Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** |
| 22.03.2014 | 1.11 | Initial version of SDD | İsmetcan Hergünşen, Mehmet Kağan Kayaalp, Nazlı Karalar, Gamze Küçükçolak, Erdi Koç |
| 23.03.2014 | 1.12 | Introduction (References, abbreviations etc.) | Nazlı Karalar, Kağan Kayaalp |
| 24.03.2014 | 1.13 | Component 1 | Gamze Küçükçolak, Erdi Koç |
| 24.03.2014 | 1.13 | Component 2 | Mehmet Kağan Kayaalp, Nazlı Karalar, |
| 24.03.2014 | 1.13 | Component 3 | İsmetcan Hergünşen |
| 25.03.2014 | 1.14 | COTS Identification | Erdi Koç, Gamze Küçükçolak, İsmetcan Hergünşen |

**TABLE OF CONTENTS**

1 Introduction 3

1.1 References 3

1.1.1 Project References 3

2 Software Architecture overview 3

3 Software design description 3

3.1 Component 1 3

3.1.1 Component interfaces 3

3.1.2 Component design description 3

3.1.3 Workflows and algorithms 3

3.1.4 Software requirements mapping 3

3.2 Component 2 3

3.2.1 Component interfaces 3

3.2.2 Component design description 4

3.2.3 Workflows and algorithms 4

3.2.4 Software requirements mapping 4

3.3 Component 3 4

3.3.1 Component interfaces 4

3.3.2 Component design description 4

3.3.3 Workflows and algorithms 4

3.3.4 Software requirements mapping 4

4 COTS Identification 4

# Introduction

SDD is a representation of the software system design such as software components and interfaces. It also shows how the software system will be structured to satisfy the requirements. The purpose of SDD is to define the detailed design for all components of CCB system.

The SDD document is organized as follows. In the first part of SDD, it describes the project references and abbreviations and in component part, it shows the design of the system with UML diagrams. In COTS, ????

## References

### Project References

| **#** | **Document Identifier** | **Document Title** |
| --- | --- | --- |
| DOC#SDP V1.0 | [1] | Software Development Plan |
| DOC#SRS V1.0 | [2] | Software Requirement Specification |
| DOC#STP  V1.0 | [3] | Software Test Plan |

### Abbreviations

|  |  |
| --- | --- |
| **Term** | **Description** |
| CCB Project | Crazy Copter Battle Project Game |
| DOC #v.1.0.x | Document version 1.0.x |
| JDA | Java Desktop Application |
| MVC | Model View Controller |
| UI | User Interface |
| GUI | Graphical User Interface |
| SDD | Software Design Document |
| SDP | Software Development Plan |
| SRS | Software Requirements Document |
| UML | Unified Modeling Language |
| STP | Software Test Plan |
| JDK | Java Development Kit |
| STR | Software Test Report |

# Software Architecture overview

Describe here the top level software components and their interactions/relationships.

Use UML diagrams.

# Software design description

CCB software system will have one package and in package content, there will be several classes whose operations are divided depending on MVC model. Therefore, the components of CCB are Model, View and Controller.

Describe each top level package/component of your software and if necessary sub-components/sub packages.

Use Class diagrams, sequence diagrams and deployment diagrams to illustrate your description.

## Component 1 - Model

### Component interfaces

Describe the interfaces of the component and input output data

### Component design description

Describe the design of the component, Use class diagrams to show the links between sub-components/sub-packages and or classes inside the component.

### Workflows and algorithms

Use sequence diagrams and activity diagrams to show the workflows of components/packages/classes inside the component.

Describe algorithms, if possible. An algorithm may be described outside this document, in this case, add the reference to that document.

### Software requirements mapping

List the SRS requirements handled by this component

## Component 2 - View

Repeat the pattern for each component.

### Component interfaces

Input: User will press the start button (controller???)

Output: Game will start. Images will be created.

Describe the interfaces of the component and input output data

### Component design description

Class: Window

Constructor

-Usage: Create a new instance of this class

Methods

* Main

**public** **static** **void** main(String[] args)

Class: Framework

Constructor

-Usage: Initialize framework, background images etc.

Methods

* Public void InitializeText()
* Public void keyReleased(KeyEvent event)
* Public void mouseClicked(MouseEvent event)

Class: Game

Constructor

-Usage: Initialize all objects of the game such as bullets, helicopter, and create content of the game.

Methods

* Public void restartGame()
* Public void updateGame()
* Public void create()
* Public void showStatistics()

Describe the design of the component, Use class diagrams to show the links between sub-components/sub-packages and or classes inside the component.

### Workflows and algorithms

Use sequence diagrams to show the workflows of components/packages/classes inside the component.

Describe algorithms, if possible. An algorithm may be described outside this document, in this case, add the reference to that document.

### Software requirements mapping

* SRS-CCB-001.1
* SRS-CCB-001.2
* SRS-CCB-002.1
* SRS-CCB-004.1
* SRS-CCB-004.2
* SRS-CCB-009.1
* SRS-CCB-009.2

List the SRS requirements handled by this component

## Component 3 - Controller

Repeat the pattern for each component.

### Component interfaces

Describe the interfaces of the component and input output data

### Component design description

Describe the design of the component, Use package diagrams and/or class diagrams to show the links between sub-components/sub-packages and or classes inside the component.

### Workflows and algorithms

Use sequence diagrams to show the workflows of components/packages/classes inside the component.

Describe algorithms, if possible. An algorithm may be described outside this document, in this case, add the reference to that document.

### Software requirements mapping

List the SRS requirements handled by this component

# COTS Identification

List external software components/libraries that your system rely on, if there are any.

Example:

COTS (commercial of the shelf) libraries used in XXX are the following:

* foo.jar, version id, download URL, License type,
* bar.jar, version id, download URL, License type,