Module 3 – Topic 3.3

Writing Requirements

Topic Outline

- Agile Requirements
  - How requirements fit into the scope of Agile software development
- Expressing Requirements
  - User Stories
  - Acceptance Tests
- Managing User Stories
  - Product Backlog
  - Story Maps

Recall the Agile Manifesto

Individuals & interactions	over	Process & tools
Working software	over	Comprehensive documentation
Customer collaboration	over	Contract negotiation
Responding to change	over	Following a plan

Agile Requirements

- Welcome changing requirements
  - requirement is an open-ended thing that can't be defined once

- Encourage face-to-face interaction
  - requirements should be elicited from your client in an open and collaborative setting

Quick Question

Which of the following express the role of requirements in Agile?

A. Requirements are moderately important. It's more important to define deadlines.
B. Requirements are not at all important. Agile is a process for writing code.
C. Requirements should be defined up front and do not change.
D. Requirements are incredibly important, and should be able to change throughout development.

User Stories

The Format of a User Story

Who: the stakeholder role for whom the requirement is being formed

As a <*role*>, I want to <*function*>, so that <*benefits*>

What: the function the stakeholder role wants to resolve using the product

Why: give context to the requirement about the value or benefit it offers

An example:
As a *kid customer*, I want to *view the kids menu*, so that *I see what meals and games interest me*.

Quick Question

You're working on a mobile game project with your client. They give you the requirement: "*I want to be able to control my character using the arrow keys*.

How would you structure this requirement to fit the form of a user story?

A. As a gamer, I want to be able to control my character using a directional input method, so that I can navigate the game world
B. I want to able to control my character using a directional input method
C. In order to navigate around the world, a character should be able to move using the arrow keys
D. Change nothing, this is already a user story

User Stories are Designed to be Compact

- Fit user stories on index cards or sticky notes

Client or You - Who Write User Stories?

- User stories are meant to be written by your client, but they usually lack experience, or training in creating user stories

- But, this doesn't mean you should not ask you client to create user stories

You help your client express their needs in user stories. You have to know how to write user stories in such a way that they're useful.

INVEST – Makes a Good User Story

Independent	A user story can be developed separately from any other user story
Negotiable	A user story should be general enough for the team to work around the implementation
Valuable	Each story should bring some sort of value to the client
Estimatable	You should be able to estimate how long it will take to design and implement a user story
Small	A user story should be able to be designed, coded, and tested within the iteration
Testable	you need to verify that every user story meets a certain set of criteria before it can be considered done

Putting a requirement in a user story form doesn't all of a sudden make it a good requirement! Check Reading 3.3 Write a Great User Story

Avoid Creating EPICs

- An epic is a vague, broad description of a requirement which certainly cannot be accomplished in a short time period

An example:
As a *customer*, I want to *pay for my bill*, so that *I can settle what I own quickly*

Why is it an epic?
- Does the user need to see the bill?
- Does the user pay for their bill within the app?
- What methods of payment are accepted?

Avoid Creating EPICs

- Break EPICs into multiple smaller stories by giving just enough information that the story can be understood well enough for a developer to implement it

Avoid too much design into requirements

As a *customer*, I want to *pay for my bill*, so that *I can settle what I own quickly*

As a *customer*, I want to *be able to see a bill*, with all of the items in order, so that *I can see how much my order will cost*

As a *customer*, I want to *be able to select a "pay now" option when I view my bill*, so that *I can pay the bill immediately*

As a *customer* I want to *be able to enter payment details for Visa or MasterCard credit cards,* so that *I can pay using a convenient method*

Quick Question

Imagine that you're building a software product in which users are able to choose eyeglasses to purchase from the client's website.

Which of the following user stories would be considered epics for this product?

A. As an eyeglass wearer, I want to see, so that I can slay dragons
B. As an eyeglass wearer, I want to purchase new eyeglasses, so that I can get them in the mail quickly
C. As an eyeglass wearer, I want to view a list of eyeglasses available for purchase, so that I can see what eyeglasses interest me
D. As an eyeglass wearer, I want to see the prices of eyeglasses available for purchase, so that I can compare the prices of eyeglasses

Acceptance Tests

Making user stories testable

Acceptance Criteria and Acceptance Tests

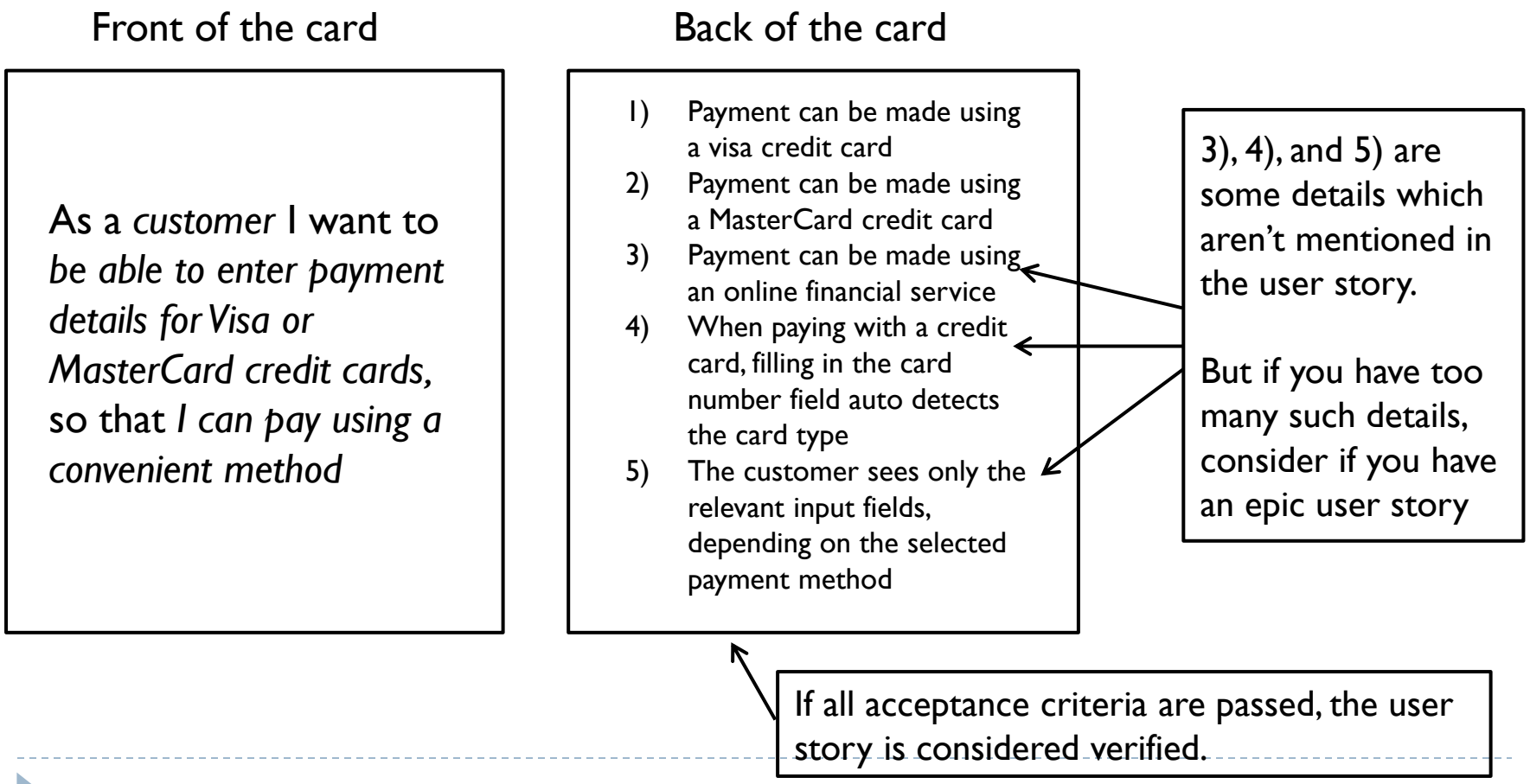
Acceptance Criteria:	Acceptance Tests :
"are specific conditions which must be met for a user story"	"are the methods for verifying whether a condition has been met or not"

In requirement elicitation, they help specify details which aren't mentioned in the user story and help avoid creating epic user story!

Once a user story is implemented, they are used to verify that the user story is done right

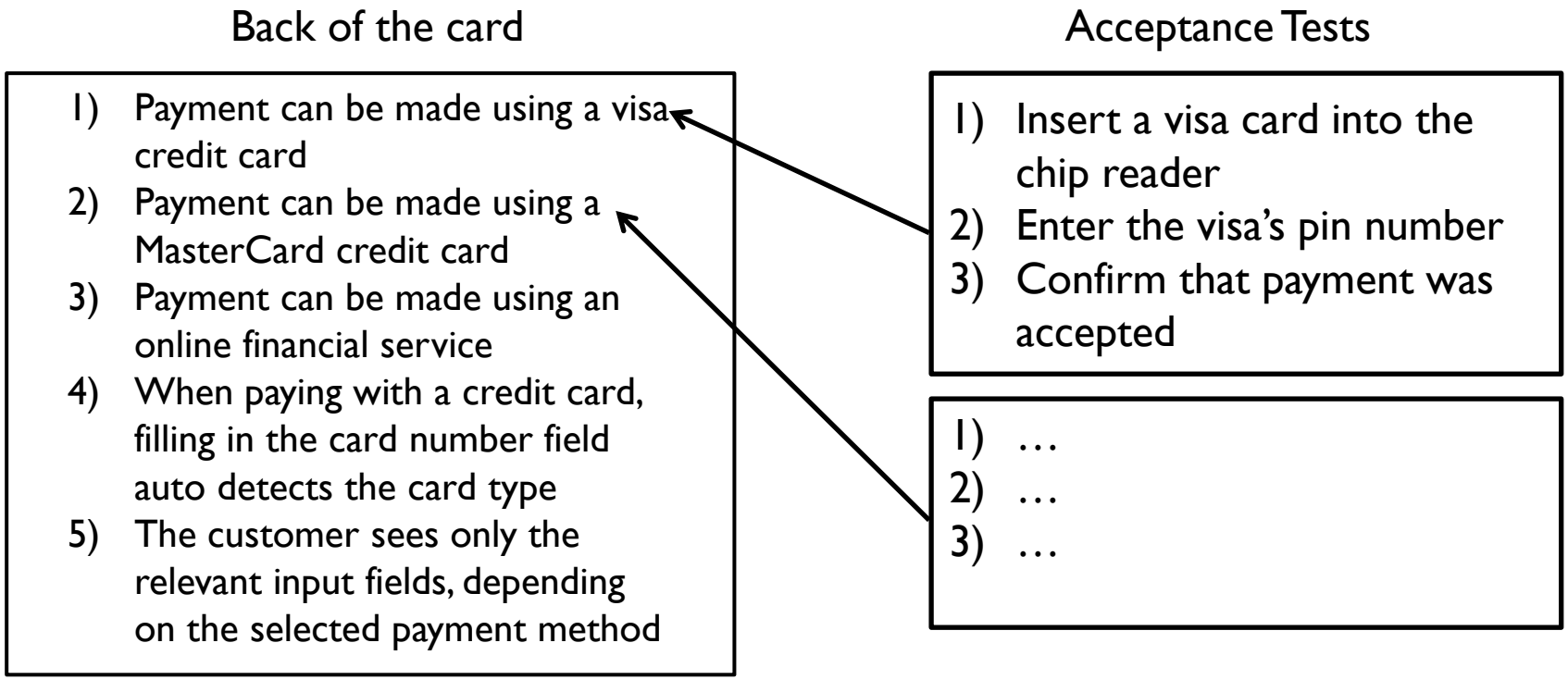
Acceptance Criteria

- Specific verifiable conditions that the user story is implemented correctly



Acceptance Tests

- Actions that show how your client will verify whether an acceptance criterion has been met or not



Quick Question

As a room service staff member I want to be able to mark a room as clean in the room overview screen, so that I can keep track of my cleaning progress.

Which of the following would be considered an acceptance test for this user story?

A. From the room overview screen, select mark room as clean
B. A room can be marked as clean from the room overview screen
C. A room should be cleaned
D. As a room service staff member, I want to be able to clean a room

Client or You – Who Create Acceptance Criteria and Tests?

- Just like user stories, acceptance criteria and tests should be formed by your client, but they usually do not know how to do it right

You create acceptance tests
alongside your client!

Subconscious Acceptance Criteria

- Subconscious acceptance criteria emerge
  - It can be difficult to articulate exactly what is considered acceptable until you see what is *not acceptable*

Discussion

As a *daddy*, I want *my kid to clean the playmat every night*, so that *the room looks neat and comfortable*

If you were my kid, what you think are acceptance criteria for a "clean" playmat?

In an agile project, when such subconscious criteria emerge, they can be discussed as future backlog items and prioritised in the context of the whole project.

Check Reading 3.4 Agile Lawn Mowing: A Story of Acceptance.

Product Backlog

Prioritize user stories by clients

Product Backlog

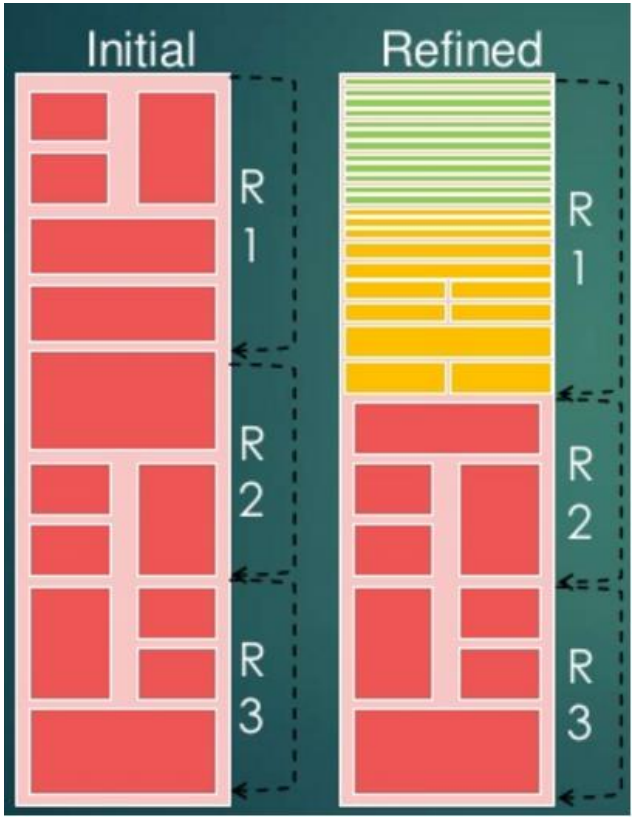
- A list of software features which you and your team intend to develop
  - User stories
  - Work tasks, e.g., set up a product testing server
  - Knowledge tasks, e.g., review options for database connection
  - Bugs

Identify user stories and other tasks in the product backlog by a unique identifier

- As an agile technique, the product backlog should be dynamic, and focused on client interaction

Prioritizing User Stories

- Initial unordered list → a refined list from highest to lowest priority



The next step is to start planning your project using these priorities as your reference point.

Check courses like COMP3120 Managing Software Development. We will recap some project planning basics in Module 5 – Topic 5.1 Agile Planning for Software Products

Be Dynamic and Responding to Changes

- It is incredibly common that requirements can change from their originally planned priorities, e.g.,
  - a requirement which was previously low priority is now extremely urgent
  - a requirement that was previously high priority is no longer desired at all
  - your client thinks of a new functionality and more user stories are added to the backlog
  - you find bugs which need to be addressed

The only place in Scrum which doesn't allow for this sort of shift, is during the current sprint, where development work is set to be done

Encourage Client Interaction

- Avoid creating a fence between the client and the development team
  - the client focuses only on prioritizing the product backlog
  - the developers focus only on estimating the efforts of implementing each user story

In Agile, we want both sides to understand the reasons why a user story has a certain priority and why an effort estimate has a certain value

Quick Question

In Scrum what role is responsible for prioritizing user stories?

A. product owner
B. development teams
C. clients
D. team leads

Quick Question

Which of the following items can be included in a product backlog?

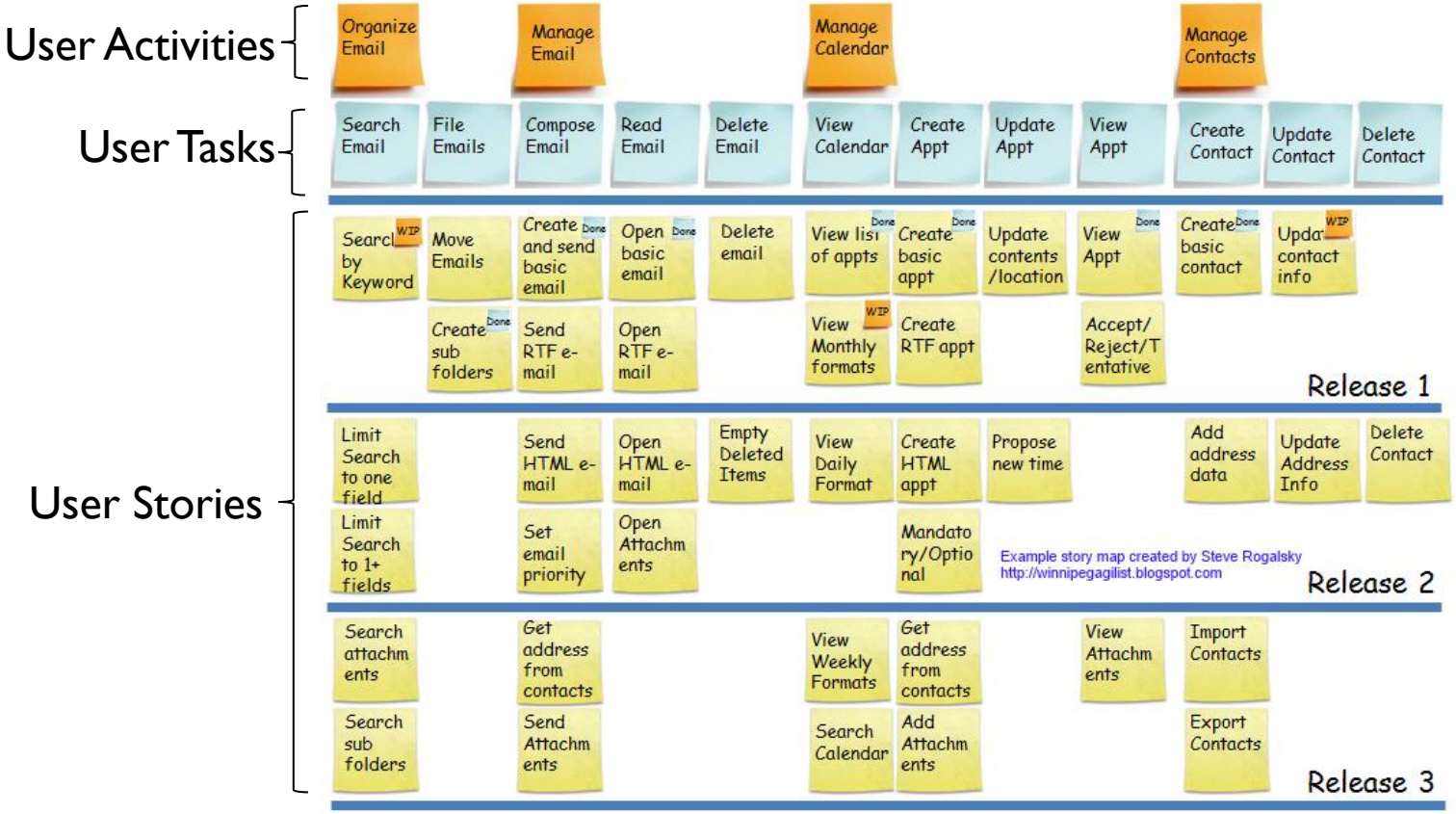
A. user requirements
B. non-work related duties, e.g., client emails
C. bugs
D. product vision
E. business requirements

Story Maps

Organizing user stories

Story Maps

- Grouping and visualizing user stories in a two dimensional structure



Quick Question

Think for a moment about what benefits creating a Story Map might bring to your project.

Please select the answers which you think reflect the benefits of a Story Map.

- saves time in the initial planning stages
- simplifies prioritization
- gives you an overall view of how the project will develop
- provides a detailed development plan

Benefits of a Story Map

- Give your backlog a more trackable, visual feel
  - How requirements fit together
- Give you perspective on how individual user stories relate
  - the developers can see how the product should be built as a whole
  - the developers can see how the product would evolve and grow as you move down the map

Covered in detail in courses like COMP3120 Managing Software Development