Software Design Description Template

Author: Uwe van Heesch
Version: 2 (23/09/16)

1 Introduction

1.1 Overall Description

<Provide a short description of the software being specified and its purpose, including relevant benefits, objectives, and goals. If a separate description of the product scope is available (e.g. in the PvA or SRS), place a link here rather than duplicating its contents here.>

1.2 Purpose of this document

< Full description of the main objectives of the SDD document.>

1.3 Definitions, acronyms, and abbreviations

Term	Description

2 Architectural Overview

<Provide a high level overview of the architectural design, for instance by means of an architectural sketch. Make sure you show

at least all sub-systems, and links to external systems. The sketch can be informal. The use of UML is not required.>

3 Detailed Design Description

<This section contains detailed design documentation of all software components. The content of this section grows iteratively during the sprints. At the end of each sprint, the diagrams shown need to be consistent.>

3.1 Deployment Diagram

<Provide a UML deployment diagram showing all physical and virtual nodes used in the system. The diagram must also contain all deployment artifacts used in the system, for instance JAR or WAR files, or web artifacts.>

3.1.1 Design Decisions related to deployment

<Describe all design decisions manifested in the deployment diagram. For instance the choice of operating systems, protocols, distribution of components over sub-systems and the like.>

3.2 Design Sub-System A

<Do not really name the section “Sub-System A”, use a name that describes the responsibility of the sub-system, instead. Provide a section for each sub-system. These sections are iteratively added and refined during the sprints. Examples of sub-systems include *Persistent Storage*, *Business Tier*, *Web Application*, *Webservice API*. The sub-sections below may be extended if you think this is useful for describing the software design. The sub-sections below are only required for object-oriented sub-systems. Use other means to describe non-OO sub-systems (for instance Javascript modules).>

3.2.1 Design Class Diagram

<Object-oriented sub-systems should be described using a class diagram. If classes or interfaces are used across sub-systems, make sure you mention this in the description of the class diagrams. If your system entails layers, make sure you indicate this in the class diagram, e.g. by means of packages. For each class diagram, make sure you also mention the deployment artifact (from the deployment diagram) it is part of.>

3.2.2 Sequence Diagrams

<Provide sequence diagrams for major object interactions within the sub-system. It is ok if sequence diagrams cross sub-system boundaries. Make sure you explain this in the description of the diagram. Sequence diagrams must be consistent with the class diagrams described above. Also, if sequence diagrams cover interaction with users, make sure the diagrams are consistent with SDDs you may have documented as part of the SRS.>

3.2.3 Activity and State Diagrams

<This section is optional. If useful, provide activity and/or state diagrams to describe complex work flows and system state transitions>

3.2.4 Design decisions made for the sub-system

<Describe all design decisions made for the sub-system. Provide at least decision descriptions for all frameworks, libraries and other technologies used. Other decisions may be related to software patterns, system-structure, adapted principles or the like.>

3.3 Design Sub-System B (and so on)

…

3.4 Database Design

<. If your system uses relational databases, make sure you provide a physical datamodel here.>

3.4.1 Design decisions related to the database

<Describe all design decisions made along the database. This could include the choice of the database management system, the use of certain triggers or stored procedures, special indexes and so on.>