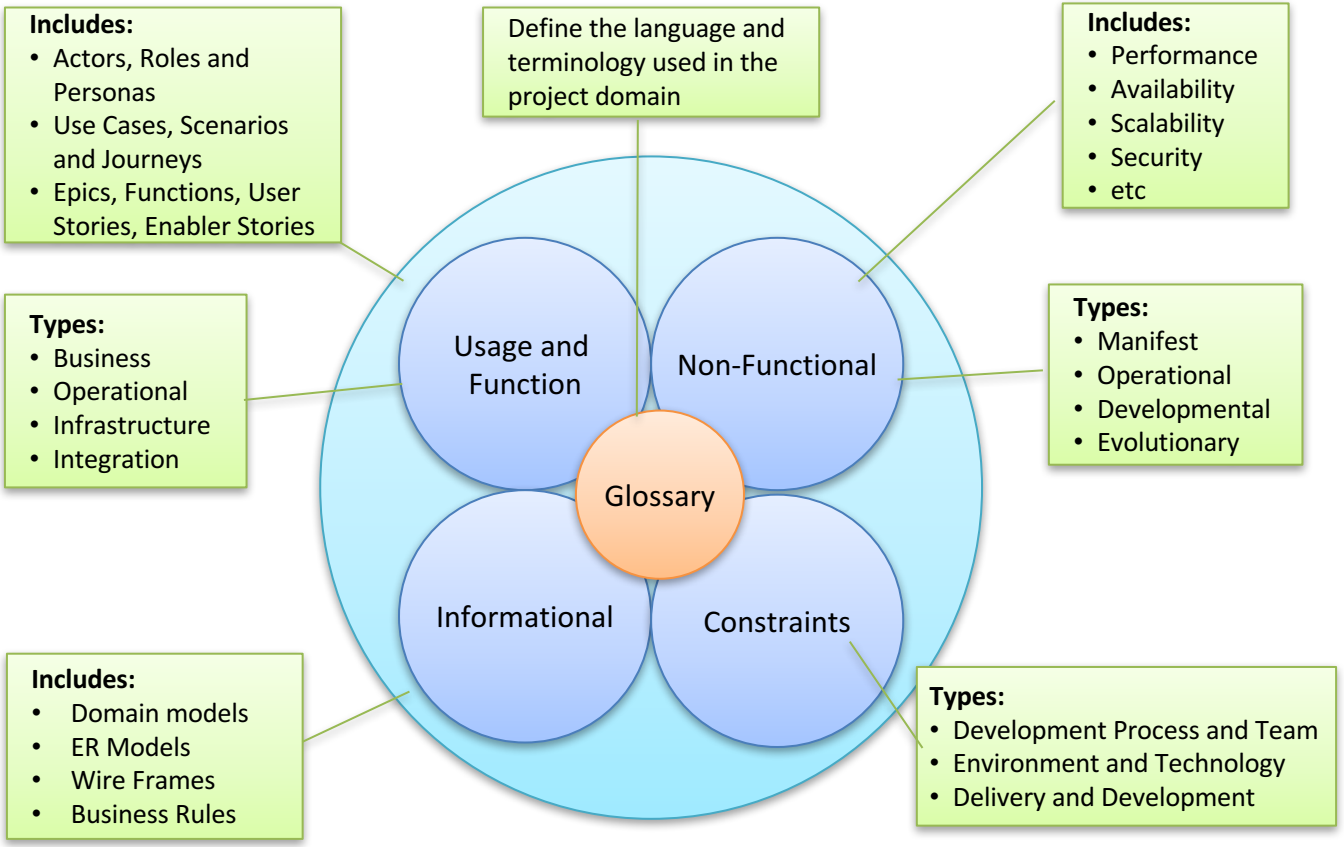
Agile Requirements Definitions

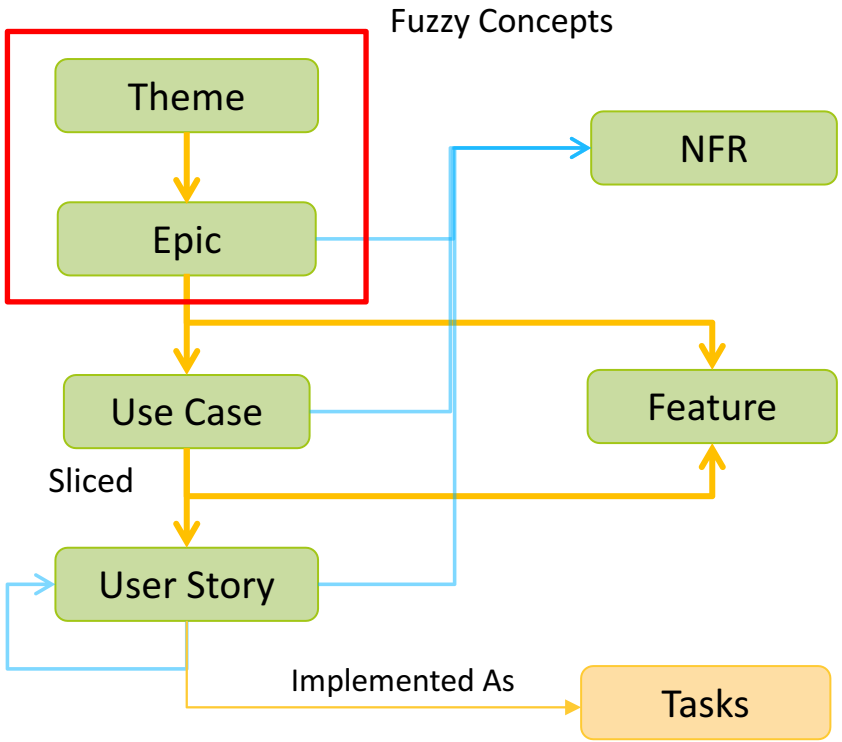
Kim Horn

Version 1.1 **1 August 2017**

Types Of Requirements



Requirements Relationships – High Level



Epics and Themes are fuzzy concepts that are commonly used as placeholders for future user stories; they have to be split before they can be implemented or tested.

Use Cases help organise and aggregate User Stories and Features and other requirements, e.g. journeys, scenarios, NFRs.

- They provide Context;

Use Cases can be sliced into User Stories (Ivar Jacobsen)

User Story

Something a stakeholder wants.
Mike Cohen : *They are descriptions of a feature told from the perspective of the person that desires that new capability.*

Used to be a simple unstructured description (like a functional spec statement), now they are commonly templated to call out the Who, What and Why:

As a	I want to	So that
Type of User (Actor)	Some Goal (Features)	Some Reason / Benefit
Who	What	Why

- Stories can be split into smaller stories, for implementation as tasks, size is variable;
- Include conditions of satisfaction; high-level acceptance tests that will be true after the user story is complete

Stories are not Requirements

A user story is an invitation to a conversation.

Ron Jeffries Stated:

"The requirement itself is communicated from customer to programmers through conversation: an exchange of thoughts, opinions, and feelings. This conversation takes place over time, particularly when the story is estimated (usually during release planning), and again at the iteration planning meeting when the story is scheduled for implementation. "

Mike Cohen Stated:

"Stories are not intended to document user needs. The story is a means to having a conversation."

Good User Stories

Six criteria (INVEST – Mike Cohen):

- **Independent**: stories should ideally be independent of the other stories, so as to facilitate prioritization.
- **Negotiable**: stories are not contracts written in stone. They form the basis of conversations, during which the story may be adjusted, or even removed.
- **Valuable:** to users or purchasers: sometimes the purchaser may have different needs from the users. E.g. a company may have requirements for consistency and cost-effectiveness that are of no interest to the users of the software.
- **Estimable**: it must be possible to estimate roughly how much time and effort would be involved in fulfilling the story.
- **Small**: If the story is too big or too small, it cannot be used for planning purposes.
- **Testable**: the story must be written in such a way that it can be tested. For example, a story to the effect that “the user must find the software easy to use” cannot be tested, whereas “the user must get a response within five seconds” can be tested.

Epics

Mike Cohen coined the term, as: *Simply a Large Story that will be broken down into smaller stories;*

- Epics are a high level description of a Business Need;
- Epics are not testable;
- **Epics can have two types:**
  - **Business**
  - **Architectural**

- My definition: “Epic” is a business digestible statement that collects and categorises Use Cases, non-functional requirements and Stories.

Themes

Mike Cohen: *A collection of User Stories*

- They are key units for funding and strategic Investment;
- Can be vague, cross systems, applications and not yet initiative or project based.
- Apply to IT as a whole;

- My definition: ”Theme” describes key value propositions that drive vision and are used for high level prioritisation and funding.

Features

IEEE standard (1990) defines the term feature as:

- (A) A distinguishing characteristic of a software item (for example, performance, portability, or functionality)
- (B) A software characteristic specified or implied by requirements documentation (for example, functionality, performance, attributes, or design constraints).

- In English it is “a distinctive attribute or aspect of something”
- Features can be viewed as aspects of the system that are orthogonal to usage or workflow; Thus User Stories cannot capture usage or workflow.
- A feature is a distinct element of functionality which can provide capabilities to the business. Features are important because they allow a customer to more readily express their needs.
- In the development of a product line they are often used to distinguish between products and establish product configurations.
- Features express requirements in terms of the solution.
- Are an orthogonal aspect to stories and epics, not hierarchical.

Scrum & Kanban Definition

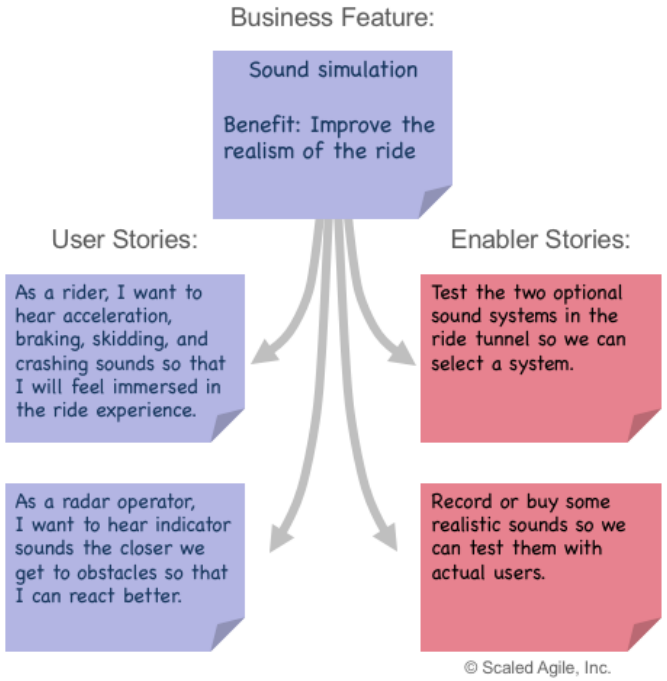
http://scrumandkanban.co.uk/theme-epic-story-task/

9

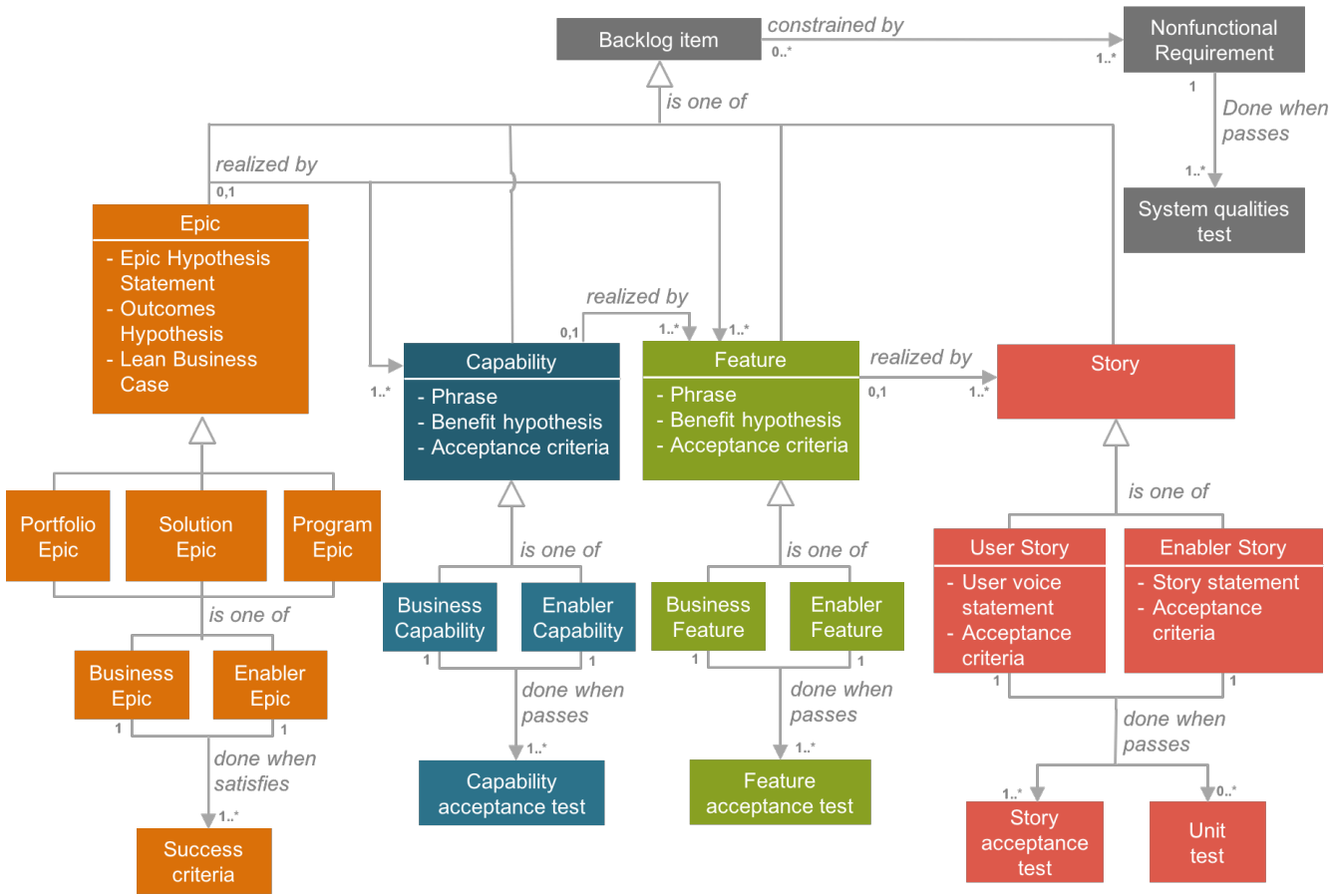
SAFE - 2 Types of Story

Two Types Of Story :

- **User Stories** - are the primary means of expressing needed functionality.
- **Enabler Stories** - Teams may need to develop the architecture or infrastructure to implement some user stories or support components of the system. In this case, the story may not directly touch any end user. These are enabler stories, and they can support exploration, architecture, or infrastructure, just like all other enablers. In these cases, the story can be expressed in technical rather than user-centric language

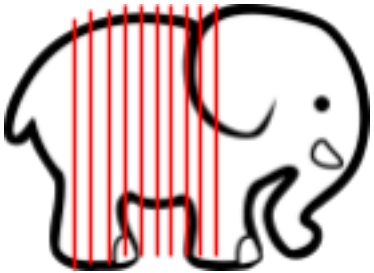


SAFE Model – Requirements require a rich language



11

Elephant Carpaccio



Ho do you slice Epics and Stories:

- Vertically or Horizontally ?
- A horizontal slice could be adding a DB component. HOWEVER:
- If you slice them into features that provide no value – THEN not releasing working software.
- Need to release a complete vertical slice from UI down….
- Alistair Cockburn created an exercise called “Elephant Carpaccio” that helps people provide nano incremental releases.
- “Elephant Carpaccio” – slice the Elephant so each slice is still elephant shaped.