CS 743 – Software Verification and Validation

Checklist for Informal Verification of Use Case Narratives

Assumption: The use case diagram is correct. No attempt will be made to verify the correctness of the diagram. The use case narratives are checked against the use case diagram.

Consistency Check

Check the following for every use case in the diagram:

- There is a corresponding use case in the narrative with the same name (Name of a use case is case sensitive; therefore 'addEntity', 'AddEntity' and 'addentity' are all different).
- The primary actor(s) mentioned in the use case narrative is (are) the same as connected with the use case in the diagram (check for spelling errors and case sensitivity as well).
- The secondary actor(s) mentioned in the use case narrative is (are) the same as connected with the use case in the diagram (check for spelling errors and case sensitivity as well).

Check the following for every use case narrative:

- The narrative has a unique use case ID within the document.
- All sections of the narrative (Use case name, purpose, …) are filled in. A field may have 'None' but cannot be empty.
- The name of every input parameter (if any) is consistency used (with no spelling errors) throughout the narrative.
- The name of every output parameter (if any) is consistency used (with no spelling errors) throughout the narrative.
- The name of every primary actor (if any) is consistency used (with no spelling errors) throughout the narrative.
- The name of every secondary actor (if any) is consistency used (with no spelling errors) throughout the narrative.
- If a use case refers to another use case in the diagram (through <<include>> or <<extend>> relationship), the name of the other use case is consistently used.
- Every parameter used inside the sections Precondition, Post-condition, Successful scenario and Exceptions is declared in the Input or Output parameters section.
- No parameter appears both in Input and Output parameters; it can be declared in only one place.

Some information can be classified as "inconsistent" or "missing". For example, if an input parameter is missing for a use case and the use case says "None" in the input parameters section, it can be stated as inconsistent because the input section has different information. However, it misses an input and so it can be reported as missing. Either of the classification is acceptable.

Completeness Check

This section requires a thorough understanding of the application domain. The verifier judges the completeness based on his/her view of the design which might be different from the design given.

Check the following for every use case narrative:

- The input and output parameters are appropriate. Some parameter(s) may not be needed or inappropriate (from the verifier's point of view).
- No input or output parameter is missing which seems to be important for this narrative (from the verifier's point of view).
- All expected constraints are specified in the Precondition section.
- The post-condition is appropriate.
- The successful scenario includes a complete description of the expected functionality of this use case.
- All expected constraints are specified in the Exceptions section.

Check whether there is any missing use case (functionality) relevant for this application.