How to use *Use Cases*

Jan Kusiak, Training Services Manager
IRM Training Pty Ltd ACN 007 219 589
Suite 209, 620 St Kilda Rd, Melbourne, Vic. 3004, Australia
Tel: +613 9533 2300

Overview

Many business analysts and business users get frustrated at the perceived lack of information in a use case diagram. "It's all very well drawing a picture" they say but what about the details – what's actually going on?

When producing project documentation, use case diagrams are rarely used on their own. They will generally be accompanied by a textual use case and if they're complex, may also have a supporting activity diagram to show what's going on "inside" the use case.

To keep things simple we'll stick to use case diagrams and textual use cases. Textual use cases can be both formal and informal. Try the following exercises (from IRM's **Modelling Requirements with Use Case & the UML** workshop) to test your skills. First we'll draw a use case diagram from a plain English description, then build on it using textual use cases.

Exercise 1 – Course registration

The following should be textually analysed and a use case diagram created containing several use cases. Identify the actors, use cases and associations.

At the start of each semester a student can request a prospectus containing a course list. Information about a course is provided, such as the tutor, department and pre-requisites.

The new system will allow students to create a schedule, then select four courses. Each student chooses two others in case their first choices become full or are cancelled. No course can have more than 10 students. No course can have less than 3 students or it will be cancelled. This will be the same functionality as available to other internal users of the system.

When registration is complete, the registration system sends a message to the billing system to send out a bill to the student.

Tutors use the system to find which classes they are teaching and who the students are. The registrar will administer the system.

For a period at the beginning of the semester the student can change their schedule. Students must be allowed to access the system during this time to add or delete courses.

Note: If you have some experience with use cases try drawing a suitable diagram. If you're new to the field, see the following page for an example answer.

Exercise 1 – Example answer

- Actors: Student, Tutor, Billing System, Registrar
- Use Cases:
  - Student
    - Register for courses
  - Tutor
    - View teaching roster
  - Registrar
    - Maintain course information
    - Maintain student information
    - Maintain tutor information
    - Maintain curriculum
    - Generate prospectus
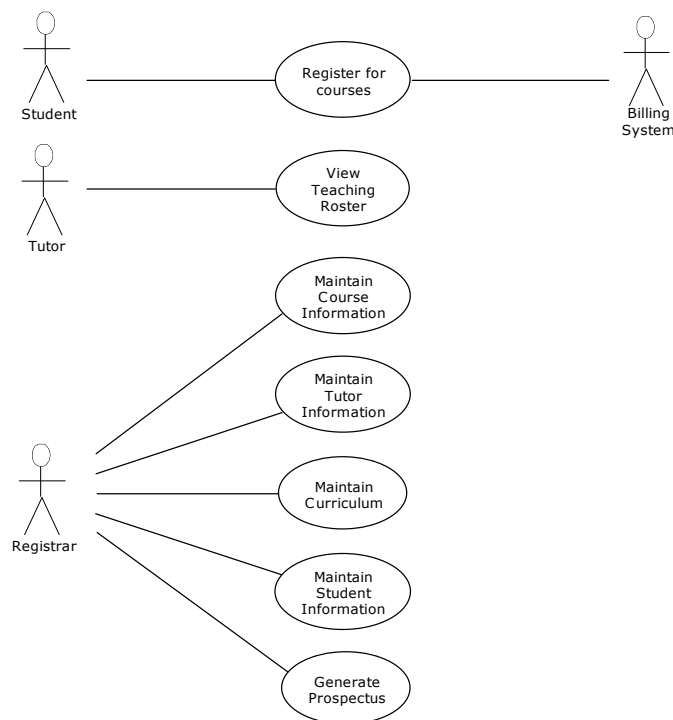
Exercise 2 – Textual use case (informal)

In the last exercise we analysed text to produce a use case diagram.

Now use the previous text and the previous example answer to write (informal) textual documentation for the use case initiated by the student.

Provide brief descriptions, primary and alternative flows of events.

Compile a list of possible scenarios which could emerge from further investigations with business users.

Note: There is an example answer on the following page.

Exercise 2 – Example answer.

Informal Style – Essential Use Case

Use Case: Register for Courses

Brief Description:
The student initiates the use case to create, read, update or delete a course for the coming semester.

Primary Flow of Events:

The student initiates the use case by entering a student number.
The system validates the student number and prompts the student for the preferred activity:

- Create a schedule
- Review the schedule
- Change the schedule by adding or deleting a course

The student indicates when finished. A schedule is printed.
The system sends details to the Billing System.

Alternative Flow of Events

If the student enters an invalid number, the student is denied access with an error message.

If a schedule already exists and a new one is created, the student will be informed and asked for another option.

Exercise 3 – Textual use case (formal)

In this final exercise we'll write a fully specified use case. Use the information from both exercises 1 and 2 and also include:

- Primary and secondary actors
- Pre-conditions
- Minimum guarantees
- Success guarantees

Note: This course registration system (like most systems) has a high number of alternate flows. Finding all of them becomes an exercise in logical thinking as you try to identify all the possible permutations and combinations of events. Don't be too disappointed if you don't find as many as are in the following example answer (congratulations if you find more!). As with all systems, it's only by working with them over time that you start to fully understand them.

Exercise 3 – Example answer.

Formal Style – Fully Specified Use Case

Use Case: Register for Courses
ID: UC-101

Brief Description:
This use case allows the student to register for courses by creating, viewing, modifying or deleting a schedule for a specified semester. Pertinent billing information is sent to the Billing System.

Primary Actor: Student
Secondary Actors: Billing System

Pre-Conditions:

1. Registrations for the Semester are open to Students

Minimum Guarantees:

1. Either a Schedule has been created/updated for a Student or the failed Validation Rule(s) displayed.

Success Guarantees:

1. A Schedule has been created/updated for a Student.

Primary Flow of Events:

1. The **Student** initiates the use case by entering a student number and password.
2. The **System** prompts the student for one of the following options:
   - Create a schedule
   - Review a schedule
   - Change a schedule (to add or delete a course offering)
3. The **Student** selects an option, completes the task and indicates when finished.
4. The **System** saves any changes made, sends billing details to the Billing System and prints the schedule.
5. The use case ends.

Alternative Flow of Events:

1a: Invalid Student Details Entered
   1. The **System** denies access and displays an error message.
   2. The use case resumes at step 1 (of Primary flow).

3a: Student Selects to Create a Schedule
   1. The **System** checks that the Student does not already have a Schedule for the upcoming semester.
   2. The **Student** selects 4 primary course offerings, 2 alternative course offerings and submits their selections.
   3. The **System** checks that the prerequisites are satisfied and adds the Student to the course offerings.
   4. The **System** generates charges associated with the selections made.
   5. The use case resumes at step 4 (of Primary flow).

3a.1a: Student has an existing Schedule

1. The **System** displays an error stating the Student already has an existing Schedule and cannot create a new one.
2. The use case resumes at step 2 (of Primary flow).

3a.3a: Prerequisites Not Satisfied
1. The **System** displays an error stating the Student does not have the required prerequisites for the selected course offering.
2. The use case resumes at step 2 (of Alternative flow 3a).

3b: Student Selects to Review a Schedule
1. The **Student** requests to view information for a given semester.
2. The **System** displays details of all the course offerings the Student is enrolled in including:
   - Course name and number
   - Timetable details
   - Location details
   - Charge details
3. The use case resumes at step 4 (of Primary flow).

3c: Student Selects to Change a Schedule – Delete a Course
1. The **Student** selects a course to delete from their schedule.
2. The **System** checks that the final date for changes has not been exceeded.
3. The **System:**
   - Removes the Student from the course offering
   - Removes the course offering from the Students schedule
   - Makes charging adjustments
4. The use case resumes at step 4 (of Primary flow).

3c.2a: Final Date Exceeded
1. The **System** displays an error stating the Student cannot delete the course offering as the final date for changes has been exceeded.
2. The use case resumes at step 1 (of Alternative flow 3c).

3d: Student Selects to Change a Schedule – Add a Course
1. The **Student** selects a course offering to add to their schedule.
2. The **System** checks that the final date for changes has not been exceeded.
3. The **System** checks that the prerequisites are satisfied and adds the Student to the course offering if it is open.
4. The **System** generates a charge associated with the new course offering added.
5. The use case resumes at step 4 (of Primary flow).

3d.2a: Final Date Exceeded
1. The **System** displays an error stating the Student cannot add the course offering as the final date for changes has been exceeded.
2. The use case resumes at step 1 (of Alternative flow 3d).

3d.3a: Prerequisites Not Satisfied
1. The **System** displays an error stating the Student does not have the required prerequisites for the selected course offering.
2. The use case resumes at step 1 (of Alternative flow 3d).

3d.3b: Course Offering Full
1. The **System** displays an error stating the selected course offering is full.
2. The use case resumes at step 1 (of Alternative flow 3d).

Related Information:
Refer to UI Specification xxx for the user interface associated with this use case.

Priority: High Status: Draft   Author: ………

----------------Last Modification Date: 01 Aug2007------------------

As you can see, producing formal use case documentation requires clear, logical thinking plus a numbering convention which allows you to track and cross-reference primary and alternate flows.

Summary

So how do we use these 3 different types of use case? For business users and stakeholders, a use case diagram together with an informal, textual use case will, in most circumstances, be sufficient to convey the essential business functions of the system. The diagram allows us to communicate with pictures whilst the primary and alternate flows allows us to summarise business functionality and key business rules.

With a use case diagram and an informal use case we're describing *what* happens in the system (the business requirements). The formal use case on the other hand is starting to bridge the gap between *what* the system does and *how* it does it. The formal use case (whilst still useful to business users needing to understand a greater level of detail) provides information which is essential to the design phase of the project.