CSC40232: SOFTWARE ENGINEERING

Professor: Jane Cleland-Huang
Term Projects + Some background
sarec.nd.edu/courses/SE2017

Department of Computer Science and Engineering

Tonight's Agenda

- The Agile Process
  About 30 minutes.
- Midterm exam

2

Agile Manifesto

"We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:

- Individuals and interactions over processes and tools.
- Working software over comprehensive documentation.
- Customer collaboration over contract negotiation.
- Responding to change over following a plan.

That is, while there is value in the items on the right, we value the items on the left more"

http://www.agilemanifesto.org

3

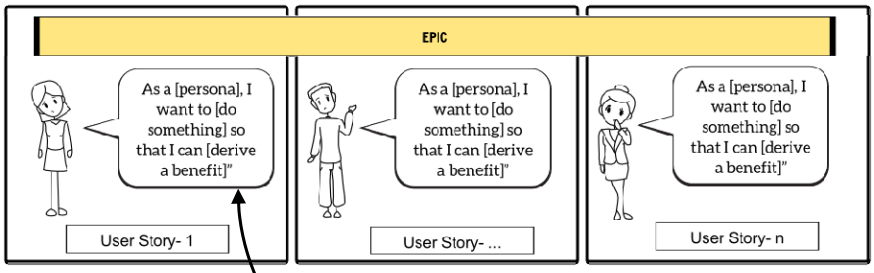
eXtreme Programming

1. Customer team member
2. User stories
3. Short cycles
4. Acceptance tests
5. Pair programming
6. Test-driven development
7. Collective ownership

8. Continuous integration
9. Sustainable pace
10. Open workspace
11. The planning game
12. Simple design
13. Refactoring
14. Metaphor

4

Epics and User Stories



Epics contain user stories.

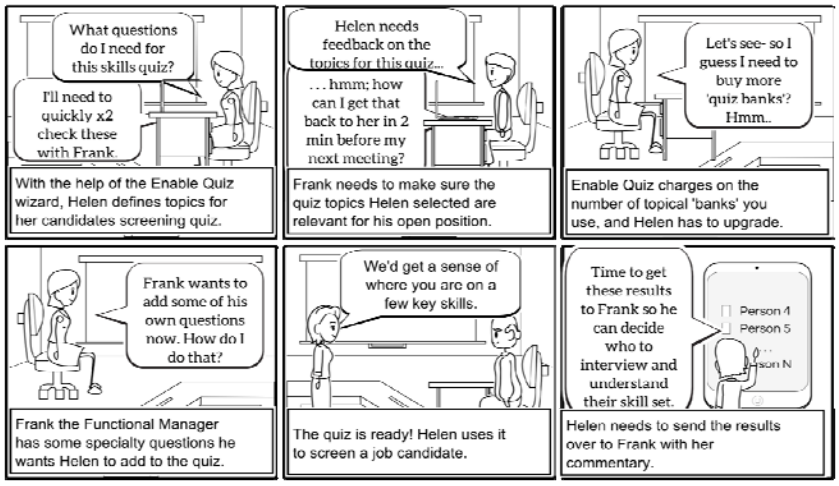
- Ubiquitous
  - Requirement is always active
- Event-driven (keyword *When*)
  - Required response to a triggering event
- State-driven (keyword *While*)
  - Required response in a specified state
- Option (keyword *Where*)
  - Applicable only if *feature* is included
- *(can use combinations of When, While and Where for requirements with complex conditional clauses)*

Example taken from http://www.alexandercowan.com/best-agile-user-story/

5

Epics

As the HR manager, I want to create a screening quiz so that I can understand whether I want to send possible recruits to the functional manager.



6

Breaking an Epic into User Stories & Tests

***User Story # 1**:* As a manager, I want to browse my existing quizzes so I can recall what I have in place and figure out if I can just reuse or update an existing quiz for the position I need now.

- Search by quiz name
- Search by quiz topics included.
- Search by creation and last used date.

User Story # 2: As an HR manager, I want to match an open position's required skills with quiz topics so I can create a quiz relevant for candidate screening.

- Search quiz topics by name.

User Story # 3: As an HR manager, I want to send a draft quiz to the functional manager so I make sure I've covered the right topics on the screening quiz.

- Add another user by email in this flow
- Include notes and customize the email
- Copy a link (for cases where HR manager wants to send their own email)

7

Using EARS in place of User Stories

- Top-level system requirements are typically written in Natural Language(NL) by individuals who are not requirements experts
- Unconstrained NL can cause problems.
- There is a need for simple, easy to apply guidance
- EARS Templates are based on industry best practice and many years of experience

All material on EARS requirements is taken from Alistair Maven, Rolls Royce.

8

Classes of Requirements

There are two *classes* of requirement
- Normal operation
- Unwanted behavior

All NL requirements can be defined using one of 5 simple templates
- 4 *normal operation* templates
- 1 *unwanted behaviour* template

Normal operation requirements define the required system behavior during *sunny day* operation. All users and all interacting systems behave as expected to meet the goals of the user

Rainy day requirements define how the system must respond to unwanted behavior.

9

Unwanted Behavior Requirements

Unwanted behavior requirements include all deviations from sunny day operation. They define the required response of the system to:

- Failures and disturbances
- Deviations from desired user behavior
- Unexpected behavior of interacting systems

10

Normal Behavior

<optional preconditions> <optional trigger> the <system name> shall <system response>

1. **Ubiquitous**
   Requirement is always active
2. **Event-driven** (keyword When)
   Required response to a triggering event
3. **State-driven** (keyword While)
   Required response in a specified state
4. **Option** (keyword Where)
   Applicable only if featureis included
5. **Hybrid**:
   Use combinations of when, while and where for requirements with complex conditional clauses.

- Simple structure adds rigor & clarity
- System response describes what the system must actually do that is *visible* at the boundary of the system

11

Ubiquitous

The <system name> shall <system response>

Defines system behavior that must be active at all times
- Continuous
- No preconditions or triggers
- Unconditional

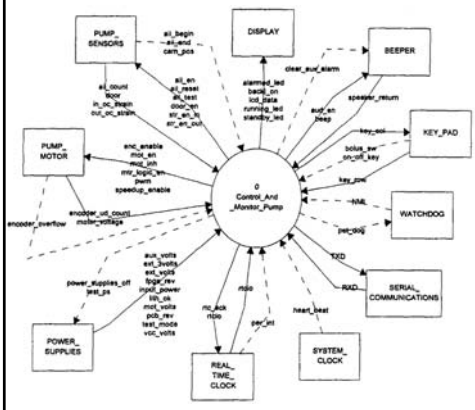
1. The car shall have a maximum retail sale price of XXX
2. The car shall be compliant with the safety requirements defined in XXX

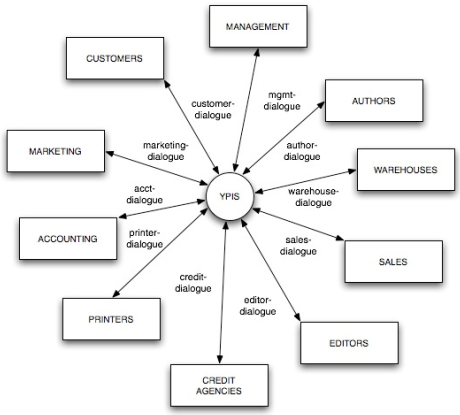
1. The laptop shall have a mass of no more than XXX grams
2. The laptop shall have a minimum battery life of XXX hours

12

Event Driven.

Before you can identify event-driven requirements you need to think about the adjacent systems – by creating an initial system level Data Flow Diagram.





Example # 1: Control and Monitor Pump

Example # 2: Yourdon Press Information System

13

Event Driven.

When <trigger> the <system name> shall <system response>

Initiated only when a triggering event is detected at the system boundary
The trigger must be something that the system itself can detect.
This often helps clarify the system boundary.

1. When the clutch pedal is depressed, the car shall disengage the driving force.
2. When the "turn indicator" command is received, the car shall operate the indicator lights on the front, side and rear of the vehicle, and provide audible and visual confirmation to the driver.

1. When the laptop is off and the power button is pressed, the laptop shall boot up.
2. When the laptop is running and the laptop is closed, the laptop shall enter "powersave" mode.

14

State Driven

While <in a specific state> the <system name> shall <system response>

Requirement is active while the system is in a defined state.
Requirement is “continuous”, but only while the system is in the specified state

1. While the ignition is on, the car shall display the fuel level and the oil level to the driver
2. While the key is in the ignition, the car alarm shall be inhibited
3. While the handbrake is applied, the wheels shall be locked

1. While the laptop is running on the battery and the battery is below XXX % charge, the laptop shall display "low battery“.
2. While an external audio output device is connected, the laptop shall mute the built-in speaker and send the audio output signal to the external audio output device.

15

Option (not so relevant for us)

Where <feature is included> the <system name> shall <system response>

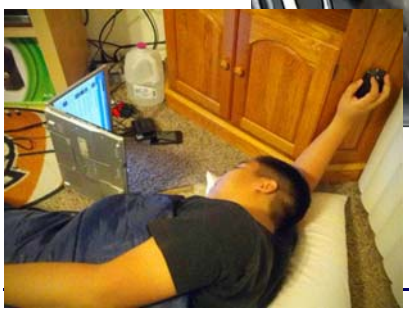
Applicable only in systems that include a particular feature. The requirement will often be “ubiquitous”, but only for systems that include the specified feature.

1. Where the car has electric windows, the electric window controls shall be on the driver's door panel.
2. Where the car includes automatic windscreen wipers, the car shall sense moisture on the windscreen and operate the windscreen wipers without driver commands.

1. Where a "long life" battery is fitted, the laptop shall have a minimum battery life of XXX hours.
2. Where the laptop is a "lightweight" model, the laptop shall have a mass of no more than XXX grams.

16

Unwanted Behavior



A variation of event-driven requirements.

17

Unwanted Behavior

If <optional preconditions> <trigger>, then the <system name> shall <system response>

Forces the separation of
- Circumstances in which the requirement can be invoked (**preconditions**)
- The initiating event (**trigger**)
- The expected system behavior (**response**)

1. If the car detects attempted intrusion, then the car shall operate the car alarm
2. If the car detects low oil pressure, then the car shall display a "low oil pressure" warning

1. If the incorrect password is entered, then the laptop shall display XXX warning message.
2. If the laptop is connected to a non-compatible device, then the laptop shall prevent transfer of data, prevent transfer of charge, display XXX warning message and not be damaged.

18

Complex Behavior

Combines if, when, while, where, and if-then

1. Where the car includes an "owner alert" system, if the car detects attempted intrusion, then the car shall send a message to the owner and activate the car alarm
2. While the car is being driven forwards above a speed of XXX, if the driver attempts to engage reverse gear, then the car shall prevent engagement of reverse gear

1. Where the laptop includes "voice input" option, while the voice input option is selected, the laptop shall accept voice input commands
2. While the laptop is running on mains electrical power, if the power cable is disconnected, then the laptop shall display a warning message

For very complex requirements consider truth tables or other non-textual notation.

19

eXtreme Programming

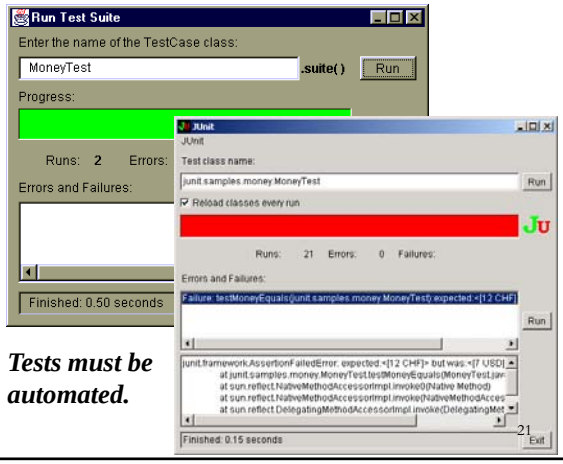
1. Customer team member
2. User stories
3. Short cycles
4. Acceptance tests
5. Pair programming
6. Test-driven development
7. Collective ownership

8. Continuous integration
9. Sustainable pace
10. Open workspace
11. The planning game
12. Simple design
13. Refactoring
14. Metaphor

20

Test Driven

- Unit tests are written before the code.
- Enough production code is written to allow the test to compile, but not pass.
- Test is executed to ensure that it fails. (ie is it a good testcase.)
- Task is completed.
- Test is rerun until it passes.



Tests must be automated.

21

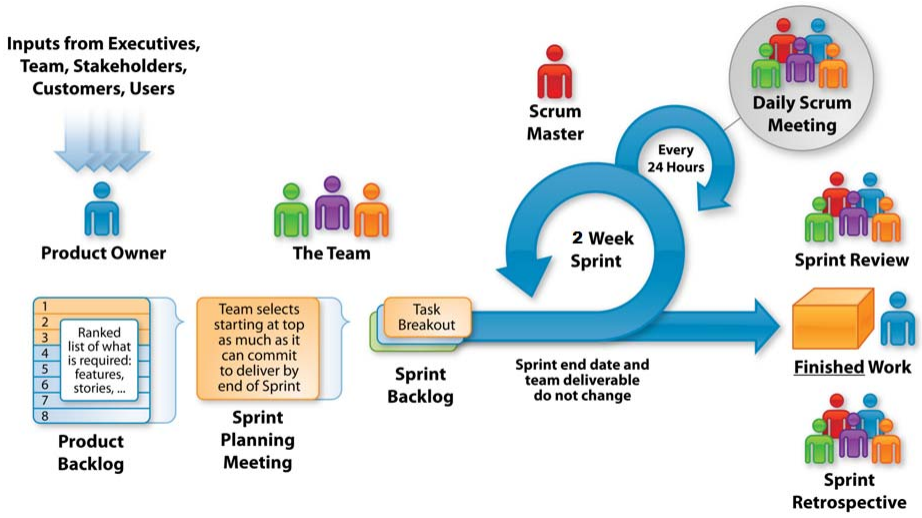
Pairs Programming

- **What is pair programming?** **"**Two programmers working collaboratively on one algorithm, design, or programming task, sitting side by side at one computer."
- **Big question:** What is the ROI of pairs programming?

Cockburn, Alistair and Williams, Laurie
.*The Costs and Benefits of Pair Programming*
in *Extreme Programming Examined*.
Addison Wesley, 2001.

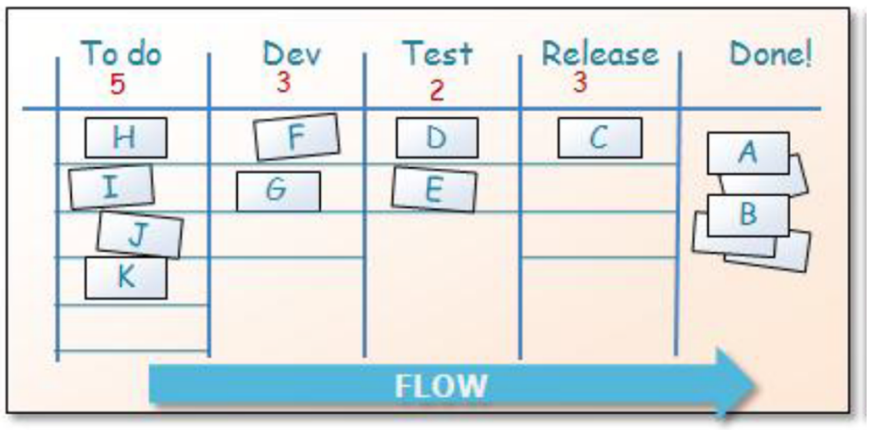
22

SCRUM



23

Kanban



24

Break User Stories into Tasks

As a vacation planner, I want to see photos of the hotels.

- Code the middle tier (8 hours)
- Code the user interface (4)
- Write Test fixtures (4)
- Code foo class (6)
- Update perform-ance tests (4)

When a call for help is received a drone shall be assigned to handle the task.

- Design the dispatch algorithm (6 hours)
- Code the dispatch algorithm (6)
- Write test fixtures (4)
- Integrate algorithm with drone management system(8)

Self-select tasks

25

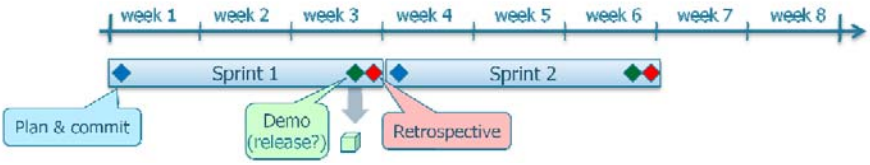
Prioritizing User Stories

Product Backlog

How do you prioritize for your project?

Sprint Planning Meeting

What goes into this sprint?



26