Software. IN CONCERT.

IBM Rational Software
Development Conference 2006

IBM

Introduction To Writing Good Use Cases

Kurt Bittner
Program Director, Emerging Technologies
IBM Rational Software
kbittner@us.ibm.com

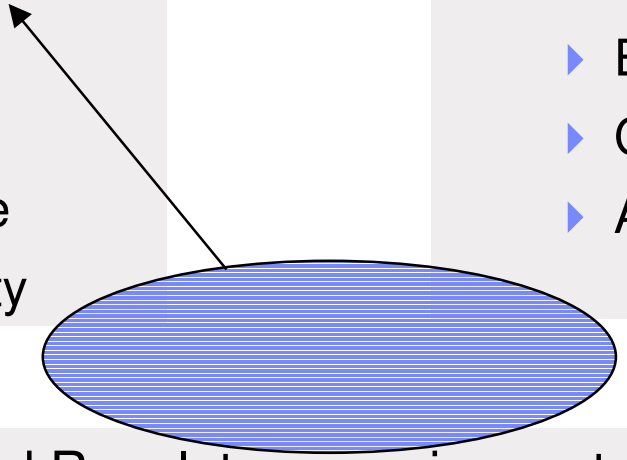
Rational. software

Agenda

- 1 - Short introduction to use cases
  - UML
  - Textual specification
- 2 - Use case styles
  - Style issues
  - RUP style
- 3 - Use case writing techniques
  - Using if-statements (or not)
  - Making choices
  - Showing iteration
  - Optional sequence of events
  - Level of detail
  - Creativity
- 4 - Use cases and Rational RequisitePro

There Can Be Lots of Types of Requirements

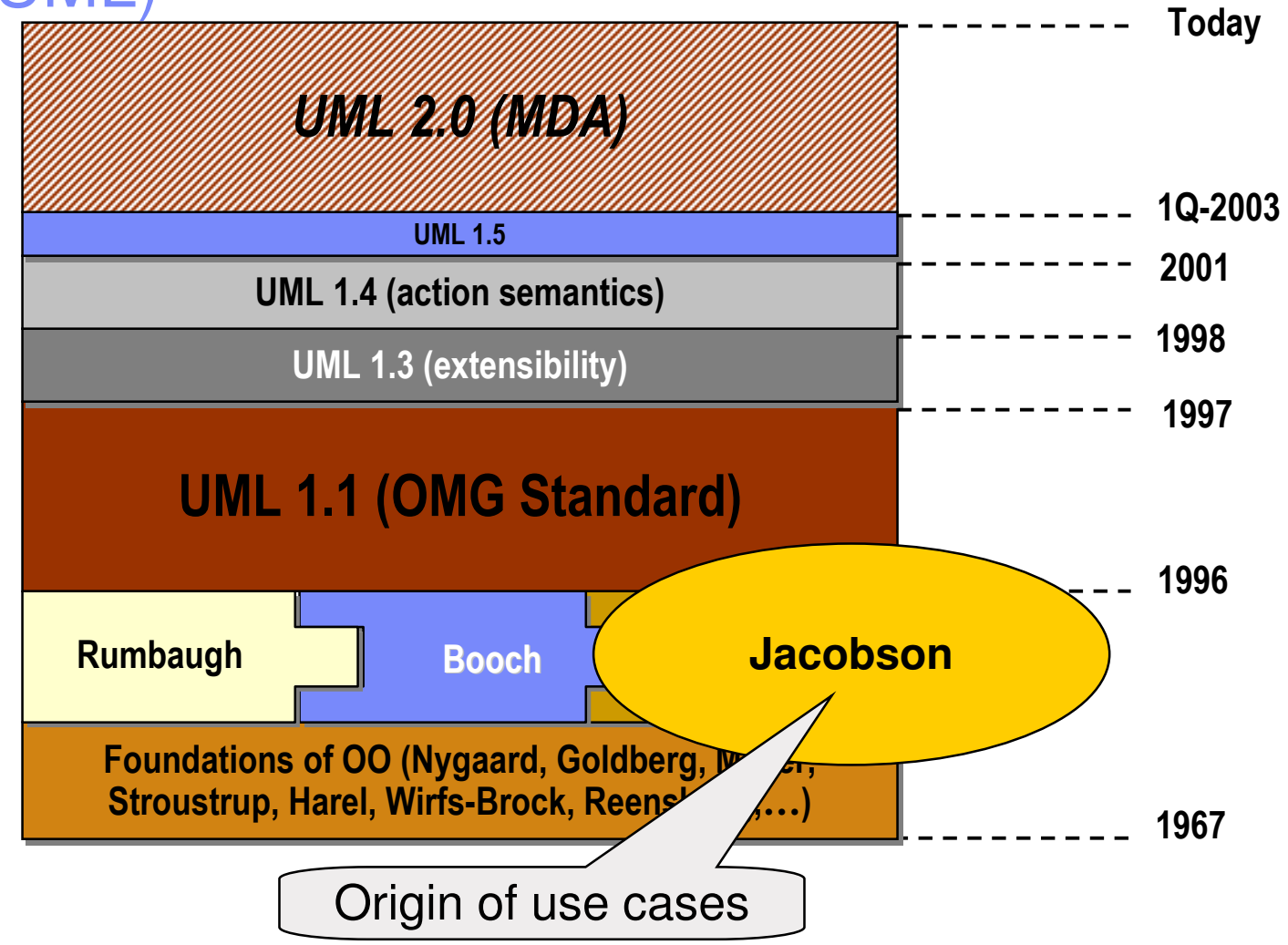
- FURPS
  - Functionality
  - Usability
  - Reliability
  - Performance
  - Supportability
- Design Constraints
  - Operating systems
  - Environments
  - Compatibility
  - Application standards



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

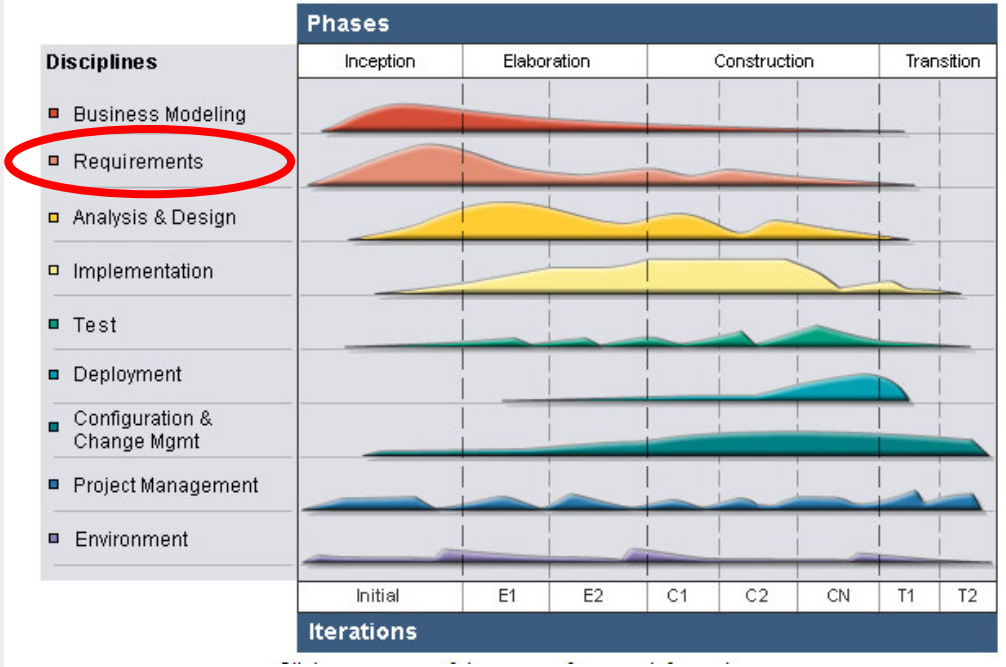
Use Cases Are Part of The Unified Modeling Language (UML)

The Evolution of UML



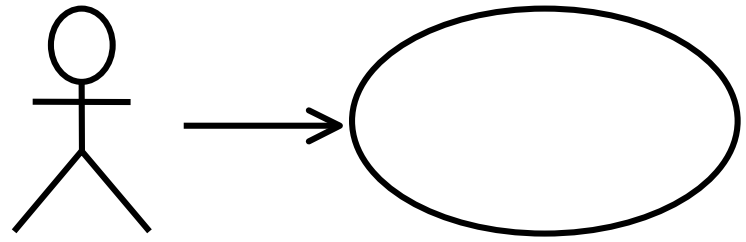
Use Case Are Part of the Rational Unified Process (RUP)

- Use cases are the basis for:
  - Analysis
  - Architecture
  - Design
  - Functional testing
  - User Interface design
- Use cases are the basis for agreement with the customer
  - And other stakeholders



What is a Use Case?

- Use cases are shown in UML 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.

Components of a Use Case Diagram

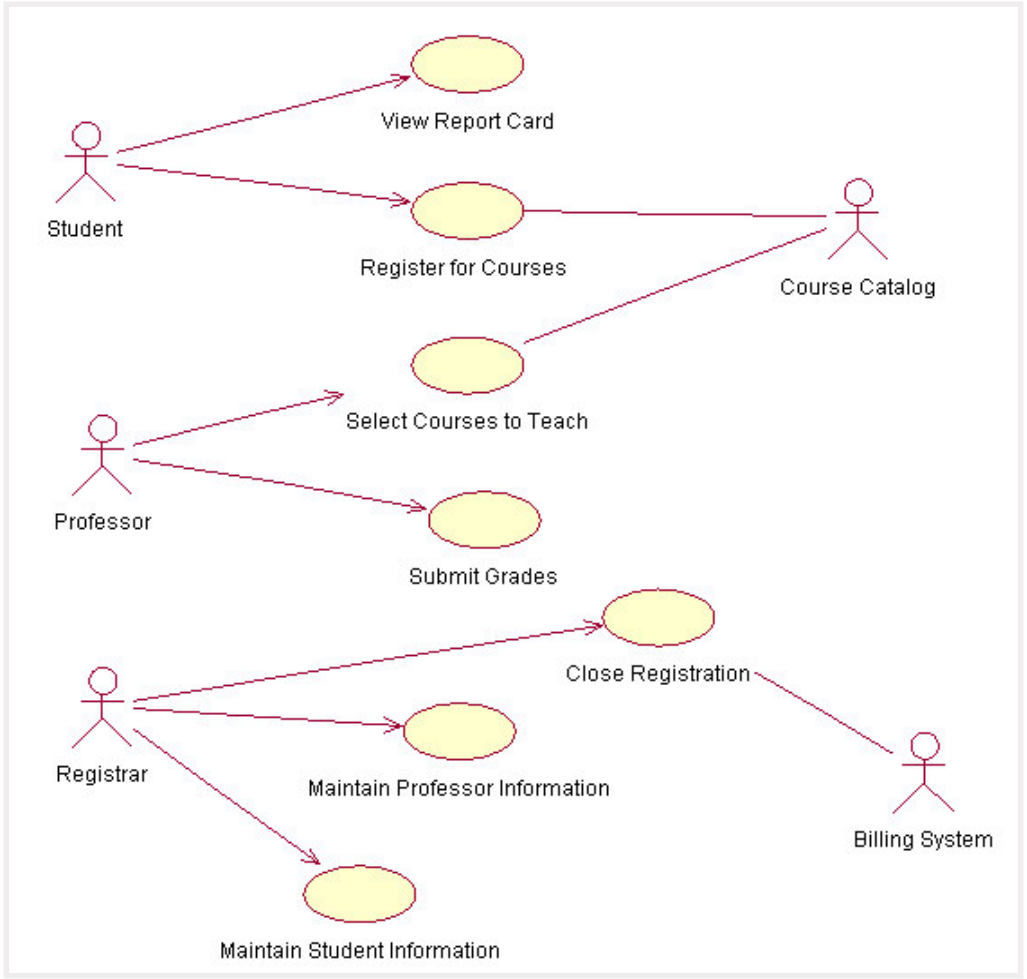
Actor:
Someone/something outside the system that interacts with the system

Use Case:
Defines a piece of functionality of the system

Communication – Association:
Shows the Actor and the Use Case communicate

Use Case Specification:
Basic flow of events, alternate flows, error flows and sub-flows as appropriate

Use Case Diagram - The Big Picture



Use Case Definition

A use case is the specification of a set of actions performed by a system, which yields an observable result that is, typically, of value for one or more actors or other stakeholders of the system (UML 2.0)

- Beware: the UML does not specify how the text of a use case should be structured, organized or written
- How you write use cases will have a large impact on how easy they are to design from and test (or not)

4 Requirements for a Use Case to be a Use Case*

1. Must provide value to a stakeholder
   - Goal orientation
2. Must be a complete narrative describing the story of how the value is provided
   - Must have main and alternative flows
3. Must stand alone
   - No sequencing of use cases
4. Must not describe design
   - What not how

*Based on the work of Ivar Jacobson, Kurt Bittner, Maria Ericsson, and others at IBM Rational Software

Use Cases vs. Declarative Statements

Declarative

- The system shall provide a list of class offerings for the current semester
- The system shall only allow registration for courses where the prerequisites are fulfilled.
- The systems shall provide a secure login.
- The system shall provide a confirmation number when the schedule is confirmed

- "The system shall…"
- Small perspective
- System orientation

Use Cases

1. The Student enters a student ID and password and the system validates the student.
2. The system displays the functions available to the student: create, modify, delete. The student chooses create.
3. The system presents a list of course offerings. The student chooses up to four…
4. The System validates the courses selected and displays a confirmation number…

- Broad perspective
- Goal orientation
- Actor (user) focus

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 Alternative Flows
    3.2.1 Alternative flow 1
      3.2.1.1 Step 1
      3.2.1.2 Step 2
      3.2.1.3 Step ...
    3.2.2 Alternative flow 2
    3.2.3 Alternative flow ..
4 Special Requirements
  4.1 Usability requirements
  4.2 Business rules
  4.x Other non-functional requirements...
5 Pre-conditions
6 Post-conditions
7 Extension Points

- One Basic Flow
  - "most likely path"
- Many Alternative Flows
  - Regular variants
  - Odd cases
  - Exceptional (error) flows

Use Case Styles

- Use cases can be considered "structured text"
- How you structure the text is the use case "style"
- Use cases can be written in many different styles
- For a given project (or organization) it is vital to choose and be consistent with one style
  - For consistency
  - For readability
  - For usability by the development team

Use Case Style Issues

- 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?

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

Goal: one "thing" per flow

Register For Courses Use Case

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

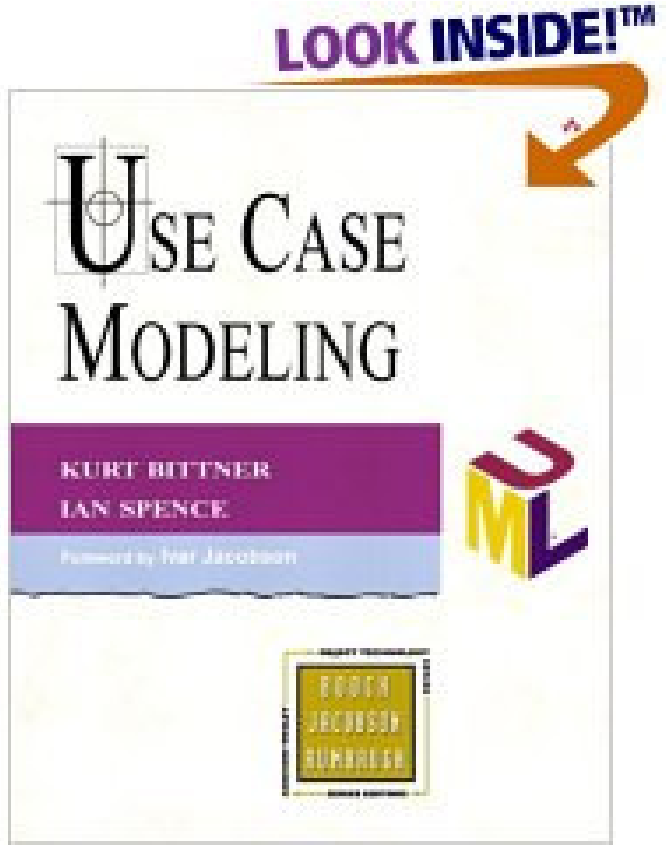
Say what causes the flow to start

Say what happens

Say where the flow resumes

More on "Style"

- See Use Case Modeling (Bittner and Spence) for guidance on handling subflows and extension points



Use Case Anti-Example 1

4.1 *Create User Group*

Brief Description	Create User Group is the functionality by which an authorized individual creates a new user group. Please note the two group types and details of their creation: • (Public) General Group: can only be created by an xxx User Administrator. • (Public) Article Group: will be created whenever a user is launched around an Article Available for User in the xxx environment.
Use Case Pre-Conditions	1. The UGC has been identified by the system. 2. The UGC has the appropriate permission to create the user group.
Use Case Post-Conditions	1. The user group has been created and is present and persistent in the system with the appropriate attributes.
List of Actors	• User Group Creator (UGC): 1. User Administrator 2. System
User Experience Links	
Basic Flow One: General Group	1. The use case begins when the User Administrator accesses the Manage User interface of xxx. 2. The User Administrator indicates that s/he wants to create a new group and provides the requested information: a. Title of user group (required) b. Description of user group (required)
Basic Flow Two: Article Group	Creation of this group is a post-condition of the Create Topic use case for "(Public) Article Topic." Please recognize that the end user of the system is oblivious to the handling of the "create (Public) Article Group" step.
Alternate Flows	

Use Case Writing Techniques

- Using if-statements (or not)
- Making choices
- Showing iteration
- Sequence of events
- Correct level of detail
- Creativity
- Use cases and Rational RequisitePro

Using "if-statements"

- Good things about "ifs"
  - Familiar to programmers
- Bad things about "ifs"
  - Can be hard to follow
  - Harder to implement and test

How would you remove the ifs?

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
  - More clear
  - 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 handled in the main flow, other choices are handled in alternative flows

Flow of Events

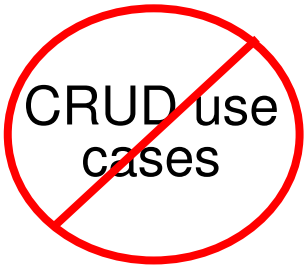
1. Basic Flow

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

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, 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 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 requests that the Seller confirm. The Seller confirms the entered *auction information*. The system stores the *auction information*.

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. The system 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.

Solution

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

Correct Level of Detail

- No GUI
  - Use prototypes or GUI specs, linked to the use cases
- No architecture
  - But use case steps may affect the architecture
- No "internal processing" unrelated to a stakeholder requirement

How much detail in a use case?
Enough to satisfy all of the stakeholders that their interests (requirements) will be satisfied in the delivered system

Correct Level of Detail?

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.

Use Case Anti-Example 2

Use-Case Specification: Delete User

1. Use-Case Delete User

1.1 Brief Description

The purpose of the delete user use-case is to provide the capability to remove a ABC user from the system.

2. Flow of Events

2.1 Basic Flow

1. From the Main Support, the ABC User will select the People option.
2. From the ABCUserMain.jsp menu, Support is notified that a user needs to be deleted from the application.
3. Support selects the ABC User's Detail option from the list of ABC Users on the ABCUserMain.jsp page.
4. Support reviews user information and selects the Delete button.
5. The QN business layer calls QNbsUsers.UserMgmt.DeleteABCUser
6. A confirm pop up is displayed with the users name in the following message; 'Click OK to delete <user> or click cancel to return to list.'
7. Support selects OK to delete the user.
8. Support is returned to the ABCUserMain.jsp page.

2.2 Alternative Flows

2.2.1 User is a task submitter

5a. Call existing KJN business layer to delete
5b. User is deleted from tblNtqSubmitter
5c. Go to 2.2.3.1 step 5

2.2.2 User owns a task folder

*2.2.2.1 New owner **is** identified*

5d. Support is prompted with a notification indicating that the user owns one or more folders.
5e. Support transfers task folder ownership to the new owner.
5f. User is deleted from tblMTFolderUser.
5g. Return to 2.1 step 6.

*2.2.2.2 New owner **not** identified*

5d. Support is prompted with a notification indicating that the user owns one or more folders.
5e. All folders are copied to a temporary table.
5f. User is deleted from tblMTFolderUser
5g. Return to 2.1 step 6.

2.2.3 Cancel Delete

6a. Support selects cancel from the confirm pop up dialog box.
6b. Return to 2.1 step 8.

Creativity

Use-Case Specification – Register for Courses (Creative alternative 1)

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 System will only show courses for which the Student has already fulfilled the prerequisites and for which the course is not already full. The system will not allow the student to register for a course for which there is a schedule conflict. . 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. A confirmation number for the schedule is displayed. The use case ends.

How you write the flows will affect the user experience

The Value of RequisitePro for Use Cases

- Increased ability to do reporting
  - "Show me all of the brief descriptions"
  - "Show me all of the High Priority use cases"
- Better tracking of history
  - Each flow or step can be tracked separately
- Better ability to keep track of the requirements artifacts
  - Documents and attributes in a known location
- Allows the ability to set up and manage traceability
  - To other requirements
  - To tests and design

Use Cases and RequisitePro
What is Marked as a Requirement?

- Steps?
- Step titles?
- Flows?
- The whole use case?

Requirements need unique identifiers
- RequisitePro fulfills this need

Use Cases and RequisitePro
What is Marked as a Requirement?

- RequisitePro usage scenarios will make a difference
  - Will Word be used? Soda? RequisiteWeb?
  - The client and the web are not as good for looking at LONG requirements text
- Understand what queries the users will want to make
  - "Show me all brief descriptions"
  - "Show me all of the preconditions"
- Understand the level of version control needed
- Understand if and how traceability will be used
  - Tracing down - Testing
  - Tracing up - features, business requirements
  - Impact analysis and coverage analysis

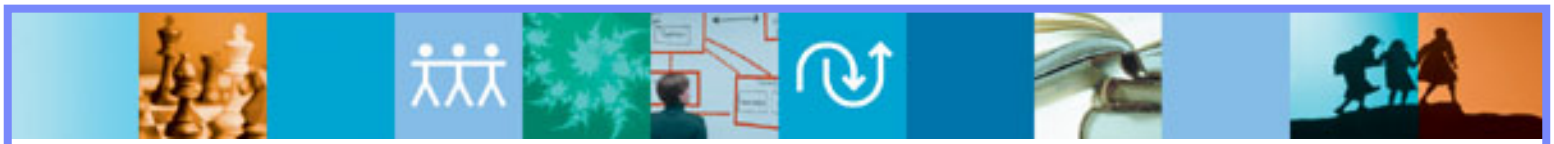
Writing Good Use Cases

- 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

For More Information

- Rational Unified Process
  - Requirements management discipline
  - Examples Overview in the tree browser
- Use Case Modeling, Bittner and Spence, Addison-Wesley, 2003.

Questions



Thank You

Jim Heumann
JimHeumann@us.ibm.com