	1.2	1.2	1.2	1.2	0	0	0	0
Owner/assigned capability and columns optional	2	0.2	2.4	2.4	2.4	2.4	0	0	0	0
Print burndown graphs	1	0.2	1.2	1.2	1.2	1.2	0	0	0	0
Sprint-3	14	0.2	16.8	17	17	17	0	0	0	0
Duplicate incomplete backlog without affecting totals	5	0.2	6	6	6	6	6	6	6	6
Note capability	6	0.2	7.2	7.2	7.2	7.2	7.2	7.2	7.2	7.2
What-if release capability on burndown graph	15	0.2	18	18	18	18	18	18	18	18
Trend capability on burndown server	2	0.2	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4
Publish facility for entire project, publishing it as HTML web pages	11	0.2	13.2	0	0	13	13	13	13	13
Future Sprints	39	0.2	46.8	34	34	47	47	47	47	47
Release-1				85	70	65	47	47	47	47

Product Backlog Issues

Arranging user stories in the order you'll build them **doesn't help to explain to others what the system does**.

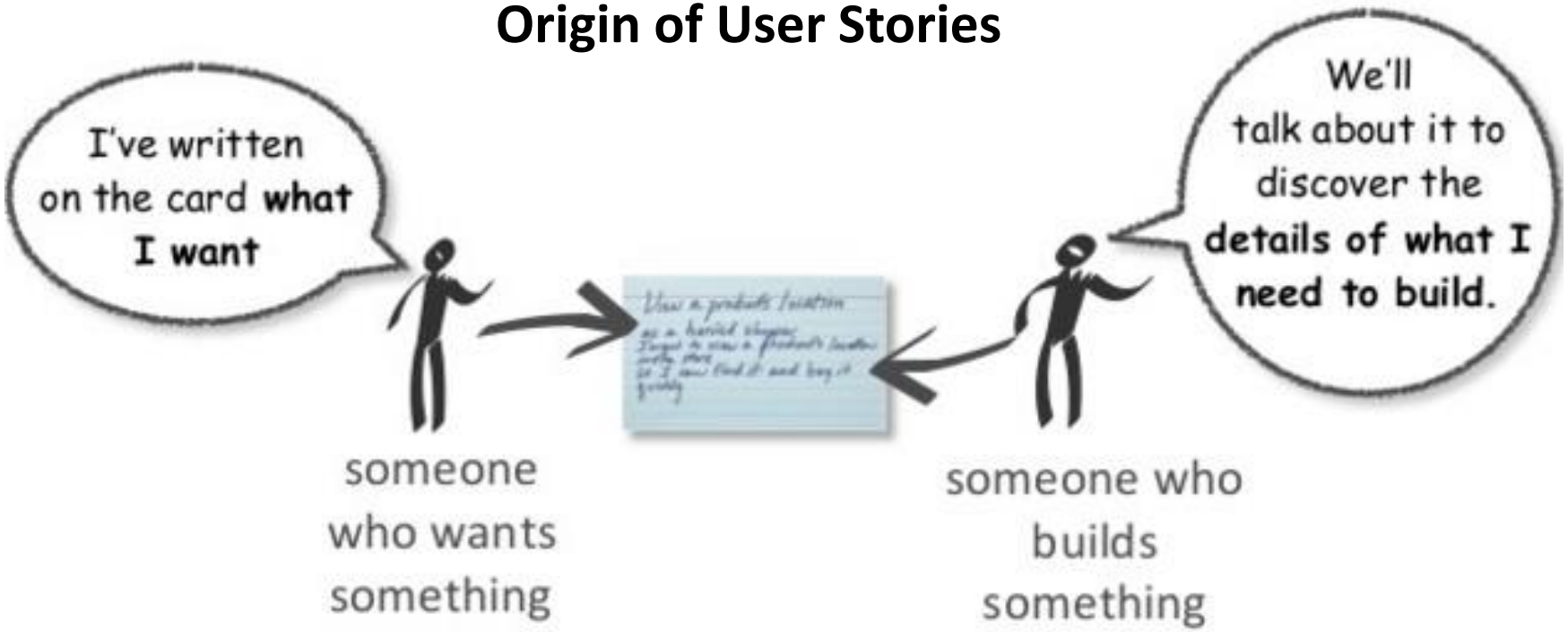
You need context in order to really tell a story about the system. BUT typical product backlogs have no context.

The typical project has a minimum of dozens of user stories, frequently over a hundred. **Stepping through each one** making an in-out decision is **tedious and time-consuming**.

Example of a typical Product Backlog

Backlog Description	Initial Estimate	Adjust-ment Factor	Adjusted Estimate	work remaining until completion						
				1	2	3	4	5	6	7
Title Import				256	209	193	140	140	140	140
Project selection or new	3	0.2	3.6	3.6	0	0	0	0	0	0
Template backlog for new projects	2	0.2	2.4	2.4	0	0	0	0	0	0
Create product backlog worksheet with formatting	3	0.2	3.6	3.6	0	0	0	0	0	0
Create sprint backlog worksheet with formatting	3	0.2	3.6	3.6	0	0	0	0	0	0
Display tree view of product backlog, releases, sprints	2	0.2	2.4	2.4	0	0	0	0	0	0
Sprint-1	13	0.2	15.6	16	0	0	0	0	0	0
Create a new window containing product backlog template	3	0.2	3.6	3.6	3.6	0	0	0	0	0
Create a new window containing sprint backlog template	2	0.2	2.4	2.4	2.4	0	0	0	0	0
Burndown window of product backlog	5	0.2	6	6	6	0	0	0	0	0
Burndown window of sprint backlog	1	0.2	1.2	1.2	1.2	0	0	0	0	0
Display tree view of product backlog, releases, prints	2	0.2	2.4	2.4	2.4	0	0	0	0	0
Display burndown for selected sprint or release	3	0.2	3.6	3.6	3.6	0	0	0	0	0
Sprint-2	16	0.2	19.2	19	19	1.2	0	0	0	0
Automatic recalculating of values and totals	3	0.2	3.6	3.6	3.6	3.6	0	0	0	0
As changes are made to backlog in secondary window, update burndown graph on main page	2	0.2	2.4	2.4	2.4	2.4	0	0	0	0
Hide/automatic redisplay of burndown window	3	0.2	3.6	3.6	3.6	3.6	0	0	0	0
Insert Sprint capability ... adds summing Sprint row	2	0.2	2.4	2.4	2.4	2.4	0	0	0	0
Insert Release capability ... adds summary row for backlog in Sprint	1	0.2	1.2	1.2	1.2	1.2	0	0	0	0
Owner/assigned capability and columns optional	2	0.2	2.4	2.4	2.4	2.4	0	0	0	0
Print burndown graphs	1	0.2	1.2	1.2	1.2	1.2	0	0	0	0
Sprint-3	14	0.2	16.8	17	17	17	0	0	0	0
Duplicate incomplete backlog without affecting totals	5	0.2	6	6	6	6	6	6	6	6
Note capability	6	0.2	7.2	7.2	7.2	7.2	7.2	7.2	7.2	7.2
What-if release capability on burndown graph	15	0.2	18	18	18	18	18	18	18	18
Trend capability on burndown server	2	0.2	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4
Publish facility for entire project, publishing it as HTML web pages	11	0.2	13.2	0	0	13	13	13	13	13
Future Sprints	39	0.2	46.8	34	34	47	47	47	47	47
Release-1				85	70	65	47	47	47	47

Origin of User Stories



Stop exchanging documents. Instead, tell me your story.

Kent Beck noticed that users sometimes tell stories about the cool new things the software does. For example, “if I type in the zip code and it automatically fills in the city and state without me having to touch a button”. He thought if you can tell stories about what software does that generate interest and vision why not tell stories before the software does it?

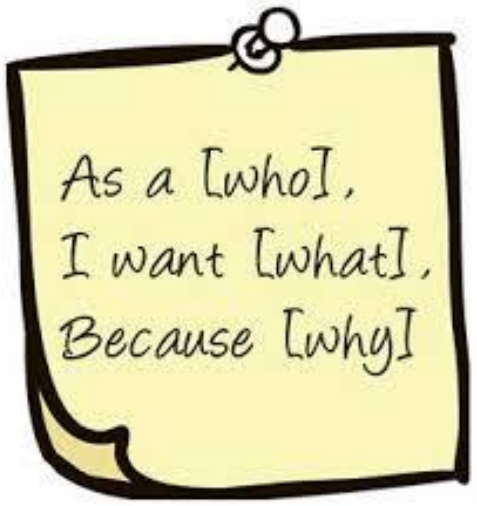
Shared understanding and alignment are the objectives of collaborative work

User stories are supposed to spark conversations.

Stories get their name from* how they should be used*, not what should be written!

A story is supposed to be a promise to have a conversation around the subject of the story, to eliminate misunderstandings through speech, not writing.

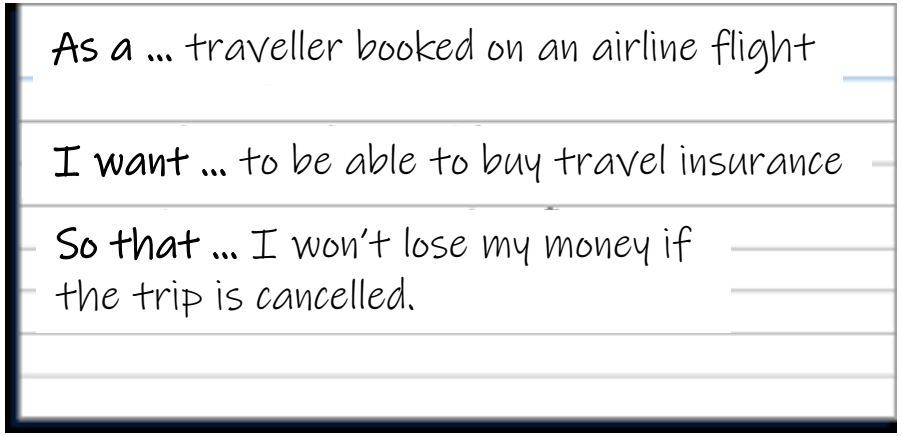
Agile User Stories



Users stories help develop a shared understanding between the customer and the development team and should be focused on the big picture. As stories are developed, they are talked about, discussed and documented. They provide a light-weight approach to managing requirements for a system. The details are figured out in future conversations between the team and the product owner or customers.

This approach facilitates just-in-time requirements gathering, analysis and design by the following activities:

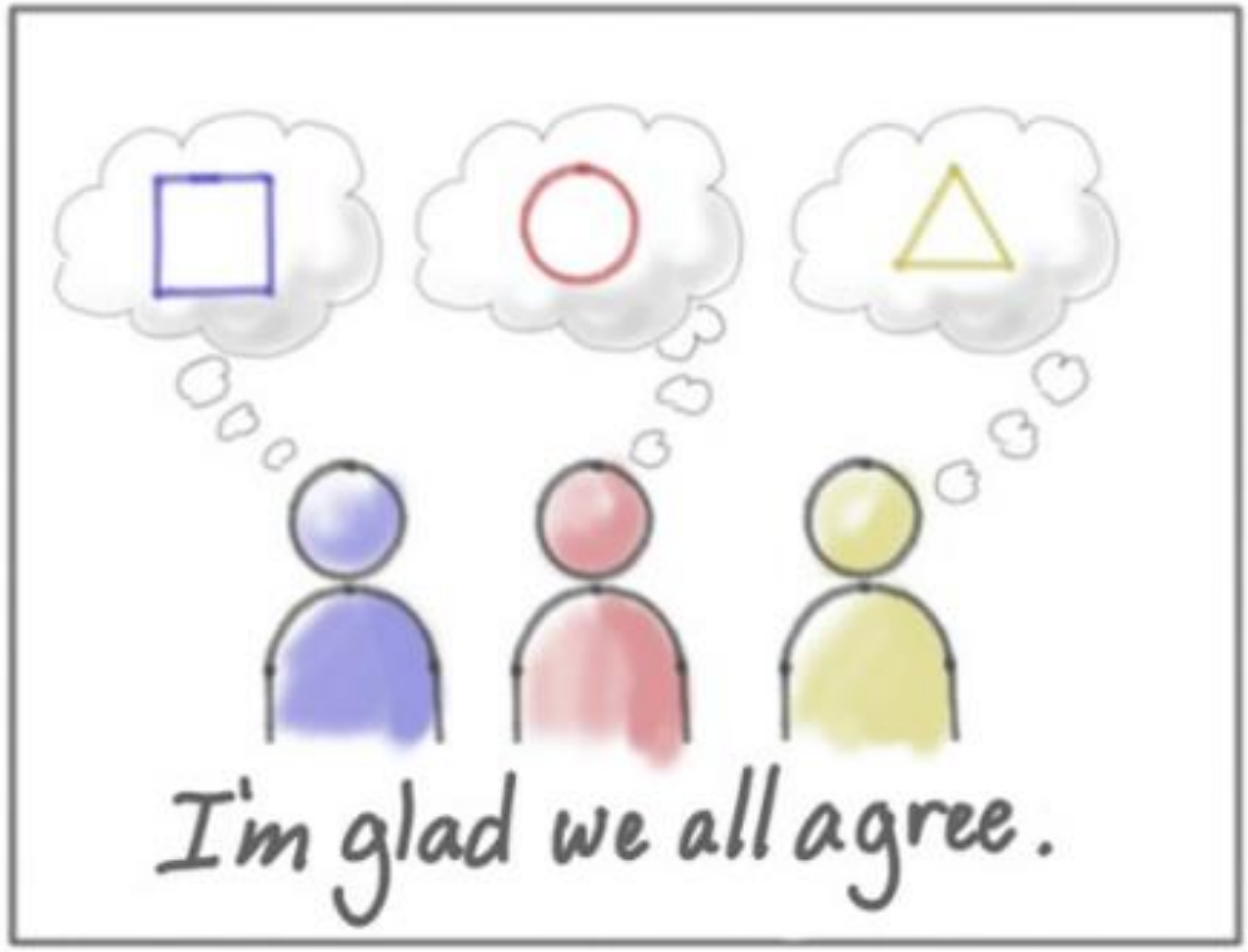
- **Slicing** user stories down in release planning
- **Tasking** user stories out in sprint planning
- **Specifying** acceptance test criteria for user stories early in development.



Frequently User Stories are documented on index cards. Writing them on a card helps externalise thinking, and makes the ideas available later to help recall what was discussed.

Describing the story informally on a card enables a team to capture ideas and to **stop them vaporising**. Generally, just a few words are written on a card immediately after thinking it. The idea is then later explained (to the whole team) who will discuss it, and refine it.

To create compelling user stories: conversations need to be deep and we need to be engaged

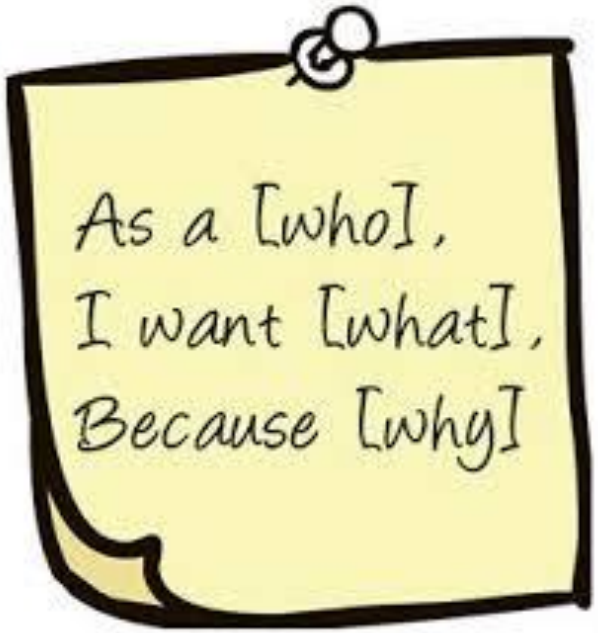


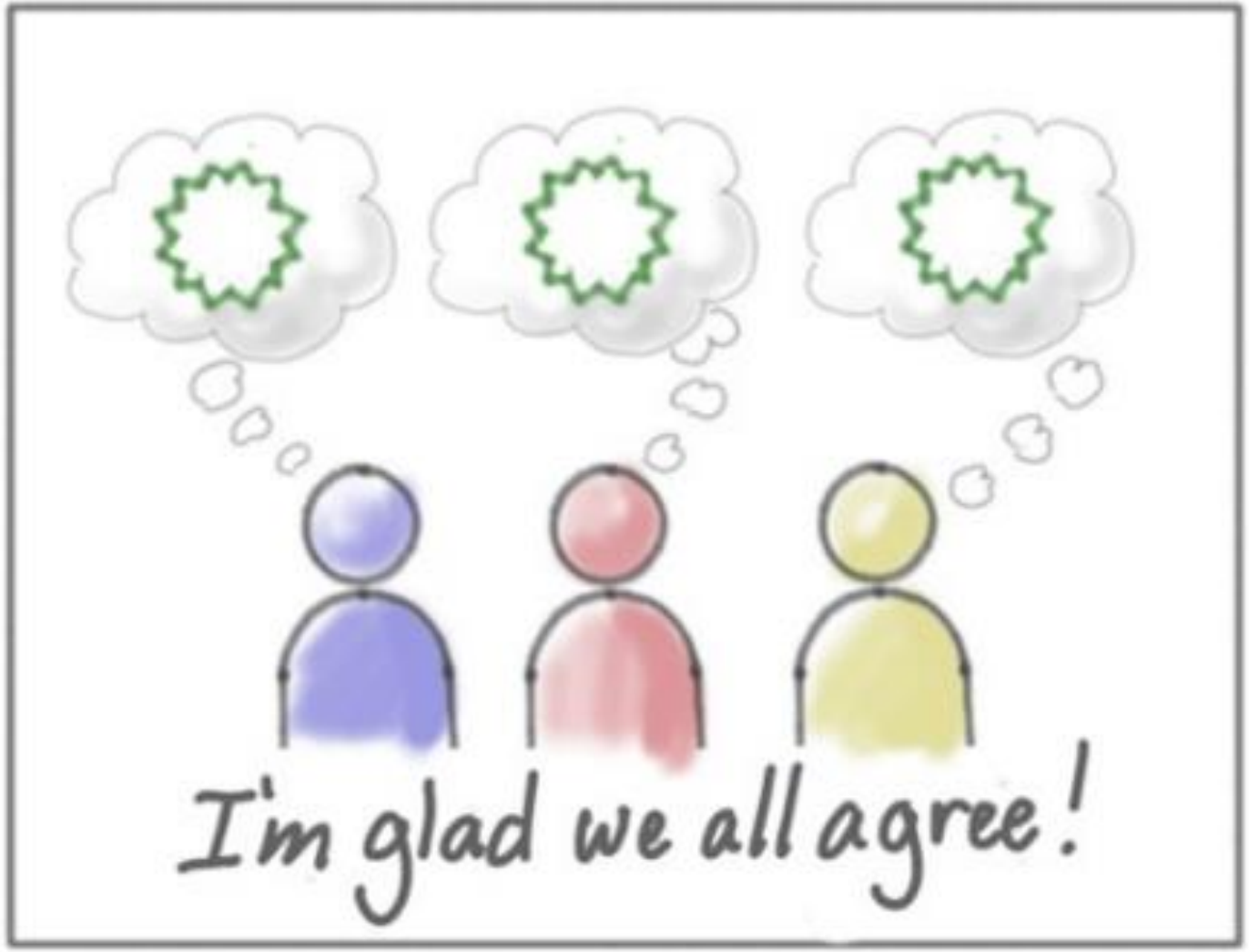
Agile User Stories

To create compelling user stories: externalise thinking with words and pictures to detect differences

To create compelling user stories: afterward, when we say the same thing, we actually mean it

To create compelling user stories: afterward, when we say the same thing, we





User stories capture who, what and where

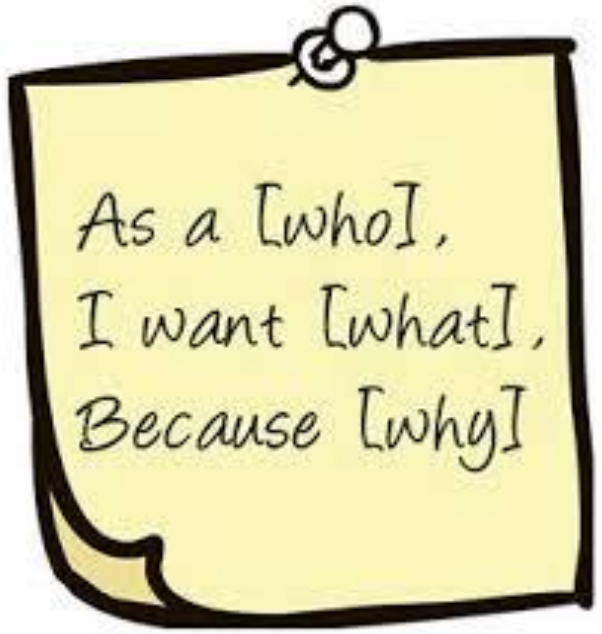
User stories help frame the product. They help teams develop and understanding of

- **Why** you are building this
- **What are the benefits** for the people who will use this
- **What problems it solves** for those people and for you

When working with stories, it is important to keep in mind the now and later model; that is things are done "just-in-time" in agile projects.

Do not provide too much detail too early. It is likely to be wrong and is a waste of time and resources.

User stories are used to capture functional and non-functional needs.



User stories capture who, what and where

A user story is a short, simple description of a feature from the perspective of the person who desires the new capability, usually a user or customer of the system.

They typically follow a simple template:

"As a <*type of user*>, I want <*some goal*> so that <*some reason*>."

and they include tests that can be used to determine when a story is complete.

User stories capture who, what and where

Who

Include ‘who’ wants the functionality and who benefits from it
Written from the point of view of a persona or role

What

Specifies the need, feature or functionality required by the ‘who’

Why

Specifies the value to the ‘who’ • Provides context • Keeps value in mind at all times

IMPORTANT -- Stories without a ‘why’ may have no value

ALSO IMPORTANT -- User stories say nothing about HOW

User stories allow us to generate user acceptance criteria

Like stories, acceptance criteria are written in simple language.

They define the conditions of success/satisfaction.

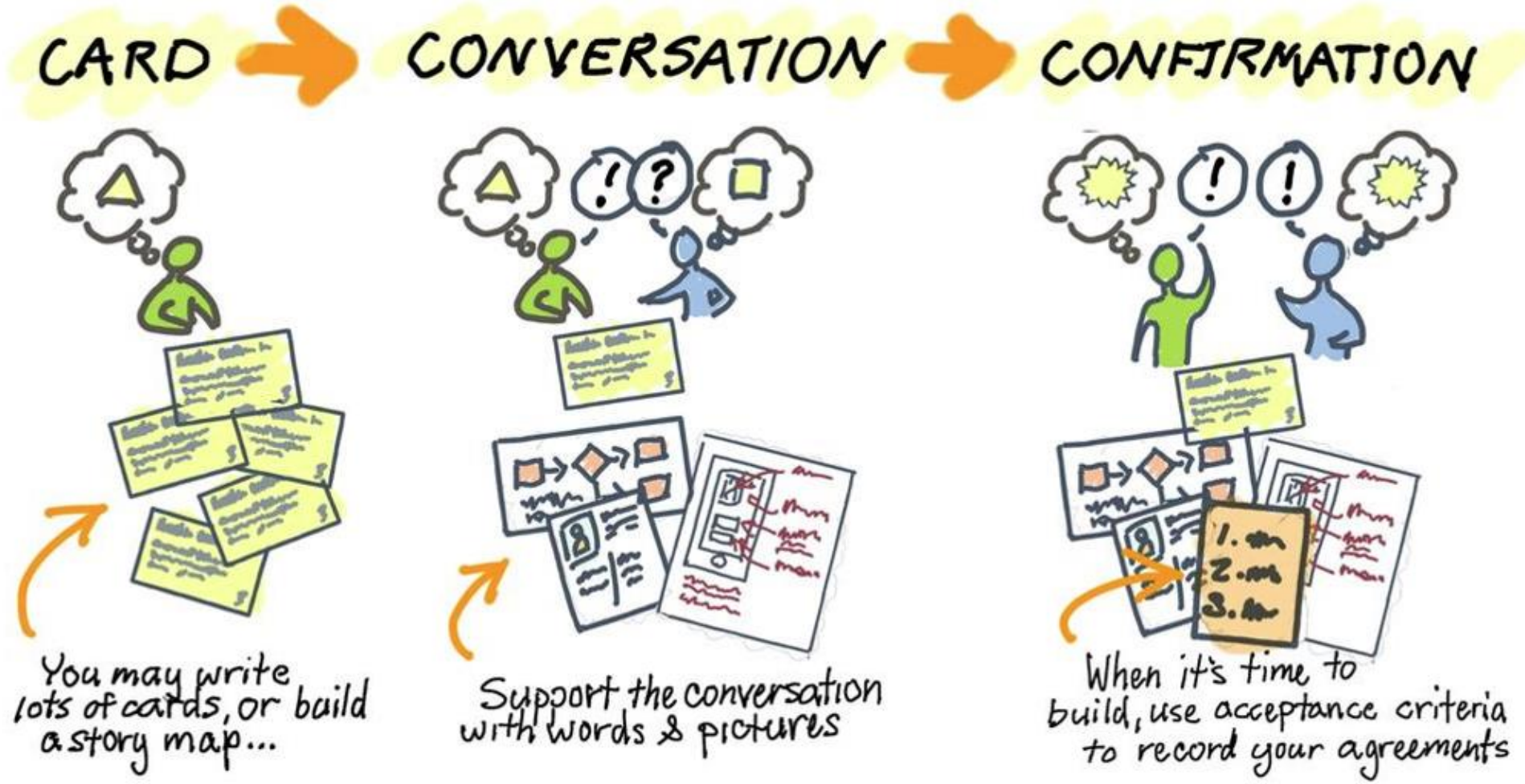
Through acceptance criteria, a good user story provides clear story boundaries and removes ambiguity by forcing the team to think through how a feature or piece of functionality will work from the user's perspective.

User stories allow us to generate user acceptance criteria

They form a checklist or template of things to consider for each story as they may include a list of impacted modules and/or documents; and/or specific user tasks, business process or functions.

Good stories help establish the basis for acceptance testing. They provide steps to test the story (given-when-then scenario) and suggest the type of testing required (manual versus automated).

The three ‘C’s


CARD
CONVERSATION
CONFIRMATION
You may write lots of cards, or build a story map...
Support the conversation with words & pictures
When it's time to build, use acceptance criteria to record your agreements

Myth or Fact: best practice requires that a user story is written using the template? Why?

Template Zombies

Tom DeMarco described "The project team [that] allows its work to be driven by templates instead of the thought process necessary to deliver products"

as "template zombies"

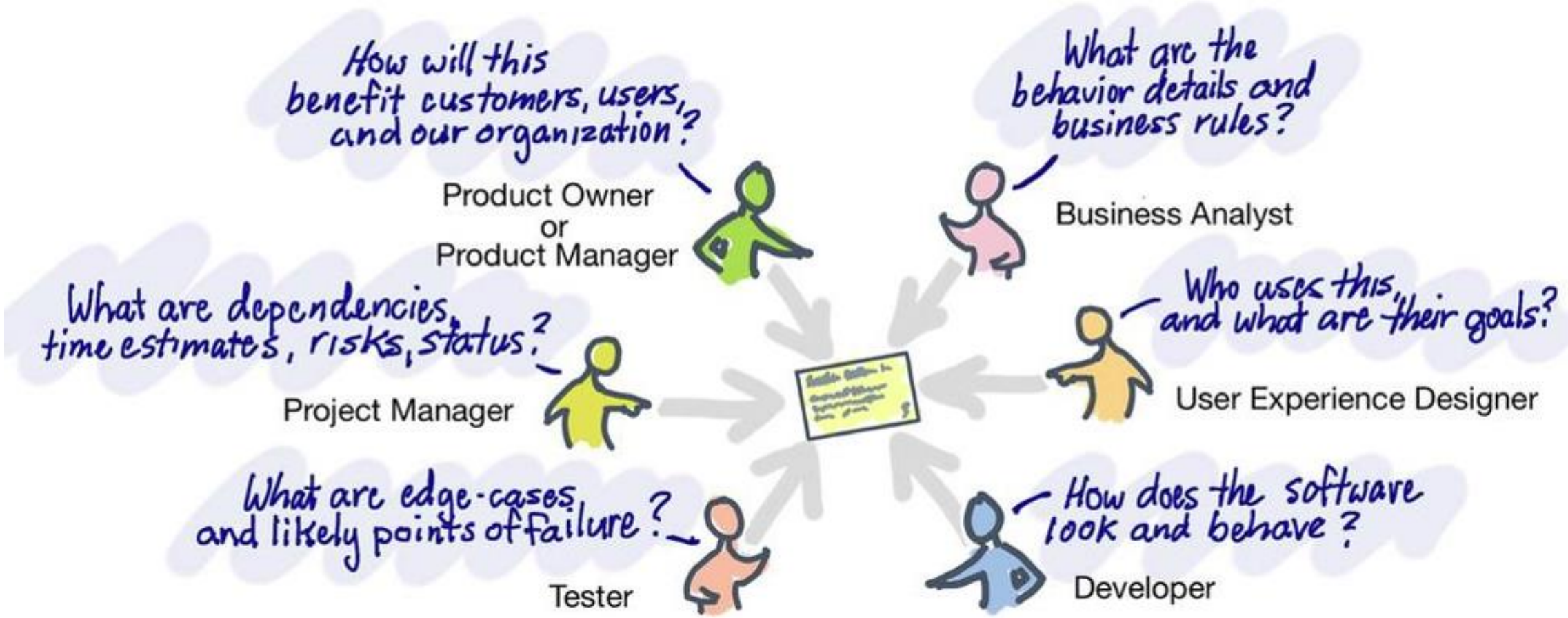
A User Story is a card which matches what a Persona wants and for what reason.

A BAD example -- "as a product owner, I want you to build a file uploader so that the customer requirements are met"

* http://www.jesusmendez.ca/tag/user-stories/
Zombie graphic sourced from https://au.pinterest.com/pin/325455510548320448/

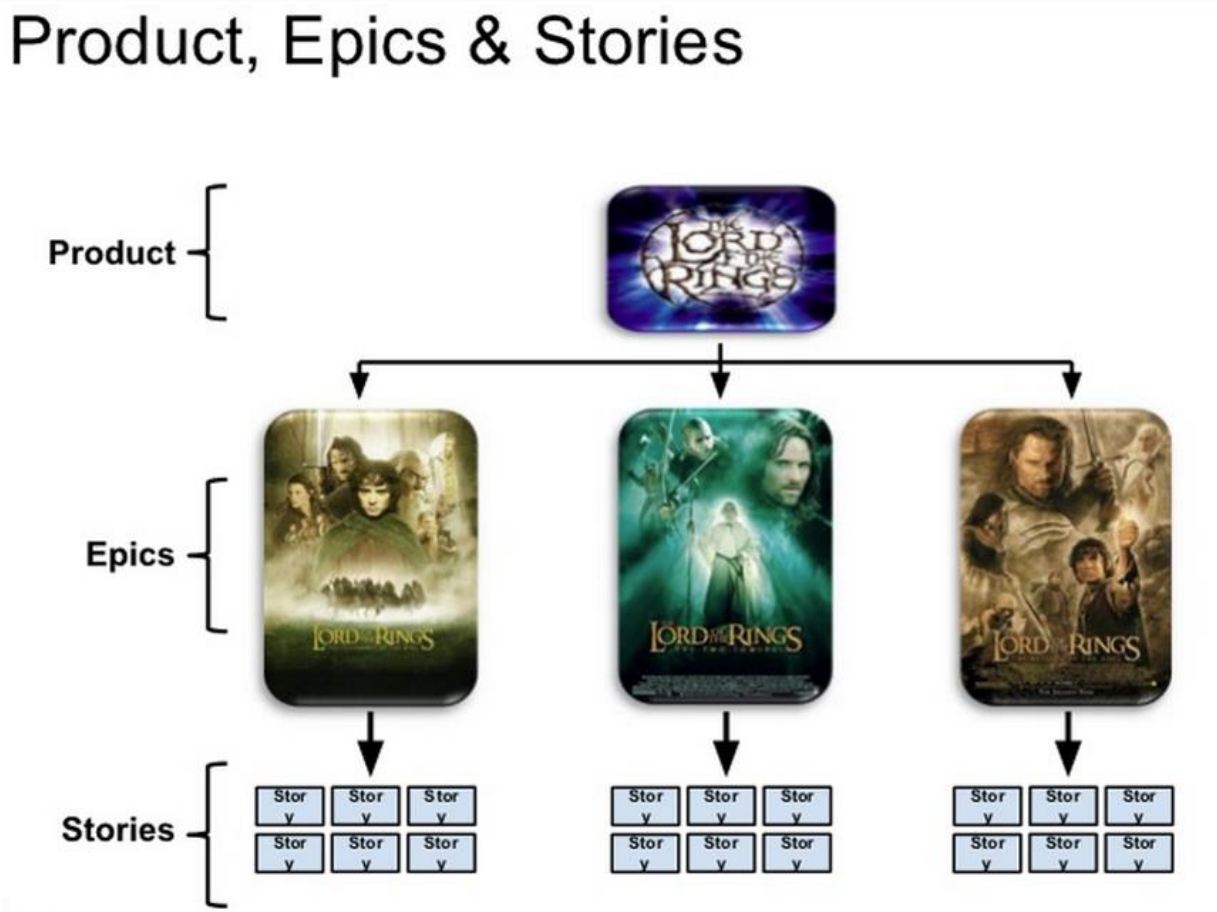
What is it about user stories that promote collaborative development?

The big picture



Why would you express the product backlog visually rather than textually?

Create a user story map



When decomposing user stories following the INVEST guideline will help you create good user stories.

I – Independent - Uuser story should be self-contained - no inherent dependency on another user story.

N – Negotiable - User stories, up until they are done, are part of an iteration (Sprint in Scrum), can always be changed and rewritten.

V – Valuable - User story must deliver value to the end user.

E – Estimable - You must always be able to estimate the size of a user story.

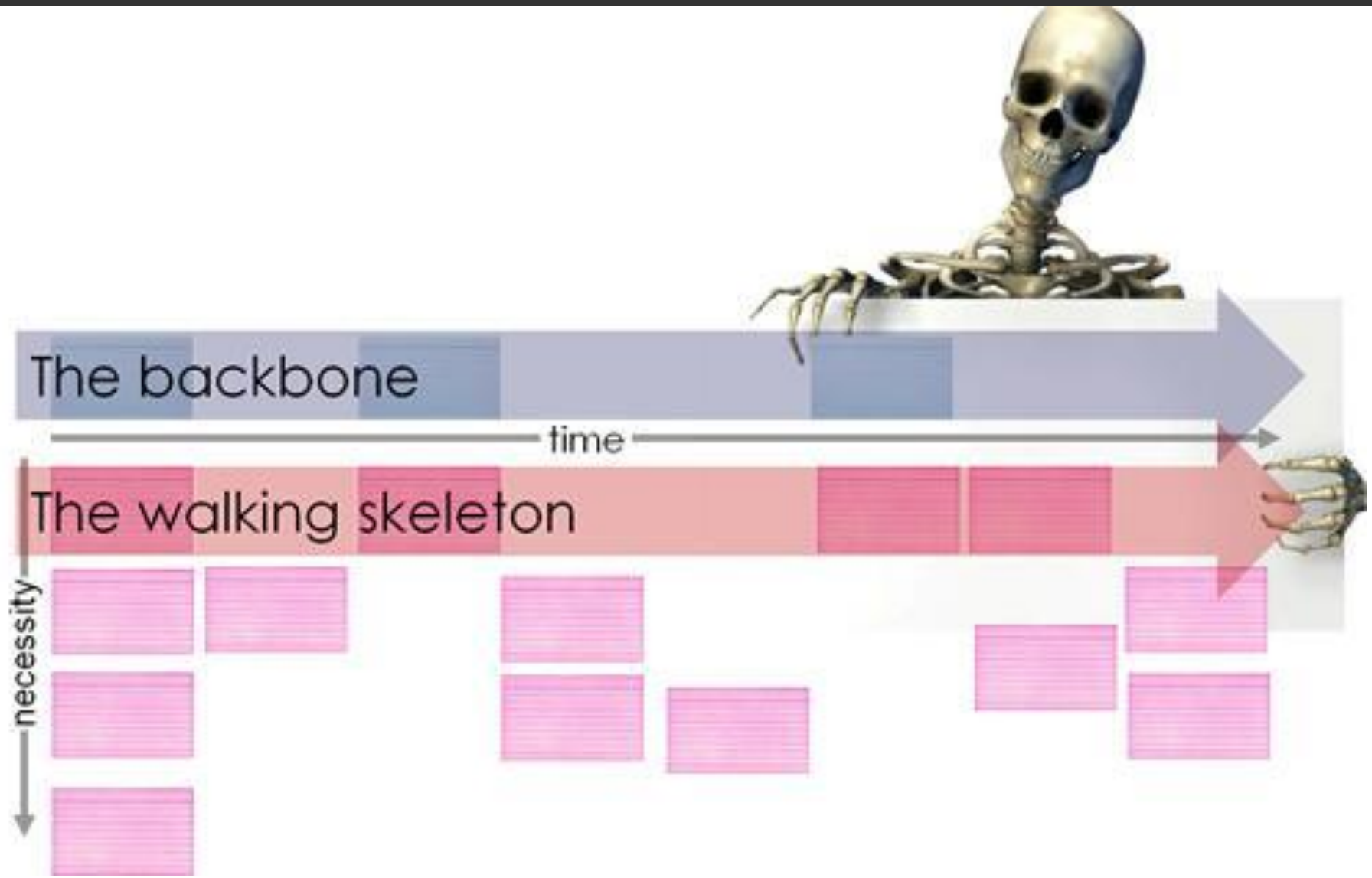
S - Sized appropriately - User stories should not be so big as to become impossible to plan/task/prioritise with some certainty.

T – Testable - The user story or its related description must provide the necessary information to make test development possible.

Planning with user story maps

Be prepared, skeleton on next page…

Planning with user story maps



Planning with user story maps

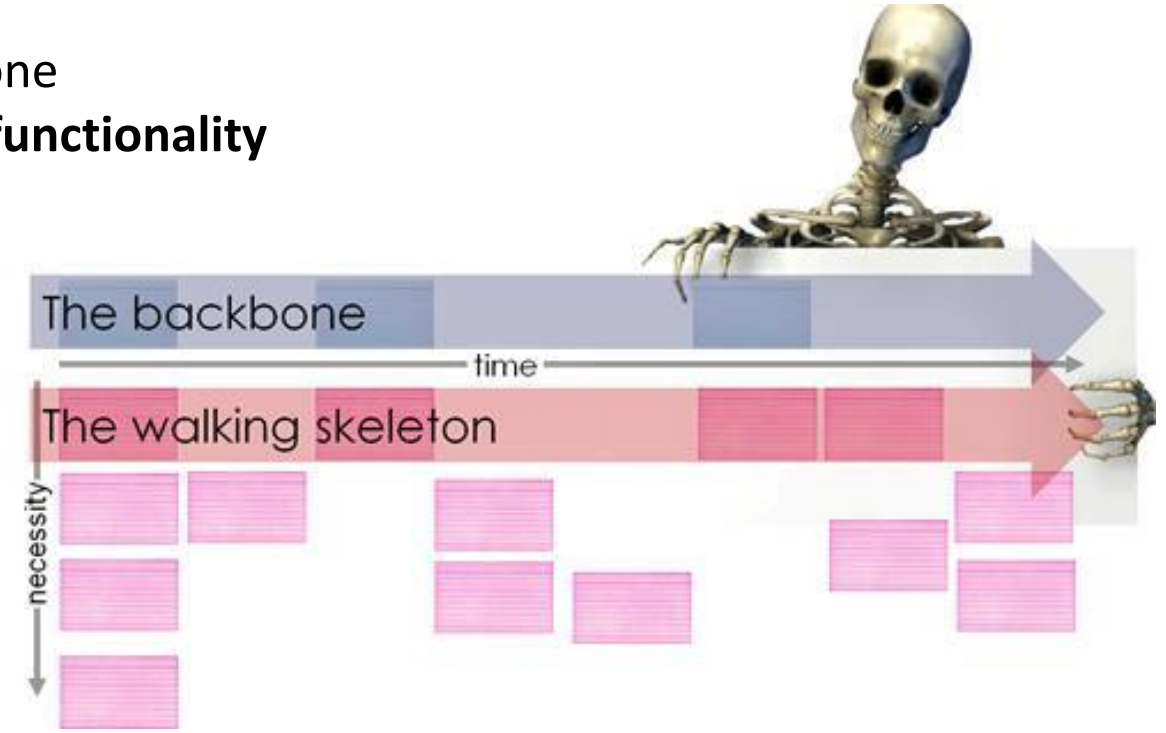
Activities are the **backbone**
- The essential activities required to deliver minimum viable product (MVP)
- Do not prioritise - without any of these you do not have an MVP

The **walking skeleton** consists of
- The highest priority tasks immediately below the backbone
- **The smallest possible system that provides end to end functionality**

Prioritize tasks below the backbone
- Move activities up or down

Releases can be marked out in “swim lanes”
- Stories can be moved in or out of releases
- Stories can have different heights within a release

Build all the major features a little at a time
- Each release **always** adds value

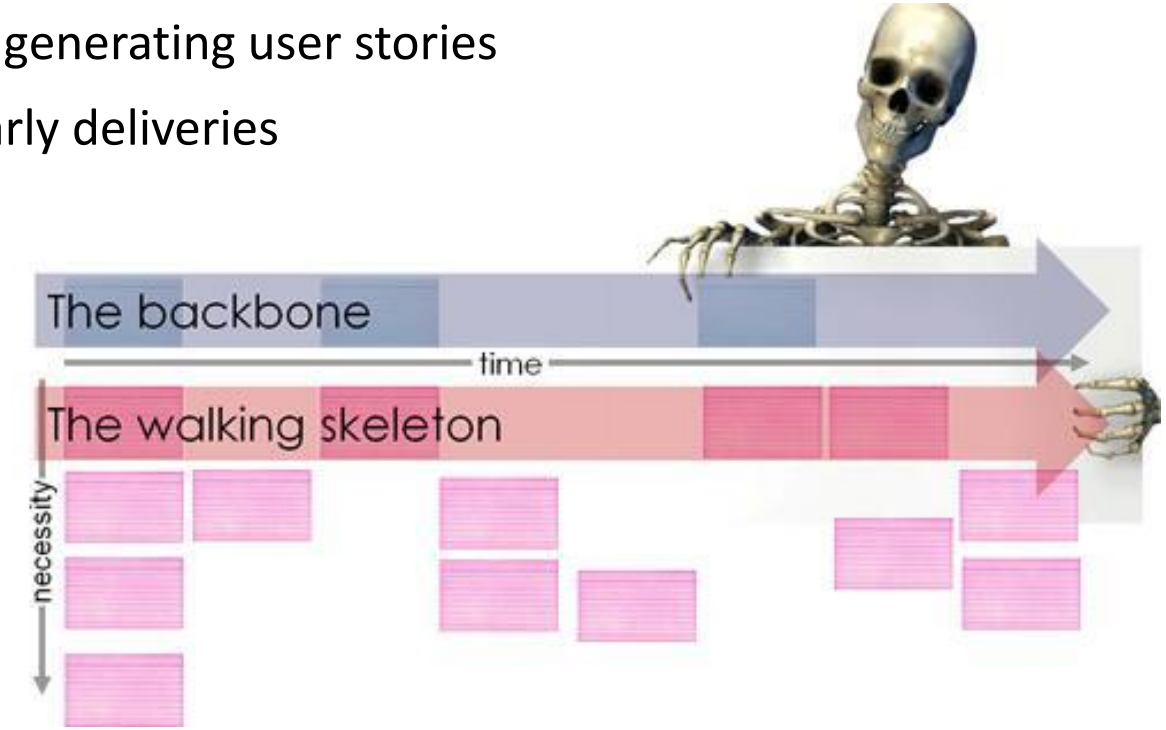


Planning with user story maps

Benefits of User Story Maps

User story maps

- big picture in your backlog,
- better tool for making decisions about grooming and prioritizing the backlog
- promote brainstorming and a collaborative approach to generating user stories
- encourage an iterative development approach where early deliveries validate your architecture and solution
- a great visual alternative to traditional project plans
- a useful model for discussing and managing scope
- visual dimension to planning, &
  real options for your project/product



Planning with user story maps

To construct a User Story Map you

Place activities – big user stories (UX),
epics at the top

An epic activity is something people do –

has lots of steps, and

doesn't always have a precise workflow

Eg. Building an email system

managing email

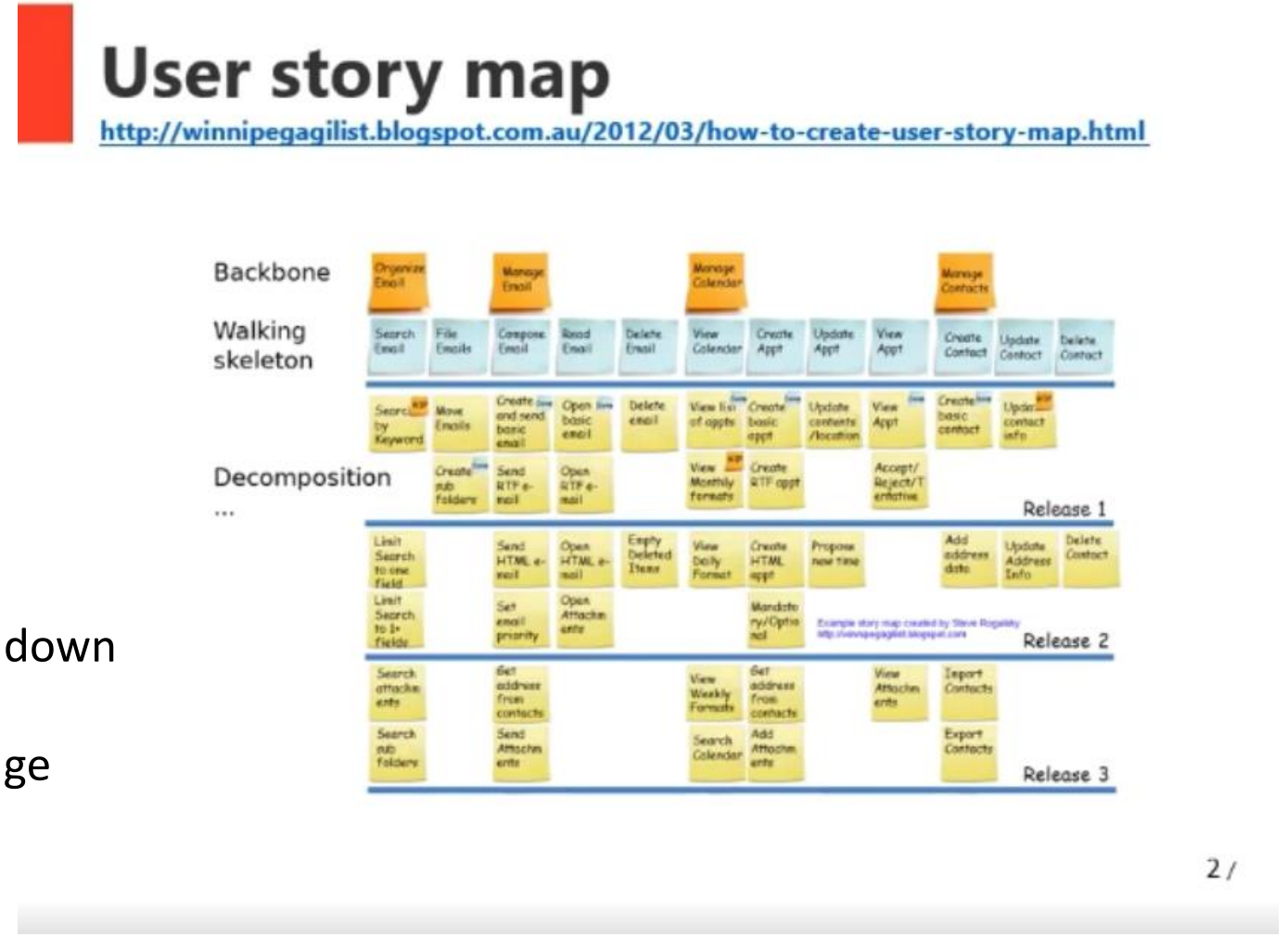
configuring email servers

Setting up out of office responses

Activities provide the context, then break activities down into user tasks i.e. smaller stories

Eg. send message, read message, delete message

A user task is something that someone does to reach a goal



Planning with user story maps

To construct a User Story Map you

Move from left to right and top to bottom

Arrange stories in the logical order in which they would be completed – grid form

Test your map
- Talk / walk it through
- Find things you have missed

Annotate it

Use different colour for different levels

Display it as an information radiator
- Use for sprint or iteration planning
- Mark of progress