MTAT.03.244
Software Economics

Product Management (2)

Dietmar Pfahl
email: dietmar.pfahl@ut.ee

Fall 2016
Announcement

• Industry Guest Lecture by Anton Keks, Codeborne, Tallinn

• Title: “Software Craftsmanship, the Codeborne Way”

• When: Friday, 2nd Dec., 10:15-11:45
• Where: J. Liivi 2, room 111
Topics Today

• Q&A on Assignment 3
• Requirements Definition: User Stories & Use Cases
• Product Cost/Effort Estimation
  • Use Case Points
  • Planning Poker -> Story Points
Q&A on Assignment 3
Assignment 3

Group forming …

Q&A during last lecture ...

<table>
<thead>
<tr>
<th>Topic</th>
<th>Votes</th>
<th>Yes (%)</th>
<th>No (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>I have attended an RE course?</td>
<td>60</td>
<td>12%</td>
<td>88%</td>
</tr>
<tr>
<td>I know what RE elicitation techniques are?</td>
<td>56</td>
<td>5%</td>
<td>95%</td>
</tr>
<tr>
<td>I know what User Stories are?</td>
<td>57</td>
<td>72%</td>
<td>28%</td>
</tr>
<tr>
<td>I know Use Case Descriptions and Models?</td>
<td>56</td>
<td>71%</td>
<td>29%</td>
</tr>
<tr>
<td>I have done requirements prioritization?</td>
<td>59</td>
<td>37%</td>
<td>63%</td>
</tr>
<tr>
<td>I have done roadmapping/release planning?</td>
<td>61</td>
<td>26%</td>
<td>74%</td>
</tr>
</tbody>
</table>
Requirements Definition: User Stories
User Story – Definition

As a tenant, I can unlock the doors to enter my apartment.

- user-role
  (benefactor)

- capability
  (functionality)

- business-value
  (motivation/rationale)

who - what - why

- Similar to NL requirements, but focus on the user benefits, instead on system characteristics (alone)
  - Unfortunately, third element (business value) is often omitted
- Should have an acceptance test
- Is a preferred technique in agile methods
User Story – Example

<Actor/Role> As a user

... I want to narrow down people search results by location

... so I can find the right person more quickly

A good User Story is:
- Independent
- Negotiable
- Valuable
- Estimable
- Small
- Testable

INVEST
User Story – Example

<Actor/Role> As a user

... 

<Action> I want to narrow down people search results by location

... 

<Value> so I can find the right person more quickly

Acceptance test:

Given I am on the search screen
And 'Paula’ is on the same indexed page with 'Tartu'

When I search for 'Paula'
Then I see 'Tartu’ in the location section of the search results
Exercise 1

(15 + 10 min)
Exercise 1 – User Stories: Discussion

• Was it difficult to come up with User Stories?
  • If so, what was difficult?
• Was it difficult to fulfill the INVEST criteria?
  • If so, what was difficult?
• Was it difficult to review/grade the User Stories of others?
  • If so, what was difficult?
• What did you observe about User Stories when reviewing?
  • What was good/bad about the User Stories you reviewed?
Requirements Definition:
Use Cases
Use Case – Definition

• For **Functional Requirements** Analysis & Specification

• A **use case** is a description of how an actor will use the system to be developed to accomplish business goals

  • Detailed use cases are usually written as **usage scenarios** or **scripts**, listing a specific sequence of actions and interactions between the actors and the system
Use Case Diagrams and Descriptions

Use Case Description:
Name of Use Case
Actors associated with Use Case
Pre-conditions
Post-conditions
Normal Flow of Events (Basic Scenario)
Alternative Flow of Events (Alternative Scenarios)

Diagram:
- Actors
- Use Cases
- Use-Case Descriptions
Types of Actors

- **Initiating actor** (also called primary actor or “user”): initiates the use case to realize a goal

- **Participating actor** (also called secondary actor): participates in the use case but does not initiate it:
  - **Supporting actor**: helps the system-to-be to complete the use case
  - **Offstage actor**: passively participates in the use case, i.e., neither initiates nor helps complete the use case, but may be notified about some aspect of it (e.g., for keeping records)
Identifying Actors

• Ask the following questions:
  • Who will be a primary user of the system? (primary actor)
  • Who will need support from the system to do her daily tasks?
  • Who will maintain, administrate, keep the system working? (secondary actor)
  • Which hardware devices does the system need?
  • With which other systems does the system need to interact with?
  • Who or what has an interest in the results that the system produces?

• Look for:
  • the users who directly use the system
  • also others who need services from the system
Finding Use Cases

For each actor, ask the following questions:

- Which functions does the actor require from the system?
- What does the actor need to do?
- Does the actor need to read, create, destroy, modify, or store some kinds of information in the system?
- Does the actor have to be notified about events in the system?
- Does the actor need to notify the system about something?
- What do those events require in terms of system functionality?
- Could the actor’s daily work be simplified or made more efficient through new functions provided by the system?
Schema for describing Use Cases

<table>
<thead>
<tr>
<th>Use Case UC-#</th>
<th>Name / Identifier</th>
<th>[verb phrase]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Related Requirements</td>
<td>List of the requirements that are addressed by this use case</td>
<td></td>
</tr>
<tr>
<td>Initiating Actor</td>
<td>Actor who initiates interaction with the system to accomplish a goal</td>
<td></td>
</tr>
<tr>
<td>Actor's Goal</td>
<td>Informal description of the initiating actor's goal</td>
<td></td>
</tr>
<tr>
<td>Participating Actors</td>
<td>Actors that will help achieve the goal or need to know about the outcome</td>
<td></td>
</tr>
<tr>
<td>Preconditions</td>
<td>What is assumed about the state of the system before the interaction starts</td>
<td></td>
</tr>
<tr>
<td>Postconditions</td>
<td>What are the results after the goal is achieved or abandoned; i.e., what must be true about the system at the time the execution of this use case is completed</td>
<td></td>
</tr>
</tbody>
</table>

Flow of Events for Main Success Scenario:

1. The initiating actor delivers an action or stimulus to the system (the arrow indicates the direction of interaction, to- or from the system)
2. The system's reaction or response to the stimulus; the system can also send a message to a participating actor, if any
3. ...

Flow of Events for Extensions (Alternate Scenarios):

What could go wrong? List the exceptions to the routine and describe how they are handled

1a. For example, actor enters invalid data
2a. For example, power outage, network failure, or requested data unavailable

The arrows on the left indicate the direction of interaction: → Actor's action; ← System's reaction
Example – Home Access Control

- Lock
- Photosensor
- Switch
- Alarm bell
- Light bulb

Backyard doors: External & Internal lock

Front doors: External & Internal lock

Central Computer
Use Case Diagram: Device Control Subsystem

First tier use cases

- UC1: Unlock
- UC2: Lock
- UC7: AuthenticateUser

Second tier use cases

- UC3: AddUser
- UC4: RemoveUser
- UC5: InspectAccessHistory
- UC6: SetDevicePrefs
- UC8: Login

System boundary

- Tenant
- Landlord
- LockDevice
- LightSwitch
- Timer

Actor

Communication

Use case

«initiate»

«participate»

«include»
## Use Case 1: Unlock

<table>
<thead>
<tr>
<th>Use Case UC-1:</th>
<th>Unlock</th>
</tr>
</thead>
<tbody>
<tr>
<td>Related Requirem’ts:</td>
<td>REQ1, REQ3, REQ4, and REQ5</td>
</tr>
<tr>
<td>Initiating Actor:</td>
<td>Any of: Tenant, Landlord</td>
</tr>
<tr>
<td>Actor’s Goal:</td>
<td>To disarm the lock and enter, and get space lighted up automatically.</td>
</tr>
<tr>
<td>Participating Actors:</td>
<td>LockDevice, LightSwitch, Timer</td>
</tr>
<tr>
<td>Preconditions:</td>
<td>• The set of valid keys stored in the system database is non-empty.</td>
</tr>
<tr>
<td></td>
<td>• The system displays the menu of available functions; at the door keypad the menu choices are “Lock” and “Unlock.”</td>
</tr>
<tr>
<td>Postconditions:</td>
<td>The auto-lock timer has started countdown from autoLockInterval.</td>
</tr>
</tbody>
</table>

### Flow of Events for Main Success Scenario:

1. **Tenant/Landlord** arrives at the door and selects the menu item “Unlock”  
2. include::AuthenticateUser (UC-7)
3. **System** (a) signals to the **Tenant/Landlord** the lock status, e.g., “disarmed,” (b) signals to **LockDevice** to disarm the lock, and (c) signals to **LightSwitch** to turn the light on
4. **System** signals to the **Timer** to start the auto-lock timer countdown
5. **Tenant/Landlord** opens the door, enters the home [and shuts the door and locks]
What about this Use Case Model?

Landlord

- Login
- AddUser
- SetDevicePrefs

?
“Sub-routine” Use Cases

BAD:

- Landlord
- Login
- AddUser
- SetDevicePrefs

GOOD:

- Landlord
- AddUser
- Login
- SetDevicePrefs

<include>
Optional Use Cases: «extend»

Example optional use cases:

Key differences between «include» and «extend» relationships

<table>
<thead>
<tr>
<th></th>
<th>Included use case</th>
<th>Extending use case</th>
</tr>
</thead>
<tbody>
<tr>
<td>Is this use case optional?</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Is the base use case complete without this use case?</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Is the execution of this use case conditional?</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Does this use case change the behavior of the base use case?</td>
<td>No</td>
<td>Yes</td>
</tr>
</tbody>
</table>

[ Source: Robert Maksimchuk & Eric Naibur: *UML for Mere Mortals*, Addison-Wesley, 2005. ]
Exercise 2

(10 min)
Example:
Motion detection and garage door control

Source: http://www.ece.rutgers.edu/%7Emarsic/books/SE/
Motion detection and garage door control: (Detailed) Requirements

• The system should turn the garage door lights on automatically when it detects motion within a given perimeter.

• The garage door opener should be possible to control either by a remote radio transmitter or by a manual button switch.

• The opener should include the following safety feature: An “electric eye” sensor, which projects invisible infrared light beams, should be used to detect if someone or something passes under the garage door while it closes. If the beam is obstructed while the door is going down, the door should not close—the system should automatically stop and reverse the door movement.
Motion detection and garage door control: Actors

- The Owner is the key actor, but the system is also activated by the motion detector and by the “electric eye” sensor. Additionally, we need a timer to schedule the automatic switching off the light.

Hence, we have four actors:

- Owner,
- MotionDetector
- Timer
- ElectricEye
Motion detection and garage door control: Use Case Model

We have four actors:

- Owner,
- MotionDetector
- Timer
- ElectricEye

We have seven use cases
Motion detection and garage door control: Use ‘AutomaticReverse’

<table>
<thead>
<tr>
<th>Use Case:</th>
<th>AutomaticReverse</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initiating Actor:</td>
<td>ElectricEye (“electric eye” sensor)</td>
</tr>
<tr>
<td>Actor’s Goal:</td>
<td>To stop and reverse the door movement if someone or something passes under the garage door while it closes.</td>
</tr>
<tr>
<td>Preconditions:</td>
<td>The garage door currently is going down and the infrared light beams have been sensed as obstructed.</td>
</tr>
<tr>
<td>Postconditions:</td>
<td>The door’s downward motion is stopped and reversed.</td>
</tr>
</tbody>
</table>

Flow of Events for Main Success Scenario:

1. ElectricEye signals to the system that the infrared light beams have been sensed as obstructed
2. System (a) stops and reverses the motor movement, and (b) disarms the ElectricEye sensor
3. System detects that the door is in the uppermost position and stops the motor
Motion detection and garage door control: Use Case ‘RemoteOpen’

For the use case RemoteOpen, the main success scenario may look something like this:

→ 1. Owner arrives within the transmission range and clicks the open button on the remote controller
→ 2. The identification code may be contained in the first message, or in a follow-up one
   ↓ 3. System (a) verifies that this is a valid code, (b) opens the lock, (c) starts the motor, and (d) signals to the user the code validity
   ↓ 4. System increments the motor in a loop until the door is completely open
← 5. User enters the garage

Alternate scenarios include:

• The receiver cannot decode the message because the remote transmitter is not properly pointed for optimal transmission

• The remote controller sends an invalid code. In this and the previous case, the system should sound the alarm after the maximum allowed number of attempts is exhausted
Product Cost/Effort Estimation:

Use Case Points
Use Case Points

For the sum of all Use Cases:

- UCP equation is composed of four variables:
  - Unadjusted Use Case Point (UUCP)
  - The Technical Complexity Factor (TCF)
  - The Environment Complexity Factor (ECF)

  \[ UCP = UUCP \times TCF \times ECF \]

- UUCP is the sum of Unadjusted Actor Weight (UAW) and Unadjusted Use Case Weight (UUCW).

  \[ UUCP = UAW + UUCW \]
# Unadjusted Actor Weight

<table>
<thead>
<tr>
<th>Actor Type</th>
<th>Description</th>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simple</td>
<td>Communicates to system through API</td>
<td>1</td>
</tr>
<tr>
<td>Average</td>
<td>Interacts with the system through some protocol (HTTP, FTP, or probably some user defined protocol), or Are data stores (Files, DBMS)</td>
<td>2</td>
</tr>
<tr>
<td>Complex</td>
<td>Interacts through HCI (GUI)</td>
<td>3</td>
</tr>
</tbody>
</table>

\[ UAW = (\text{Total No. of Simple actors} \times 1) + (\text{Total No. Average actors} \times 2) + (\text{Total No. Complex actors} \times 3) \]
### Unadjusted Use Case Weight

<table>
<thead>
<tr>
<th>Use Case Type</th>
<th>Description</th>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simple</td>
<td>1 to 3 transactions</td>
<td>5</td>
</tr>
<tr>
<td>Average</td>
<td>4 to 7 transactions</td>
<td>10</td>
</tr>
<tr>
<td>Complex</td>
<td>8 or more transactions</td>
<td>15</td>
</tr>
</tbody>
</table>

\[
\text{UUCW} = \left(\text{Total No. of Simple Use Cases x 5}\right) + \left(\text{Total No. Average Use Case x 10}\right) + \left(\text{Total No. Complex Use Cases x 15}\right)
\]
## Technical Complexity Factor – TCF

<table>
<thead>
<tr>
<th>Technical Factor</th>
<th>Description</th>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>TF(1)</td>
<td>Distributed System</td>
<td>2</td>
</tr>
<tr>
<td>TF(2)</td>
<td>Performance</td>
<td>1</td>
</tr>
<tr>
<td>TF(3)</td>
<td>End User Efficiency</td>
<td>1</td>
</tr>
<tr>
<td>TF(4)</td>
<td>Complex Internal Processing</td>
<td>1</td>
</tr>
<tr>
<td>TF(5)</td>
<td>Reusability</td>
<td>1</td>
</tr>
<tr>
<td>TF(6)</td>
<td>Installability</td>
<td>0.5</td>
</tr>
<tr>
<td>TF(7)</td>
<td>Usability</td>
<td>0.5</td>
</tr>
<tr>
<td>TF(8)</td>
<td>Portability</td>
<td>2</td>
</tr>
<tr>
<td>TF(9)</td>
<td>Modifiability</td>
<td>1</td>
</tr>
<tr>
<td>TF(10)</td>
<td>Concurrency</td>
<td>1</td>
</tr>
<tr>
<td>TF(11)</td>
<td>Includes special security requirements</td>
<td>1</td>
</tr>
<tr>
<td>TF(12)</td>
<td>Provides direct access by third parties</td>
<td>1</td>
</tr>
<tr>
<td>TF(13)</td>
<td>Special User training facilities are required</td>
<td>1</td>
</tr>
</tbody>
</table>

Each TF(i) can have a value from 0 (factor is irrelevant) to 5 (factor is essential)

TCF = 0.6 + TF/100 with TF = \( \sum_{i=1}^{13} (TF(i) \times Weight(i)) \)
# Environmental Complexity Factor - ECF

<table>
<thead>
<tr>
<th>Factor Number</th>
<th>Description</th>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>EF(1)</td>
<td>Familiarity with system development process in use</td>
<td>1.5</td>
</tr>
<tr>
<td>EF(2)</td>
<td>Application experience</td>
<td>0.5</td>
</tr>
<tr>
<td>EF(3)</td>
<td>Object-oriented experience</td>
<td>1.0</td>
</tr>
<tr>
<td>EF(4)</td>
<td>Lead analyst capability</td>
<td>0.5</td>
</tr>
<tr>
<td>EF(5)</td>
<td>Motivation</td>
<td>1.0</td>
</tr>
<tr>
<td>EF(6)</td>
<td>Requirements stability</td>
<td>2.0</td>
</tr>
<tr>
<td>EF(7)</td>
<td>Part time staff</td>
<td>-1.0</td>
</tr>
<tr>
<td>EF(8)</td>
<td>Difficulty of programming language</td>
<td>-1.0</td>
</tr>
</tbody>
</table>

Each EF(i) can have a value from 0 (no experience) to 5 (expert)

ECF = 1.4 + (-0.03*EF) with EF = \(\sum_{i=1}^{8}(EF(i) * \text{Weight}(i))\)
Effort based on Use Case Points

For the sum of all Use Cases:
- Effort equation is composed of four variables:
  - Unadjusted Use Case Point (UUCP)
  - The Technical Complexity Factor (TCF)
  - The Environment Complexity Factor (ECF)
  - The Productivity Factor (PF)

\[
\text{Effort} = \text{UUCP} \times \text{TCF} \times \text{ECF} \times \text{PF} \quad \text{[person-hours/UCP]}
\]

- UUCP is the sum of Unadjusted Actor Weight (UAW) and Unadjusted Use Case Weight (UUCW).

\[
\text{UUCP} = \text{UAW} + \text{UUCW}
\]

Either 20 person-hours/UCP or 28 person-hours/UCP
Productivity Factor - PF

- If the sum of (number of factors E1 through E6 assigned value < 3) and (number of factors E7 and E8 assigned value > 3) ≤ 2
  \[ PF = 20 \]

- Else If the sum of (number of factors E1 through E6 assigned value < 3) and (number of factors E7 and E8 assigned value > 3) = 3 or 4
  \[ PF = 28 \]

- Else
  Rethink project; it has too high a risk of failure

- Example:
  http://en.wikipedia.org/wiki/Use_Case_Points
Exercise 3
(10 min)
Product Cost/Effort Estimation:

Planning Poker -> User Story Points
Planning Poker

Participants in planning poker include all of the developers on the team

• **Step 1:** Give each estimator a deck of cards

• **Step 2:** Moderator reads description of User Story to be estimated.

• **Step 3:** Product owner answers any question the estimators may have about the User Story.

• **Step 4:** Each estimator privately selects a card representing his or her estimate. Cards are not shown until each estimator has made a selection.
Planning Poker (cont’d)

- **Step 5:** When everyone has made an estimate, the cards are simultaneously turned over.

- **Step 6:** If estimates differ, the highest and lowest estimates are explained by the estimators - otherwise the estimation is completed for this User Story.

- **Step 7:** The group can discuss the story and their estimates for a few more minutes. The moderator can take any notes he/she thinks will be helpful when this story is being programmed and tested. After the discussion, each estimator re-estimates by selecting a card.

-> Go to Step 5.

Note: In many cases, the estimates will already converge by the second round. But if they have not, continue to repeat the process. The goal is for the estimators to converge on a single estimate that can be used for the story. It rarely takes more than three rounds, but continue the process as long as estimates are moving closer together.
Project Estimation using User Story Points

Points assigned to individual user stories

Total work size estimate:

\[ \text{Total size} = \sum \text{points-for-story } i \ (i = 1..N) \]

Velocity (= Productivity) estimated from experience

Estimate the work duration:

\[ \text{Project duration} = \frac{\text{Total size}}{\text{Velocity}} \ [\text{time unit}] \]
Agile Effort Estimation & Sprint Allocation

Work backlog

1) ST-4: Unlock 15 days (9pts)
2) ST-2: Lock 5 days (3pts)
3) ST-5: Manage Users 16 days (10pts)
4) ST-7: Preferences 10 days (6pts)
5) ST-6: View History 10 days (6pts)
6) ST-...

List prioritized by the customer

Items pulled by the team into an iteration

Estimated work duration

1st iteration
2nd iteration
n-th iteration

Estimated completion date

21 days

5 days

MTAT.03.244 / © Dietmar Pfahl 2016
Exercise 4

(10 min)