



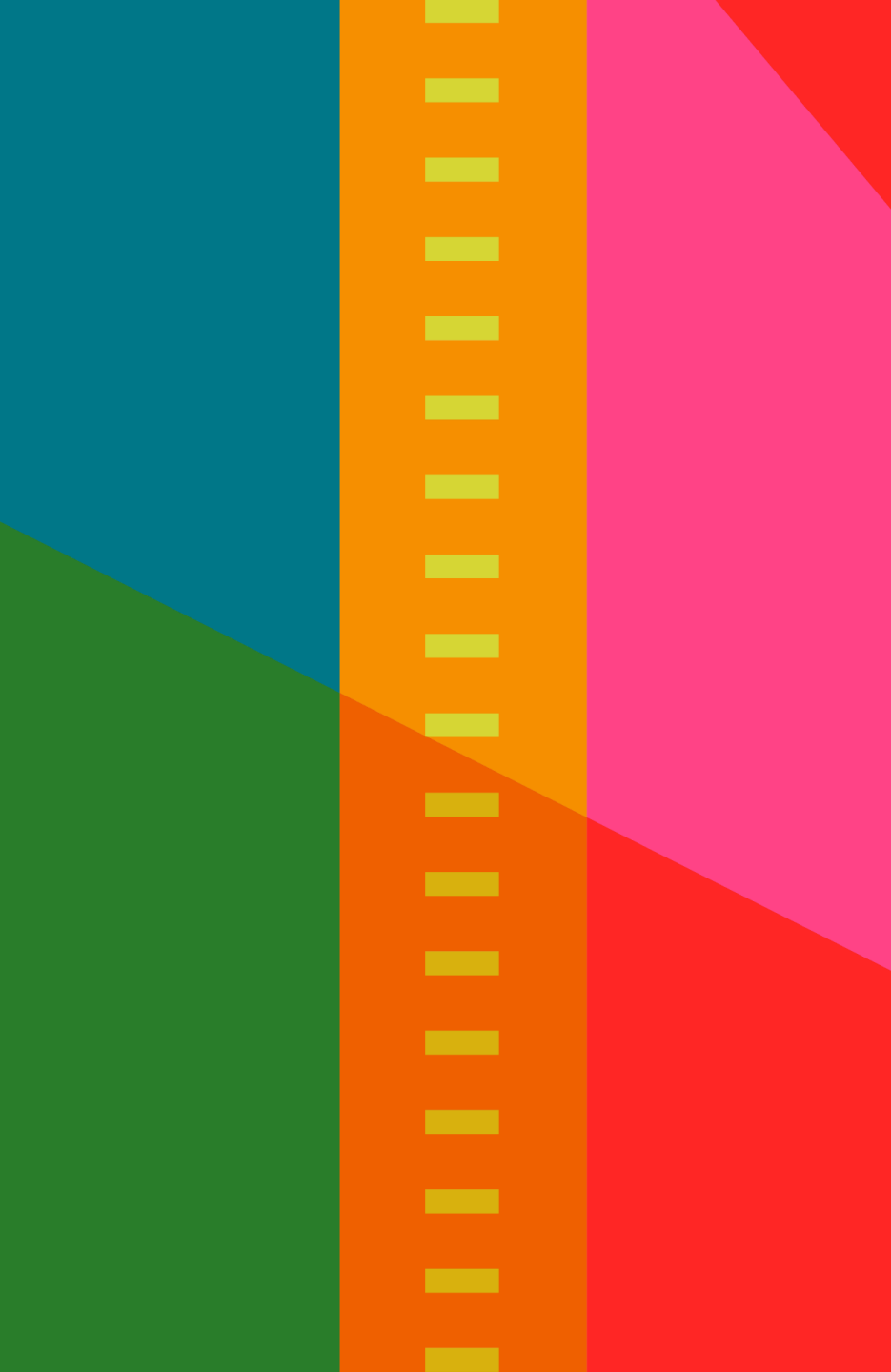
CO1605

Systems Analysis and Database Design

Julie Allen

JAllen17@uclan.ac.uk

C&T Building Room CM214



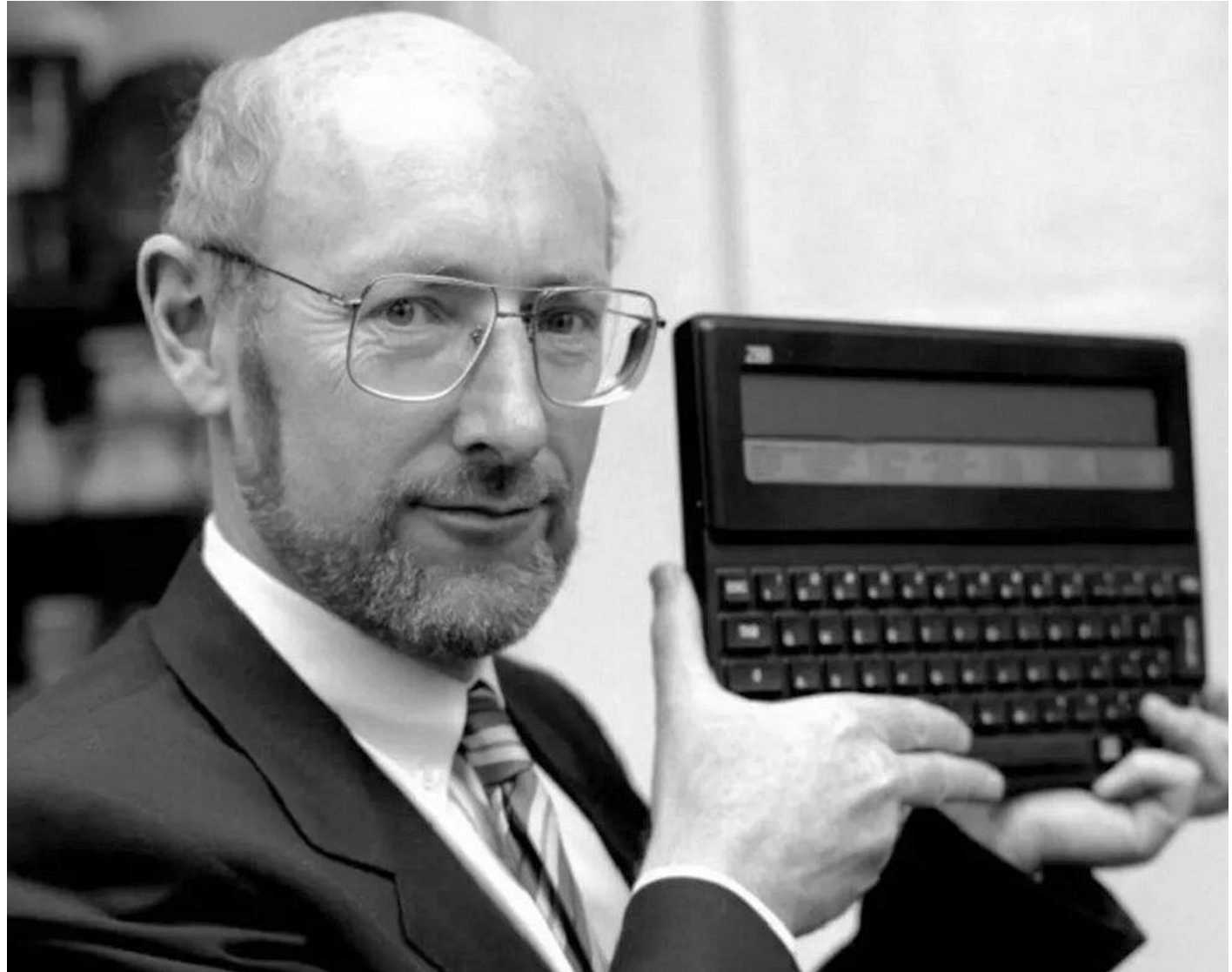
Learning Outcomes

Students will be able to:

- Understand what a requirement is
- Explain a requirement in the form of a user story
- Specify a functional requirement
- Specify a non-functional requirement

Requirements

**Clive Sinclair:
Inventor
of the
ZX80, ZX81,
and
ZX Spectrum
computers**



The Sinclair C5



Sinclair C5



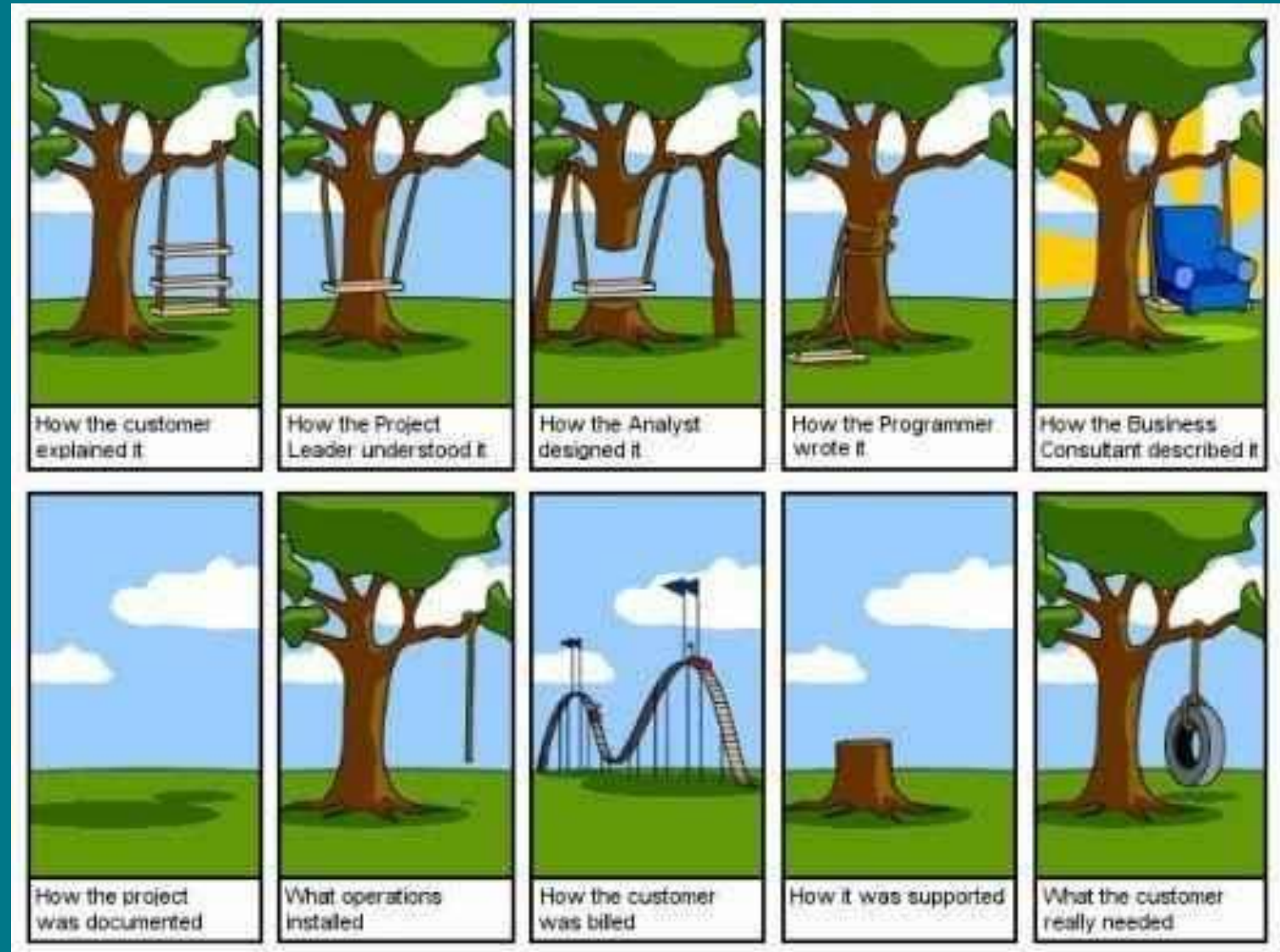
Portable TV's



Maybe the world just wasn't ready...

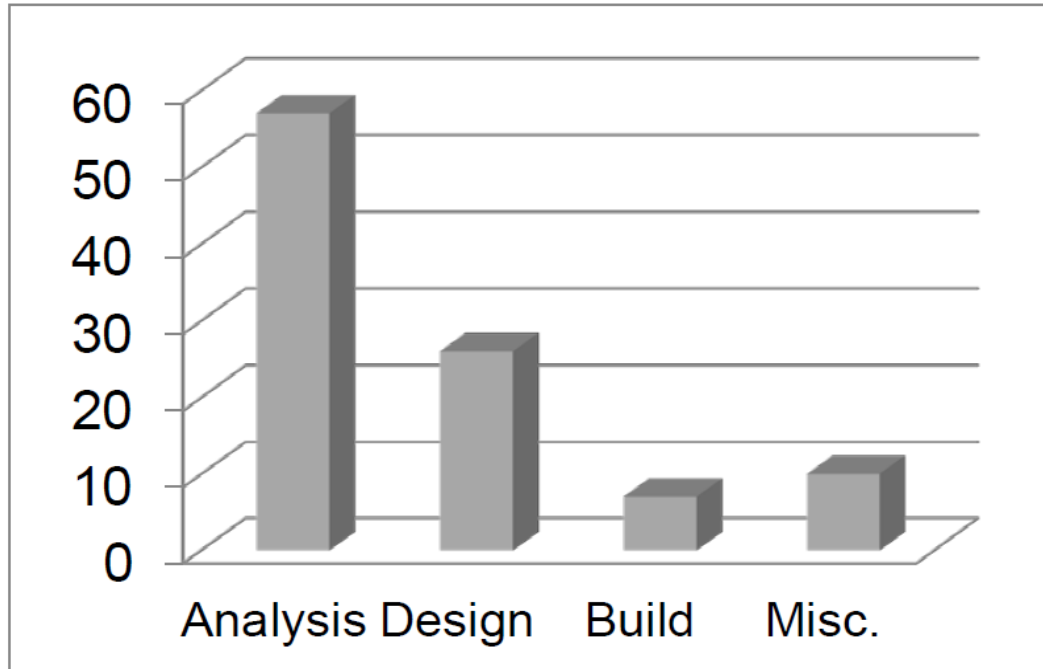


Software Development is difficult

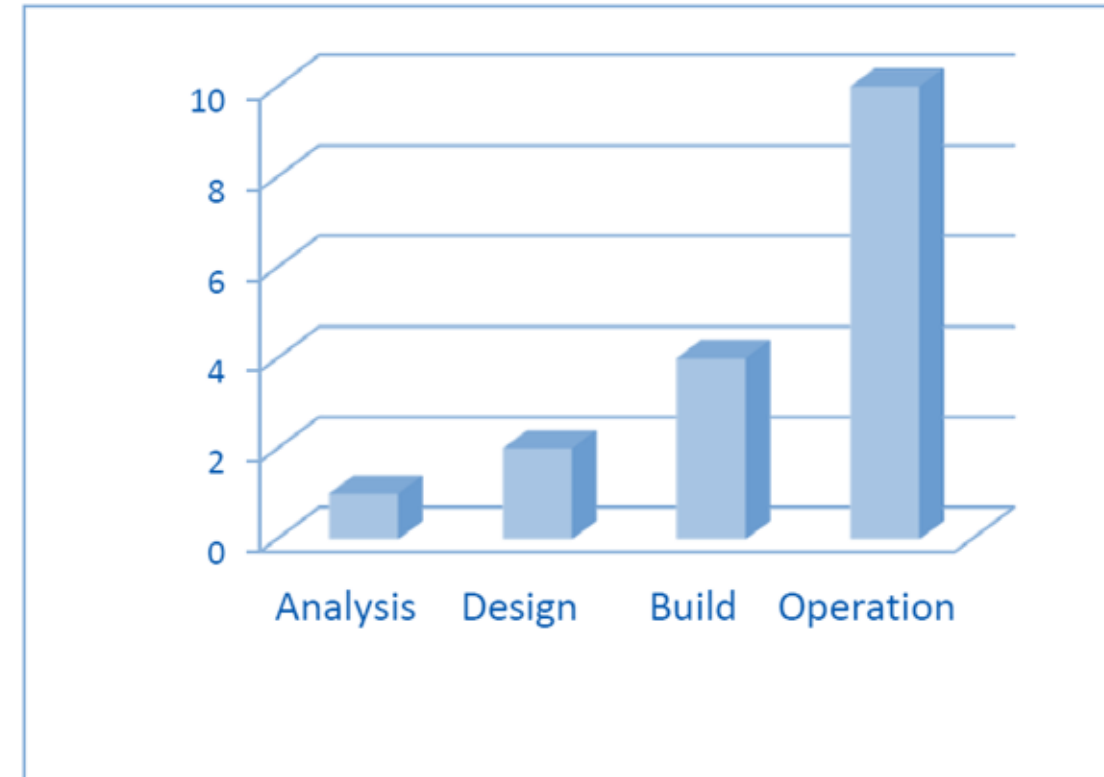


Origin of Errors

% Origin of conceptual errors found at acceptance testing – James Martin



Relative cost of correcting errors – Barry Boehm



Software Project Success Rates



System Requirements

A set of characteristics that a system must have to meet the needs of its users

- What does the application need to do?
- What is the problem that you are trying to solve?
- Why are you building the application?



2 Different Types of Requirement...

Functional

Describes **what** the system will do - the functionality:

- Inputs
- Interactions
- Outputs
- The data being stored

Non-Functional

Describes **how** the system will do it. How well it provides the functional requirements:

- Reliability
- Usability
- Maintenance
- Support
- Security
- Legal, etc.

Functional Requirements - examples

The system must...

- Capture the details of a sale to a customer.
- Allow a new member to register.
- Allow a customer
- Allow a customers details to be updated.
- Keep an up-to-date stock record.
- Create a report of customers who have purchased within the last year.

Non-Functional Requirements - examples

The system should...

- Allow for concurrent access of 100 users.
- Support 3000 transactions per hour.
- Allow only the Accounts Supervisor to change a customer's credit limit.



Problems in the Requirements Process



Problem not well defined



Users unsure what is needed



Ambiguity



Contradiction /conflict between requirements



Overlapping/ duplicate requirements



Unrealistic requirements



Not testable



Not linked to business objective

How to write good requirements (yes, there are requirements for writing requirements)

Valuable

Concise

Implementation
agnostic

Achievable

Complete

Consistent

Unambiguous

Verifiable

Atomic

Prioritising Requirements

Start with the minimum Features...

- The things that are an absolute must – the core functionality - leave out the extras, the optional features for now
- This is called the **Minimum Viable Product**
- A first pass at identifying requirements is not meant to be perfect – it can be refined later on.



MoSCoW Prioritisation



MUST HAVE

Non-negotiable
Minimum viable
product
Product cannot be
delivered without
these



SHOULD HAVE

Important but not
vital
May be painful to
leave out but the
product can be
delivered without



COULD HAVE

Desirable but less
important than
Should Have
Only do if there is
time and budget



WON'T HAVE

Won't include this
time around
Not enough budget
Would be nice but
has no real impact

An Example - The Teasmade

An automatic
all-in-one
teamaker,
alarm clock and
bedside light





Functional Requirements

The system must...

- Make a pot of tea
- Allow the user to set the alarm time
- Start making the tea 10 minutes before the alarm time
- Sound the alarm at the specified alarm time
- Allow the user to switch the alarm sound off
- Allow the user to make a tea without the alarm function
- Allow the user to use the alarm feature without making tea
- Allow the user to switch the light on and off



Non-Functional Requirements

The system should...

- Be easy to use (this is ambiguous)
...we need to define what
easy to use is here - be specific!
- Allow tea to be poured without dripping
- Keep the tea hot for... (specify how long)
- The user will be able to turn the Teasmade off with one button
- Be able to use with eyes closed
(Good/Bad?)



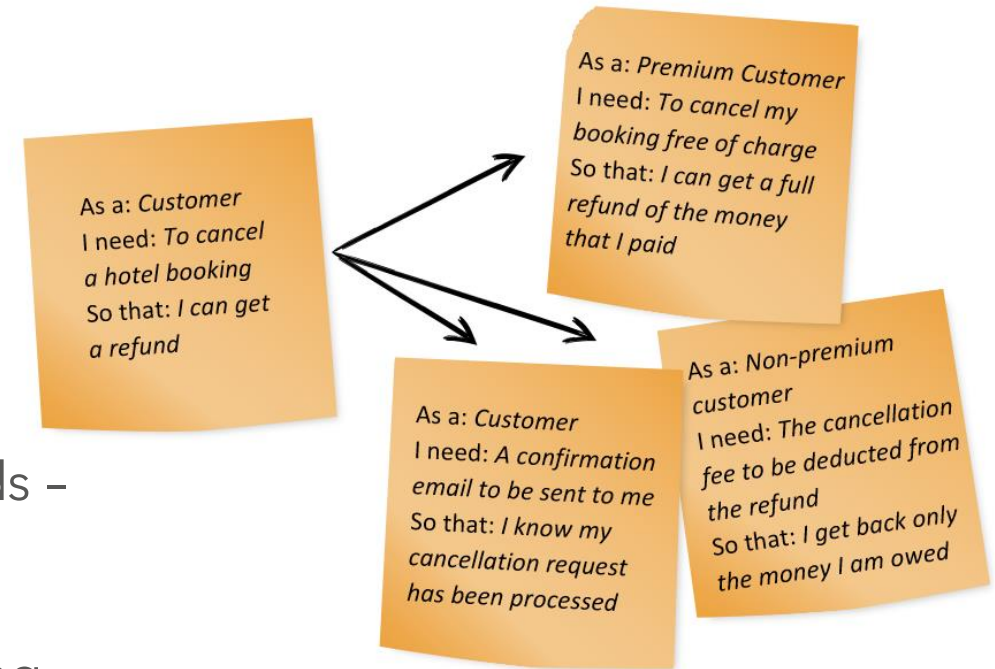
User Stories

User Story

- Describes WHO, WHAT and WHY
- Describes the customer's view of value
- A blueprint from which to collaborate
- High level acceptance criteria

Story Cards

- Traditionally written on physical index cards – nowadays can be digital or physical cards
- Keep brief and transient
- Aids agile estimating, planning and tracking



User Stories help justify and define the requirements...

As a *[who]* I want *[what]* so that *[why]*

Examples:

- As a **Stock keeper** I want an up-to-date stock record so that I know what stock needs re-ordering
- As a **Sales assistant** I need a price list so that I charge the correct price
- As a **Customer** I need a receipt so that I can get a refund

The Three C's



Card – short-hand, physical token for planning and management



Conversation – between customer and developer about the detail



Confirmation – through redefined acceptance criteria and tests



Story Cards – modify insurance quote

Front of Card

As a: *car owner*

I want: *to adjust the type of cover and the excess payment*

So that: *I can get a refund*

Back of Card

- Will my modifications appear in less than one second?
- Can I see all of the things that I want to change on the screen at the same time?

A story card is one unit of delivery for an Agile team

INVEST in User stories

I

Independent - of each other

N

Negotiable — the start point for collaboration

V

Valuable — to the customer/purchaser

E

Estimatable — able to be estimated

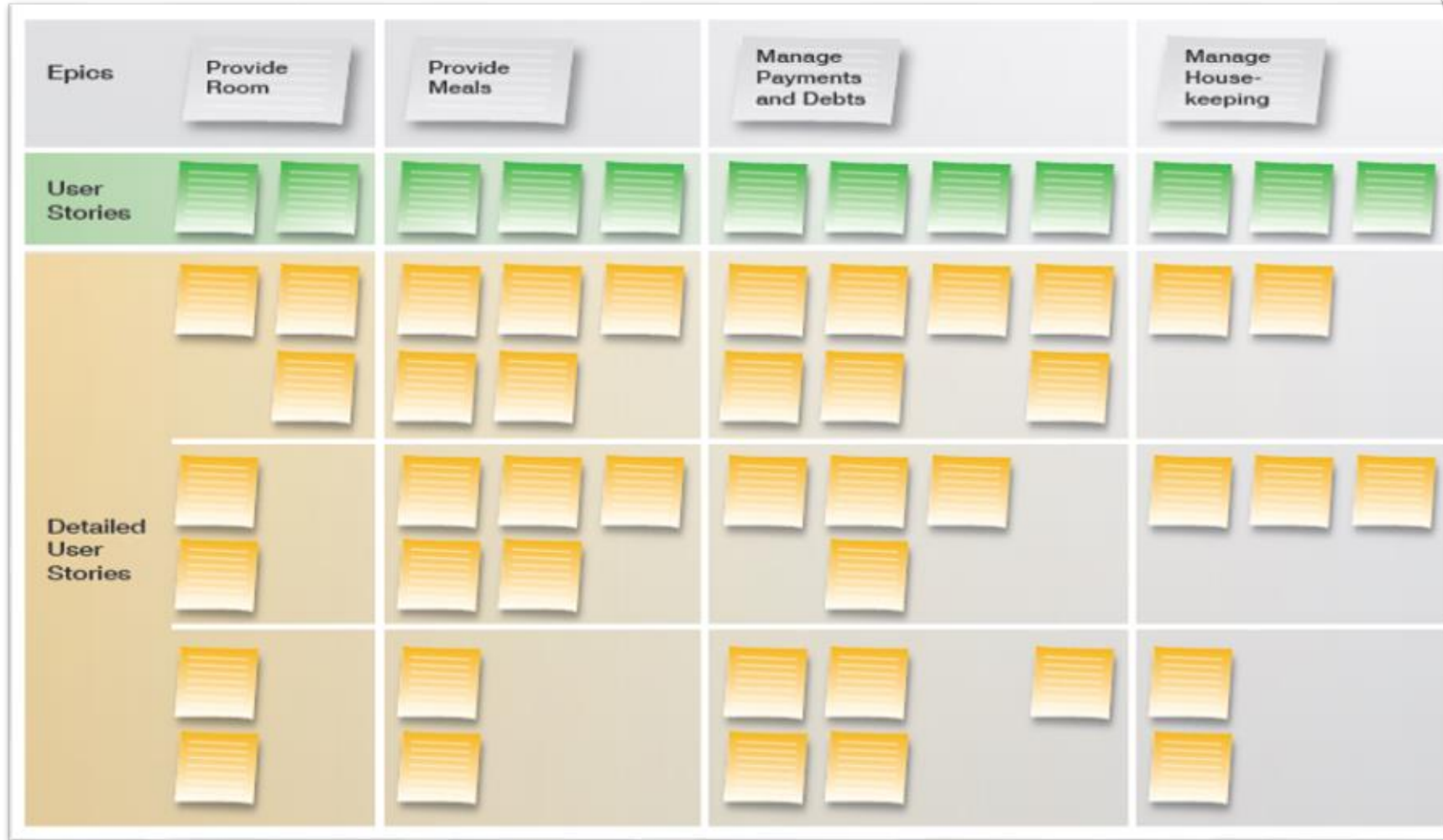
S

Small enough - 1-5 days effort

T

Testable — every story must be testable

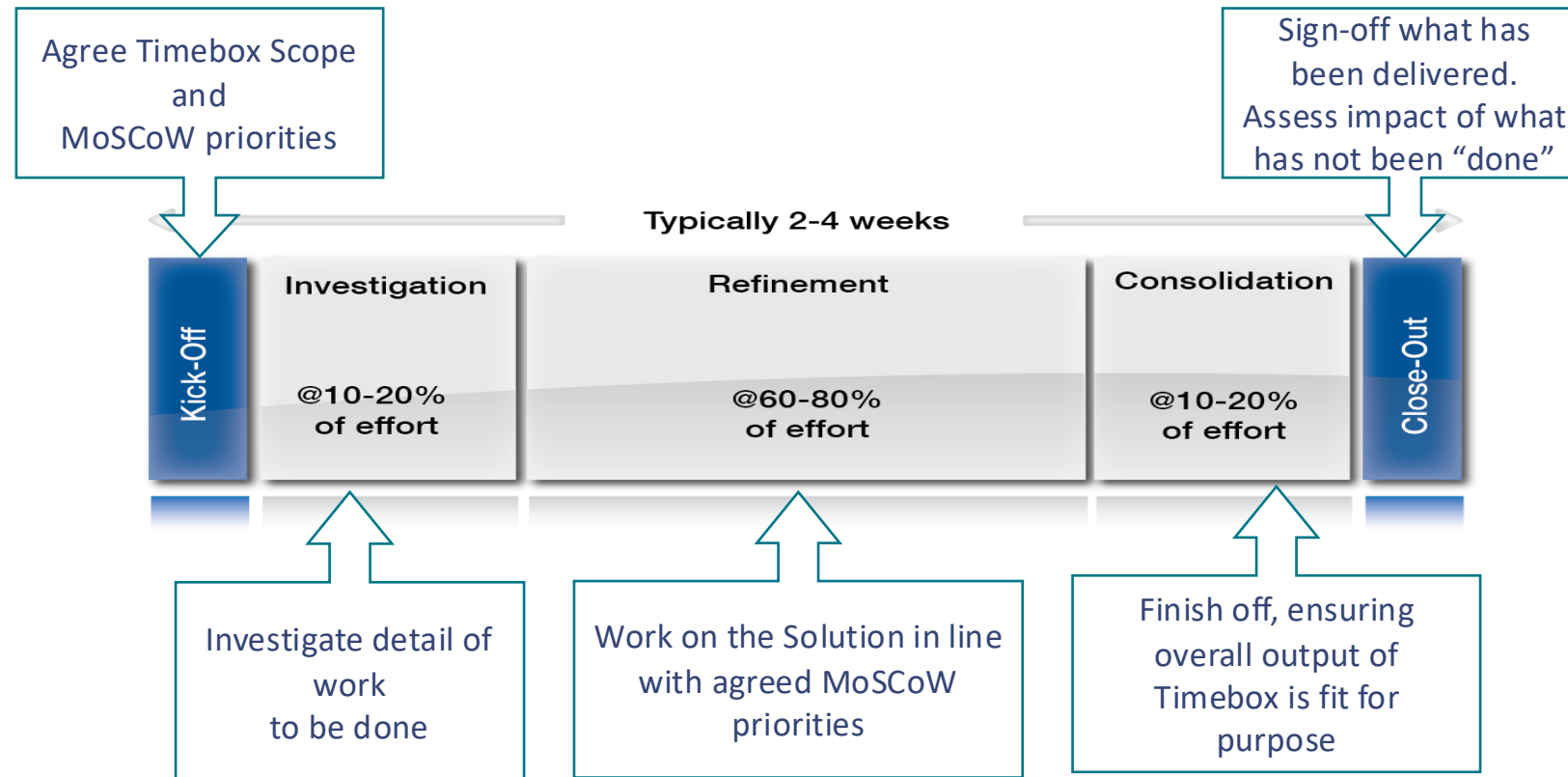
User story Mapping



Structured Timebox



Structured Timebox



Estimating



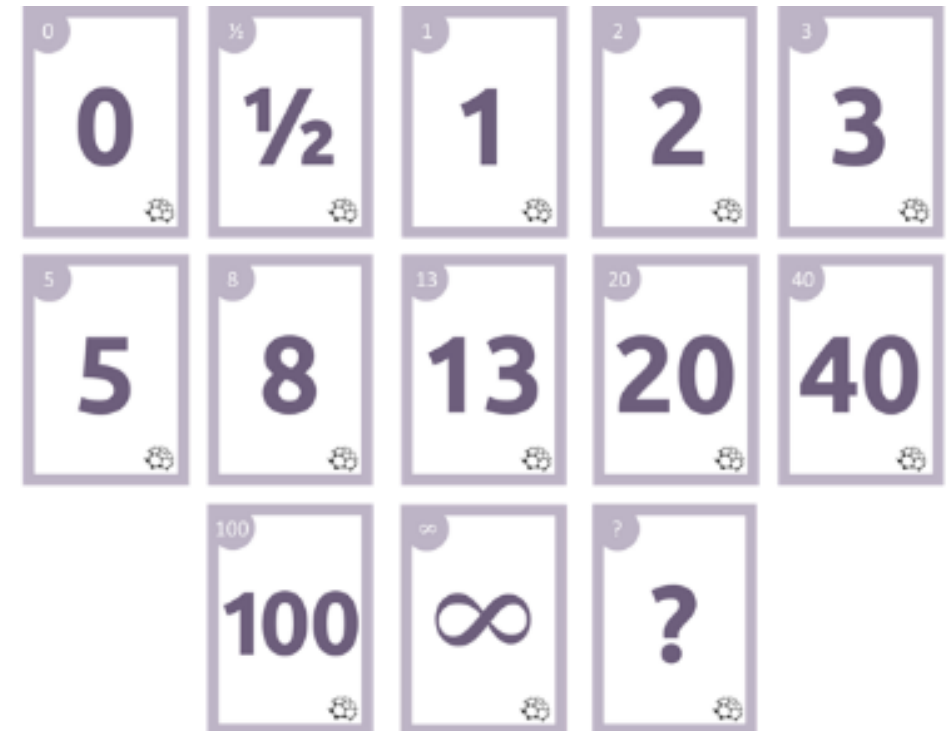
- Agile estimation of a piece of work (user story) is in terms of size, not a time-period – relative estimation not absolute estimation.
- Story Points – an abstract measure of the effort – a number which assigns a difficulty level to the story



Planning Poker

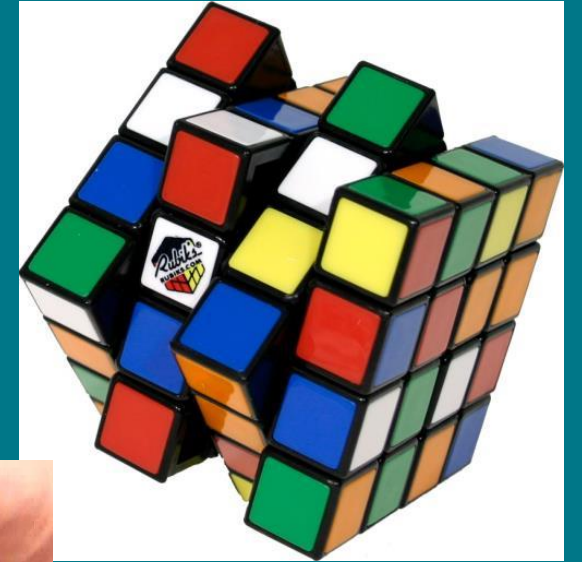


- A consensus-based approach to estimating, using relative estimates to size User Stories against each other.
- Each participant has a deck of Planning Poker cards with values in a series close to the Fibonacci series.
- The values can be used to represent story points (a relative measure), ideal days, or other units agreed by the team who are making the estimates.

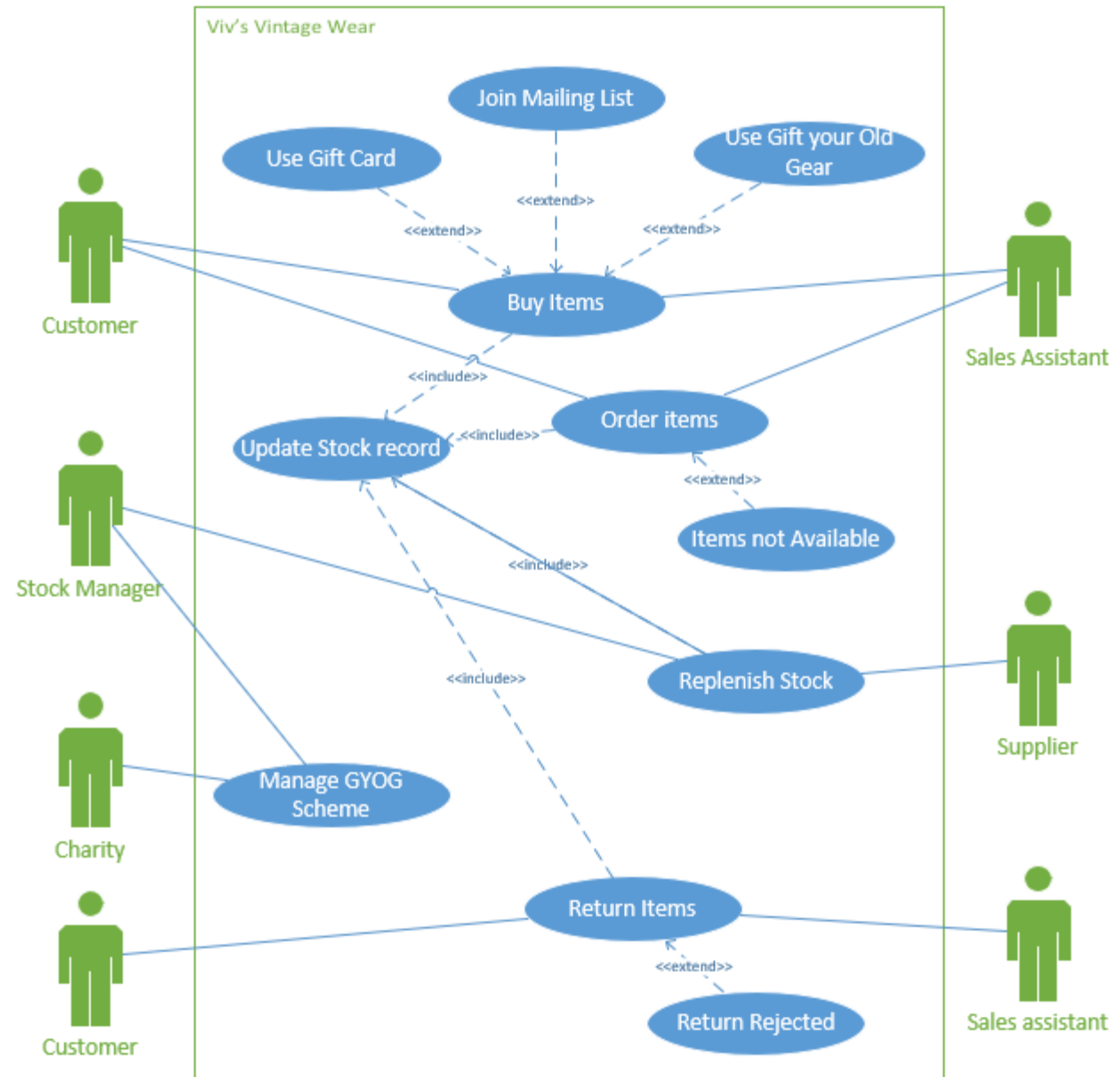


So, to sum up...

**Writing
Requirements
is Hard!**



My Use Case Diagram Solution - Viv's Vintage Wear (from last weeks lab)



Textual Description: Buy Items



Use Case Name	Buy Items
Description	Capture details of a sale and its payment
Actors	Customer (initiator), Sales assistant
Pre-conditions	Customer wants to purchase items
Post-conditions	Customer has purchased items
Main Success Scenario	1. Use case begins when Customer browses items in shop. 2. Customer selects items. 3. Sales assistant gives total amount. 4. Customer pays for items. 5. Sales Assistant issues a receipt. 6. Sales Assistant adds Customer to the mailing list. 7. Use case ends when Customer leaves the shop with items.
Alternate Paths	2a. Customer leaves without buying anything. Use case ends. 2b. Items wanted not available. 2bi. Sales Assistant orders from another store. 4ai. Customer presents a gift card 4aii. Sales assistant applies gift card discount to balance. Return to step 4. 4bi. Customer has a GYOG discount 4bii. Sales Assistant applies GYOG discount. Return to step 4. 6a. Customer does not want to be added to the mailing list.

Assignment

Released today: 27th November 2023

Hand in: 9th February 2024 at 11:59pm (submission via Blackboard)

Details are on the CO1605 Blackboard space in the Assessments folder



Any Questions?