Business Process
Flow and Use Cases

Prepared by Alice Bouché
President, Bouché & Associates, Inc.

January 23, 2006
Overview

• Business Process Flow
  - Swim lane approach

• Use Cases
  - Unified Modeling Language (UML)
  - Textual specification
  - Style
  - Techniques
IT Project Activity Overview

High Level Business Process

Gap Analysis

Phased Implementation and Role Based Training Plan

Phase I  Phase II  Phase III
IT Project Activities

1. Document Ideal Business Process Flow
2. Document Use Cases
   - Next level of requirements detail, but still not implementation specific
   - Use Cases drive the Gap Analysis and Role Based Training
3. Perform Gap Analysis
   - Identify the gaps between the software functionality vs. the use case requirements
   - Estimate work/cost required to resolve identified gaps
   - Prioritize & Schedule work into Phases
4. Implement in Phases as Prioritized
   - Coding and/or Configuration to address usability issues
   - Develop Role Based Training
   - Conduct User Acceptance Testing
   - Deploy Role Based Training
Business Process Flow
Business Process Flow

• Swim Lane Approach
  ▪ What’s a swim lane?
    • Represents a Discipline or Role (actions performed by Actors go into a swimlane)
    • Time moves from left to right
  ▪ Shape of objects
    • Diamond – a question, assumption or decision point
    • Rectangle – an action
    • Objects are connected by arrows to move from action to action
  ▪ Shaded objects
    • Blue Diamonds (asked and answered earlier in the process)
    • Gray (actions performed by the system) in lieu of another swim lane
    • Dashed outline around logical time-boxed functions (for readability)
Business Process Flow

1. Define at a High Level the Ideal Business Process
   - Core Team representation from key areas
   - Swim lane approach with Brown Paper and Post-It Notes

2. Review with broader base of stake holders for buy in and validation
   - Formalize the Post It Notes and Brown Paper

3. Present findings to management (Steering Committee)
Use Cases

- Use cases are part of the Unified Modeling Language (UML)
- Use cases are part of the Rational Unified Process (RUP) and are object oriented in design
  - Use cases are the basis for the work of the development team
    - 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?

A use case describes a sequence of actions a system performs that yields an observable result of value to a particular actor.

- Use Cases can be shown in UML diagrams
- Use Cases can be described in text

**Student** → **Register for Courses**

---

**Use Case Flow**

1. Brief Description
2. Actors
3. Flows of Events
   3.1 Main (basic) flow
   3.2 Alternate Flows
     3.2.1 Alternate flow 1
     3.2.2 Alternate flow 2
     3.2.3 Alternate flow ...
     3.2.4 Alternate flow n
4. Special Requirements
   4.1 Business rules
   4.2 Usability requirements
   4.3 Data definitions
5. Pre-conditions
6. Post-conditions
What makes a Use Case?

Four Requirements for a Use Case

1. Must provide value to a stakeholder
   • Goal orientation

2. Must be a complete narrative describing 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
   • Use Cases describe what not how
Use Cases vs. Declarative Statements

Use Cases (Do)
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

Declarative (Don’t)
• 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 system shall provide a secure login.
• The system shall provide a confirmation number when the schedule is confirmed.
   - Small perspective
   - System orientation
   - “The system shall” – where’s the value add?
Contents of a Use Case

Use Case Name

1. Brief Description
2. Actors
3. Flows of Events
   3.1 Main (basic) flow
   3.2 Alternate Flows
      3.2.1 Alternate flow 1
      3.2.2 Alternate flow 2
      3.2.3 Alternate flow ...
      3.2.4 Alternate flow n
4. Special Requirements
   4.1 Business rules
   4.2 Usability requirements
   4.3 Data definitions
5. Pre-conditions
6. Post-conditions

• One Basic Flow
• “Happy Day” Scenario
• Many Alternative Flows
  • Regular variants
  • Odd cases
  • Exceptional (error) flows
Use Case Styles

• 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?
RUP Style Use Case

RUP Style Main (Basic) Flow of Events

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

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.
Alternative Flows

RUP Style Alternative Flows of Events

Alternative flows are flat

They can have steps

They have names

Register For Courses Use Case

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

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
Using if statements

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

21. MODIFY A SCHEDULE
   AT BF CREATE SCHEDULE if the Student already has a schedule that has been saved, the system reveals 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.

22. DELETE A SCHEDULE
   AT BF CREATE SCHEDULE if the Student has an existing schedule and chooses to delete it, the system verifies and displays the Student's current schedule. The system prompts the Student to verify the deletion. If the 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 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.

23. 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.

24. 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, if courses that are not marked as "enrolled in" are marked as "selected in the schedule," the system saves the schedule. The use case ends.

25. CANNOT ENROLL
    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 there are schedule conflicts, the system will not enroll the student in the course. The system displays a message to the student and the use case continues at BF SELECT COURSES.

26. 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.
Without “if-statements”

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 its use cases

---

www.WeDeliverValue.com

Bouché & Associates, Inc.
Level of Detail

- Show iteration
- Show sequence of events when relevant
- No GUI
- No Architecture
  - However, use case steps may affect the architecture
- No “internal processing” unrelated to a stakeholder requirement (references to existing programs by name, etc.)
- How much detail in a use case?
  - Enough to satisfy all of the stakeholders that their interests (requirements) will be satisfied in the delivered solution
Summary

• Use cases are a very effective way to specify requirements for a system.
• They should be easy to use by the stakeholders and the development team.
• Use cases specify functional requirements
• Use cases are essential to the process:
  ▪ Communicate with and get agreement from stakeholders
  ▪ Used by the development team to architect, design and test