Levels of Documentation

Documentation is ordinarily found at two levels: High and low level. High level documentation is coarse-grained text that describes without a lot of detail. Low-level documentation consists of fine-grained descriptions with detailed text.

If a professional were to document a C++ class thoroughly, they would provide high-level descriptions of the members in the header (.h) file for consumption by application programmers who used the class and low-level descriptions of the function implementations in the .cpp file for developers for the purpose of upgrading or further developing the class.

Function Examples

In describing a linear search function on an array that returns the index of the search key *key* in an array *list* with capacity *cap* that can hold zero or more values, we would have:

High level: This function returns the location (index) of *key* in array *list*. If *key* is not found, the function returns -1.

Low level: This function traverses the array *list*, left-to-right via a for loop, applying the equality relational operator (==) between the search key and the element at the index indicated by the loop's control variable, until a match is found or the entire used portion of the array has been traversed. If a match was found, the location, given by the value of the loop control variable, is returned. Otherwise, -1 is returned to indicate failure.

Example: Software Design Document Guidelines

A project's high level design document should provide a *complete* (though still high level at this stage) specification of the design for the software that will be developed in the project. The high level design description should contain at a minimum the names of the classes that will be developed during the low level design and implementation stages, the important associations between those classes (for example, inheritance, ownership, reference counting, etc.), and any details of those classes that are firmly determined by the high level design stage. The high level design document should also describe the classes, associations, and any appropriate details, that will be involved in testing and evaluating the project according to the *evaluation plan* defined in the project's requirements document.

More importantly, a project's high level design document should provide a *narrative* describing how the design forces that were defined in the preceding requirements stage are resolved by that design specification, and in particular which design patterns were applied (and in what order) to generate the design specification from the design forces arising from the project's requirement specification. The high level design document should also describe the design patterns that were applied to arrive at the testing design, which may require further thought about the design forces that are implied by the statements in the evaluation plan.

Whereas the requirements document was based mainly on English sentences (e.g., requirements would contain words like "should", "will", "if", etc.), the design document should begin to move from English descriptions (which are of course still important in explaining the design narratives described above) toward more technical representations like class structure diagrams, interaction diagrams, etc. An important issue in using these more technical representations is that it must be clearly shown how the design emerged from the requirements

As with anticipating requirements, in practice it is sometimes difficult to anticipate all of the considerations that will come up during design and implementation up front, and so it is sometimes necessary and beneficial to revisit high level design during later stages. Beware, however, of letting the scope of the project design change (sometimes known as "scope creep") as that can put the timeline and quality of the project in jeopardy.

A low level design document should contain a listing of the declarations of all the classes, non-member-functions, and class member functions that will be defined during the implementation stage, along with the associations between those classes and any other details of those classes (such as member variables) that are firmly determined by the low level design stage. The low level design document should also describe the classes, function signatures, associations, and any other appropriate details, which will be involved in testing and evaluating the project according to the *evaluation plan* defined in the project's requirements document

Each project's low level design document should provide a *narrative* describing how the high level design is mapped into its detailed low-level design, which is just a step away from the implementation itself. This should be an English description of how the technical diagrams (and text descriptions) found in the high level design were converted into appropriate class and function declarations in the low level design. Be especially careful to explain how the class roles and their methods were combined in the low level design, and any changes made in combining and refining them.