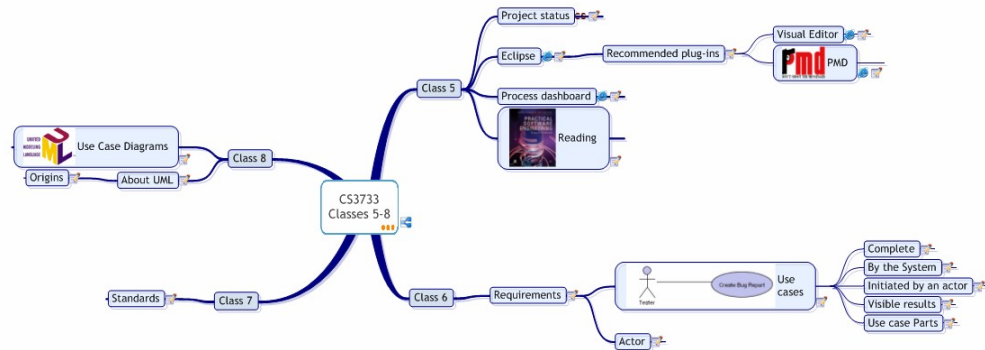
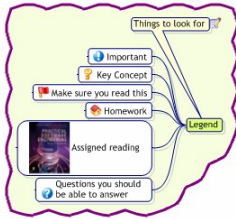


CS3733 Classes 5-8



See document: [CS3733 SoftwareEngineering.mmap](#)

Class 5

Project status

See document: [ProjectStatusReport.htm](#)

Each week on Friday, you will be expected to post your project status on your group's myWPI pages. There is a discussion forum for Project Status where you will post the information.

Your project status must include:

- What did you plan to do this week
- What did you do
- What didn't get done, why, and what are you going to do about it
- What do you plan to do for next wee

Use the template provided by clicking on the link.

Eclipse

See document: www.eclipse.org

Eclipse is the software development environment you will use for your project. The Eclipse home page can be found at the link associated with this topic.

You should use Eclipse 3.0, or 3.1 for your work.

Recommended plug-ins

While you don't need to have anything more than the basic Eclipse Java development tools, there are a couple of plug-ins that will make life easier for you.

Visual Editor

See document: [vep](#)

The visual editor will help you create GUI elements quickly and consistently. It requires a couple of other Eclipse plug-ins, like GEF and MEF. This is identified on the download link associated with this topic.

PMD

See document: pmd.sourceforge.net

PMD is a great plug-in for helping you produce high-quality code by performing static analyses of your source files.

Process dashboard

See document: processdash.sourceforge.net

Process dashboard is another open source tool that can help you track your own productivity and thereby make better estimates. While this is not a required tool for this course, I highly recommend that you give it a try. You can download it from the associated link.

Reading

Sections 7.3 and 7.4

Class 6

Requirements

Requirements are more than just customer desires, or needs. They are specific, well-defined statements of what your system must do.

While you will not, in most cases, be able to obtain a complete requirements specification as soon as you start your project, you need to try to get most of the *big* requirements that reflects the most important characteristics of your system, and then continually gather more requirements, and manage changing requirements as you go.

Use cases

Use cases have become a popular method for describing *functional requirements*. Functional requirements are those requirements that define the functionality that must be implemented in the system.

Use cases were invented by Ivar Jacobson and a key part of his Objectory process. They have become a standard part of the requirements management toolkit.

So, what is a use case? There are many similar definitions for a use case. The one we will use for this class is:

A use case is a complete set of actions performed by the system, initiated by an actor, that provides visible results (to the actor).

Complete

The first thing to notice is that a use case is complete. That means that it describes, in enough detail, a complete set of related actions. A use case does not map 1-to-1 to a feature, but it should implement parts of many features, a part of a single feature, or a complete feature. However, just implementing a feature like, "the ability to deposit salary directly into the

employee's bank account," is not a use case. That feature might be part of a larger use case, such as "pay an employee."

By the System

The purpose of a use case is to describe the actions of the system. In doing so, it must describe the interactions with the outside world, but not the details of what goes on outside of the system.

As you refine the use cases, you will add more and more detail to the use case until you have described it in enough detail that the software that implements the use case may be written and verified.

Initiated by an actor

Use cases describe the purpose of the system. They just don't randomly occur without some trigger. That trigger is the actor, or something *outside of the system* under consideration. There are times when you might have to fabricate an actor, such as "time" that causes a use case to occur on a schedule, but in general, the actor is a person or other system external to the system you are describing.

Visible results

Any use case must provide value to someone. If it doesn't, what is its purpose? The value, in terms of visible, or tangible results may be value to the actor that initiates the use case, or to some other external entity.

Use case Parts

Besides the UML diagram that consists of stick figures for the actors, ovals for the use case, and a solid line (with optional arrow) showing the association between the two, use cases are described textually. There are typically several possible parts in a use case description. Some of the more common ones are:

- Name. Every use case has a name that should be in the form <verb> <noun>. It should be an active name like "Withdraw money."
- Actor(s). The actors that participate in the use case are identified in this section.
- Brief description. The brief description describes the purpose of the use case in a couple of sentences. It should indicate the value that the use case delivers to the actor. An example is "This use case lets a bank account owner withdraw money from an ATM machine."
- Preconditions. If there are any, the preconditions describe a state that must exist *before the use case can start*.
- Postconditions. If there are any specified, the postconditions describe the state *after the use case completes successfully*.
- Basic flow of events. This is also called the *happy day* flow, or *sunny day* flow. It describes the most common successful case.
- Alternate flows of events. Also called *subflows*. These describe alternate paths that deviate from the basic flow of events. These do not have to be complete descriptions of the use case actions. They only have to describe the difference for the specific subflow.
- Exception flows. Some use case authors will divide subflows into those that are normal, successful alternate actions and those that occur when something goes wrong. These are called *exception* or *error* flows.

Actor

An actor is something outside of the system that interacts with the system. It does not have to be a person. It can be a hardware device or another software system.

Class 7

Standards

This class is a working session where the class will work on identifying the standards needed to continue progress on the term project.

Class 8

Use Case Diagrams

The use case diagram is the easiest diagram to create in UML. By itself, the UML diagram does not convey a lot of detailed information. It must be supported by text. However, it can be very useful to provide an overview of the system you are going to build. It also helps you bound the system. If you draw a box around all of the use cases, with the actors outside of the box, the system you will build is inside the box and the actors indicate interface points with the system.

In UML, use cases are represented by an oval with the use case name written inside of the oval. Actors are represented as stick figures, and they are connected with a solid line, called an *association*. If you want to show a direction of the association, you use an open arrowhead at the appropriate end.

About UML

UML has become the "lingua franca" for describing software designs. It is a visual modeling language that has a well-defined set of semantics. Today there are tools that support UML, more than just drawing the diagrams, but generating code from the diagrams and keeping the code and diagrams in synchronization. UML is also extensible and there have been several extensions for particular domains, such as real-time software systems.

Origins

UML is a standard that has been adopted by the Object Management Group (OMG). It was originally a combination of several existing notations, but most notably the three developed by Grady Booch, Ivar Jacobson, and James Rumbaugh (the three amigos).

Jacobson was the inventor of the Objectory process and notation. Booch had developed the Booch design method and notation. Rumbaugh is the creator of the OMT analysis and process notation.