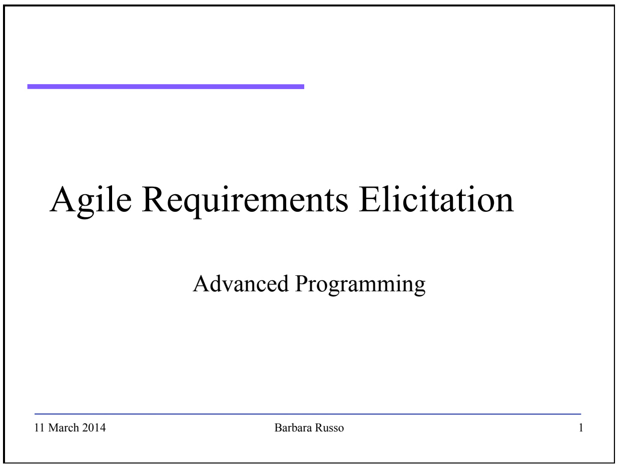
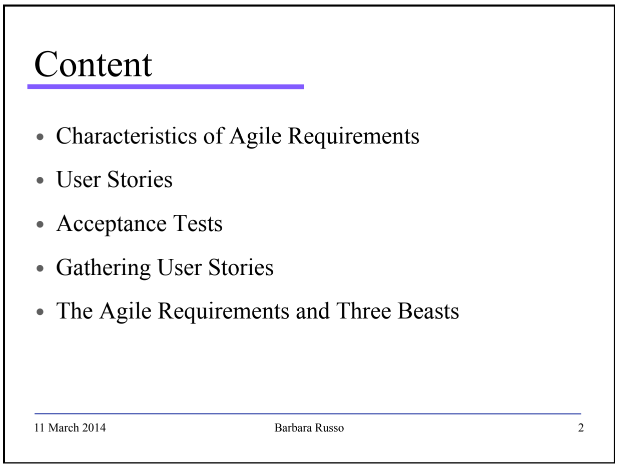


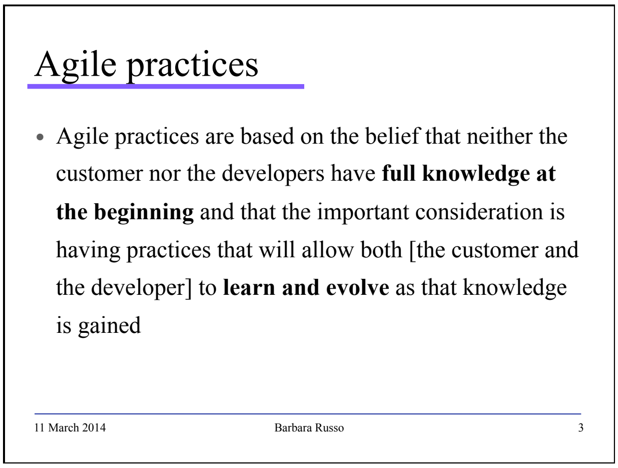
Agile Requirements Elicitation

Advanced Programming



Content

- Characteristics of Agile Requirements
- User Stories
- Acceptance Tests
- Gathering User Stories
- The Agile Requirements and Three Beasts



Agile practices

- Agile practices are based on the belief that neither the customer nor the developers have **full knowledge at the beginning** and that the important consideration is having practices that will allow both [the customer and the developer] to **learn and evolve** as that knowledge is gained

Agile Requirements (Rs)

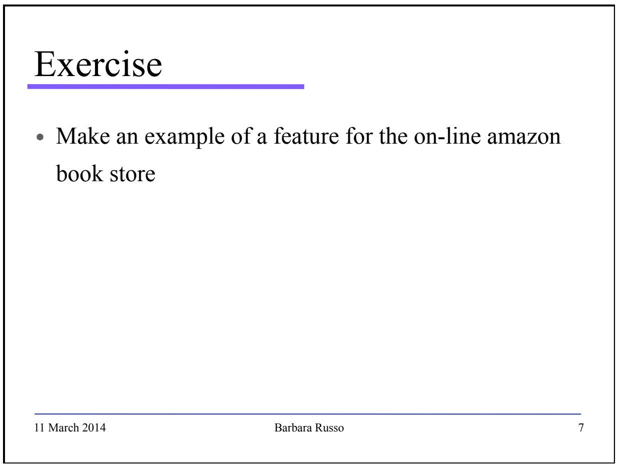
- Suitable for projects with **a lot of change**, not for stable projects with safety-critical implications
- Agile requirements are expressed as high-level, brief written statements of the best information fairly easily available
- Not fully documented form of the requirements

Characteristics - agile Rs 1/2

- Frequent, personal interactions with customers and/or stakeholders
  - for example, in XP: Customer On-Site
- Frequent delivery of software to customers
  - If customer can actually provide feedback

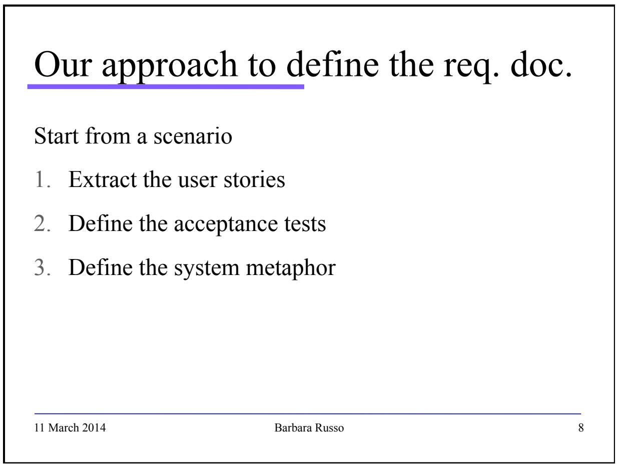
Characteristics - agile Rs 2/2

- Expressing Requirements as set of **Features**
  - A feature is a short phrase that describes what the customer wants
    - E.g. Items on auction can be added, deleted, and modified
- Modelling features as **User Stories**
  - Used in eXtreme Programming (XP)



Exercise

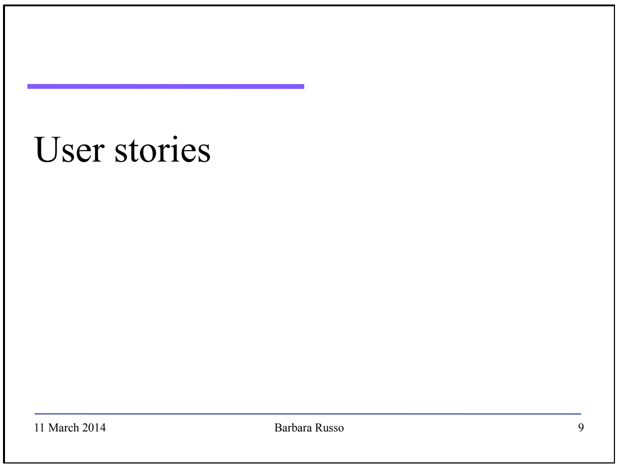
- Make an example of a feature for the on-line amazon book store



Our approach to define the req. doc.

Start from a scenario

1. Extract the user stories
2. Define the acceptance tests
3. Define the system metaphor



User stories

A user story (how many features?)

Title	viewTransactionHistory
Priority	2
Story Points	1
Risk Level	1
Description	The user must be registered and logged in the service. The user selects an interval of time or a type of transaction to display a list of transactions.
Related User Stories	SelectTransaction
Acceptance Test	ViewTransactionHistoryTest

Another layout

viewTransactionHistory

Acceptance Test: **ViewTransactionHistory** Priority: 2 Point: 1 Risk: 1

The user must be registered and logged in the service. The user selects an interval of time or a type of transaction to view the list of transactions.

Related US: SelectTransaction

User Stories basics 1/2

- User stories are a simple, few-sentence description of a **functional requirement (features)** that provide value to the customer
- User stories are written by (with) the **customer**
- User stories are usually written on an **index card** which is passed to the team member who will work on it

User Stories basics 2/2

- Developers estimate how long the stories might take to implement in **story points** 1, 2, or 3 weeks
  - ideal development time: how long it would take to implement the story in code without distractions, other assignments and problems
  - Longer than 3 weeks the user story must be split
- Each story should be implemented in one iteration
  - http://www.extremeprogramming.org/rules/userstories.html

Gathering User Stories

- Goal-oriented approach:

Auction System: user can place bid and get items

  - Identify the **goal of the system**
  - What **steps (by feature)** does the **user** need to do to achieve the goal?

1. the user log-ins (registered user?); 2. browses items; 3. selects item(s); 4. places a bid; 5. if he wins, purchases the item else starts from 2.

  - Describe them in **user stories**

User log-in
If the user is registered, the user enters the system supplying its credentials. If credentials are correct the user is prompted with the welcome page . If is not ….

Additional requirements

- Non-functional requirements and the constraints are not written as stand alone user stories
- The user story cards can be augmented with a listing of non-functional requirements and constraints
  - for example: the system must be 96% reliable

Non-functional Rs and constraints

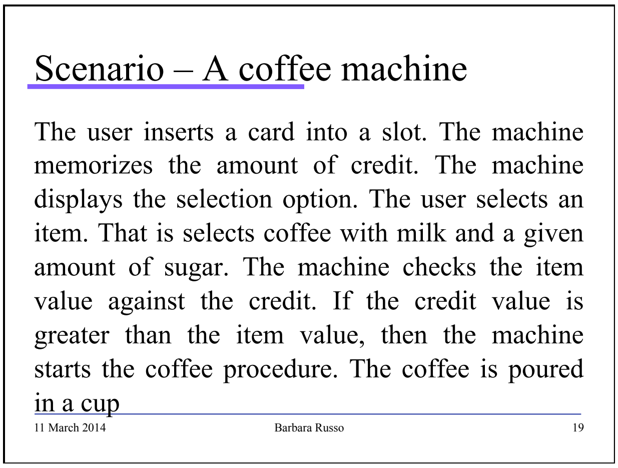
- Non-functional requirements
  - All transactions must complete within 5 seconds of the submit button being clicked.
  - The mean-time-to-failure (MTTF) must be a minimum of 1000 hours. Failure downtime must not exceed 10 minutes.
- Constraints
  - The system shall be run on a web server that uses the Tomcat application server.
  - The system shall be fun on a web server that uses has a MySQL database.

Example of User Story (too detailed)

register		
Acceptance Test: registerTest	Priority: 2	Point: 3
The users can register online. A registration page asks the user's information, including the name of the user, a nickname (serves as the user ID), birthday, address, phone number, and email address. After registration, the user will get a temporary password that he/she may change later. If the nickname is already registered, an error message will show and the user will have to choose another nickname. To be registered as a seller, in addition to the user information, he/she also has to provide the SSN. If the SSN is already registered, the user will not be allowed to register again.		

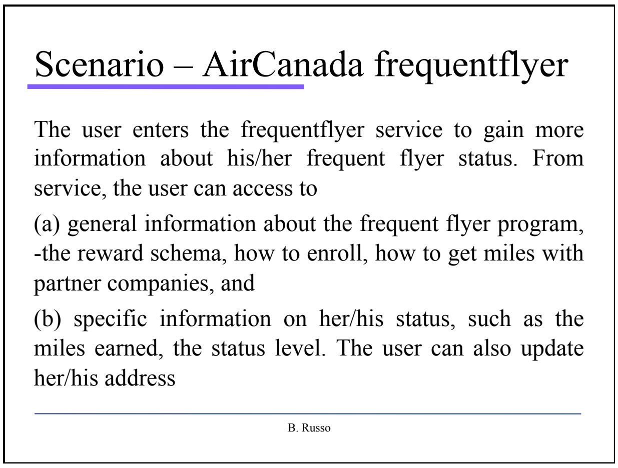
Example of User Story

register		
Acceptance Test: registerTest	Priority: 2	Point: 3
The user can register online providing user's information. The user can register as generic user or seller. After registration, the user will get a temporary password that he/she may change later. If the nickname is already registered, the user will have to choose another nickname. To be registered as a seller, in addition to the user information, he/she also has to provide extra information. If the user is already registered as seller, the user will not be allowed to register again.		



Scenario – A coffee machine

The user inserts a card into a slot. The machine memorizes the amount of credit. The machine displays the selection option. The user selects an item. That is selects coffee with milk and a given amount of sugar. The machine checks the item value against the credit. If the credit value is greater than the item value, then the machine starts the coffee procedure. The coffee is poured in a cup

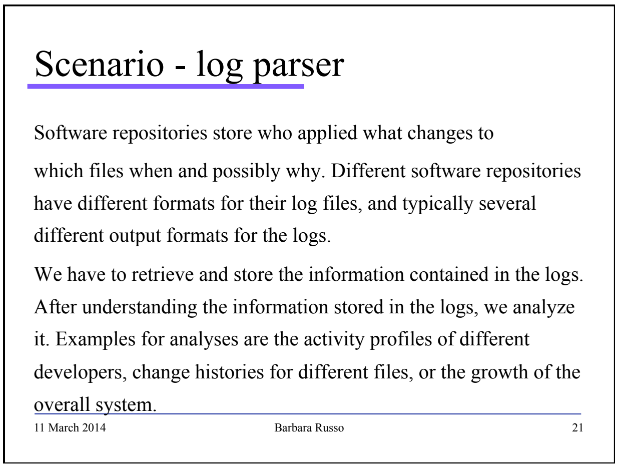


Scenario – AirCanada frequentflyer

The user enters the frequentflyer service to gain more information about his/her frequent flyer status. From service, the user can access to

(a) general information about the frequent flyer program, -the reward schema, how to enroll, how to get miles with partner companies, and

(b) specific information on her/his status, such as the miles earned, the status level. The user can also update her/his address



Scenario - log parser

Software repositories store who applied what changes to which files when and possibly why. Different software repositories have different formats for their log files, and typically several different output formats for the logs.

We have to retrieve and store the information contained in the logs. After understanding the information stored in the logs, we analyze it. Examples for analyses are the activity profiles of different developers, change histories for different files, or the growth of the overall system.

Scenario – log parser

- As a user of the system, I specify an input data source, the time frame of the logs and the level of information of the source.
- I can also load previously stored information. Once the datastream from the source has been parsed, I can:
- – choose different analysis methods,
- – get summary information about the analysis, and
- – export the data underlying the analysis for further analysis in other tools.

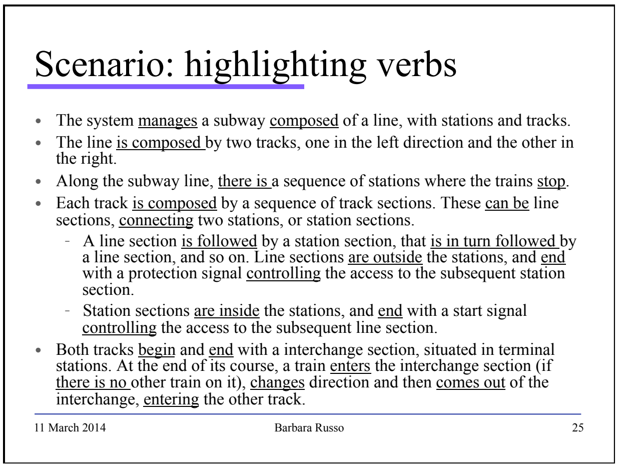
Extract the user stories

1. From the scenario highlight verbs
2. Not all the verbs describe an action to the system

Who is the actor in the scenario?

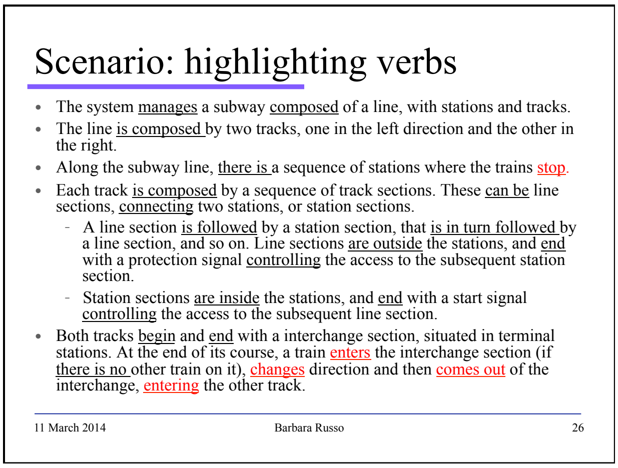
Scenario: subway simulation system

- **(Goal)** The system manages a subway composed of a line, with stations and tracks.
- **(Steps)** The line is composed by two tracks, one in the left direction and the other in the right.
- Along the subway line, there is a sequence of stations where the trains stop.
- Each track is composed by a sequence of track sections. These can be line sections, connecting two stations, or station sections.
  - A line section is followed by a station section, that is in turn followed by a line section, and so on. Line sections are outside the stations, and end with a protection signal controlling the access to the subsequent station section.
  - Station sections are inside the stations, and end with a start signal controlling the access to the subsequent line section.
- Both tracks begin and end with a interchange section, situated in terminal stations. At the end of its course, a train enters the interchange section (if there is no other train on it), changes direction and then comes out of the interchange, entering the other track.



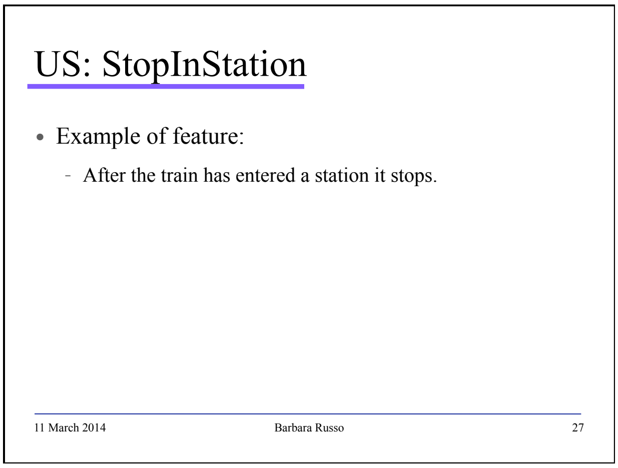
Scenario: highlighting verbs

- The system manages a subway composed of a line, with stations and tracks.
- The line is composed by two tracks, one in the left direction and the other in the right.
- Along the subway line, there is a sequence of stations where the trains stop.
- Each track is composed by a sequence of track sections. These can be line sections, connecting two stations, or station sections.
  - A line section is followed by a station section, that is in turn followed by a line section, and so on. Line sections are outside the stations, and end with a protection signal controlling the access to the subsequent station section.
  - Station sections are inside the stations, and end with a start signal controlling the access to the subsequent line section.
- Both tracks begin and end with a interchange section, situated in terminal stations. At the end of its course, a train enters the interchange section (if there is no other train on it), changes direction and then comes out of the interchange, entering the other track.



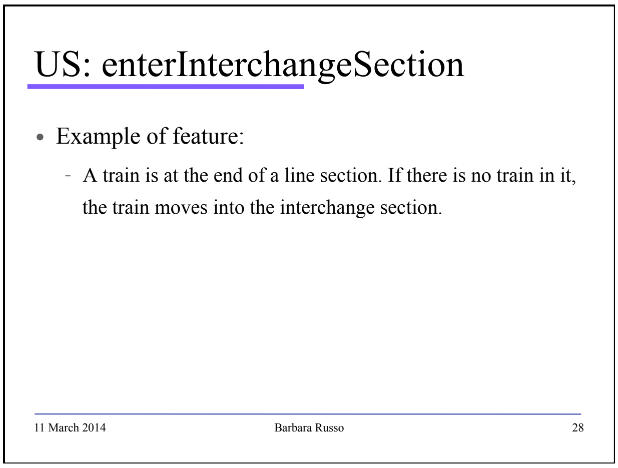
Scenario: highlighting verbs

- The system manages a subway composed of a line, with stations and tracks.
- The line is composed by two tracks, one in the left direction and the other in the right.
- Along the subway line, there is a sequence of stations where the trains stop.
- Each track is composed by a sequence of track sections. These can be line sections, connecting two stations, or station sections.
  - A line section is followed by a station section, that is in turn followed by a line section, and so on. Line sections are outside the stations, and end with a protection signal controlling the access to the subsequent station section.
  - Station sections are inside the stations, and end with a start signal controlling the access to the subsequent line section.
- Both tracks begin and end with a interchange section, situated in terminal stations. At the end of its course, a train enters the interchange section (if there is no other train on it), changes direction and then comes out of the interchange, entering the other track.



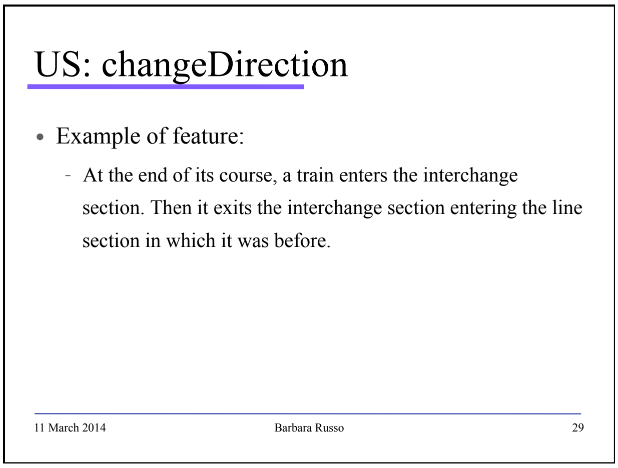
US: StopInStation

- Example of feature:
  - After the train has entered a station it stops.



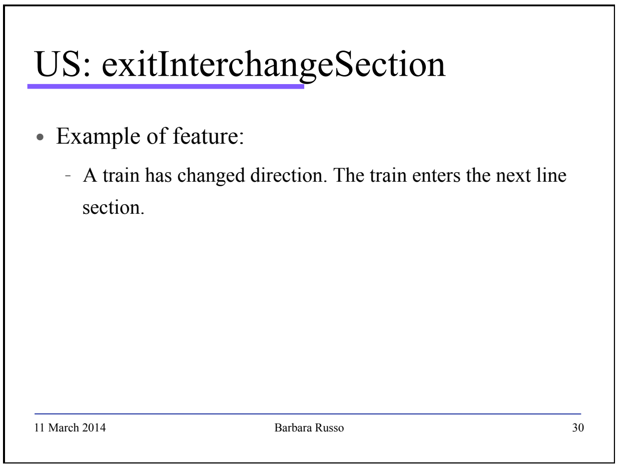
US: enterInterchangeSection

- Example of feature:
  - A train is at the end of a line section. If there is no train in it, the train moves into the interchange section.



US: changeDirection

- Example of feature:
  - At the end of its course, a train enters the interchange section. Then it exits the interchange section entering the line section in which it was before.



US: exitInterchangeSection

- Example of feature:
  - A train has changed direction. The train enters the next line section.

US: enterTrack

- Example of feature:
  - A train is at the end of an interchange section. The train enters the next track when the start signal allows.

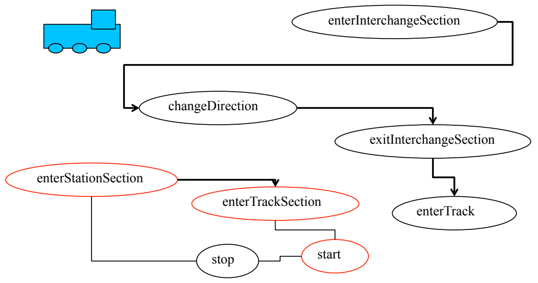
Not enough!

- The set of user stories we have drawn is incomplete:
  - E.g., There is no indication on how the train enters in station sections or track sections
- Exercise: Complete the set of uses of the system and draw a diagram

Example: US: enterStationSection

- A train is at the end of a line section. Before entering next station section, the train stops and waits the protection signal to go. The train enters the station section when the signal allows.

User stories diagram



Scenario: Which verbs correspond

- The system manages a subway composed of a line, with stations and tracks.
- The line is composed by two tracks, one in the left direction and the other in the right.
- Along the subway line, there is a sequence of stations where the trains stop.
- Each track is composed by a sequence of track sections. These can be line sections, connecting two stations, or station sections.
  - A line section is followed by a station section, that is in turn followed by a line section, and so on. Line sections are outside the stations, and end with a protection signal controlling the access to the subsequent station section.
  - Station sections are inside the stations, and end with a start signal controlling the access to the subsequent line section.
- Both tracks begin and end with a interchange section, situated in terminal stations. At the end of its course, a train enters the interchange section (if there is no other train on it), changes direction and then comes out of the interchange, entering the other track.

Scenario of your lab project

- Scenario

Acceptance Test

- An acceptance test is a test case written by the customer (in partnership with the developers)
  - **Verifies that the user story has been correctly implemented**
- Traceability between the user story and the acceptance test(s) used to verify that user story
- Acceptance tests are **black-box** tests where only input and output are known

Acceptance Test

- At least one acceptance test case per user story
- Often written on the back of the User Story card
- Better if they are automated
- Used in XP for testing functionality
  - http://www.extremeprogramming.org/rules/functionaltests.html

Acceptance Test

- The Acceptance Test should be **more specific** then the User Story description, and **define** better the customer's conditions of satisfaction
- Acceptance tests are different from UnitTests in that UnitTests are modelled and written by the **developer** of each class
- They should not include things like verification, boundary conditions, or "if then" statements.

Example of Acceptance test

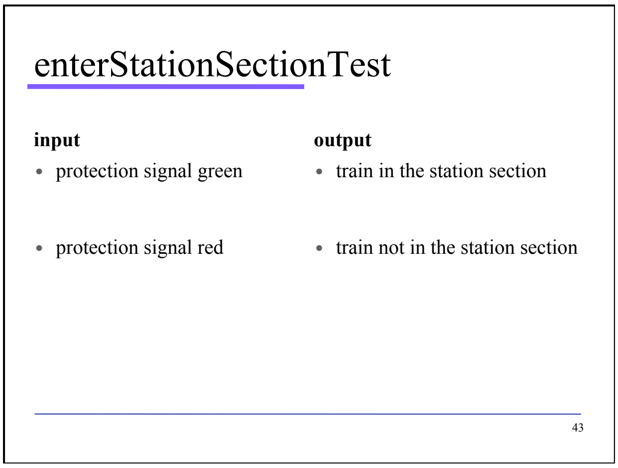
RegisterTest		
Input	Description	Output
name of the user, a nickname (serves as the user ID), birthday, address, phone number, and email address.	Preconditions: The user has accessed the on-line system. User can enter the credentials. Registration completes if credentials are correct and the user is not already registered .	temporary password (access) Or Error message (Incomplete/ incorrect credentials retry) Or Error message (choose another nickname)
Credentials + SSN	Precondition: as before. User enters credentials and SSN. Registration completes if credentials are correct and the user is not already registered as seller.	Temporary password (access) Or as before Or Error message (Wrong SSN retry) Or Error Message (User already registered cannot register twice)

Further examples from Ambler

- http://www.agilemodeling.com/artifacts/acceptanceTests.htm

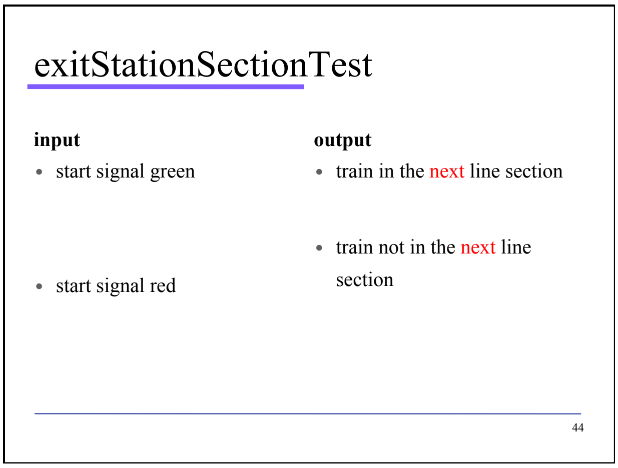
Exercise

- Describe the requirement “enterStationSection” and define one acceptance test for it



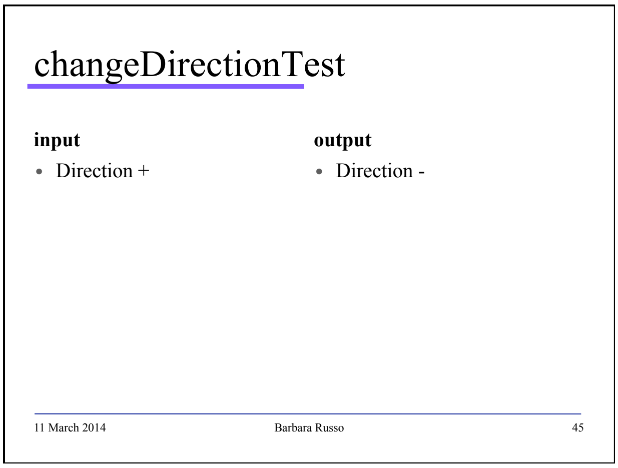
enterStationSectionTest

input	**output**
• protection signal green	• train in the station section
• protection signal red	• train not in the station section



exitStationSectionTest

input	**output**
• start signal green	• train in the next line section
• start signal red	• train not in the next line section



changeDirectionTest

input	**output**
• Direction +	• Direction -

changeDirectionTest

input

- protection signal green
- protection signal red
- start signal green
- start signal red

output

- train in the interc. section
- train not in the interc. section
- train in the line section of opposite direction
- train not in the line section of opposite direction

User Story Template

Title:

Acceptance Test:	Priority:	Story Points:

Acceptance test template

Title:

Input	Description	Output

User stories summary

User Stories	Priority	Points	Acceptance Tests
Browse auction items	1	7	BrowseTest
View item information	1	3	ItemPageTest
Register	2	3	RegisterTest
Login	2	2	LoginTest
View/Update personal information	2	3	PersonalInfoTest
Place Bid	2	4	BidTest
Add items	2	3	ItemAddTest
Access appropriate page	3	2	MainPageTest
Update item status	3	4	ItemUpdateTest
Maintain Categories	3	3	CategoryUpdateTest
Maintain user information	3	2	UserInfoTest
Total Story Points		38	

3. Define the System Metaphor

- Kent Beck "The system metaphor is a story that everyone - customers, programmers, and managers - can tell about how the system works"

System metaphor example

This metaphor emphasizes the logic of the best bid

- Auction system
- As a *Sorting Algorithm*: The system takes the selected items and ranks them in terms of a specific criterion (best offer). The top best item is provided to the user
- As a *Shop*: The system provides an item based on the request of the user and the availability of the item.

This metaphor emphasizes data flows and storage in the system

Another example: Coffee machine

Scenario.

- The user inserts a prepaid card into a slot. The machine reads the card and memorizes the amount of credit. The machine displays the selection option. The user selects an item. That is selects "coffee with milk and a given amount of sugar". If the credit value of the card is greater than the item value, then the machine starts the coffee procedure. At the end the coffee with milk is provided.

Naive metaphor: selecting and adding items

- The machine selects the amount of powder. An amount of water is selected and added. A plastic cup is selected and the coffee water is added into the cup. The milk powder is added into the cup. A given amount of sugar is added.

! It is not a scenario nor a user story!

- A scenario is a description of the way the system is **used**
  - the system is a whole entity providing services in a given environment
- User stories describe cases of use in a light way
  - Use cases diagram represents the user's actions to the system
- The metaphor describes how the system **works in reaction** to its use

Illuminating example

- http://xp123.com/articles/the-system-metaphor/

ARs and the three beasts

- Uncertainty
  - Agile requirements embraces change, allowing the uncertain to resolve itself as project development proceeds
- Irreversibility
  - Since agile requirements consume less resource, the requirements may be economically reversible
- Complexity
  - By attacking each requirement in turn rather than trying to digest the entire project at once, project development can be kept within the intellectual control of the team

Exercise

- Define a system metaphor for the subway system