Assignment 2: Requirements analysis

Requirements Engineering 2011

Description:

The purpose of this assignment is for you to get deeper understanding of what is a good or a bad requirement through analysis of a fellow students solution to assignment 1. To aid you in the analysis this document contains a checklist of aspects that should be considered for a good or a bad requirement or requirements specification. Your task is to use the checklist to find what requirements are good and which are bad in your fellow students solution and motivate why the requirements you chose are good or bad and why the entire set of requirements is good or bad. The solutions are in the continuation of this description referred to as mini-Software Requirements Specifications (SRS)

In particular you should answer the three questions,

1. What was good/bad with the requirements in the mini-SRS, and why, according to the checklist?
2. What was good/bad with the mini-SRS as a whole?
3. What was good/bad in the mini-SRS according to your own opinion?

Note that the checklist does not fully describe the quality aspects, only gives a brief summary, and that it is expected of you as a student to find further information about the quality aspects yourself. Further information can be found in the course book, course slides, academic articles and other material online. Make sure to add references to all the material you use in your analysis, especially material you find outside the course literature!

Important information:

- The submission shall be done through the FIRE system.
- The deadline is 9/9 at 18.00. No late submissions will be accepted
- The assignment is individual
- The assignment should be written as a technical paper using either Latex or the Word template available on the webpage.
- The report should be maximum 3 pages long.
- Use your anonymous code, NO names!

Checklist for individual Requirements:

In order for a requirement to have high quality (note difference from quality requirements!) it should be,

1. Correct: Valid from a customer's point of view, e.g. requirements that the customer wants/needs.
2. Achievable/Feasible: The requirement is implementable given current technology, timeframe, etc. (For this assignment we can assume the technology required to build a Terminator is current state-of-practice and available to us)
3. Annotated: A requirement should be traceable to where it originated, i.e. who or what required it and why.

4. Traceable: A requirement should include information that makes it possible to track it over different versions of the system, how the requirement is connected to code, tests, etc.
5. Unambiguous: The requirement should only have ONE interpretation.
6. Verifiable: A requirement is verifiable if it can be tested that it is fulfilled in a cost effective and finite way.

Checklist for SRS:

In order for a SRS to have high quality it should be,

1. Complete: All the functionality needs of the customer should be included in the SRS, i.e. the SRS should define a complete system.
2. Correct: All requirements in the SRS model functionality that the software should meet.
3. Unambiguous: All requirements in the SRS only have one interpretation
4. Consistent: The requirement does not conflict with another requirement or a set of requirements.
5. Ranked for Importance and/or Stability: All requirements have identifiers that rank them according to importance.
6. Verifiable: All requirements in the SRS are verifiable.
7. Modifiable: Changes to the requirements should be easy to make.
8. Traceable: All the requirements should be forwards and backwards traceable.

Questions relevant for the analysis can be found in APPENDIX A. Note that the APPENDIX is supposed to serve as a guide and not as something that you need to fully incorporate into your solution.

APPENDIX A:

Organization

- Does the Requirements Document follow the template of Requirements Document provided in the CS 389 class? *Each section must be present.*
- Is the implementation priority of each requirement included (*with descriptions of priorities provided*)?
- Are all internal cross-references to other requirements correct?

Completeness

- Have algorithms intrinsic to the functional requirements been defined? Are all requirements written at a consistent and appropriate level of detail?
- Does the SRS include all of the known customer or system needs? *Are all the tasks the user want being specified?*
- Is any necessary information missing from a requirement? If so, is it identified as TBD? Is the expected behavior documented for all anticipated error conditions?

Correctness

- Do any requirements conflict with or duplicate other requirements?
- Is each requirement written in clear, concise, unambiguous language?
- Is each requirement verifiable by testing, demonstration, review, or analysis?
- Is each requirement in scope for the project?
- Is each requirement free from content and grammatical errors?

- Can all of the requirements be implemented within known constraints?
- Are any specified error messages unique and meaningful?

Quality Attributes (Non functional requirements)

- Are all performance objectives properly specified? Are all security and safety considerations properly specified?
- Are other pertinent quality attribute goals explicitly documented and quantified, with the acceptable tradeoffs specified?

Traceability

- Are requirement dependencies being identified?
- Is each requirement uniquely and correctly identified?
- Is each requirement traceable to its source?

Special Issues

- Do the requirements provide an adequate basis for design and testing?
- Are all requirements actually requirements, not design or implementation solutions?
- Have internationalization issues been adequately addressed?