Black Dog

Supplementary Specification

Version 1.0

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** |
| 13/03/2020 | 1.0 | First look at Supplementary Specification. | Victor Padurean |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Table of Contents

1. Introduction 4

2. Non-functional Requirements 4

2.1 Availability 4

2.2 Performance 4

2.3 Security 4

2.4 Testability 4

2.5 Usability 4

3. Design Constraints 4

Supplementary Specification

# Introduction

[The introduction of the **Supplementary Specification** provides an overview of the entire document.

The **Supplementary Specification** captures the system requirements that are not readily captured in the use cases of the use-case model. Such requirements include:

Legal and regulatory requirements, including application standards.

Quality attributes of the system to be built, including usability, reliability, performance, and supportability requirements.

Other requirements such as operating systems and environments, compatibility requirements, and design constraints.]

The Supplementary Specification attributes depends strictly on the application to be implemented, and they refer to the design requirements that are not easily defined in the Use Case Model. Quality attributes are realized non-functional requirements used to evaluate the performance of a system.

# Non-functional Requirements

*[Define system quality attributes in terms of scenarios according to the following template:*

* *Quality attribute definition*
* *Source of stimulus: the entity (human or another system) that generated the stimulus or event*
* *Stimulus: a condition that determines a reaction of the system*
* *Environment: the current condition of the system when the stimulus arrives*
* *Artifact: is a component that reacts to the stimulus. It may be the whole system or some pieces of it*
* *Response: the activity determined by the arrival of the stimulus*
* *Response measure: the quantifiable indication of the response*
* *Tactics*

*]*

## Availability

* Availability is concerned with system failure and its associated consequences
* Source of stimulus: internal, external to the system
* Stimulus: crash, timing, unanticipated message
* Environment: normal operation at run time, also at design time
* Artifact: process, system
* Response: deploy/notify modification, continue or not
* Response measure: no downtime

## Performance

* Performance refers to timing, when events (interrupts, messages, requests from users etc.) occur and the system must respond to them
* Source of stimulus: independent sources, users
* Stimulus: initiate transactions
* Environment: under normal operations
* Artifact: system
* Response: transactions are processed
* Response measure: with average latency

## Security

* Security is a measure of the system’s ability to resist unauthorized usage while still providing its services to legitimate users
* Source of stimulus: correctly identified individual
* Stimulus: tries to modify information
* Environment: under normal operations
* Artifact: data within the system
* Response: system maintains audit trail, blocks the access
* Response measure: correct data is restored within a day

## Testability

* Testability refers to the ease with which software can be made to demonstrate its faults through testing
* Source of stimulus: unit tester/developer
* Stimulus: performs unit test
* Environment: at the completion of a component
* Artifact: component of the system
* Response: component has interface for controlling behavior and output of the component is observable
* Response measure: path coverage of a specific percentage is achieved

## Usability

* Usability refers to how easy is for the user to accomplish a desired task and the kind of user support the system provides
* Source of stimulus: users
* Stimulus: minimize impact of errors
* Environment: at runtime
* Artifact: system
* Response: wishes to cancel the current operations
* Response measure: cancellation takes less than a second

# Design Constraints

[This section needs to indicate any design constraints on the system being built. Design constraints represent design decisions that have been mandated and must be adhered to. Examples include software languages, software process requirements, prescribed use of developmental tools, architectural and design constraints, purchased components, class libraries, and so on.]

* The application should be developed using Java or C# programming languages.
* The application should use the Model-View-Controller architectural pattern and the Factory Method design pattern.
* It should be a desktop application.
* The project should also have a client-server architectural style and the Observer design pattern.
* The data on the server will be stored in a database.
* The code doesn't need to violate any good practices/conventions.