How to create a use case diagram and specify all the use cases in the diagram?

<http://argouml-stats.tigris.org/documentation/manual-0.34/ch04s04.html>

**4.4.6.  Documentation**

ArgoUML has some simple documentation facilities associated with model elements on a diagram. In general these should be used only to record the location of material in documents that can be handled by a mainstream editor or word processor, not the actual documentation itself.

Documentation for a particular model element is recorded through the documentation tab in the details pane (the quadrant of the user screen at the bottom right).

In addition annotation may be added to diagrams using the text icon on the editing pane toolbar ( http://argouml-stats.tigris.org/documentation/manual-0.34/images/icons/text.gif).

The recommendation is that a use case diagram should use the documentation tab of actors to record information about the actor, or if the actor is complex to refer to a separate document that holds information about the actor.

The documentation tab of use cases should record the location of the use case specification. The information in a use case specification (for all but the simplest use cases) is too complex to be placed directly in the tab.

The project should also have a separate vision document and supplementary requirements specification. A text annotation on diagrams may be used to refer to these if the user finds this helpful.

**4.3.3.  The Use Case Specification**

Each use case must be documented to explain in detail the behavior it is specifying. ArgoUML assists in this area through the generation of graphic files for inclusion in this documentation. This document is known by different names in different processes: *use case specification*, *use case scenario* or even (confusingly) just *use case*.

A typical use case specification will include the following sections.

* *Name*. The name of the use case to which this relates.
* *Goal*. A one or two line summary of what this use case achieves *for its actors*.
* *Actors*. The actors involved in this use case, and any context regarding their involvement.

|  |  |
| --- | --- |
| [Note] | **Note** |
| This should not be a description of the actor. That should be associated with the actor on the use case diagram. |

* *Pre-condition*. These would be better named “pre-assumptions”, but the term used everywhere is pre-conditions. This is a statement of any simplifying assumptions we can make at the start of the use case.

In the ATM example we might make the assumption for the“Maintain Equipment” use case that an engineer is always available, and we do not need to worry about the case where a routine maintenance visit is missed.

|  |  |
| --- | --- |
| [Caution] | **Caution** |
| Avoid pre-conditions wherever possible. You need to be absolutely certain that the pre-condition holds under all possible circumstances. If not your system will be under specified and hence will fail when the pre-condition is not true. Alternatively, when you cannot be certain the pre-condition is always true, you will need to specify a second use case to handle the pre-condition being false. In the first case, pre-conditions are a source of problems, in the second a source of more work. |

* *Basic Flow*. The linear sequence of steps that describe the behavior of the use case in the “normal” scenario. Where a use case has a number of scenarios that could be normal, one is arbitrarily selected. Specifying the basic flow is described in more detail in[Section 4.3.3.1, “ Specifying the Basic Flow ”](http://argouml-stats.tigris.org/documentation/manual-0.34/ch04s03.html#s3.tut.basic_flow) below.
* *Alternate Flows*. A series of linear sequences describing each of the alternative behaviors to the basic flow. Specifying alternate flows is described in more detail in [Section 4.3.3.2, “ Specifying the Alternate Flows ”](http://argouml-stats.tigris.org/documentation/manual-0.34/ch04s03.html#s3.tut.alternate_flows).
* *Post-conditions*. These would be better named “post-assumptions”. This is a statement of any assumptions that we can make at the end of the use case. Most useful where the use case is one of a series of subsidiary use cases that are included in a main use case, where they can form the pre-conditions of the next use case to be included.

|  |  |
| --- | --- |
| [Caution] | **Caution** |
| Like pre-conditions, post-conditions are best avoided. They place a burden on the specification of the use case flows, to ensure that the post-condition always holds. They therefore are also a source of problems and extra work. |

* *Requirements*. In an ideal world the vision document, use case diagrams, use case specifications and supplementary requirements specification would form the requirements for a project.

For most market-led developments, where ownership of requirements is within the same business as the team who will do the development, this is now usually the case. The marketing department can learn use case based requirements capture and analysis to link to their customer facing activities.

However for external contract developments, customers may insist on a traditional “list of features” as the basis of the contract. Where this is the case, this section of the use case specification should link to the contract features that are covered by the use case.

This is often done through a third party tool that can link documents, providing automated checking of coverage, in which case this section is not needed, or may be generated automatically.

The final size of the use case specification will depend on the complexity of the use case. As a rule of thumb, most use cases take around 10-15 pages to specify, the bulk of which is alternate flows. If you are much larger than this, consider breaking the use case down. If you are much smaller consider whether the use case is addressing too small a chunk of behavior.