Software Engineering Group 11

SE.QA.05A.S

Design Specification Standards Summary

Author: Theo Goree (tcg2)

Config Ref: SE.QA.05A.S

Date: 2014-10-18

Version: 1.0

Status: Release

Department of Computer Science

Aberystwyth University

Aberystwyth

Ceredigion

SY23 3DB

CONTENTS

SE.QA.05A – Design Specification Standards

1. Introduction

1.1. Purpose of Document

The Purpose of this document is to provide an outline and summary of the content included in SE.QA.05A [1] for group members to use as a basic reference when creating the design specification.

1.2. Scope

The Summary aims to act as a basic guide and a quick referencing material for all members of the group and is to be used in conjunction with the original SE.QA.05A [1] provided.

1.3. Objectives

SE.QA.05A [1] identifies the format of, and information which must be supplied in the design specification. It covers:-

- Structure of the design specification
- Creation of the design specification

2. Relevant QA Documents
   - The design spec must be produced in accordance with SE.QA.01[2]
   - Must be produced as a part of a task in SE.QA.02[3]
   - And be maintained in a CMS according to SE.QA.08[4]
   - The general layout and form must adhere to SE.QA.03[5]
3. Outline Structure
   - 1. Intro – defined in SE.QA.03[5]
   - 2. Decomposition Description
     - 2.1 Programs in system
     - 2.2 Significant classes in each program
       - 2.2.1 significant classes in program 1
       - 2.2.2 significant classes in program 2
       - 2.2.n significant classes in program n
     - 2.3 Modules shared between programs
     - 2.4 Mapping from requirement to classes
   - 3. Dependency Description
     - 3.1 Component Diagrams
       - 3.1.1 component diagram for program 1
       - 3.1.2 component diagram for program 2
     - 3.2 Inheritance Relationships
   - 4. Interface Description
     - 4.1 class 1 interface specification
     - 4.2 class 2 interface specification
   - 5. Detailed Design
     - 5.1 Sequence Diagrams
     - 5.2 Significant algorithms
     - 5.3 Significant data structures
   - 6. References – Defined in SE.QA.03[5]

4. Decomposition Description
    - The decomposition description records the division of software into programs and then the modules which make them programs.
    - Describes the structure and purpose of each function of each program and significant module.
    - Gives an overview and justification for design.

    4.1 – Programs in the System

    - If only one program in design then simplify headings and brief summary of what it does
    - If more than one program should explain relationship between them

    4.2 – Significant Classes

    - Each program should have main classes named and shortly explained
    - Statement of purpose should be a plain English description a few lines long
    - Spec for each module not given here
    - If a set of classes shared between several programs then should be made into a Package and a section describing each Package

    4.3 – Table Mapping Requirements onto Classes

    - To understand the implications of changes in requirements or design
    - To check all requirements met
    - Produce a table mapping functional requirements onto the classes which contribute to requirements for e.g.

Requirement	Classes providing requirement
FR1	HeatingClass, DiaryClass, DiaryObject
FR2	LogClass, DiaryClass, DiaryObject

5. Dependency Description
    - Specifies relationship and dependencies between modules
    - Helps reader see how parts of system fit together as described in the Decomposition Description
    - Describes general architecture such as
        - Model-view-controller
        - UML provides a formalism called Component Diagrams
    - These are appropriate ways of describing relationships between modules in java
    - The spec should contain a subsection containing a component diagram for each program
    - Should show method links between modules
    - Can show compilation dependencies between modules
    - Can show inheritance dependencies between classes

6. Interface Description
    - Should provide everything designers, programmers and testers need to know to use the facilities of the module.
    - Outline spec of each class or interface in the system which should include
        - Name and type of class or interface (private/public, abstract for a class)
        - classes or interfaces which it extends and why
        - public methods implemented by the class
        - parameter names and types for each method should be given and a short summary of the method
    - simplest way of specifying each task may be a java outline
    - tools are available for generating code automatically from UML class descriptions
    - Java coding standards should be used as in SE.QA.09[6] and should compile into a system
    - Won't do a lot as only contains outlines of classes
    - Where possible a java interface should be specified rather than class, together with one or more factory classes which contain static factory methods which return an instance of the interface type.
    - Ordering of class descriptions in this document should follow a sensible convention (e.g. alphabetical or order by program to match decomposition stage)

7. Detailed Design
    - It is not needed to provide all of the internal details of each module
    - You should however consider the difficult parts of the design and the way classes work together
    - For any module that the internal workings are not self-evident detail is needed to demonstrate feasibility and coherence of the overall design
    - Specifying the interface of an intractable module leave the whole design intractable
    - UML sequence diagrams are one good way of documenting how classes work together for major operations in the program
    - State diagrams and activity diagrams may also be of use
    - During design we will have experimented and done theoretical investigation to reduce risk decisions on difficult parts of the system should be documented here
    - For algorithms easier way is to do a textual description of what is to be done
    - If code exists it should be referred to but not included here
    - Significant data structure will occur when a number of objects are linked in a complex structure
    - Class diagrams showing entity relationships should be drawn where appropriate
    - Object diagrams should show how static relationship in class diagrams work in real examples
    - The mechanism to support any persistent data must be described
    - Language specific mechanisms may be cited and need not to be explained

REFERENCES

[1] QA Document SE.QA.05A – Design Specification

[2] QA Document SE.QA.01 – Quality Assurance Plan

[3] QA Document SE.QA.02 – Project Management Standards

[4] QA Document SE.QA.08 – Operating Procedures and Configuration Management Standards.

[5] QA Document SE.QA.03 – General Documentation Standards

[6] QA Document SE.QA.09 – Java Coding Standards

DOCUMENT HISTORY

Version	CCF No.	Date	Changes made to document	Changed by
1.0	N/A	18/10/14	Document Created and Structured	TCG2