

CS 2000 – Systems Analysis and Design
Final Exam – Fall 2005

Introduction:

One of the goals of this course was to expose you to the Unified Process and demonstrate how it can be effectively used to transform fuzzy features expressed in a vision statement into working application code, while at the same time minimizing risk and providing for frequent user feedback. A vital component of the UP is use cases, which is used as the primary vehicle for capturing the systems functional requirements. Numerous use cases were presented for class discussion and if you recall, some of them were well written, while others not. Most importantly, we have learned that if the project's inputs are bad (in this case its requirements), then there is a high probability that its outputs (i.e. what the customer will be paying for) will be just as bad, or at minimum, its outputs will be unpredictable.

What follows is an email by Scott Ambler, a noted author and proponent of Agile methods. The author makes two observations about use cases: 1) when used properly, they can be an effective form of requirements and 2) from an industry perspective; they may have a negative impact on overall productivity. He then goes on to give two examples of the same use case: one of them "good enough" for the agile developer and the other a "heavyweight" alternative.

Assignment:

One of the most important threads that ran throughout the semester is that as a Systems Analyst, your fundamental goal is to communicate effectively with people, not computers. Your assignment is to read the email and answer the following questions:

1. Which version of the use case do you feel communicates more effectively¹? In your response, you might want to consider who the audience of a use case would be, and what each member of this audience is looking for. You may also want to consider yourself. That is, as an author of a use case, which version would you want to present to the project community. Finally, you may also want to consider the artifacts that are used to support a use case, such as a data definition document, etc.
2. Do you feel the author is making a convincing point? Taking into consideration the techniques and guidelines we learned about use cases, would you be influenced any differently by the author's opinions? If not, why not and if yes, how?
3. What is your overall impression of the authors writing style? Do you think he is presenting a fair and balanced view? Knowing what we learned about use cases, do you think you were treated as intelligent person? Depending on your answer, who do you think this author's intended audience is?
4. Do you think Mr. Ambler is a fair and balanced representative of Agile methods? You may want to search the web for more information on agile methods. A good place to start is www.agilemanifesto.org. Do you think the UP, as applied like we learned in class is an Agile method ?

Your opinions on the subject matter and/or the author will not influence your grade. What will influence your grade is how effectively you are able to express your opinions. I am looking for clear, concise, well-formed and justified responses. There is no set length of your paper, but a good rule of thumb is no more than one page per question. You can of course go over this, but the extra pages may or may not add any value to your grade.

Extra Credit (10 points; requires both questions to be answered):

1. List the things you learned in this class. From this list, which items do you think will be most important to you as a professional software developer? Which ones were not important and why do you think they will not be important? Are there any lessons learned that you can apply in general, i.e. not specific to software development?

¹ You also have the option of giving me your own version of the use case (attached as an appendix).

2. Comment on your experience on working with your team. What did you like about it and what didn't you? If you were to work on another team assignment, what might you do continue doing and what might you want to do differently? If you feel you need to sharpen your interaction skills, how might you want to do that?

Use Cases of Mass Destruction

By Scott W. Ambler

Use cases are the primary requirements artifact for several software development methodologies -- including ICONIX, the Rational Unified Process (RUP) and the Enterprise Unified Process (EUP) -- and for many developers they're synonymous with the concept of requirements. It's high time someone examined the effectiveness of use cases; therefore, I have two fundamental observations to share with you:

1. Use cases, when used properly, can be an effective form of requirements.
2. From an industry perspective, use cases may have a negative impact on overall productivity.

First and foremost, use cases can be quite valuable. I've worked on many projects in which use cases were vital tools for discovering usage-based requirements, and I have no doubt that most of you have been on projects that benefited from the application of use cases had similar experiences. There are many helpful books and Web pages on the subject, and a good starting point is Alistair Cockburn's site (<http://click.sd.email-publisher.com/maabmxmaaZVAba9GOfCb/>).

Unfortunately, you need only to spend some time on modeling- related mailing lists to discover that many organizations are struggling with use cases: a challenge that's relatively easy to overcome. A more insidious problem is process-related -- the bureaucrats often want you to make your use cases more complex and wordy than they need to be. For example, I've included two examples following this article (scroll down for details) -- Use Case A: Informal Version presents an agile use case, whereas Use Case B: Narrative Style presents a much wordier, more traditional version of a use case. See the difference? Imagine how easy it would be to write the agile version -- and then compare it to the traditional version. Is the greater investment of resources worth it? I'm not so sure.

Although the agile alternative achieves the same fundamental goal as its heavyweight counterpart, bureaucrats will rarely allow you to go lean in this way. Not realizing that developers are intelligent beings who don't require extensive documentation, they insist that you write "nearly perfect" use cases to ensure that you understand the requirements. The bureaucrats also want to review the use cases to ensure that they're correct, comprehensive and compliant with the bureaucrats' onerous standards. These reviews also justify the dubious necessity of additional bureaucrats nosing into your quality assurance group -- after all, paper doesn't push itself.

I believe that this process is the crux of the use-case problem. In many organizations, "well-intentioned" bureaucrats have turned a potentially simple and effective model into an onerous, unnecessarily complex rigmarole. It's simple to teach developers to write agile use cases like the informal version of "Enroll in Seminar" (Use Case A), but it's not so easy to teach them how to write use cases in the formal narrative style presented in Use Case B.

Do agile use cases really work? Yes, for the vast majority of projects, if the team is allowed to succeed. Agile developers understand that models don't need to be perfect; just barely good enough. Use Case A is just barely good enough, while Use Case B is bureaucratic overkill. Is your primary goal to build working software that meets the needs of your stakeholders -- or to justify the bureaucrats' existence?

Another common use-case challenge is the literal acceptance of "use case-driven" marketing rhetoric. Use cases are one of many modeling techniques: They're very good for exploring usage requirements, but not so good for exploring user-interface or data requirements, business rules, or constraints (to name just a few examples). Instead of adopting several modeling techniques and using each when appropriate, some people

assume that use cases must include all requirements. This approach produces large use cases that are difficult to understand and manage, decreasing your development flexibility.

Use-case modeling specialists are also part of the problem: When someone's job is to write use cases, chances are good that they'll write use cases -- whether you need them or not. They'll also write overly complex use cases, whether you need them or not, because they need to look busy. They'll actively lobby to ensure that use cases are a critical part of anything you do, whether it's appropriate or not.

My advice? Chill out and understand that use cases don't need to be perfect and that they're only one of several modeling artifacts. You should also promote the concept of generalizing specialists within your organization: People who have one or two specialties and a general understanding of software development are better able to understand the role of use cases (and any other development artifact, for that matter) in your project, and accurately judge how much effort should be invested in them. Specialists are too narrowly focused to make this type of decision.

It's possible to employ use cases effectively, but you must choose to do so. I've seen several projects in which use cases have added value, but many others were complete disasters. I don't have any hard figures, but my gut tells me that use cases have had a negative impact on our overall productivity across the IT industry. I see too many people arguing about formatting issues, content issues, traceability issues and so on, and suspect that the "use case thrash" more than outweighs the benefits we've achieved to date; in fact, someone in the research community should investigate this issue.

USE CASE A

Informal version of the "Enroll in Seminar" system use case.

Name: Enroll in Seminar

Identifier: UC 17

Basic Course of Action:

- Student inputs her name and student number
- System verifies the student is eligible to enroll in seminars. If not eligible, student informed and use case ends.
- System displays list of available seminars.
- Student chooses a seminar or decides not to enroll at all.
- System validates the student is eligible to enroll in the chosen seminar. If not eligible, student is asked to choose another.
- System validates the seminar fits student's schedule.
- System calculates and displays fees.
- Student verifies the cost and indicates she wants to enroll or not.
- System enrolls the student in the seminar and bills her for it.
- The system prints enrollment receipt.

USE CASE B

"Enroll in Seminar" written in formal narrative style.

Name: Enroll in Seminar

Identifier: UC 17

Description:

Enroll an existing student in a seminar for which she is eligible.

Preconditions:

The Student is registered at the University.

Postconditions:

The Student will be enrolled in the course she wants if she is eligible and room is available.

Basic Course of Action:

1. The use case begins when a student wants to enroll in a seminar.
2. The student inputs her name and student number into the system via UI23 Security Login Screen.
3. The system verifies the student is eligible to enroll in seminars at the university according to business rule BR129 Determine Eligibility to Enroll. [Alt Course A]
4. The system displays UI32 Seminar Selection Screen, which indicates the list of available seminars.
5. The student indicates the seminar in which she wants to enroll. [Alt Course B: The Student Decides Not to Enroll]
6. The system validates the student is eligible to enroll in the seminar according to the business rule BR130 Determine Student Eligibility to Enroll in a Seminar. [Alt Course C]
7. The system validates the seminar fits into the existing schedule of the student according to the business rule BR143 Validate Student Seminar Schedule.
8. The system calculates the fees for the seminar based on the fee published in the course catalog, applicable student fees and applicable taxes. Apply business rules BR 180 Calculate Student Fees and BR45 Calculate Taxes for Seminar.
9. The system displays the fees via UI33 Display Seminar Fees Screen.
10. The system asks the student if she still wants to enroll in the seminar.
11. The student indicates she wants to enroll in the seminar.
12. The system enrolls the student in the seminar. 13. The system informs the student the enrollment was successful via UI88 Seminar Enrollment Summary Screen.
13. The system bills the student for the seminar, according to business rule BR100 Bill Student for Seminar.
14. The system asks the student if she wants a printed statement of the enrollment.
15. The student indicates she wants a printed statement.
16. The system prints the enrollment statement UI89 Enrollment Summary Report.
17. The use case ends when the student takes the printed statement.

Alternate Course A: The Student Is Not Eligible to Enroll in Seminars.

- A.3. The registrar determines the student is not eligible to enroll in seminars.
- A.4. The registrar informs the student she is not eligible to enroll.
- A.5. The use case ends.

Alternate Course B: The Student Decides Not to Enroll in an Available Seminar

- B.5. The student views the list of seminars and does not see one in which she wants to enroll.
- B.6. The use case ends.

Alternate Course C: The Student Does Not Have the Prerequisites

- C.6. The registrar determines the student is not eligible to enroll in the seminar she chose.
- C.7. The registrar informs the student she does not have the prerequisites.
- C.8. The registrar informs the student of the prerequisite she needs.
- C.9. The use case continues at Step 4 in the basic course of action.

Note: Use Case A and B were modified from the forthcoming book, The Object Primer 3rd Edition: Agile Model Driven Development with UML 2. For more information, go to <http://click.sd.email.publisher.com/maabmxmaaZVAca9GOfCb/>