Use Case Diagram Cheat Sheet

A use case diagram provides an external (black box) view of a process, that is, the entities that make it work. It is **STRUCTURAL** and as such contains no temporal information. You can not tell by looking at a use case diagram what happens when; that is provided either textually or with sequence or activity diagrams. A use case can be non-testable (e.g. “manage documents”) or testable (e.g.” delete document”). A non-testable use case is marked as ABSTRACT, and is tested by testing all of its included and extending concrete use cases.

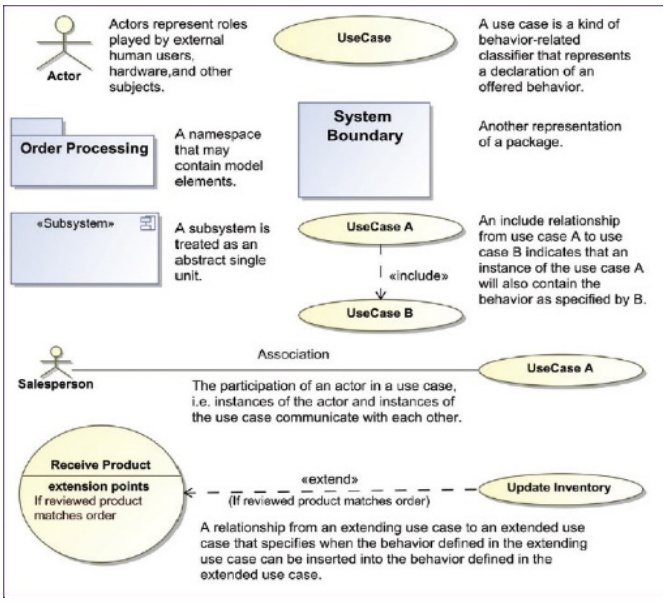


Figure 1 Use case symbols

To create a diagram in Magic Draw, first you have to create a new project. Thereafter, you can use the project as a file cabinet to hold all of your ASE 6001 Drawings. After bringing up Magic Draw, select CREATE NEW PROJECT:

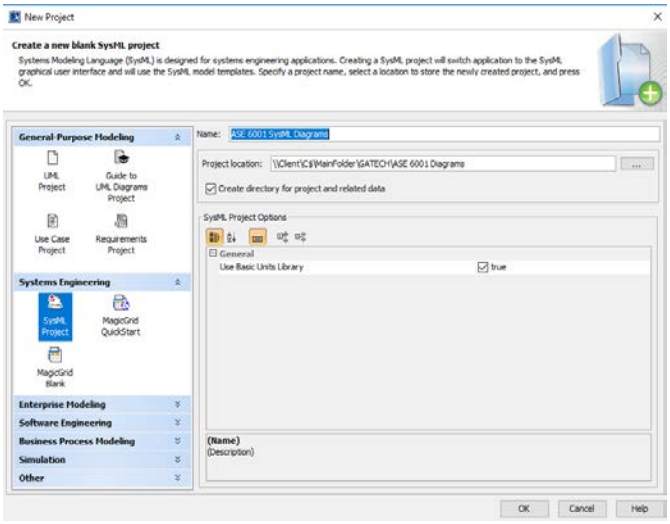


Figure 2 Creating a project to hold your diagrams

Once you have created the project (figure 2), you can reuse it to hold all your diagrams. Here we select diagrams>create new diagram>SysML use case diagram. Note that UML and SysML use case diagrams are slightly different. Then we left click on an item on the left tool bar, and left click again in the drawing pallette to drop it onto the drawing area. To make a use case abstract, right click on the use case, select Specification and then mark the use case as abstract (figure 3). One very common technique is to create an entity (here an abstract use case), then create a diagram under the symbol that holds all the details about the symbol. Right click on the symbol, select “create diagram” then select SysML Use case diagram.

Note that the diagram name will be the same as the entity that is its parent. Then go to browser, select the use case and drag it into the child diagram. You can now add all the lower level entities associated with this use case. Note also that you can drag any diagram onto another diagram to create a hyperlink.

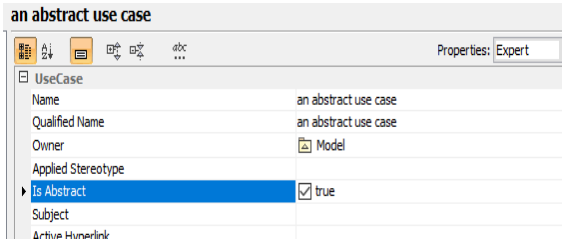
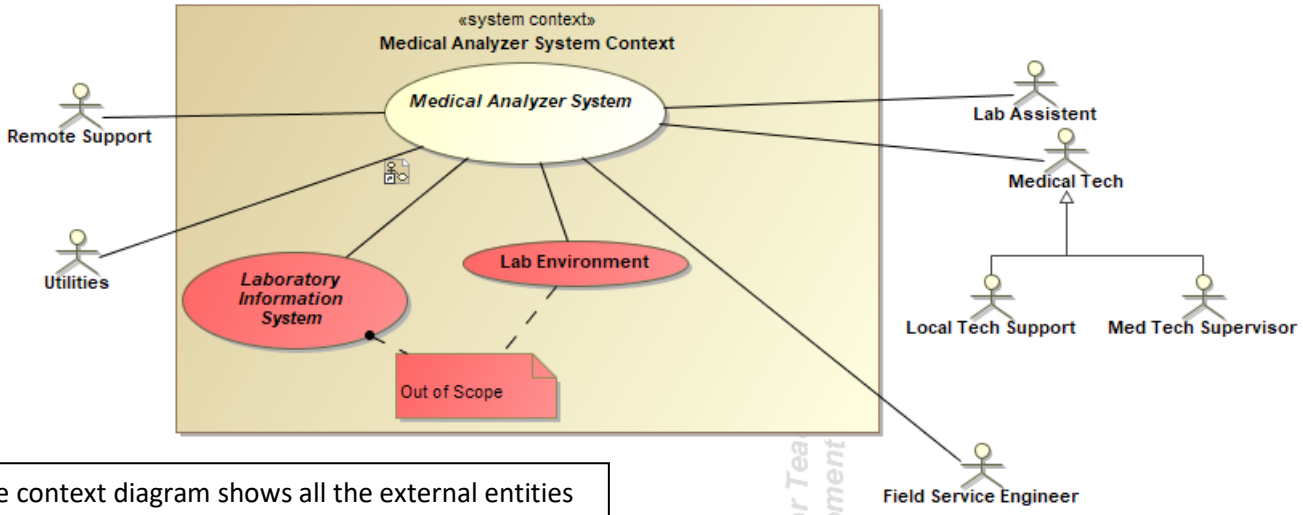
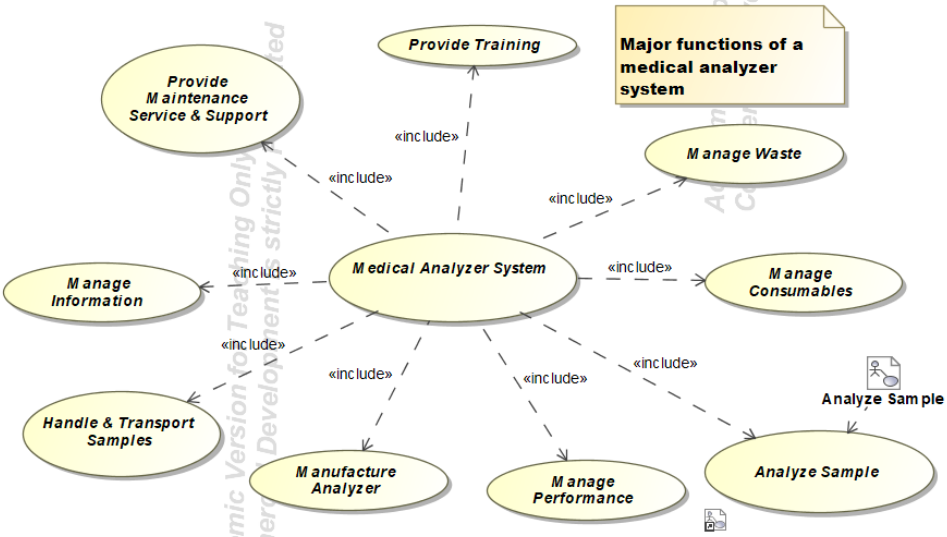


Figure 3 Marking a use case as abstract

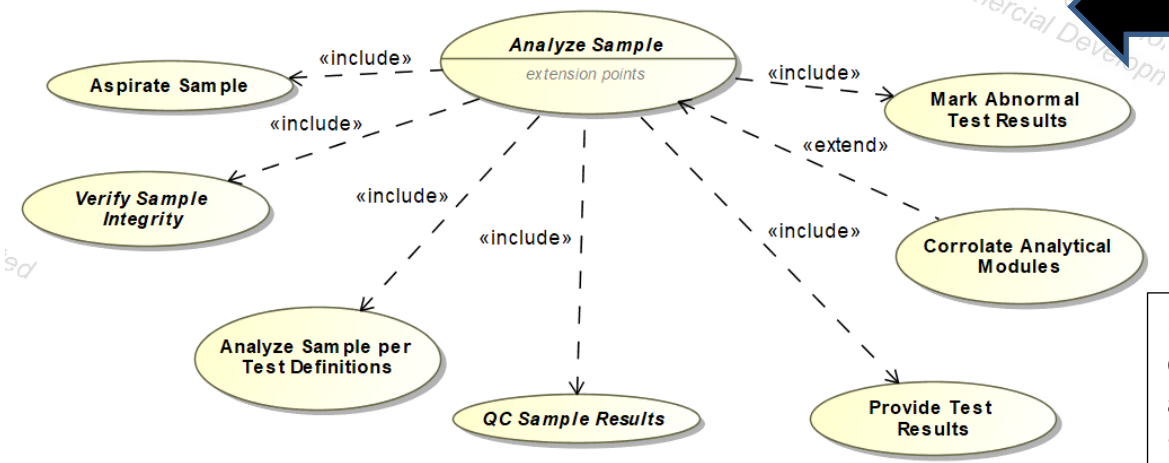
On the back side of this sheet is an example, starting with a context diagram and descending two levels.



The context diagram shows all the external entities that interact with the system. Note the use of color to mark functions out of scope



Here we are defining the high-level functions of the analyzer



Here we are decomposing the abstract function "analyze sample" into testable functions