IBM Software Group

Writing Good Use Cases

Terry Quatrani
UML Evangelist

Rational. software

e business on demand software

Agenda

- Use Cases and Requirements
- Do's and Don'ts for Use Cases
- Use Case Authoring Lifecycle
- Use Case Writing Styles
- TQ Book Review

What is a Requirement?

\Re*quire"ment\ (-ment), n.

1. The act of requiring; demand; requisition.
2. That which is required; an imperative or authoritative command; an essential condition; something needed or necessary; a need.

Webster's Revised Unabridged Dictionary, © 1996, 1998 MICRA, Inc.

Many Types of Requirements

- FURPS
  - Functionality
  - Usability
  - Reliability
  - Performance
  - Supportability

- Design Constraints
  - Operating systems
  - Environments
  - Compatibility
  - Application standards

- Legal and Regulatory requirements
  - Federal Communication Commission
  - Food and Drug Administration
  - Department of Defense

What is a Use Case?

- A use case is the specification of a set of actions performed by a system which yields an observable result that is of value to one or more actors or other stakeholders of the system
  - UML 2.0 Specification

- TQ translation – a use case is a fairly large piece of functionality that makes an actor happy

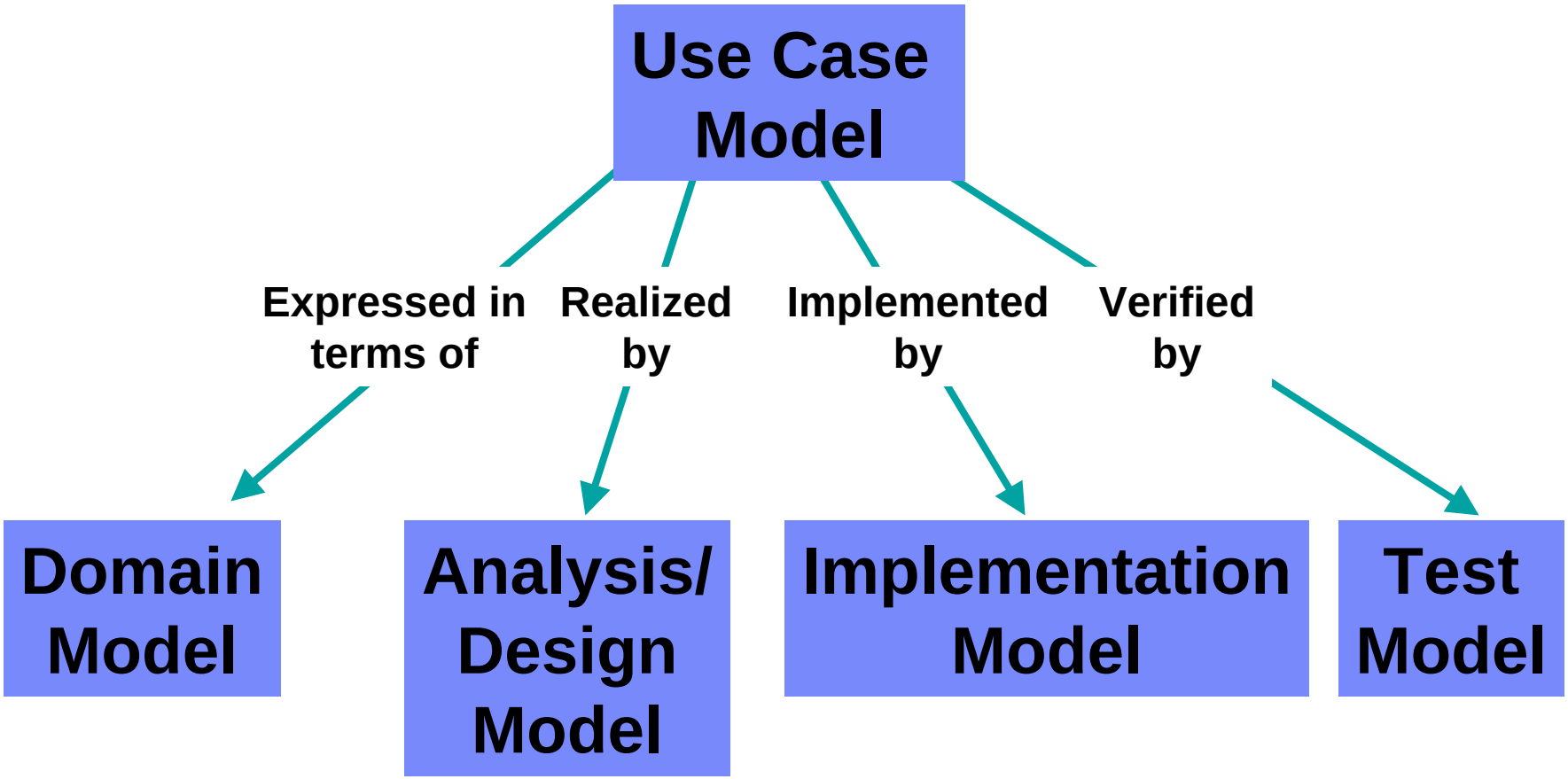
Use Cases and Requirements

- FURPS
  - Functionality
  - Usability
  - Reliability
  - Performance
  - Supportability
- Design Constraints
  - Operating systems
  - Environments
  - Compatibility
  - Application standards
- Legal and Regulatory requirements
  - Federal Communication Commission
  - Food and Drug Administration
  - Department of Defense

Why Create Use Cases?

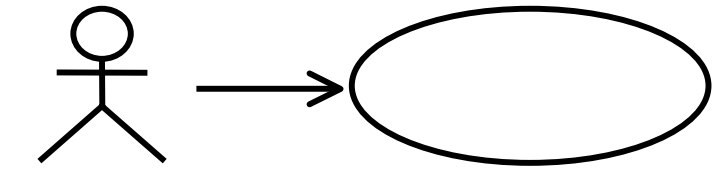
- Link stakeholder needs to system requirements
- Define clear boundaries of a system
- Capture and communicate desired behavior of the system
- Identify who & what interacts with the system
- Validate/verify requirements
- As a planning instrument

Use Cases Drive Analysis, Design, Testing

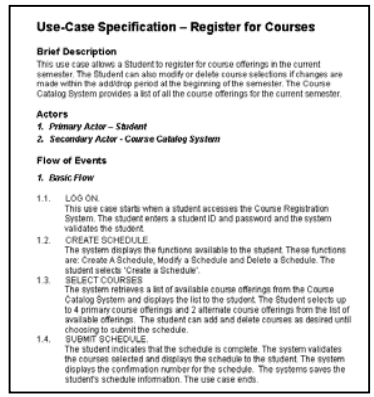


Documenting Use Cases

- Use cases are shown in diagrams



- Use cases are described in text



Use-Case Specification – Register for Courses

Brief Description

This use case allows a Student to register for course offerings in the current semester. The Student can also modify or delete course selections if changes are made within the add/drop period at the beginning of the semester. The Course Catalog System provides a list of all the course offerings for the current semester.

Actors

1. *Primary Actor – Student*
2. *Secondary Actor - Course Catalog System*

Flow of Events

1. *Basic Flow*

1.1. LOG ON.
This use case starts when a student accesses the Course Registration System. The student enters a student ID and password and the system validates the student.

1.2. CREATE SCHEDULE.
The system displays the functions available to the student. These functions are: Create A Schedule, Modify a Schedule and Delete a Schedule. The student selects 'Create a Schedule'.

1.3. SELECT COURSES
The system retrieves a list of available course offerings from the Course Catalog System and displays the list to the student. The Student selects up to 4 primary course offerings and 2 alternate course offerings from the list of available offerings. The student can add and delete courses as desired until choosing to submit the schedule.

1.4. SUBMIT SCHEDULE.
The student indicates that the schedule is complete. The system validates the courses selected and displays the schedule to the student. The system displays the confirmation number for the schedule. The systems saves the student's schedule information. The use case ends.

But…. The UML does not specify how a use case should be described.

What Should NOT be in a Use Case

- Implementation details
  - "The student information is written to a relational database"
  - "The student information is saved"
- GUI information
  - "The student presses the Select button to select a course"
  - "The student selects a course"
- Internal processing unrelated to a stakeholder request
  - "The system calculates a confirmation number using a hashing algorithm based on the date and the student id. The number is then displayed to the student"
  - "The system displays a confirmation number to the student"
- Non- functional requirements
  - "The system shall respond to the student course selection within 1 minute"
  - "The system shall be available 20/7"
  - "The system shall be able to handle scheduling requests from 1000 concurrent users"

But… the User Interface is Important

- Use cases are great sources for user interface requirements
  - What the UI must provide not how to do it!
- Separate document should contain UI details
  - Use visual or even physical mock-ups to describe UI
  - Describe navigation using *storyboards*
  - UML class and sequence diagrams are great aids here
- The UI Specification and use cases should be documented in parallel

What About Business Rules?

- "The student enters his/her student identification number. The identification number must be 6 to 8 alpha-numeric characters. It must contain one number that cannot be the first or last character.

- Keep your use case statements simple
  - "The student enters his/her student identification number"
- Document business rules in a separate supplementary specification
  - "Student identification numbers are numbers that are used to verify that a student is eligible to register for courses. The identification number must be 6 to 8 alpha-numeric characters. It must contain one number that cannot be the first or last character."

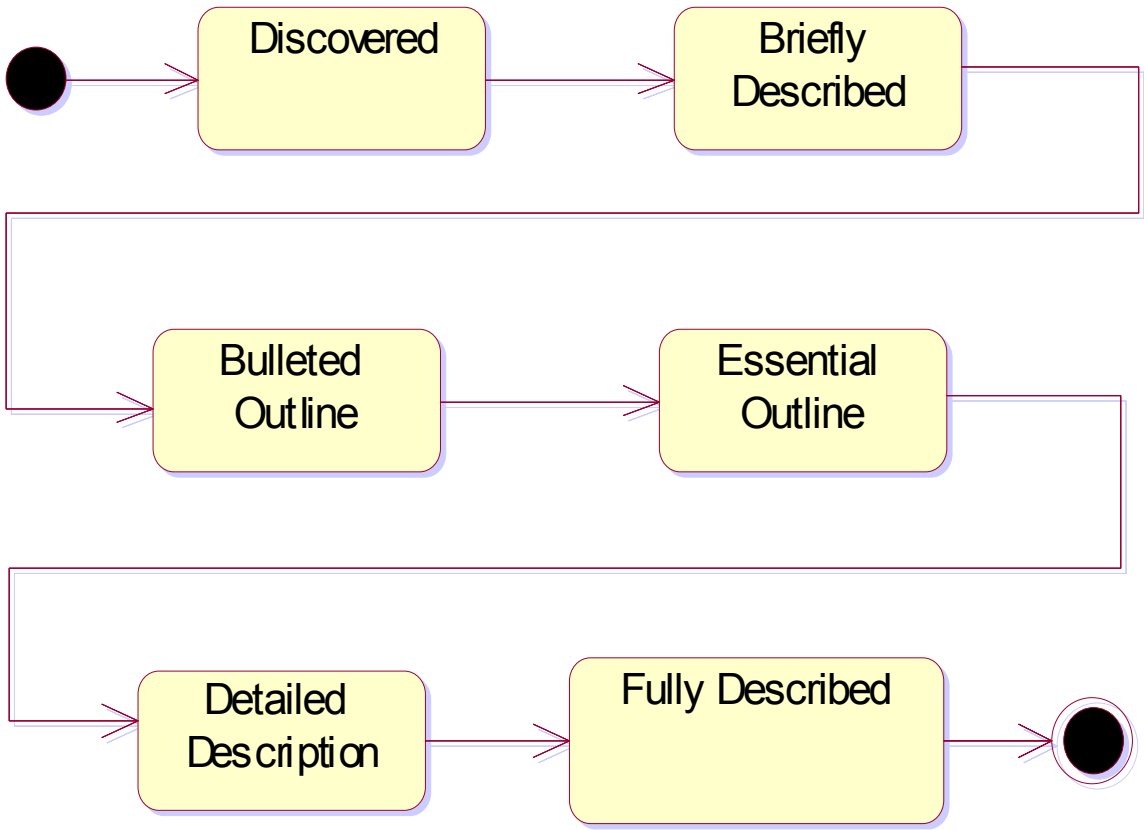
What should be IN a Use Case?

- Brief Description
- Flow of Events
  - Basic Flow
  - Alternate Flow(s)
- Special Requirements
- Pre-Conditions
- Post-Conditions
- Extension Points

- “Good use cases are balanced, describing essential system behavior while providing only the necessary details about the interactions between a system and its users”

Patterns for Effective Use Cases

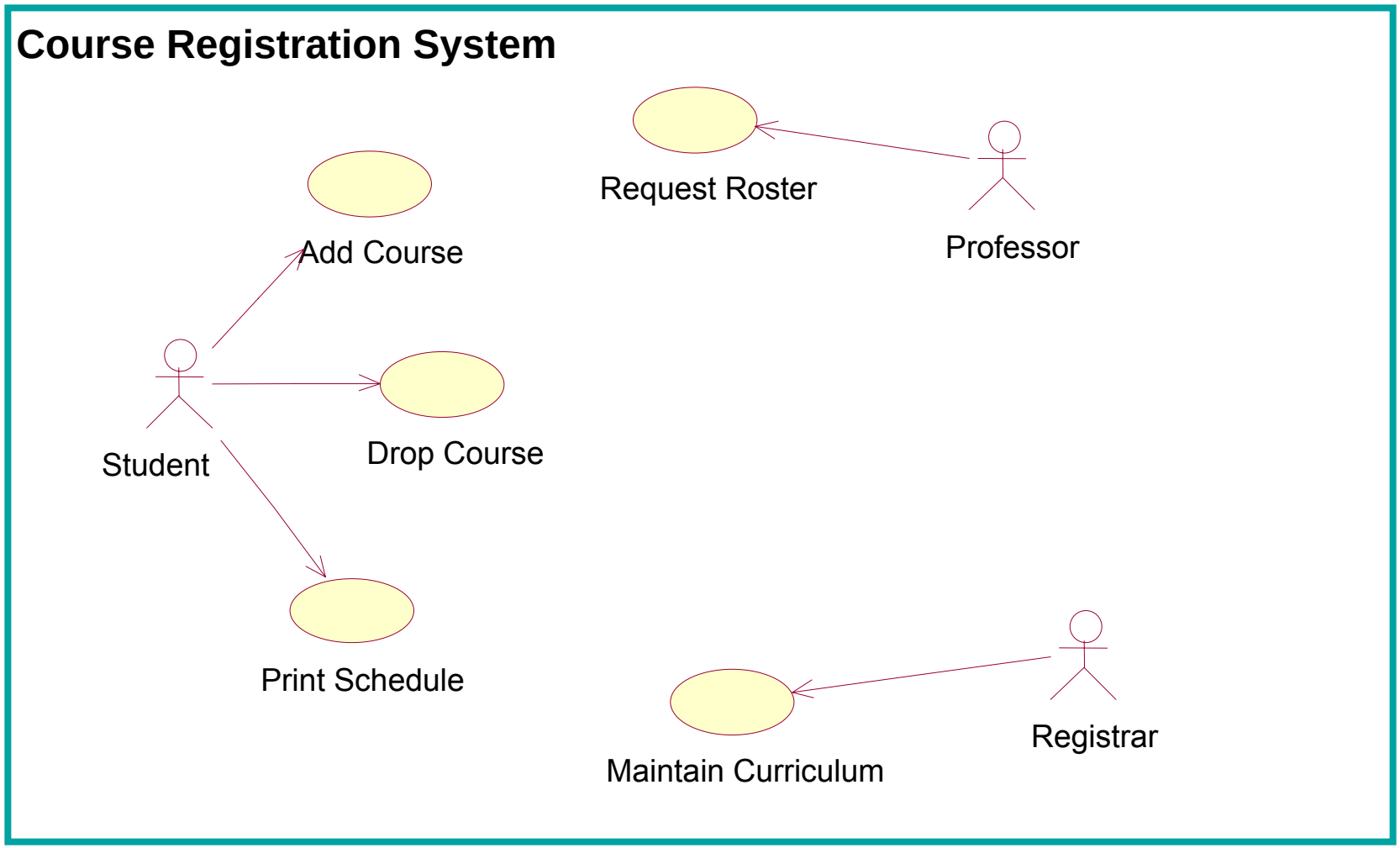
The Use Case Authoring Cycle



Discovery

- Examine the identified actors and ask the following questions
  - Why do they want to use the system?
- Vision document is a great source of input
  - Why is the system being built?
  - What problems is the system suppose to solve?
- Create a use case diagram showing actors and use cases
  - Only use communication associations at this point
  - Do NOT try to structure the use cases using ‘include’ and ‘extend’ relationships
    - Only leads to trouble ☺

Use Case Diagram



Brief Description

- A paragraph that describes
  - Who interacts with the use case
  - The purpose of the use case
- The *Add Course* use case is initiated by the Student. It allows the student to add a course to a schedule for a specified semester.

Bulleted Outline

- Time ordered outline of the steps in the use case
  - Typically just simple sentences
- Concentrate on the steps in the basic flow
- List major alternate flows and exceptions

- This is just a first cut at the use case flow of events
- Benefits
  - Allow you to get a handle on the complexity of the use case (more steps typically more complexity)
  - Allow you to get early buy in from the customer that you are building the right system
  - Provide basis for prototyping

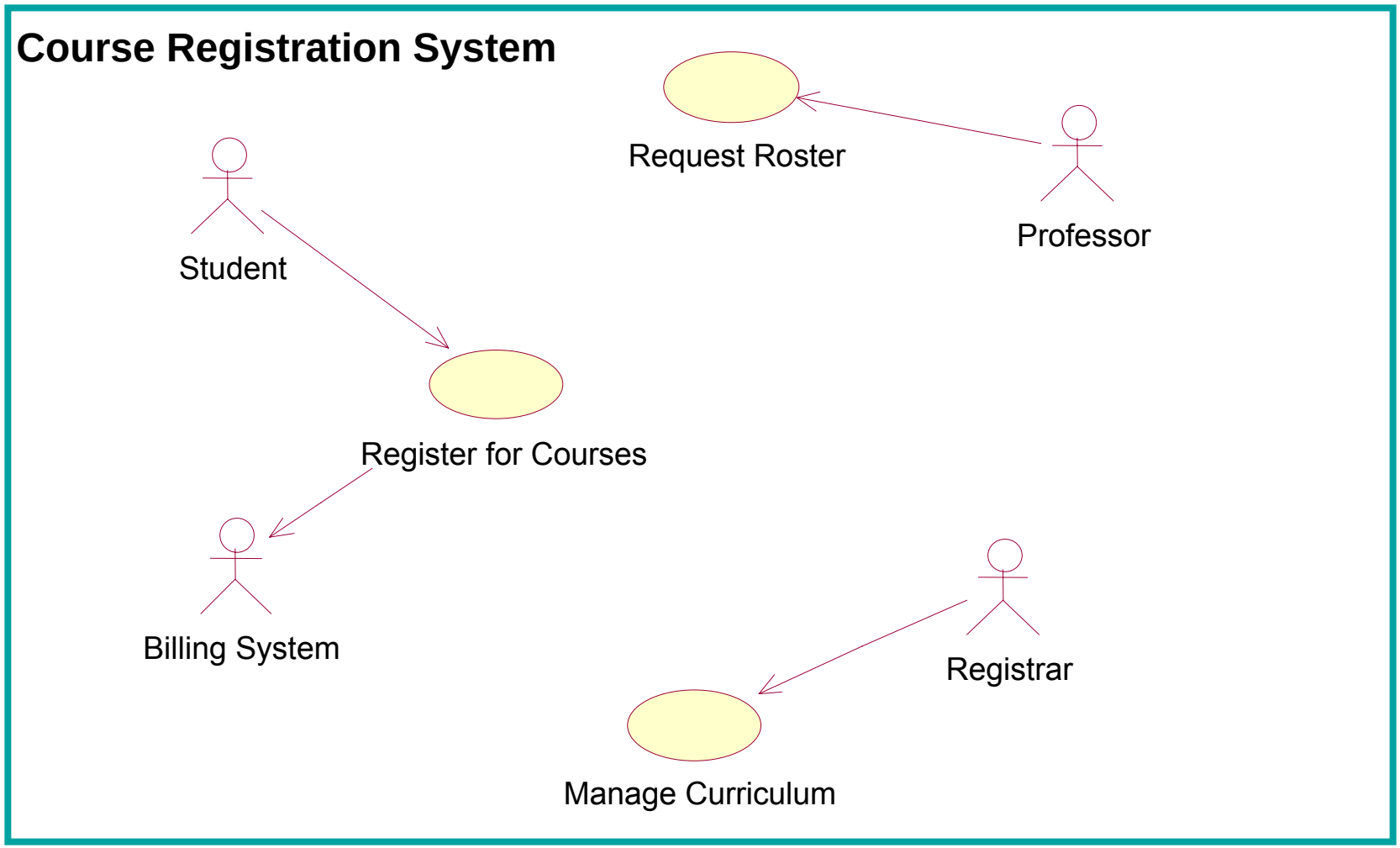
Add a Course Bulleted Outline

- Basic Flow
  - Student logs onto system
  - Student opts to add a course to a schedule
  - Student enters a course number
  - System verifies that student has satisfied pre-requisites for the course
  - System displays a list of open course offerings
  - Student selects an offering
  - Student is registered for the course
- Alternate Flows
  - Pre-requisites not satisfied
  - No course offerings available
  - Student cannot be registered for the course offering

Drop a Course Bulleted Outline

- Basic Flow
  - Student logs onto system
  - Student opts to delete a course from a schedule
  - Student enters a course number
  - System verifies that student is registered for the course
  - Student is deleted from the course
- Alternate Flows
  - Schedule not available
  - Student cannot be registered for the course offering

Use Case Diagram



Register for Courses Bulleted Outline

- Basic Flow
  - Student logs onto system
  - Student opts to create a schedule
  - Student enters a course number
  - System verifies that student has satisfied pre-requisites for the course
  - System displays a list of open course offerings
  - Student selects an offering
  - Student is registered for the course
- Alternate Flows
  - Student opts to modify an existing schedule
  - Student opts to print an existing schedule
  - Student opts to delete an existing schedule
  - Pre-requisites not satisfied
  - No course offerings available
  - Student cannot be registered for the course offering

Essential Outline

- Focus on the most important system behavior
  - How the system responds to an actor action/request
  - Great inputs for user interface prototypes
- “Black box” view of they system
  - Ensures that the needs of the actor are met
- Make sure you do not get bogged down with details… details come later
- Typically only done for the most important use cases

Register for Courses Essential Outline

User Action	***System Response***
1. Student enters a course number	Display list of course offerings
2. Student selects a course offering	Display course name, location and professor
3. Student confirms selection	Add student to the course roster

Detailed Description and Fully Described

- Iteratively add more detail to the outline to specify the steps of the use case
  - Each step should clearly state who is performing the action and the result of the action
- You are done when the use case has a complete flow of events, has terminology described in a support glossary and defines all inputs to and outputs from the use case
- Question to ask: Is there enough information specified to complete the system analysis, design and test?

Use Case Styles

- A use case is only as good as its author
- "…a well-written use case is relatively easy to read. People may suppose that easy-to-read also means easy-to-write, but that is a mistake"

  Patterns for Effective Use Cases

- Style questions:
  - Does the main flow reference other flows or not?
  - Do steps in the flows have numbers or titles or both?
  - Do alternative flows have numbers or titles or both?
  - How do you reference one part of a use case from another?
  - Can flows have embedded flows?
  - How do alternative flows tell what happens when they are done done?

Use Case Styles – Three Examples

- Rational Unified Process (RUP) Style
- Tagged Style
  - Based on the RUP style; Includes additions to resolve issues some have found with the RUP style
  - Bittner/Spence book: Use Case Modeling
- Table style

The Contents of a RUP Style Use Case

Use Case Name

1 Brief Description

2 Actors*

3 Flows of Events

- 3.1 Main (basic) Flow
  - 3.1.1 Step 1
  - 3.1.2 Step 2
  - 3.1.3 Step ...
- 3.2 Alternate Flows
  - 3.2.1 Alternate flow 1
    - 3.2.1.1 Step 1
    - 3.2.1.2 Step 2
    - 3.2.1.3 Step ...
  - 3.2.2 Alternate flow 2
  - 3.2.3 Alternate flow ...

4 Special Requirements

5 Pre-conditions

6 Post-conditions

7 Extension Points

*Actors are missing from the formal RUP template

- One basic flow
  - Happy-Day Scenario
- Many alternative flows
  - Regular variants
  - Odd cases
  - Exceptional (error) flows

RUP Style Main (Basic) Flow of Events

Use-Case Specification – Register for Courses

Brief Description

This use case allows a Student to register for course offerings in the current semester. The Student can also modify or delete course selections if changes are made within the add/drop period at the beginning of the semester. The Course Catalog System provides a list of all the course offerings for the current semester.

Actors

1. ***Primary Actor – Student***
2. ***Secondary Actor - Course Catalog System***

Flow of Events

1. ***Basic Flow***

1.1. LOG ON.
This use case starts when a student accesses the Course Registration System. The student enters a student ID and password and the system validates the student.

1.2. CREATE SCHEDULE.
The system displays the functions available to the student. These functions are: Create A Schedule, Modify a Schedule and Delete a Schedule. The student selects 'Create a Schedule'.

1.3. SELECT COURSES
The system retrieves a list of available course offerings from the Course Catalog System and displays the list to the student. The Student selects up to 4 primary course offerings and 2 alternate course offerings from the list of available offerings. The student can add and delete courses as desired until choosing to submit the schedule.

1.4. SUBMIT SCHEDULE.
The student indicates that the schedule is complete. The system validates the courses selected and displays the schedule to the student. The system displays the confirmation number for the schedule. The systems saves the student's schedule information. The use case ends.

Main flow shows the actor succeeding in his/her goal

Structure the flow into steps

Number and title each step

Describe steps (1-3 sentences)

Don't refer to alternate flows in the main flow

RUP Style Alternative Flows of Events

Alternative flows are flat

They can have steps

They have names

They should specify one thing

2. Alternate Flows

2.1. MODIFY A SCHEDULE.
At BF CREATE SCHEDULE, the Student already has a schedule that has been saved; the system retrieves and displays the Student's current schedule (e.g., the schedule for the current semester) and allows him/her to use it as a starting point. The use case resumes at BF SELECT COURSES.

2.2. DELETE A SCHEDULE.
AT BF CREATE SCHEDULE the Student has an existing schedule and chooses to delete it. The system retrieves and displays the Student current schedule. The system prompts the Student to verify the deletion. The Student verifies the deletion. The system deletes the schedule. The use case ends

2.3. UNIDENTIFIED STUDENT.
At BF LOG ON, the system determines that the student is not valid, an error message is displayed and the use case ends.

2.4. QUIT.
The Course Registration System allows the student to quit at any time during the use case. The Student chooses not to save any partial schedule information. The use case ends

2.5. *QUIT AND SAVE.*
The Student chooses to quit creating a schedule and chooses to save a partial schedule before quitting. All courses that are not marked as "enrolled in" are marked as "selected" in the schedule. The system saves the schedule. The use case ends.

2.6. CANNOT ENROL.
At BF SUBMIT SCHEDULE the system determines that prerequisites for a selected course are not satisfied, or that the course is full, or that there are schedule conflicts, the system will not enrol the student in the course. The system displays a message to the student and the use case continues at BF SELECT COURSES.

2.7. COURSE CATALOG UNAVAILABLE.
At BF SELECT COURSES, the system determines that the Course Catalog system is not available. The system displays an error message and the use case ends.

2.8. REGISTRATION CLOSED.
At BF LOG ON, the system determines that registration is closed; the system indicates that the student can no longer select courses and the use case ends.

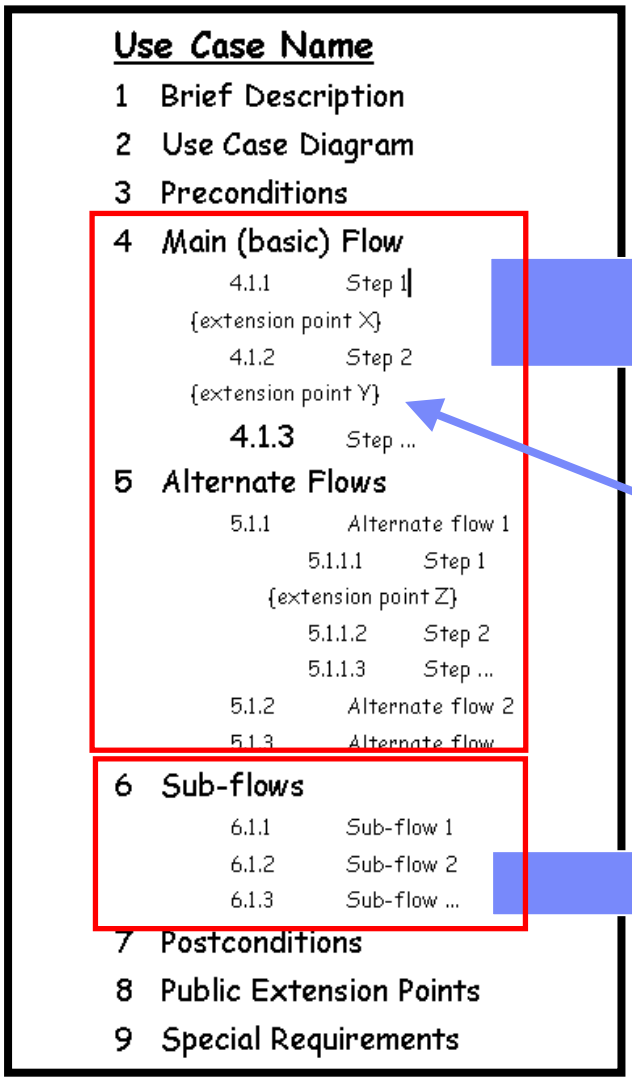
Say in which step or alternate flow the flow starts

Say what causes the flow to start

Say what happens

Say where the flow resumes

The Contents of a Tagged Style Use Case



- One basic flow
  - Happy-Day Scenario
- Many alternative flows
  - Regular variants
  - Odd cases
  - Exceptional (error) flows
- Extension points
- 0..many sub-flows
  - Reusable steps
  - Complex steps

Tagged Style - Main Flow

Main flow steps do reference other parts of the use case

Extension point "labels" are used for reference from elsewhere

Flow of Events

1. Basic Flow

{Log on}

1.1. This use case starts when a student accesses the Course Registration System. The student enters a student ID and password and the system validates the student.

{Create schedule}

1.2. The system displays the functions available to the student. These functions are: Create A Schedule, Modify a Schedule and Delete a Schedule. The student selects 'Create a Schedule'.

1.3. Perform subflow ***Select Courses***

{Submit Schedule}

1.4. The student indicates that the schedule is complete. The system validates the courses selected and displays the schedule to the student.

1.5. The system displays the confirmation number for the schedule.

1.6. The systems saves the student's schedule information.

{Use Case Ends}

1.7. The use case ends.

Structure the flow into steps

Number each step

Describe steps briefly

Tagged Style - Alternative Flows

Say at which extension point they start

Say what causes the flow to start

Say what happens

Say at which extension point the flow resumes

2. Alternative Flows

2.1. MODIFY A SCHEDULE.
2.1.1. At {Create Schedule}, the Student already has a schedule that has been saved; the system retrieves and displays the Student's current schedule (e.g., the schedule for the current semester) and allows him/her to use it as a starting point.
2.1.2. Perform subflow **Select Courses**.
2.1.3. The use case resumes at basic flow {Submit Schedule}

2.2. DELETE A SCHEDULE.
2.2.1. At {Create Schedule} the Student has an existing schedule and chooses to delete it.
2.2.2. The system retrieves and displays the Student's current schedule.

{Delete}

2.2.3. The system prompts the Student to verify the deletion. The Student verifies the deletion.
2.2.4. The system deletes the schedule.
2.2.5. The use case ends

2.3. DELETE A SUBMITTED SCHEDULE
2.3.1. At {Delete} the student's schedule has already been submitted.
2.3.2. The system warns the student that deleting the schedule after it has been submitted will incur a 10% processing fee,
2.3.3. The use case continues at {Delete}.

Tagged Style - Subflows

Do not say where they start – they are explicitly called

Subflows always have steps

They always resume at the next step from where they are called

3. Subflows

3.1. SELECT COURSES.

3.1.1. The system retrieves a list of available course offerings from the Course Catalog System and displays the list to the student.

3.1.2. The Student selects up to 4 primary course offerings and 2 alternative course offerings from the list of available offerings.

3.1.3. The student can add and delete courses as desired until choosing to submit the schedule.

Purpose: Break out complex or reused steps

Write the main and alternate flows first, then factor out the subflows

Use Cases in Tables

Hard to understand the sequence

Programmers love this format

Use-Case Specification – Register for Courses (Table Version)

Brief Description

Same as previous versions.

Actors

1. **Primary Actors**
   1.1. **Student**
2. **Secondary Actors**
   2.1. **Course Catalog System**
   2.2. **Accounting System**

Flow of Events

1. **Basic Flow**

Step	Actor Action	System Response
1	This use case starts when a student accesses the Course Registration System. The Student enter a student id and password.	The system the system validates the student.
2	The student selects 'Create a Schedule'.	The system displays the functions available to the student. These functions are: Create A Schedule, Modify a Schedule and Delete a Schedule
3	The Student selects up to 4 primary course offerings and 2 alternative course offerings from the list of available offerings. The student can add and delete courses as desired until choosing to submit the schedule	The system retrieves a list of available course offerings from the Course Catalog System and displays the list to the student
4	The student indicates that the schedule is complete	The system validates the courses selected and displays the schedule to the student. The system displays the confirmation number for the schedule. The systems saves the student's schedule information. The use case ends.

Some things don't neatly fit in a column

Non-technical stakeholders hate this format

Use Case Styles

- It is important to make a conscious decision about a style
- Decide up-front and enforce the chosen style
- Document the style in the Use Case Modeling Guidelines document

Further Writing Considerations

- Use of “if” statements
- Actor choices
- Iteration
- Sequence of events

Using “if-statements”

Good things about "ifs"

- Familiar to programmers

Bad things about "ifs"

- Can be hard to follow
- Harder to implement and test

2. Alternative Flows

2.1. MODIFY A SCHEDULE.
At BF CREATE SCHEDULE, if the Student already has a schedule that has been saved; the system retrieves and displays the Student's current schedule (e.g., the schedule for the current semester) and allows him/her to use it as a starting point. The use case resumes at BF SELECT COURSES.

2.2. DELETE A SCHEDULE.
AT BF CREATE SCHEDULE if the Student has an existing schedule and chooses to delete it the system retrieves and displays the Student current schedule. The system prompts the Student to verify the deletion. If the schedule has already been submitted then the system warns the student that deleting the schedule after it has been submitted will incur a 10% processing fee and the system notifies the accounting system that the schedule was deleted. The Student verifies the deletion. The system deletes the schedule. The use case ends

2.3. UNIDENTIFIED STUDENT.
At BF LOG ON, if the system determines that the student is not valid, an error message is displayed and the use case ends.

2.4. QUIT.
The Course Registration System allows the student to quit at any time during the use case. If the Student chooses not to save any partial schedule information the use case ends without the system saving anything. Otherwise all courses that are not marked as "enrolled in" are marked as "selected" in the schedule. The system saves the schedule. The use case ends.

2.5. CANNOT ENROL.
At BF SUBMIT SCHEDULE if the system determines that prerequisites for a selected course are not satisfied, or if that the course is full, or if that there are schedule conflicts, the system will not enrol the student in the course. The system displays a message to the student and the use case continues at BF SELECT COURSES.

2.6. COURSE CATALOG UNAVAILABLE.
At BF SELECT COURSES, if the system determines that the Course Catalog system is not available the system displays an error message and the use case ends.

No “if-statements”

Good

- Clearer
- Easier to read
- Easier to define scenarios

Bad

- More alternative flows

Decide up-front whether your team will use if-statements in it's use cases

2. Alternative Flows

2.1. MODIFY A SCHEDULE.
At BF CREATE SCHEDULE, the Student already has a schedule that has been saved; the system retrieves and displays the Student's current schedule (e.g., the schedule for the current semester) and allows him/her to use it as a starting point. The use case resumes at BF SELECT COURSES.

2.2. DELETE A SCHEDULE.
AT BF CREATE SCHEDULE the Student has an existing schedule and chooses to delete it. The system retrieves and displays the Student current schedule. . The system prompts the Student to verify the deletion. The Student verifies the deletion. The system deletes the schedule. The use case ends

2.3. DELETE A SUBMITTED SCHEDULE
At AF DELETE A SCHEDULE the student's schedule has already been submitted. The system warns the student that deleting the schedule after it has been submitted will incur a 10% processing fee, The student confirms the deletion. The system notifies the accounting system that the schedule was deleted. The use case ends.

2.4. UNIDENTIFIED STUDENT.
At BF LOG ON, the system determines that the student is not valid, an error message is displayed and the use case ends.

2.5. QUIT.
The Course Registration System allows the student to quit at any time during the use case. The Student chooses not to save any partial schedule information. The use case ends.

2.6. QUIT AND SAVE.
The Student chooses to quit creating a schedule and chooses to save a partial schedule before quitting. All courses that are not marked as "enrolled in" are marked as "selected" in the schedule. The system saves the schedule. The use case ends.

2.7. CANNOT ENROL.
At BF SUBMIT SCHEDULE the system determines that prerequisites for a selected course are not satisfied, or that the course is full, or that there are schedule conflicts, the system will not enrol the student in the course. The system displays a message to the student and the use case continues at BF SELECT COURSES.

Actor Choices

One choice in the main flow, other choices are handled in alternative flows

CRUD use cases

Flow of Events

1. Basic Flow

1.1. LOG ON.
This use case starts when a student accesses the Course Registration System. The student enters a student ID and password and the system validates the student.

1.2. CREATE SCHEDULE.
The system displays the functions available to the student. These functions are: Create A Schedule, Modify a Schedule and Delete a Schedule. The student selects 'Create a Schedule'.

1.3. SELECT COURSES
The system retrieves a list of available course offerings from the Course Catalog System and displays the list to the student. The Student selects up to 4 primary course offerings and 2 alternative course offerings from the list of available offerings. The student can add and delete courses as desired until choosing to submit the schedule.

1.4. SUBMIT SCHEDULE.
The student indicates that the schedule is complete. The system validates the courses selected and displays the schedule to the student. The system calculates the confirmation number using a hashing algorithm based on the student ID and adds it to the student record. The system displays the confirmation number for the schedule. The system saves the student's information in the Student Information Database. The use case ends.

2. Alternate Flows

2.1. UC 1.3 MODIFY A SCHEDULE.
At BF CREATE SCHEDULE, the Student already has a schedule that has been saved; the system retrieves and displays the Student's current schedule (e.g., the schedule for the current semester) and allows him/her to use it as a starting point. The use case resumes at BF SELECT COURSES.

2.2. DELETE A SCHEDULE.
AT BF CREATE SCHEDULE the Student has an existing schedule and chooses to delete it. The system retrieves and displays the Student current schedule. The system prompts the Student to verify the deletion. The Student verifies the deletion. The system deletes the schedule. The use case ends

2.3. UNIDENTIFIED STUDENT.
At BF LOG ON, the system determines that the student is not valid, an error message is displayed and the use case ends.

Showing Iteration

Flow of Events

1. Basic Flow

1.1. LOG ON.
This use case starts when a student accesses the Course Registration System. The student enters a student ID and password and the system validates the student.

1.2. CREATE SCHEDULE.
The system displays the functions available to the student. These functions are: Create A Schedule, Modify a Schedule and Delete a Schedule. The student selects 'Create a Schedule'.

1.3. SELECT COURSES
The system retrieves a list of available course offerings from the Course Catalog System and displays the list to the student. The Student selects up to 4 primary course offerings and 2 alternative course offerings from the list of available offerings. The student can add and delete courses as desired until choosing to submit the schedule.

1.4. SUBMIT SCHEDULE.
The student indicates that the schedule is complete. The system validates the courses selected and displays the schedule to the student. The system calculates the confirmation number using a hashing algorithm based on the student ID and adds it to the student record. The system displays the confirmation number for the schedule. The system saves the student's schedule information in the Student Information Database. The use case ends.

The Sequence of the Events

There is no reason why the credit card information must be entered at exactly this point in the sequence…

Use Case Specification: Sell Item

1. Brief Description

The Sell Item use case allows a Seller to create a new online auction. The Seller specifies *auction information*, and an online auction is created for the item.

2. Actors

2.1 Seller

The Seller is the user whose goal is to sell one or more items via an on-line auction.

3. Flow of Events

3.1 Basic Flow

3.1.1 START.

This use case starts when the Seller chooses to create an auction in order to sell an item.

3.1.2 ENTER INFORMATION.

The Seller enters the auction information [start time/duration of the auction, product information (title, description, a picture), starting price (i.e., minimum initial bid price), minimum bid increment, auction category in which to list the auction]

3.1.3 VALIDATE INFORMATION.

The System validates the entered auction information

3.1.4 SELLER CONFIRMS.

The System displays the auction information and requests that the Seller confirm. The Seller confirms the entered *auction information*. The system stores the *auction information*.

3.1.5 CHOOSE CREDIT CARD.

The system requires that the Seller provide *credit card information* to be used for payment of the auction fees and presents a choice to use a credit card on file. The Sellers chooses a card.

3.1.6 CREATE.

The system creates an auction with the entered information.

3.1.7 DISPLAY SUCCESS.

The system indicates that the auction was created successfully and display all of the auction information. The auction is now open and ready for bidding. The use case ends.

The Sequence of the Events Can Be Optional

Solution one

3.1 Basic Flow

3.1.1 START.

This use case starts when the Seller chooses to create an auction in order to sell a[illegible].

3.1.2 ENTER INFORMATION.

The Seller enters the *auction information*: start time/duration of the auction; product information (title, description, a picture); starting price (i.e., minimum initial bid price); minimum bid increment; and the auction category in which to list the auction. The Seller enters his or her *credit card information* to be used to pay the auction fees.

3.1.3 VALIDATE INFORMATION.

The System validates the entered auction information

3.1.4 SELLER CONFIRMS.

The System displays the auction information and requests that the Seller confirm[illegible] *auction information*. The system stores the *auction information*.

3.1.5 CREATE.

The system creates an auction with the entered information.

3.1.6 DISPLAY SUCCESS.

The system indicates that the auction was created successfully and display all of[illegible] auction is now open and ready for bidding. The use case ends.

Solution two

3.1 Basic Flow

3.1.1 START.

This use case starts when the Seller chooses to create an auction i[illegible] sell an item.

3.1.2 ENTER INFORMATION.

The Seller enters the auction information: start time/duration o[illegible]ction; product information (title, description, a picture); starting price (i.e., minimum initial bid price); minim[illegible] increment; and the auction category in which to list the auction. The Seller enters his or her credit card inf[illegible]on to be use to pay the auction fees.

3.1.3 VALIDATE INFORMATION.

The System validates the entered auction information

3.1.4 SELLER CONFIRMS.

The System displays the auction information and reque[illegible]s that the Seller confirm. The Seller confirms the entered *auction information*. The system stores the *auction inf[illegible]rmation*.

3.1.5 CHOOSE CREDIT CARD.

This step can occur at any time prior to the creation of the auction.

The system requires that the Seller provide *credit card information* to be used for payment of the auction fees and presents a choice to use a credit card on file. The Sellers chooses a card.

3.1.6 CREATE.

The system creates an auction with the entered information.

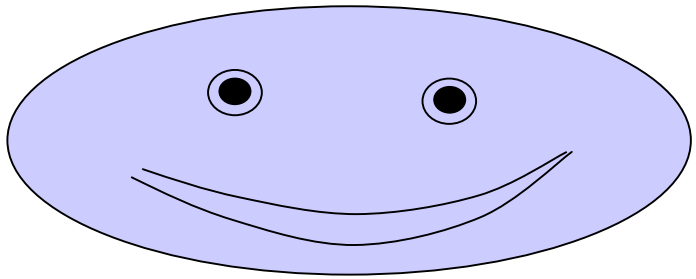
3.1.7 DISPLAY SUCCESS.

The system indicates that the auction was created successfully and display all of the auction information. The auction is now open and ready for bidding. The use case ends.

If you don't state otherwise, the developers will assume the sequence is a requirement

The Bottom Line

- How you write a use case affects its usability
  - By stakeholders
  - By the development team
- Choose a style, and stick to it
  - Make sure everyone uses the chosen style
- Think about, and use, good use case writing techniques
  - Use cases easier to write
  - They will also be easier to understand



TQ Book Review

- Use Case Modeling, Bittner and Spense, Addison-Wesley, 2003
  - Great book, my bible, lots of good practical advice
- Patterns for Effective Use Cases, Adolph and Bramble, Addison-Wesley, 2003
  - Lot of good advice, some bad advice, a bit hard to follow
- Use Cases, Requirements in Context by Kulak and Guiney, Addison-Wesley, 2004
  - Some good advice, really like their chapter on classic mistakes
- Writing Effective Use Cases, Cockburn, Addison-Wesley, 2001
  - Do not like the approach for use case modeling taken in this book, levels of use cases with funny symbols tend to lead to functional decomposition in my mind


Thank
You