Case Diagram Explanation

Today’s lesson will not have a PowerPoint. Instead, I will just give you the information on case diagrams. I will go through some of the importance of what a case diagram is, why it is used, and some guidelines for using it.

So, to start, let’s bring up this website called draw.io. This is a site that can be used to create case diagrams and allow you and others to work on it together if needed. I like this site because it allows multiple user to make changes at the same time, and everyone sees the changes immediately. For CIT260, we use this and shared it through Google drive. There are other programs, such as Visual Paradigm and Lucid chart, that you can use to create case diagrams. It’s really up to you on which one to use. Just use one that you are comfortable with.

Now let’s talk about what is a case diagram. Case diagrams are used by the Unified Modeling Language, or UML, as the primary form of showing system or software requirements for developing software. It is really just a sketch of WHAT you want the software to do, and not HOW to do it. It should only summarize some of the relationships between cases, actors, and systems. The case diagram will not show the order in which the steps are performed or give details as to how to accomplish those steps. This technique was first developed in 1986 by Ivar Jacobson, and popularized in 1992 when he wrote a book about it.

So why do we use this? One of the reasons we use this, is because it helps us design a system from the end user’s perspective. This makes it extremely valuable, because we are designing our software for the end user, and with a diagram, we can represent it to them in non-technical terms. That helps the developer and end user to be on the same page. Now you can both specify and capture the context and requirements of a system.

Now let’s start using this software to create a case diagram. I already have one, so I will open it and walk you through the parts and explain what they are. This particular diagram is to show a management system for wireless devices used by an organization.

The first thing I want to show you is this large rectangle. This considered the boundary of the system. Everything you want the system to do, should be encased in this rectangle. An effective case diagram will not have more than 20 use cases, some resources say 10 should suffice. If you have a complex system and need to use more than, consider changing each module to be its own system.

Next, we have the use cases. These are represented by these ROUNDED rectangles. As you can see, they only contain a brief description of what it will need to do. It does not show any specifics how to do it. What I want my system to do is get inventory inputted in to it, allow it to change the status of said inventory, and then provide reports that show me what equipment I have, where my equipment is, and what my cost are.

Now here are the actors. This is going to be who uses this system. Here I’ve identified a supply manager, a communications officer, a budget officer, and the director. Their responsibility is to interact with the system and provide input that triggers the USE cases. While all actors must be linked to a use case, not all use cases need to be linked to an actor. Additionally, not all actors need to be able to access all USE cases.

Lastly, let’s talk about some of these communication links. The solid links indicate that an actor and a use case will interact. This can be done by messages to each other such as the director clicking run distribution report, and the system outputting the report to him or her.

The arrow lines indicate that the there is a relationship between two USE cases. On these top cases, you can see where I indicate that when creating or updating records, we will need to make sure that the data being entered is validated. The term INCLUDE and USES are used interchangeably in the resources I read, so I labeled as such. They both mean the same thing. An include relationship is depicted with a directed arrow having a dotted line. The tip of arrowhead points to the child use case and the parent use case connected at base of the arrow.

In the middle USE cases, I am showing that when an action is taken on the service status, then the record is automatically updated. However, as you can see, if the update includes a service change, then update record run extends to calculate cost. This will only happen if the change in the record is a service change. If it is any other type of change, then it will not run calculate cost. An extend link is depicted with a directed arrow. The tip of arrowhead points to the base USE case and the child USE case is connected at the base of the arrow.

Any questions? If not, then this is the end of my presentation.