CS-230 Software Engineering

L17: Prototypes, User Stories, and Scrum

Dr. Liam O'Reilly

Semester 1 – 2020

Previously in CS 230...

Go Team!

Previously in CS 230... (2)

- Is Facebook programmed by a single person?
    - No, it's programmed by several people
    - This is why a group work based software engineering module is important
- What do you guys think makes a good team?

Previously in CS 230... (3)

- Many of the Software Lifecycle Models require prototypes

Prototyping

Prototyping Motivation

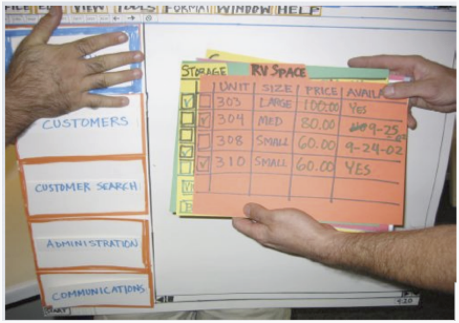
- Sometimes requirements aren't clear.
    - Users don't know what they want.
    - Therefore, cannot formalise requirements or good design choices.
- Early prototypes help figure out what users want.
- Converge on an acceptable solution.

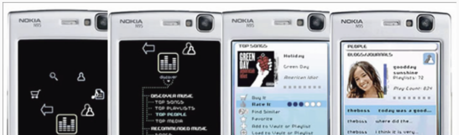
Prototypes

- A prototype is an initial version of a system used to demonstrate concepts and try out design options.
- A prototype can be used in:
  - The requirements engineering process to help with requirements elicitation and validation;
  - In design processes to explore particular solutions and develop UI designs.
    - E.g., Paper-based mockups (which are obviously not executable).
    - Versions of software which implement in a rough way a feature for the purpose of demonstrating the feature.

Prototypes (2)

- Concrete but partial system implementation
- Quick to generate and easy to throw away
  - low fidelity - omits many details
  - high fidelity - near finished product





Low Fidelity

- Throw away prototypes that focus on underlying ideas.
- Produced quickly and thrown away.
- Generates many possible ideas.

High Fidelity

- Built in software for automation and similar to final product.
- Detail accuracy is important and used in realistic scenarios.
- Helps client acceptance.

Paper Prototypes

- Low fidelity prototype involving screens sketched on paper
    - pointing with finger = mouse click
    - writing on paper = typing
- Designer moves sheets around to indicate interaction
- Usually made with stationary, markers, tape, and glue
- What can be learned?
    - Do users understand it?
    - Does it do what users need? Missing features?
    - Can users find their way around?
    - Do users understand the interface?
    - Is the presentation appropriate?

Computer Prototypes

- Interactive software simulation
- High-fidelity in look & feel
- Low-fidelity in depth
    - computer prototype provides shallow functionality
    - computer prototype may be horizontal
    - covers most features, but no back end

Computer Prototypes (2)

- May be based on rapid prototyping languages or tools.
- May involve leaving out functionality
    - Prototype should focus on areas of the product that are not well-understood;
    - Error checking and recovery may not be included in the prototype;
    - Focus on functional rather than non-functional requirements (such as reliability and security).

Computer Prototypes (3)

- Computer prototypes should be discarded after development as they are not a good basis for a production system:
    - It may be impossible to tune the system to meet non-functional requirements;
    - Prototypes are normally undocumented;
    - The prototype structure is usually degraded through rapid change;
    - The prototype probably will not meet normal organisational quality standards.
    - The prototype may not work exactly the way the final version will work, e.g., a prototype may be slow (as easy by inefficient coding was used).

Benefits of Prototyping

- Improved system usability.
- A closer match to users' real needs.
- Improved design quality.
- Improved maintainability.
- Reduced development effort.

User Stories

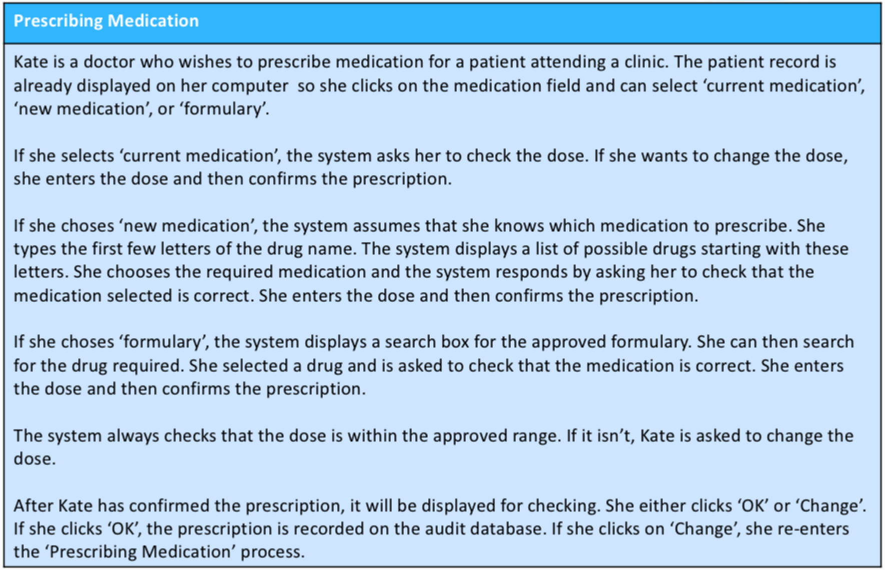
First - Recap - Test Driven Design

- Writing tests before code clarifies the requirements to be implemented.
- Tests are written as programs (e.g., JUnit) rather than data (on paper) so that they can be executed automatically. The test includes a check that it has executed correctly.
    - Usually relies on a testing framework such as JUnit.
- All previous and new tests are run automatically when new functionality is added, thus checking that the new functionality has not introduced errors.
    - i.e., regression testing

Requirements Scenarios

- In XP, a customer or user is part of the XP team and is responsible for making decisions on requirements.
- User requirements are expressed as scenarios or user stories.
- These are written on cards and the development team break them down into implementation tasks. These tasks are the basis of schedule and cost estimates.
- The customer chooses the stories for inclusion in the next release based on their priorities and the schedule estimates.

A ‘Prescribing Medication’ Story



Prescribing Medication

Kate is a doctor who wishes to prescribe medication for a patient attending a clinic. The patient record is already displayed on her computer so she clicks on the medication field and can select ‘current medication’, ‘new medication’, or ‘formulary’.

If she selects ‘current medication’, the system asks her to check the dose. If she wants to change the dose, she enters the dose and then confirms the prescription.

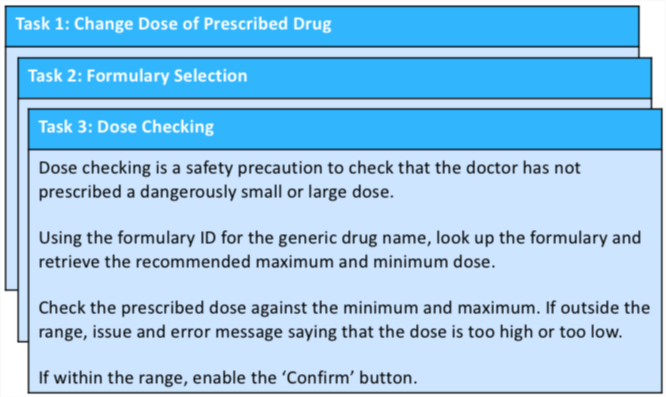
If she choses ‘new medication’, the system assumes that she knows which medication to prescribe. She types the first few letters of the drug name. The system displays a list of possible drugs starting with these letters. She chooses the required medication and the system responds by asking her to check that the medication selected is correct. She enters the dose and then confirms the prescription.

If she choses ‘formulary’, the system displays a search box for the approved formulary. She can then search for the drug required. She selected a drug and is asked to check that the medication is correct. She enters the dose and then confirms the prescription.

The system always checks that the dose is within the approved range. If it isn’t, Kate is asked to change the dose.

After Kate has confirmed the prescription, it will be displayed for checking. She either clicks ‘OK’ or ‘Change’. If she clicks ‘OK’, the prescription is recorded on the audit database. If she clicks on ‘Change’, she re-enters the ‘Prescribing Medication’ process.

Examples of Task Cards for Prescribing Medication


Task 1: Change Dose of Prescribed Drug

Task 2: Formulary Selection

Task 3: Dose Checking

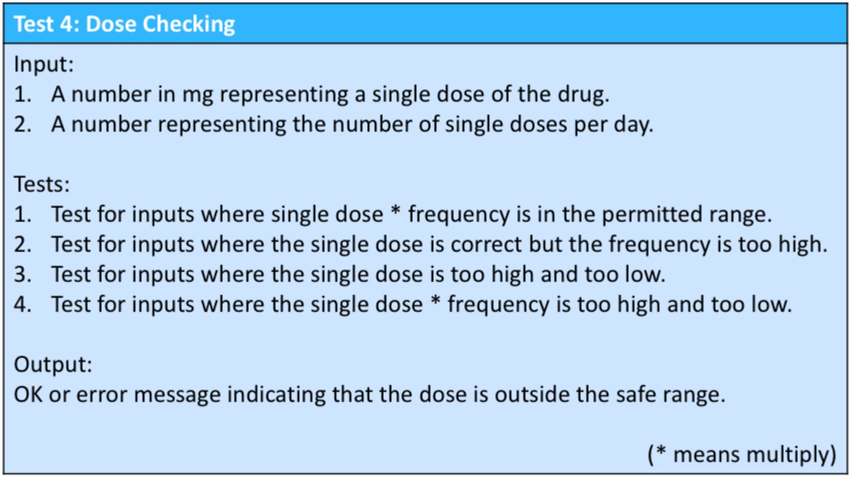
Dose checking is a safety precaution to check that the doctor has not prescribed a dangerously small or large dose.

Using the formulary ID for the generic drug name, look up the formulary and retrieve the recommended maximum and minimum dose.

Check the prescribed dose against the minimum and maximum. If outside the range, issue and error message saying that the dose is too high or too low.

If within the range, enable the 'Confirm' button.

Test Case Description for Dose Checking



Test 4: Dose Checking

Input:

1. A number in mg representing a single dose of the drug.
2. A number representing the number of single doses per day.

Tests:

1. Test for inputs where single dose * frequency is in the permitted range.
2. Test for inputs where the single dose is correct but the frequency is too high.
3. Test for inputs where the single dose is too high and too low.
4. Test for inputs where the single dose * frequency is too high and too low.

Output:

OK or error message indicating that the dose is outside the safe range.

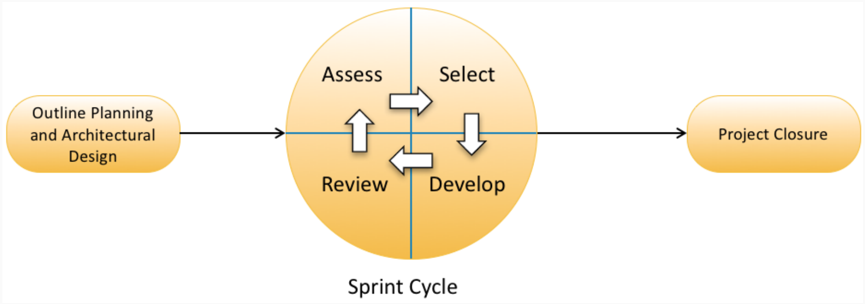
(* means multiply)

Scrum

Scrum

- The Scrum approach is a general agile method but its focus is on managing iterative development rather than specific agile practices.
- Massively popular!
- There are three phases in Scrum.
  - The initial phase is an outline planning phase where you establish the general objectives for the project and design the software architecture.
  - This is followed by a series of sprint cycles, where each cycle develops an increment of the system.
  - The project closure phase wraps up the project, completes required documentation such as system help frames and user manuals and assesses the lessons learned from the project.

The Scrum Process



The Sprint Cycle

- Sprints are fixed length, normally 2 weeks. They correspond to the development of a release of the system in XP.
- The starting point for planning is the product backlog, which is the list of work to be done on the project.
- The selection phase of a sprint involves all of the project team who work with the customer to select the features and functionality (i.e., stories) to be developed during the sprint.

The Sprint Cycle (2)

- Once these are agreed, the team organise themselves to develop the software. During this stage the team is isolated from the customer and the organisation, with all communications channelled through the so-called Scrum master.
- The role of the Scrum master is to protect the development team from external distractions.
- At the end of the sprint, the work done is reviewed in a meeting called the sprint retrospective and is then presented to stakeholders. The next sprint cycle then begins.

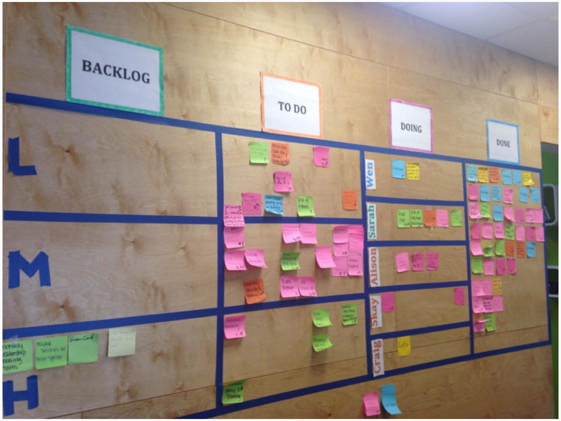
Scrum Master and the Daily Standup

- The Scrum master is a facilitator who arranges daily meetings, tracks the backlog of work to be done, records decisions, measures progress against the backlog and communicates with customers and management outside of the team.
- The whole team attends short daily meetings, known as the Daily Standup, where all team members share information, describe their progress yesterday, problems that have arisen and what is planned for today.
    - This means that everyone on the team knows what is going on and, if problems arise, can re-plan short-term work to cope with them.

Scrum Board

- Scrum uses a large board to keep track of the current sprint (next slide).
- Each task normally has an estimate of how many hours are left on it.
    - The task hours are estimates made by the developers. They should get more accurate over time.
- At the start of the sprint the hours on the board should total:
    - 8 * 10 * number of developers in team. (assuming you work 8 hours a day , Mon – Fri).
- Each member tries to reduce the board by 8 hours work each day.

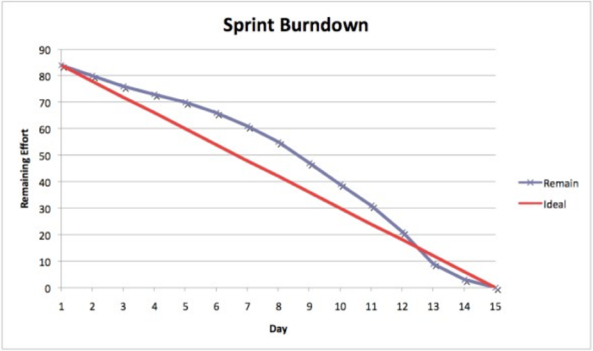
Example Scrum Board



- Here backlog and ToDo are prioritised Low Medium and High.
- Each morning team meet 9am.
- Each member:
    - Discuss what they did previous day.
        - Problems that they need help with.
    - Move post-its along and update them.
    - Select new tasks from ToDo.

Sprint Burndown

- Next to the Scrum board you have this.
- Produced by the scrum master
- Shows where we are currently and how behind or ahead the development is.



Scrum Benefits

- The product is broken down into a set of manageable and understandable chunks.
- Unstable requirements do not hold up progress.
- The whole team have visibility of everything and consequently team communication is improved.
- Customers see on-time delivery of increments and gain feedback on how the product works.
- Trust between customers and developers is established and a positive culture is created in which everyone expects the project to succeed.