Requirements Specification

Thomas Fritz

Many thanks to Reid C. Holmes and Elisa Baniassad

Agenda

1. Requirements Specification
2. Documenting Requirements (Reification)
3. Writing user stories (Class Activity)
4. Validation
5. Questions & More

Note:

- *Have a Java IDE available for next week!*

Examinable skills

At the end of this class, you should be able to…

- Describe the role of software specifications in software development
- Describe challenges and provide examples of why providing good specifications is hard
- Describe different ways for specifying requirements
- Specify requirements and write user stories with the correct format and meaning, including definitions of done
- Critique a user story / sets of user stories

Requirements Specification

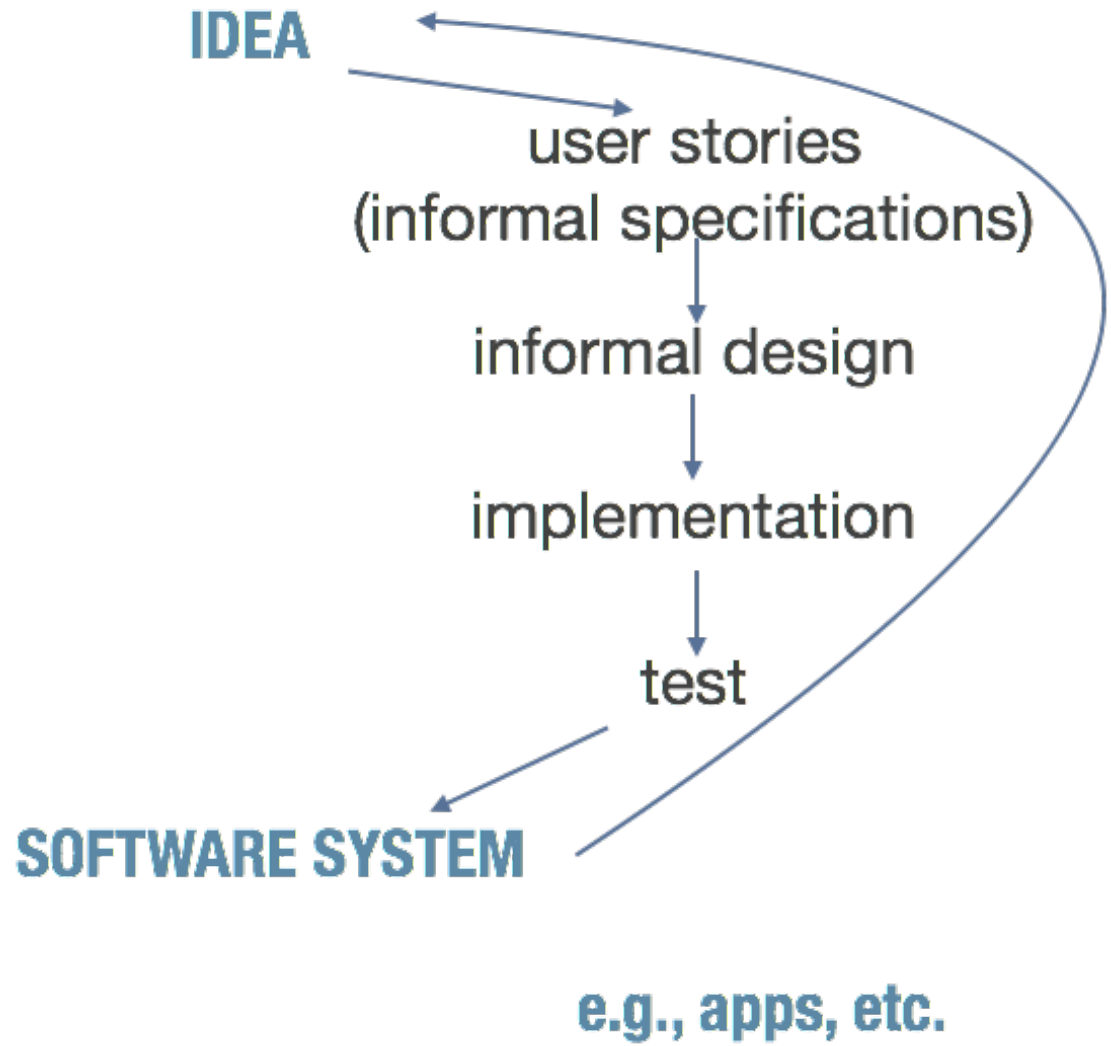
Learning Objectives

Be able to:

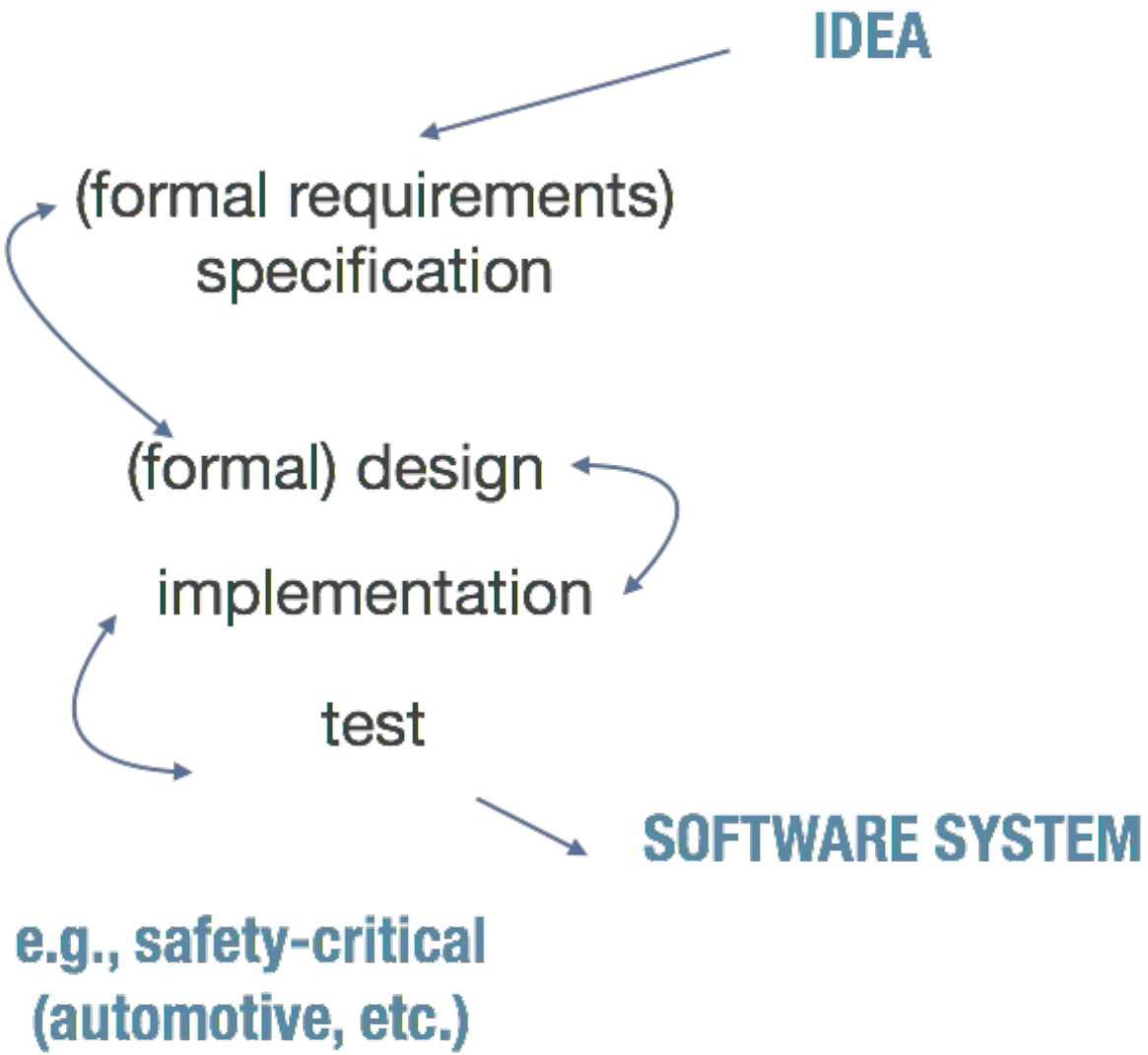
- Describe the role of software specifications in software development
- Describe challenges and provide examples of why providing good specifications is hard

Software development

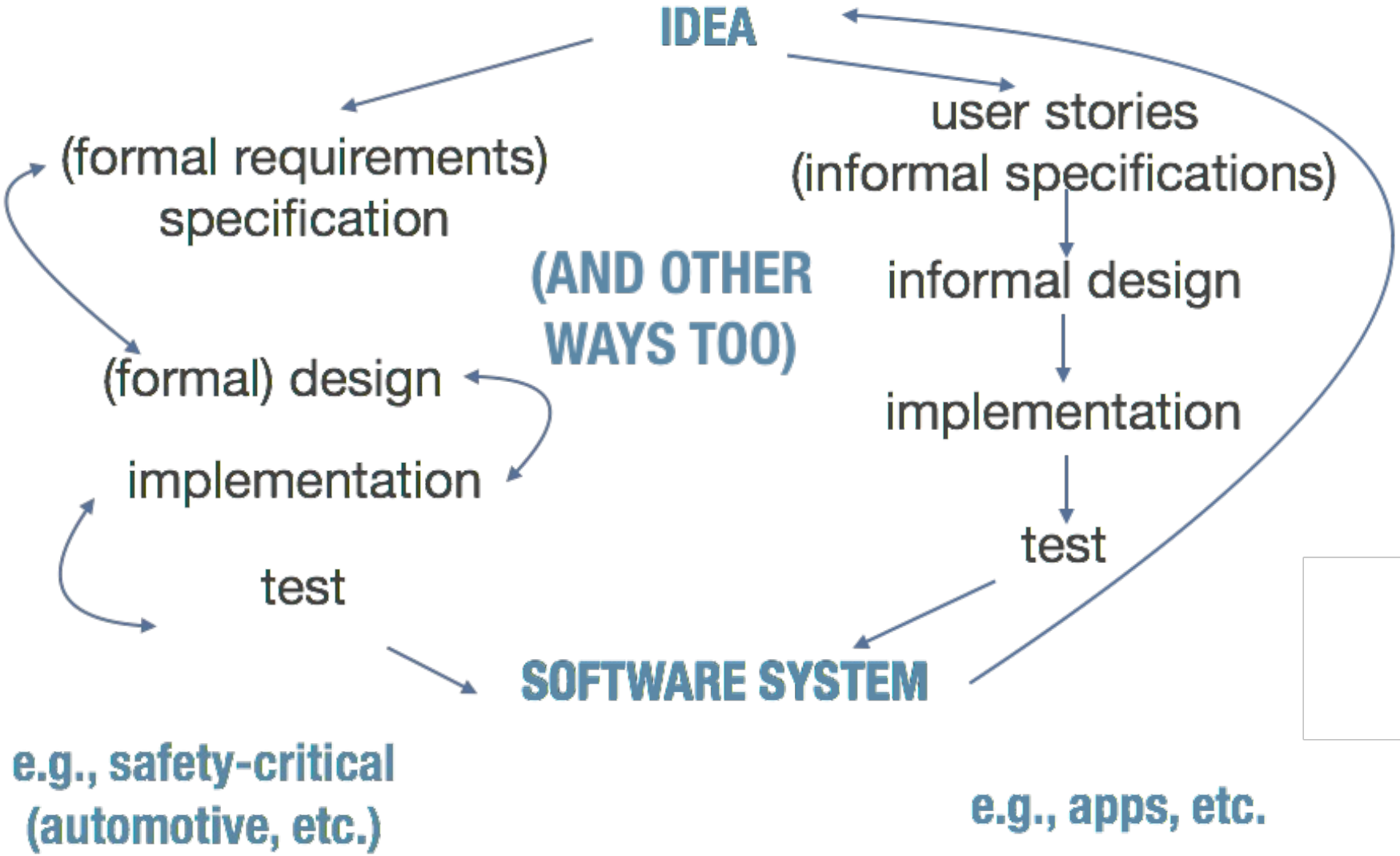
Software development



Software development



Software development



Software development – a ‘wicked problem’

A ‘wicked problem’ can only be clearly defined and understood by solving it

> *“one must solve a problem once to define it and then solve it again to create a solution that works” (Peters and Tripps)*

- Software development often considered a wicked problem

  high-level requirements can have important ramifications for low-level implementation and vice versa

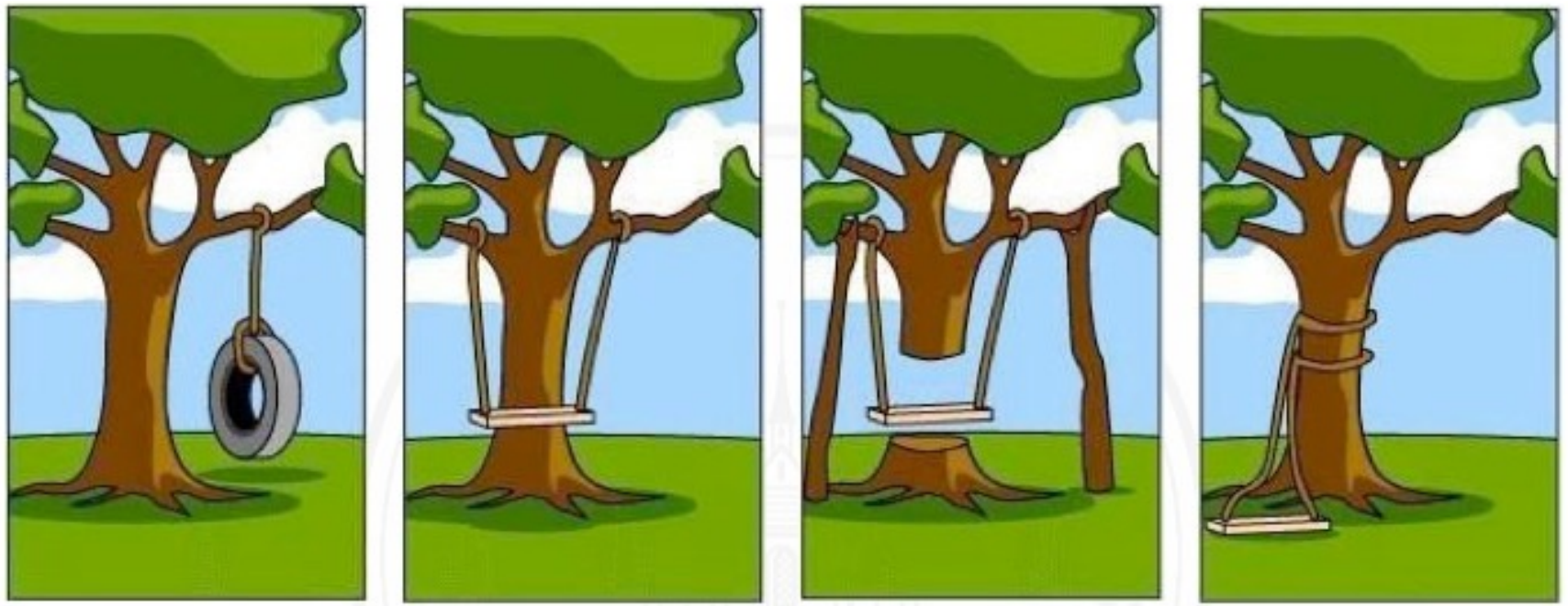
- Common characteristics of wicked problems:
  - No definitive formulation
  - No stopping rule (never really ‘done’)
  - Solutions are not right or wrong, generally just better or worse (benefits/trade-offs)

→ Discuss and interact: consensus emerges through the process of laying out alternative understandings, competing interests, priorities and constraints

Software Requirements

- Define **what** the system must do (not how it is to be achieved)
- Help to understand, communicate, connect and define when done
- Everyone must understand them, e.g. designers, developers, testers, customers, …

Telephone effect



Need	Analysis	Design	Deployed System
What the customer wanted	What the analyst understood	What the architect designed	What the programmers implemented

Software Requirements

- Define **what** the system must do (not how it is to be achieved)
- Help to understand, communicate, connect and define when done
- Everyone must understand them, e.g. designers, developers, testers, customers, …
- Specification captures
  - **Functional** requirements – what the system must do
  - **Non-functional** requirements – properties the system must have, often called ‘ilities’
- Good requirements are essential for a project to be successful, failures are costly

Class Activity – classify as functional or non-functional

Kahoot!

Kahoot

1) The system must allow the user to undo the last changes [Functional]
2) As a customer, I want to be able to run the game on all versions of Android [Non-Functional]
3) As a user, I want to be able to change the sensitivity of the mouse to adjust the moving speed [Functional]
4) As a user I want to see a warning when the connection fails to know what is going on [Functional]
5) As a user, I want the site to be available 99.9 % of the time I try to access it so that I don’t get frustrated and switch to another site [Non-Functional]

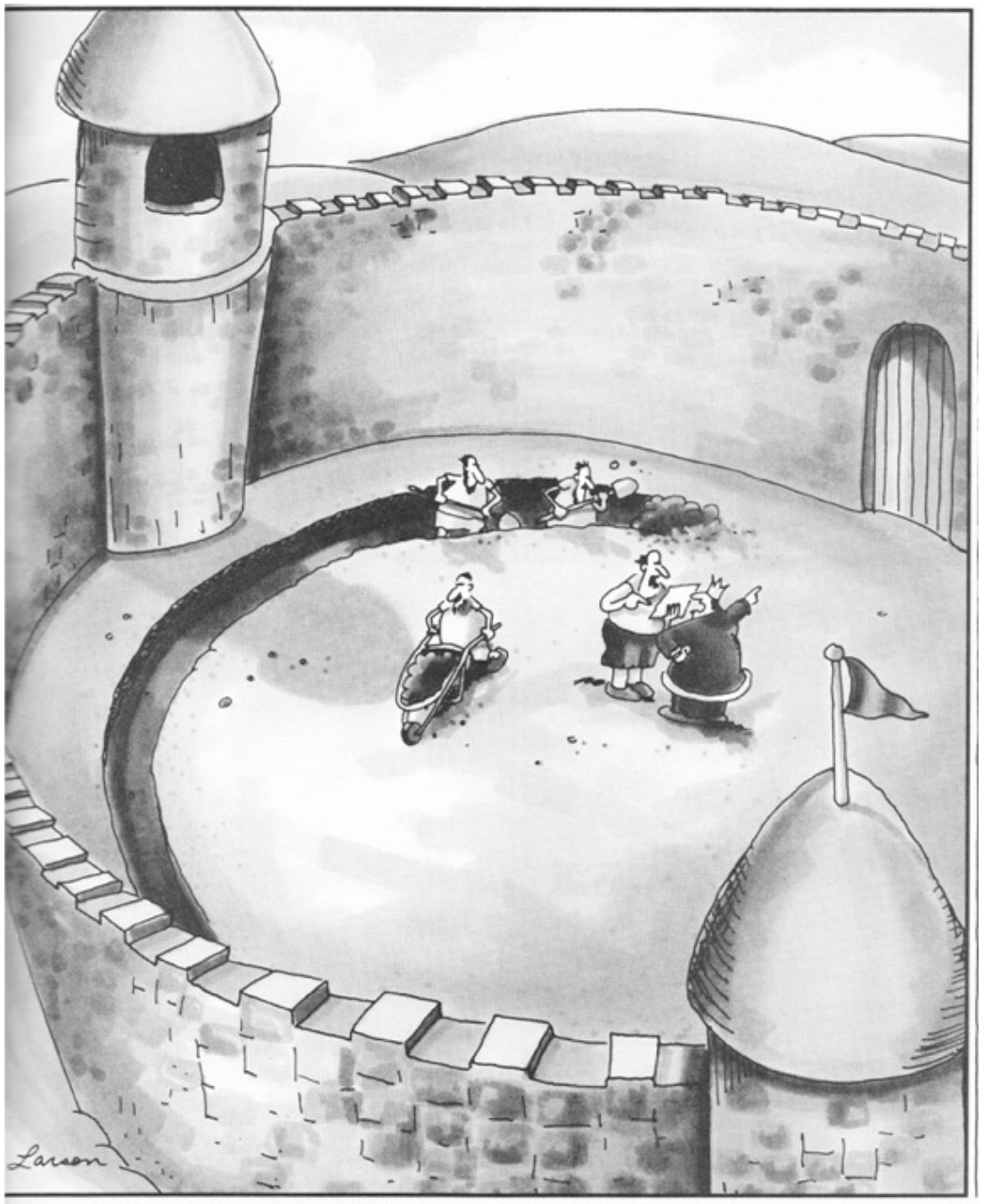
Software Requirements

- A perfect implementation is no good if it solves the wrong problem. Validation of a specification is making sure the right problem is solved
- A good specification is:
  - Complete
  - Consistent
  - Precise
  - Concise
  - Unambiguous

Imagine…

- that you are a King in medieval times, and you have asked a construction crew to build a ditch/moat around the castle that you intend to fill with water

- In your head: sketch a castle and a moat meeting this description



Vague requirements

©Gary Larson

Imagine…

- You are creating an exam for Info I and you ask students to write a function to compute the square of a number
- Is this a valid answer?

```
public static int square(int n) {
        return n*n;
}
```

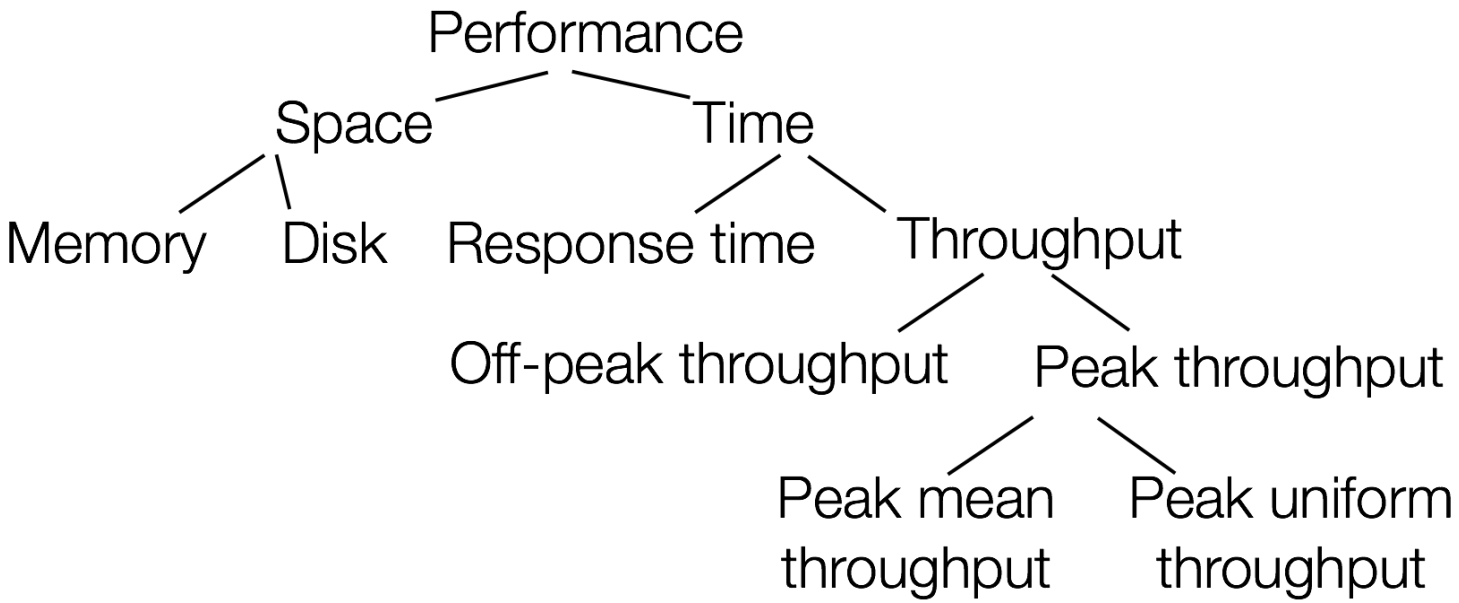
Ambiguous Requirements

©Bill Watterson
January 09, 1995

Requirements and the ability to validate/test them

‘To measure is to know. If you can not measure it, you can not improve it.’
—Lord Kelvin

Explicitly quantify requirements; these explicit measures also provide explicit goals



Examples

- Memory: the system will use less than 128MB memory while processing 100 payments
- Response time: creating a report shall take less than 20 seconds for 95% of the cases

Validating requirements – usability examples

Training effort

Task time (e.g. novice vs experienced user)

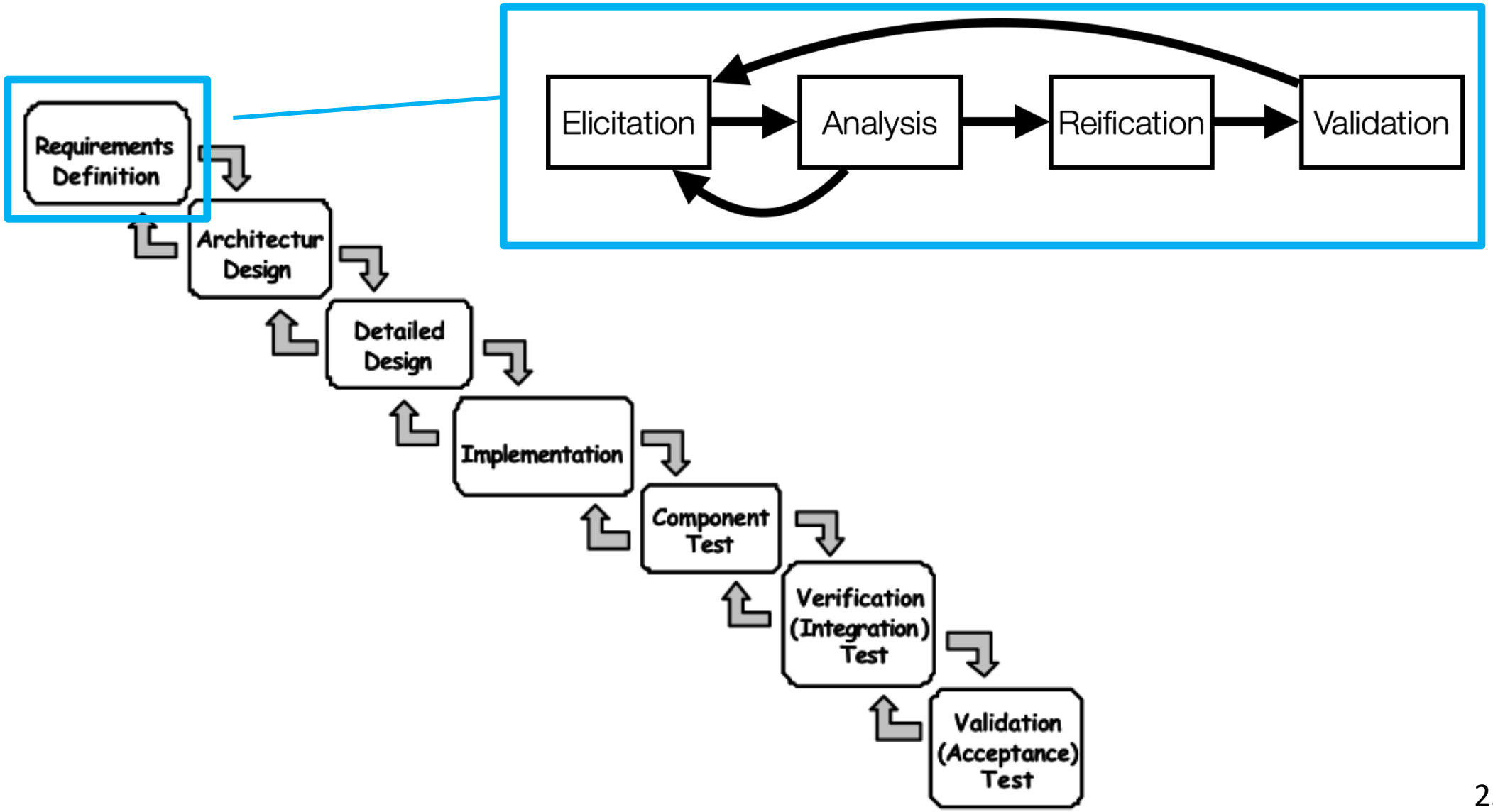
Error rate (correctness of usage)

…

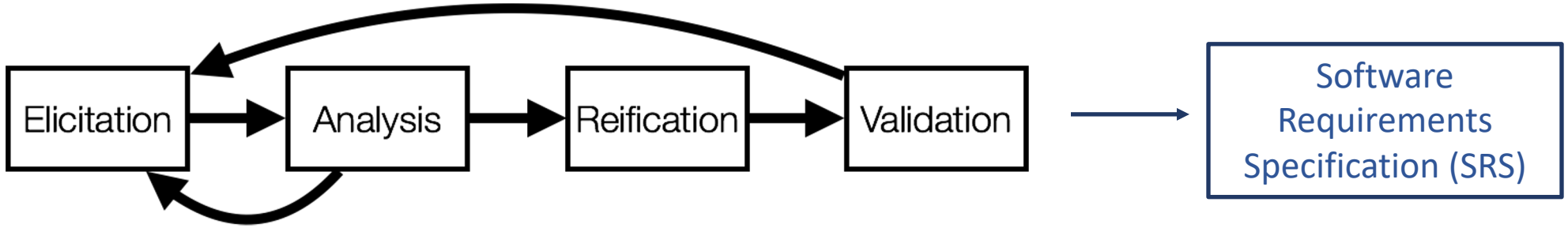
Example:

The Microsoft Windows style guide will be adhered to. Error messages must give the user specific instructions for recovery. The help system will explain all functions and how to achieve common tasks. Users will be able to consistently locate appropriate help information. All keyboard shortcuts will be customisable.

Requirements specification in the software process



Requirements process

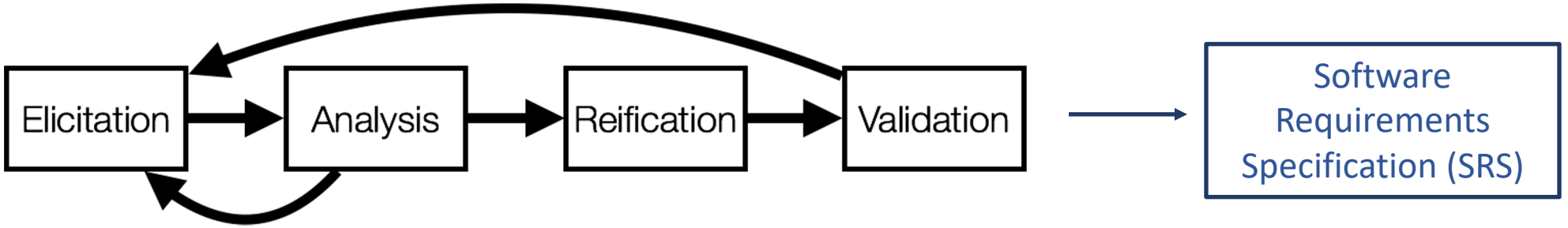


- Elicitation: gather requirements from various stakeholders

  e.g., questionnaire, interviews, brainstorm, focus group, ethnographic analysis, document study, mock-ups

- Analysis: understand and model desired behavior, check for consistency
- Reification/Specification: document the behavior of the proposed software system
- Validation: check that your specification matches the users' requirements

Requirements process



Collecting the user's requirements

Understanding and modeling the desired behavior

Documenting the behavior of the proposed system

Checking that our specification matches the users' requirements

We are going to focus on specification / reification

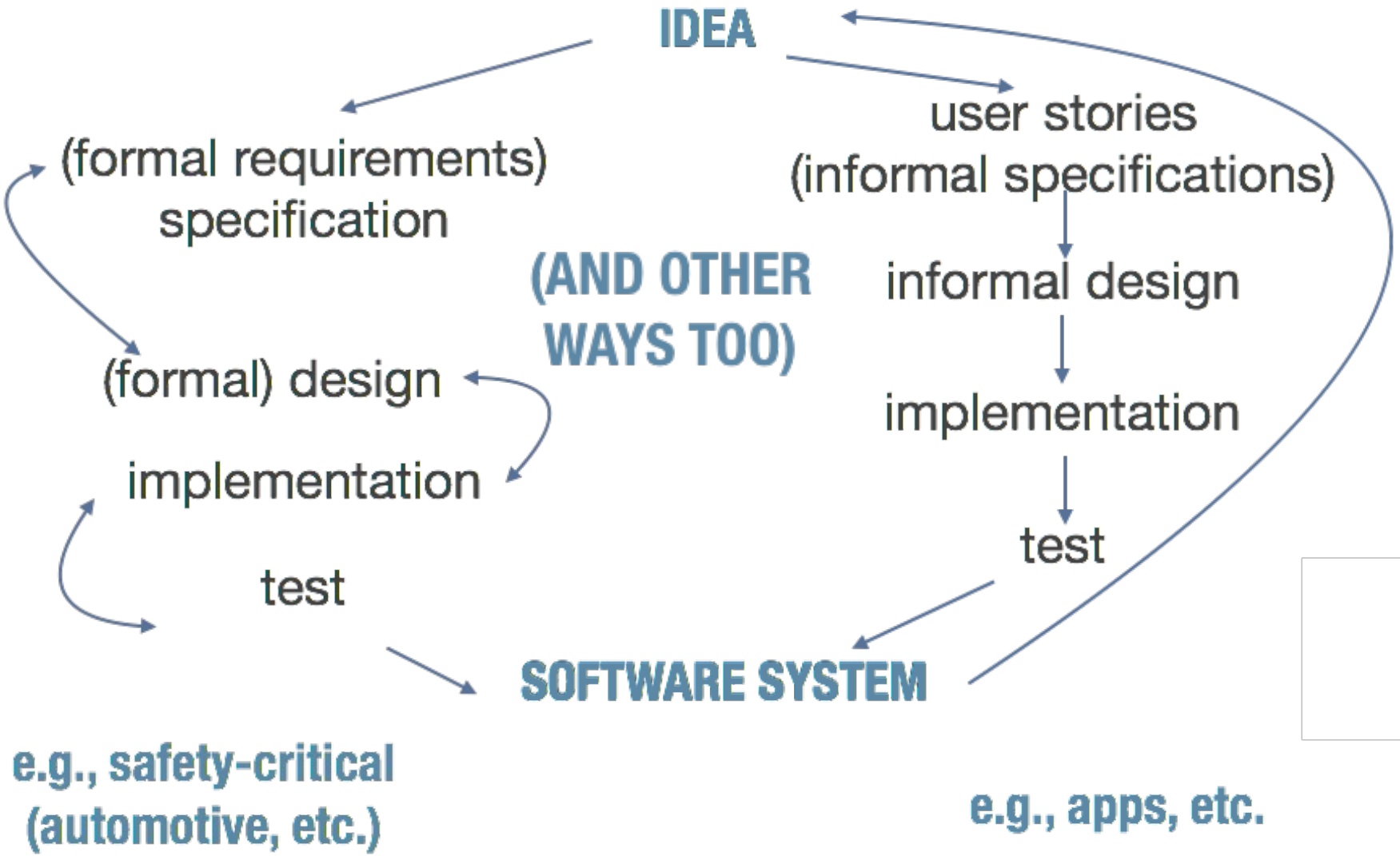
Documenting Requirements (Reification)

Learning Objectives

Be able to:

- Describe different ways for specifying requirements
- Specify requirements and write user stories with the correct format and meaning, including definitions of done

Software development



Capturing requirements

Many ways to capture / express requirements as a specification

- Formal (e.g. mathematical languages)
  - Alloy: structural modelling based on first order logic; Z; VDM; …
- Informal (e.g., natural language)
  - IEEE Standard 830
  - Use Cases
  - User stories

Requirements specification example - NASA

NASA

National Aeronautics and Space Administration

NSTS 08271

Lyndon B. Johnson Space Center

Houston, Texas 77058

SPACE SHUTTLE
FLIGHT SOFTWARE
VERIFICATION AND VALIDATION REQUIREMENTS
NOVEMBER 21, 1991

FOREWORD

Efficient management of the Space Shuttle program dictates that effective control of program activities be established. To provide a basis for management of the program requirements, directives, procedures, interface agreements, and information regarding system capabilities are to be documented, baselined, and subsequently controlled by the proper management level.

Program requirements to be controlled by the Director, Space Shuttle (Level I), have been identified and documented in Level I program requirements documentation. Program requirements controlled by the Deputy Director, Space Shuttle Program (Level II), are documented in, attached to, or referenced from Volume I through XVIII of NSTS 07700.

This document, which is to be use
community, defines the Space Shu

2.16.3.f) While acting as the bus controller, the C&C MDM CSCI shall set the e, c, w, indicator identified in Table 3.2.16-II for the corresponding RT to "failed" and set the failure status to "failed" for all RT's on the bus upon detection of transaction errors of selected messages to RTs whose 1553 FDIR is not inhibited in two consecutive processing frames within 100 msec of detection of the second transaction error if; a backup BC is available, the BC has been switched in the last 20 sec, the SPD card reset capability is inhibited, or the SPD card has been reset in the last 10 major (10 sec) frames, and either:

1. the transaction errors are from multiple RTs, the current channel has been reset within the last major frame, or
2. the transaction errors are from multiple RT's, the bus channel's reset capability is inhibited, and the current channel has not been reset within the last major frame.

Cassini Requirement: If Spacecraft Safing is requested via a CDS (Command and Data Subsystem) internal request while the spacecraft is in a critical attitude, then no change is commanded to the AACS (Attitude and Articulation Control Subsystem) attitude. Otherwise, the AACS is commanded to the homebase attitude.

```
saf: THEORY
% Example is excerpted from saf theory.
% Spacecraft safing commands the AACS to homebase mode, thereby
% stopping delta-v's and desat's.
BEGIN

aacs_mode:   TYPE = {homebase, detumble}
attitude:    TYPE

cds_internal_request:   VAR bool
critical_attitude:    VAR bool
prev_aacs_mode:     VAR aacs_mode

aacs_stop_fnc (critical_attitude, cds_internal_request, prev_aacs_mode):
        aacs_mode =
        IF critical_attitude
          THEN IF cds_internal_request
               THEN prev_aacs_mode
               ELSE homebase
               ENDIF
          ELSE homebases
        ENDIF

aacs_safing_req_met_1:  LEMMA
      (critical_attitude AND cds_internal_request)
      OR (aacs_stop_fnc (critical_attitude, cds_internal_request,
         prev_aacs_mode) = homebase)

END saf
```

Requirements specification – Use Cases (casual)

Use Case 1: Buy something

The Requestor initiates a request and sends it to her or his Approver. The Approver checks that there is money in the budget, check the price of the goods, completes the request for submission, and sends it to the Buyer. The Buyer checks the contents of storage, finding best vendor for goods. Authorizer: validate approver's signature . Buyer: complete request for ordering, initiate PO with Vendor. Vendor: deliver goods to Receiving, get receipt for delivery (out of scope of system under design). Receiver: register delivery, send goods to Requestor. Requestor: mark request delivered..

At any time prior to receiving goods, Requestor can change or cancel the request. Canceling it removes it from any active processing. (delete from system?) Reducing the price leaves it intact in process. Raising the price sends it back to Approver.

Cockburn, A. *Writing Effective Use Cases.* Addison-Wesley

Fully-dressed Use Case

Use Case 1: Buy something

Context of use: Requestor buys something through the system, gets it.

Scope: Corporate - The overall purchasing mechanism, electronic and non-electronic, as seen by the people in the company.

Level: Summary

Preconditions: none

Success End Condition: Requestor has goods, correct budget ready to be debited.

Failed End Protection: Either order not sent or goods not being billed for.

Primary Actor: Requestor

Trigger: Requestor decides to buy something.

Main Success Scenario

1. **Requestor**: initiate a request
2. **Approver**: check money in the budget, check price of goods, *complete request for submission*
3. **Buyer**: check contents of storage, find best vendor for goods
4. **Authorizer**: *validate approver's signature*
5. **Buyer**: *complete request for ordering, initiate PO with Vendor*

more of it….

6. Vendor: deliver goods to Receiving, get receipt for delivery (out of scope of system under design)

7. Receiver: *register delivery*, send goods to Requestor

8. Requestor: *mark request delivered*.

Extensions

1a. Requestor does not know vendor or price: leave those parts blank and continue.

1b. At any time prior to receiving goods, Requestor can change or cancel the request.
Canceling it removes it from any active processing. (delete from system?)
Reducing price leaves it intact in process.
Raising price sends it back to Approver.

2a. Approver does not know vendor or price: leave blank and let Buyer fill in or call back.

2b. Approver is not Requestor's manager: still ok, as long as approver signs

2c. Approver declines: send back to Requestor for change or deletion

3a. Buyer finds goods in storage: send those up, reduce request by that amount and carry on.

3b. Buyer fills in Vendor and price, which were missing: gets resent to Approver.

4a. Authorizer declines Approver: send back to Requestor and remove from active processing.

5a. Request involves multiple Vendors: Buyer generates multiple POs.

5b. Buyer merges multiple requests: same process, but mark PO with the requests being merged.

6a. Vendor does not deliver on time: System does *alert of non-delivery*

7a. Partial delivery: Receiver marks partial delivery on PO and continues

7b. Partial delivery of multiple-request PO: Receiver assigns quantities to requests and continues.

8a. Goods are incorrect or improper quality: Requestor does *refuse delivered goods*. (what does this mean?)

8b. Requestor has quit the company: Buyer checks with Requestor's manager, either *reassign Requestor*, or return goods and *cancel request*.

Still more

Deferred Variations

none

Project Information

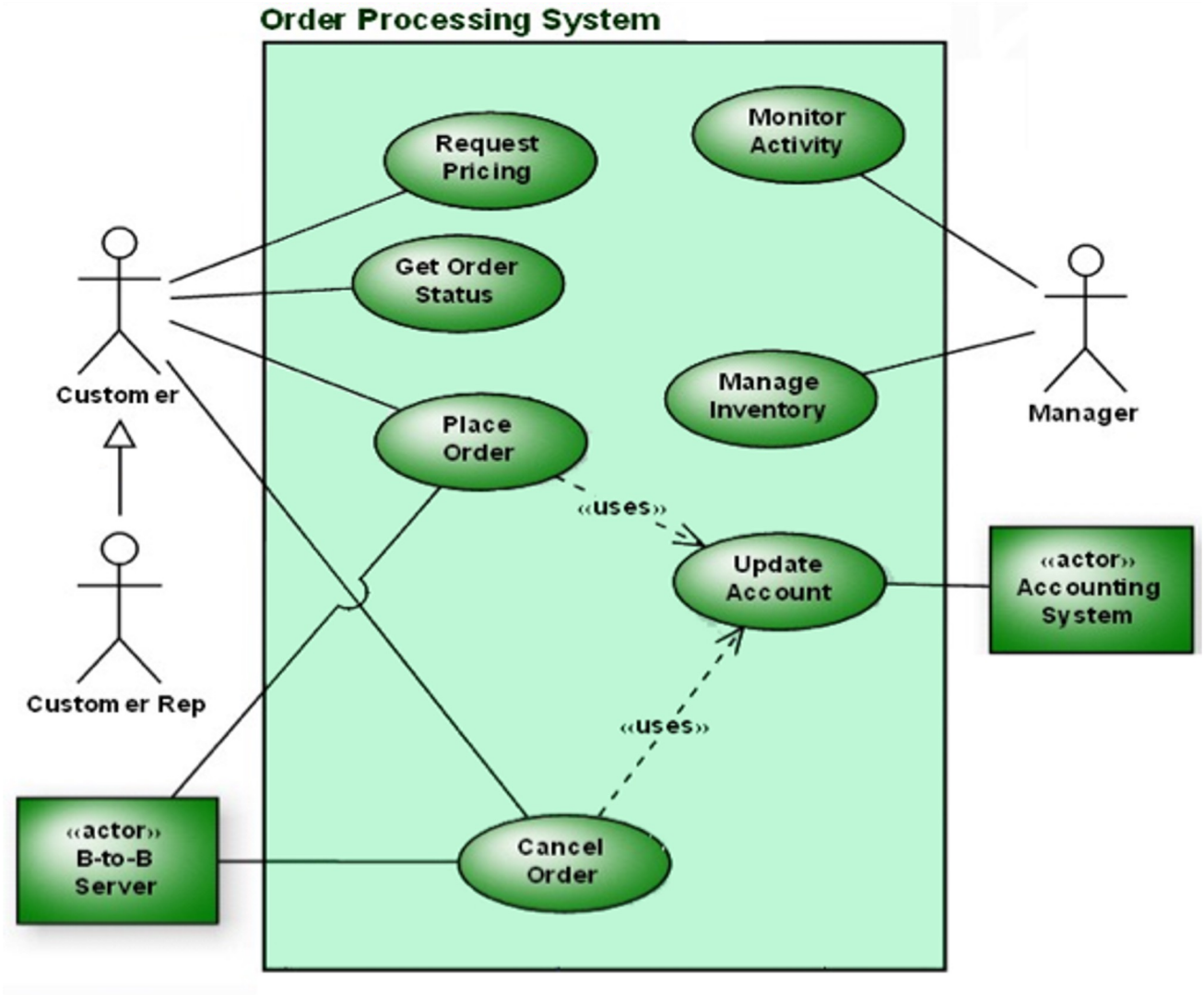
Priority	Release Due	Response time	Freq of use	
Various	Several		Various	3/day

Calling Use Case: none
Subordinate Use Cases: see text
Channel to primary actor: Internet browser, mail system, or equivalent
Secondary Actors: Vendor
Channels to Secondary Actors: fax, phone, car
Open issues

When is a canceled request deleted from the system?
What authorization is needed to cancel a request?
Who can alter a request's contents?
What change history must be maintained on requests?
What happens when Requestor refuses delivered goods?

Use Case diagrams

Show packaging, decomposition, actors,…



Writing User Stories

Learning Objectives

Be able to:

- Describe challenges and provide examples of why providing good specifications is hard
- Specify requirements and write user stories with the correct format and meaning, including definitions of done

User Story

- ~3-sentence description of what a software feature should do, focused on the value or result that a user will receive from doing this thing
- Written in the customer's language
- Should only provide enough detail to make a low-risk time estimate (a few days)
- Used a lot in agile development

User Story – Role-Goal-Benefit

- Title written in customer's language and in the format:

  "As a <ROLE>, I want to <GOAL> in order to <BENEFIT>"

  e.g. As a user, I want to login to the game with name & password in order to play

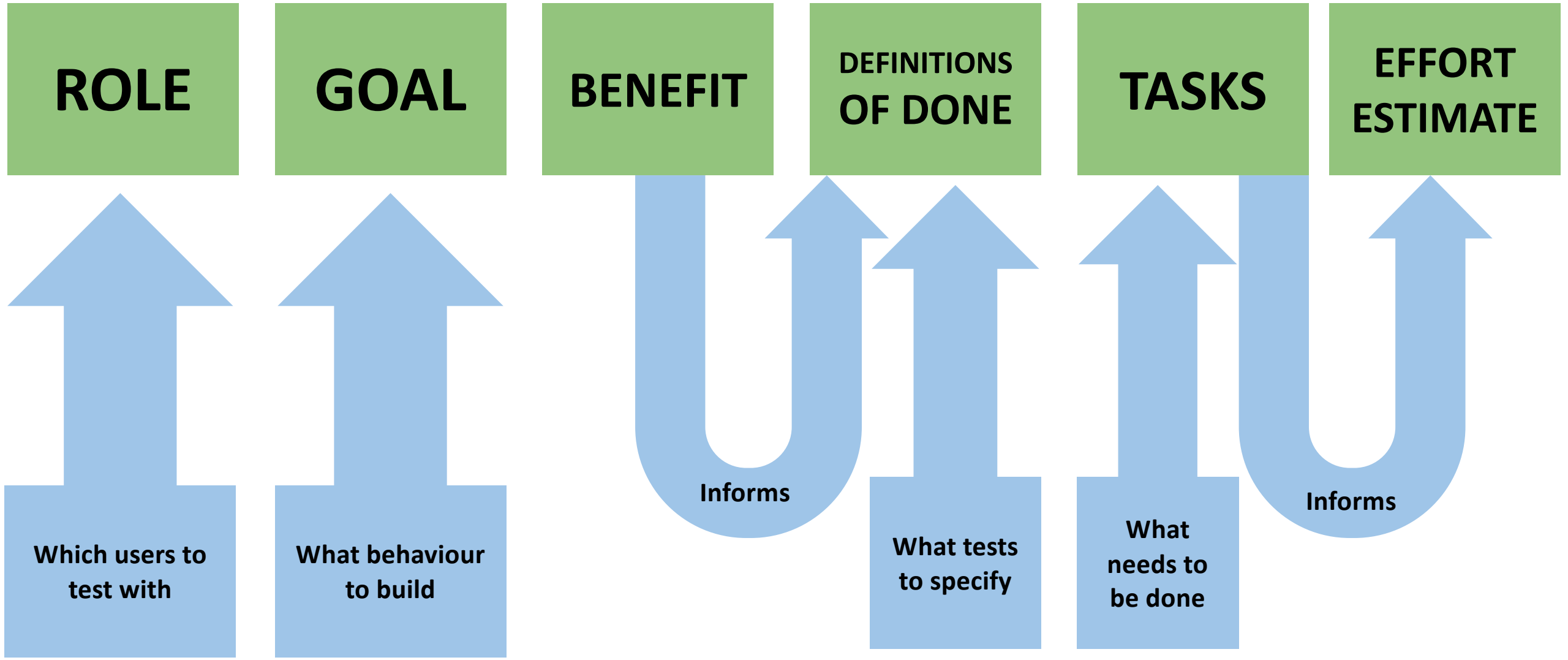
- Notes / Description / Limitations
- Definition of Done / Acceptance Tests / Acceptance Criteria

  specific description of how the story can be validated by developer and product owner to ensure it's completed correctly; also helps avoid unnecessary functionality

- Tasks

  SE tasks / coding that you will need to do to implement the whole user story

- Effort estimate (overall cost)


ROLE
GOAL
BENEFIT
DEFINITIONS OF DONE
TASKS
EFFORT ESTIMATE
Which users to test with
What behaviour to build
Informs
What tests to specify
What needs to be done
Informs

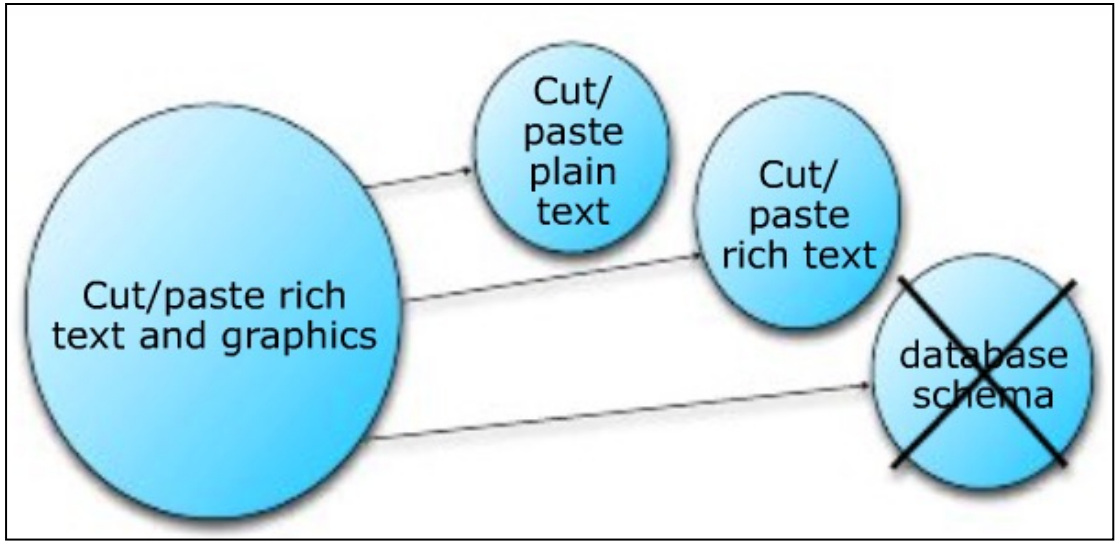
User Story – Examples

As a user, I want to send messages to communicate with a friend.

As a user, I want to receive messages to communicate with friends.

As a customer, I want to search for product items, so I can buy them.

As an employer, I want to post a job on the website so people can apply for it.



Good definitions of done

These are your contracts with your clients

This is how you know you have completed a user story, and can mark it resolved

This is how you know whether you can test your user story (if you can't, you need a different user story!)

Definition of done:
User clicks the button buy, and it appears in their purchased items, and is shipped to their home.

User Story – Examples (without description, notes,…)

User story: As a user, I want to be able to buy something to get it

Definition of done: User clicks the button buy, and it appears in their purchased items, and is shipped to their home.

User story: As a user, when I'm told that I'm not approved for purchase by the system, I want to be able to click "request approval" and then receive confirmation that the approval request has been sent

Definition of done: User is seeing "not approved", and clicks "request approval", and then user's ID appears in the list of approval requests, and an email is sent back to the user that their request is in processing

Properties of good User Stories

Follows INVEST:

- **I**ndependent
- **N**egotiable
- **V**aluable to users or customers
- **E**stimable
- **S**mall
- **T**estable

Example of a good user story

- As a company, I can use the corporate credit card so I can pay to upload job postings to the jobs database
- Notes: Accepts Visa, MasterCard, Amex. Consider: Discover
- Definitions of done:
  - With valid cards of all types (Visa, Amex, Diner's Club, M/C) test that paying is successful and that the user is then able to upload the job postings
  - Same test but with bad, expired, and missing card ID numbers
- Tasks: <listing of all the implementation steps to achieve this>
- Estimate: 28 hours.

Example #2

- As a Creator, I want to upload a video from my local machine so that any user can view it.
- Notes: …
- Definitions of done:

  Test
  - Click the “upload” button
  - Specify a video file to upload
    - Check that .flv, .mov. … extensions are supported
    - Check that files larger than 100MB or longer than 10 mins result in an error
  - Click ”upload video” button
  - Check that progress is displayed in real time
  - Check that users can then view the video
- Tasks: …
- Estimate: 20 hours.

User story examples

RGB: As a shopper, I want to be able to buy something and then see it in my purchased list so that I can spend money on the site.

Definition of done: User clicks the button buy, and it appears in their purchased items, and is shipped to their home and the user will see the money deducted from their account.

Bad User story: As a buyer, When I'm told that I'm not approved for purchase by the system, I want to be able to click "request approval" ***(solution domain🗲)*** and then receive confirmation that the approval request has been sent. *(no benefit! How valuable is this?)*

Definition of done: User is seeing "not approved", and clicks "request approval", and this triggers a react function which makes user's ID appear in the list of approval requests, and an email is sent back to the user that their request is in processing

(DoD is user-level: should not mention code; DoD is client-oriented solution domain)

Not user stories

- Implement contact list view ContactListView.java
- Define product table database schema
- Design team creation interface
- Add extra tests
- Refactor the code to make it more readable.

These are engineering tasks.

Let’s examine:

Architectural context: zoom

User story: As a host I want to be able to ask participants a question in a poll, and see the results showing a count of the responses for each choice so I can easily and efficiently gather opinions from participants.

Definitions of done: User sees a button to create a poll, clicks that, popup shows up with edit/add/remove options for the poll, text box to type the question, and then launch poll to public and close the poll. The launched poll pops up in each participants window, they can select the choices, hit submit. Once the poll is launched, a window pops up to show results in realtime to the host.

This is way too large and could be an “epic”, a collection of multiple user stories!

Possible split:

- As a host I want to be able to have a pop-up with a message appear to all participants so that I can communicate a message to each of them whether or not they can see their chat windows *(press the popup-message button, enter message, click show, message appears to all)*
- As a host I want to be able to have the participants enter text into the popup window and have all that text appear in a results window…
- As a host I want to be able to pose poll options to participants in a popup, and have their responses appear raw in a results window…
- As a host I want to be able to see summarised responses to poll options…

Writing user stories – Class Activity

Learning Objectives

Be able to:

- Write user stories with the correct format and meaning, including definitions of done
- Critique a user story / sets of user stories

Class Activity – User stories

- Creating and critiquing user stories for a set of features for a “WhatsApp” like messaging app
- Split into groups
- Round 1:
  - I’ll give you some features
  - You write the user story(ies)
  - We’ll discuss some as a class
- Round 2:
  - Critique user stories.

Round 1 – Writing user stories (10-15 mins) [https://bit.ly/3wGGdyJ]

- In groups of 3 to 4
- Write user story(ies) for WhatsApp like app that:
  - Allow a customer to create and maintain a profile. Profiles can include a name, a picture and interests.
  - Connect as a contact with other users on the system.

Round 1 – Wrap-Up

What user stories did you come up with?

Round 2 – Critiquing user stories if time allows [5-10mins] [https://bit.ly/48Gwxl5]

- Exchange user stories from Round 1 with another group (take the one that's below your entry in the google form, last -> first)

- Critique user stories using "INVEST"

Round 2 – Wrap-Up

Let’s find examples of each part of INVEST

Round 3 – New user stories [optional]

- Write new user stories (that build on the ones you have been assigned)
  - Start a group chat
  - Receive notifications of interesting events in the app

Validation

Validation: a holistic look at all requirements

Remember from before:

- A perfect implementation is no good if it solves the wrong problem. Validation of a specification is making sure the right problem is solved
- A good specification is:
  - Complete
  - Consistent
  - Precise
  - Unambiguous
  - Concise

Validation

Most common errors:

- Omission: requirement is missing
- Inconsistency: conflicting requirements
- Incorrect facts
- Ambiguity

Inspection is the primary way to validate requirements (SRS, user stories,…); should include various stakeholders.

User stories to tasks

SCRUM backlogs and tasks

Product Backlog: (prioritized) list of all items that need to be done within a project

Sprint Backlog: subset/list of negotiated PBIs for sprint; based on priorities and team's perception of required time

Tasks: User stories are then broken down by developers into engineering tasks. These are NOT from a user perspective -- they are just things that need to be done to get the work completed (finish the parser; investigate the JSON library; set up the database; setup mocks for testing; etc)

Sprint Backlog Items – Examples

Real Weather App:

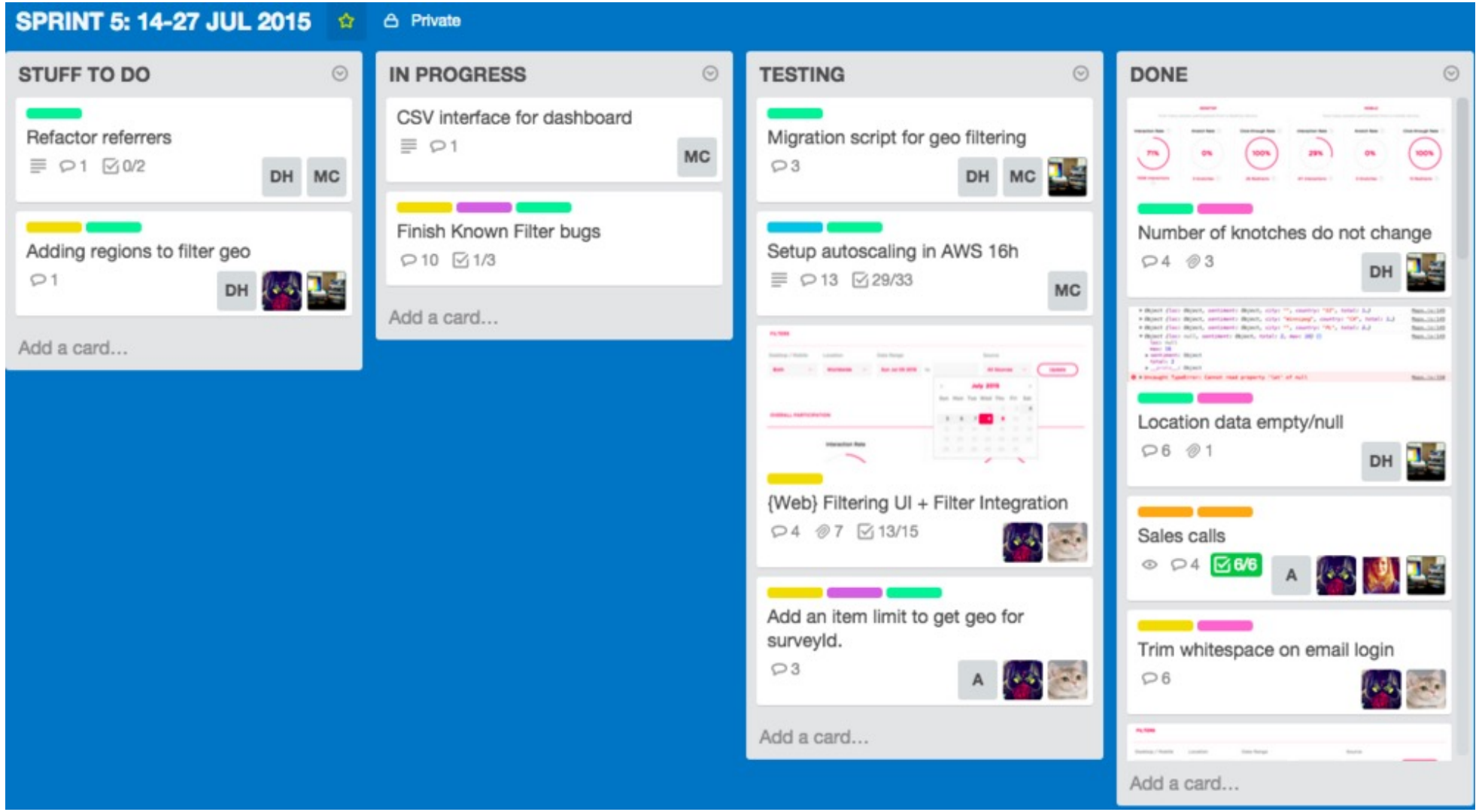
PBI: As a subscriber, I want to see a 10-day forecast of conditions so that I can plan at least a week ahead

- T1: Parse the weather data in day packs
- T2: Push several days data to the client

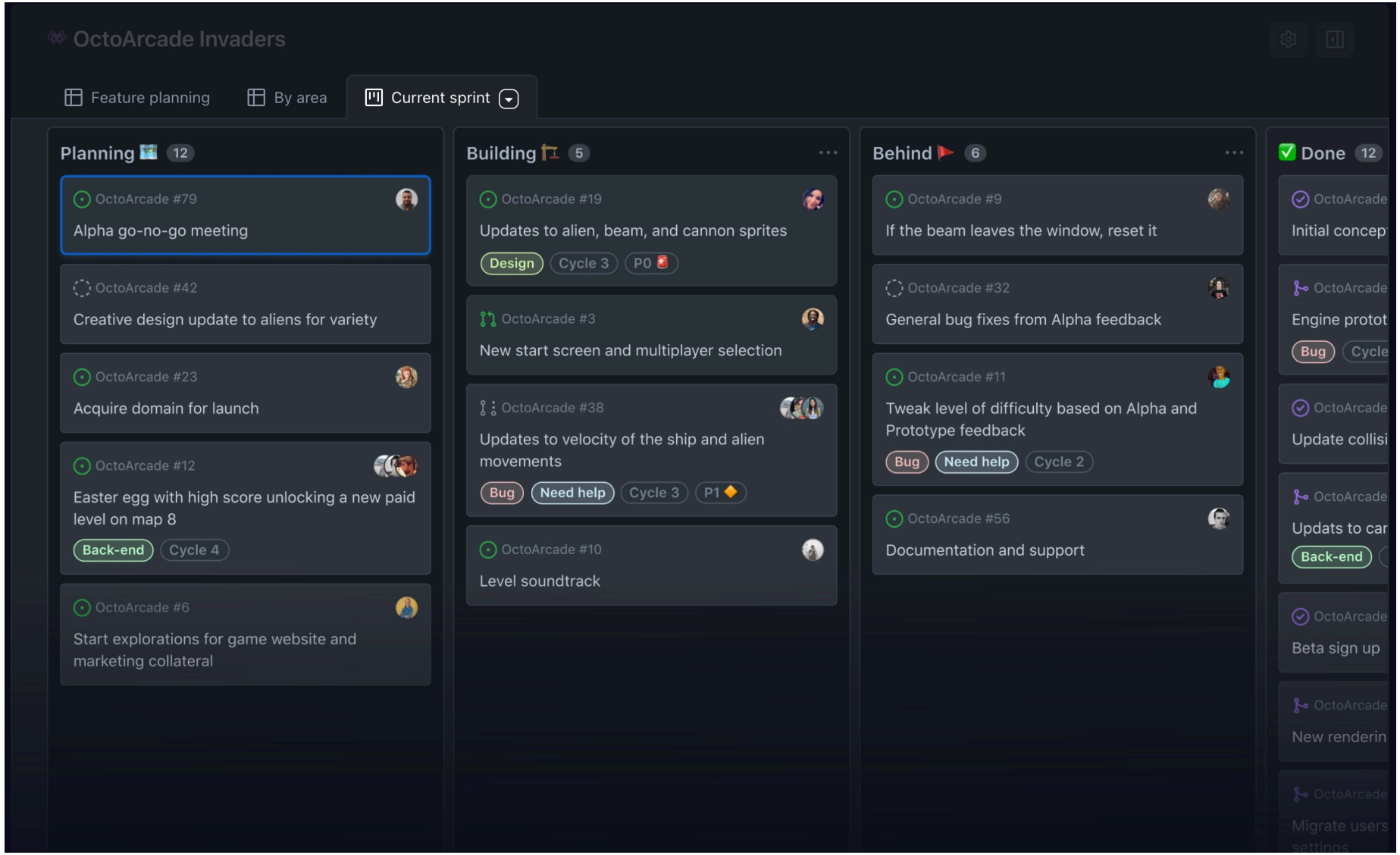
PBI: As a subscriber, I want to see precipitation accumulations and forecast so that I can plan my activities.

- T1: Parse snow/rain data from the provider's data
- T2: Push the snow/rain data to the client
- T3: Redesign client screen a bit
- T4: Refactor the server code

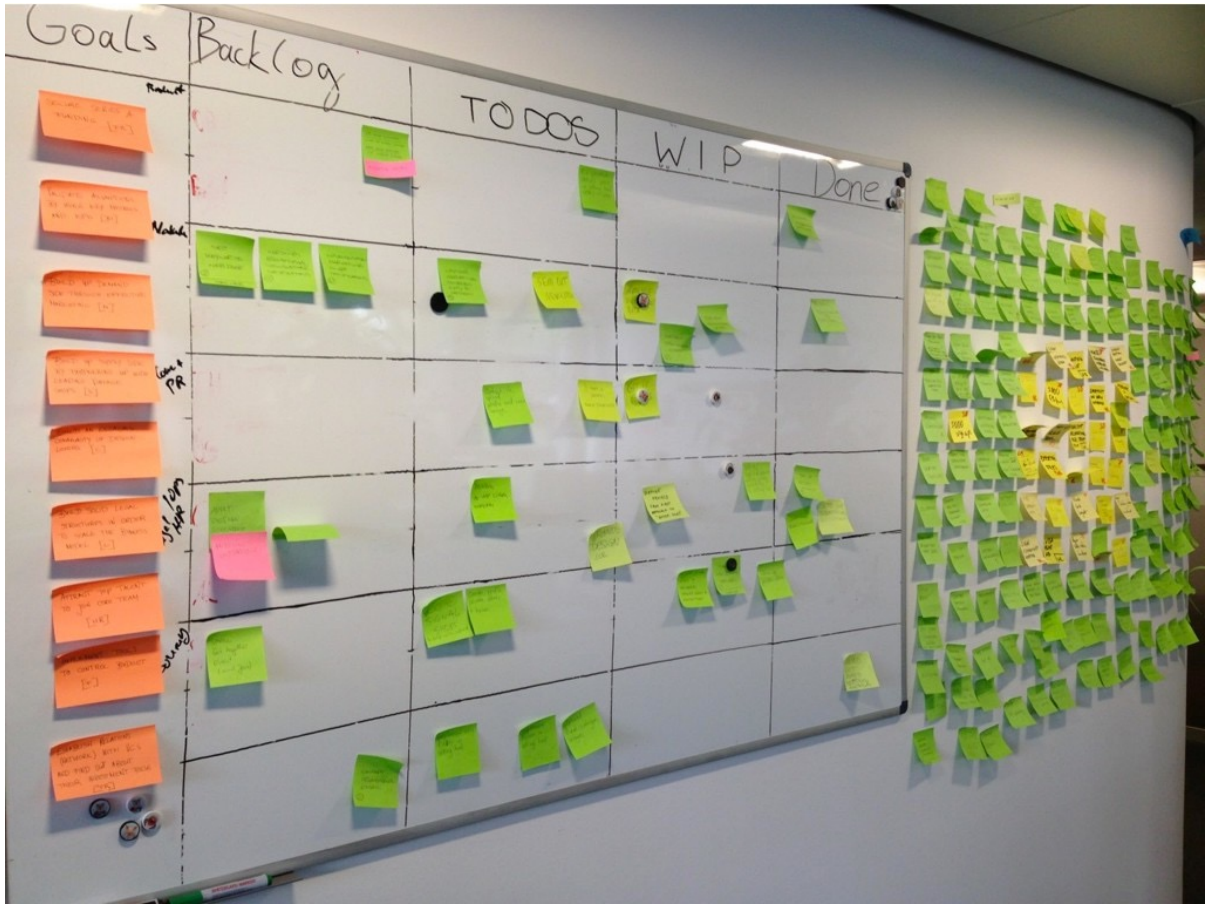
User stories and sprint (dashboard)



Example of one sprint plan - Github




KAN BAN
Queue
E+A
In Prog.
Done!
EXPRESS LANE
DOCKS/ ON WATER
SAIL BOATS
BARN/ SHOP/ ON LAND


Goals
Backlog
TODOS
WIP
Done

Questions & Quiz

Questions (from previous courses)

1. How do engineers get the user stories? (e.g. do you do surveys with users themselves?)
2. How can you be sure you captured all requirements in the elicitation phase?
3. Can there be too many user stories? How many user stories should you have?
4. Are there alternatives to user stories?
5. Who validates the user stories and when?
6. Would it be good to always add more time to an estimate?

Quiz

1. Good user stories should be (select all that apply)
    - Small
    - Estimable
    - Testable
    - Ambiguous

2. Select whether the following statement is true or false:

"The more requirements we have for a system, the better."

    - True / False

Quiz

3. Select all functional requirements out of the following list (note that the requirements do not necessarily refer to the same application but are rather independent):
   - The web application should load and render on the web page in less than 3 seconds for a total number of simultaneous users <= 5000
   - As a visually impaired user, I want the document editing application to provide a mode that enlarges the text, icons and more throughout the application so that I am able to read it easily.
   - As a user of the document editing application I want to be able to change the font size of a paragraph that I have written in my document.
   - As a user, I want to be able to review items in the cart, change their number, or remove them before checkout

Quiz

4. A good user story using Role-Goal-Benefit format should contain:
   - Effort estimate
   - Definition of Done
   - Limitations
   - Detailed technical specifications of the feature

Next Week:
Design –
Modularity & High-Level Design

Things to do THIS WEEK

- Listen to module videos and read readings

- Fill in Quiz (next Wednesday, 10:00 – 10:20am)

Have a Java IDE accessible/available next week!