The Change Counter

# Software Requirement Specification

## Gangof4 Team

Abishek Minde
Joao Carlos Almeida
Krishnan Ganesan
Nisha Narayan

**Version** : 1.0

**Date** : April 3rd, 2009

# Document Revision History

| Version | Date | Author | Reviewer | Updates |
|---|---|---|---|---|
| **1.0** | April 3rd, 2009 | Krishnan Ganesan | Joao Almeida | |

# Table of Contents

# 1 Introduction

## 1. Purpose

The purpose of this document is to describe the functional and quality requirements of the change counter that would serve as a guide for developers to build the system and help the end users verify if the developers are building the system that they intended.

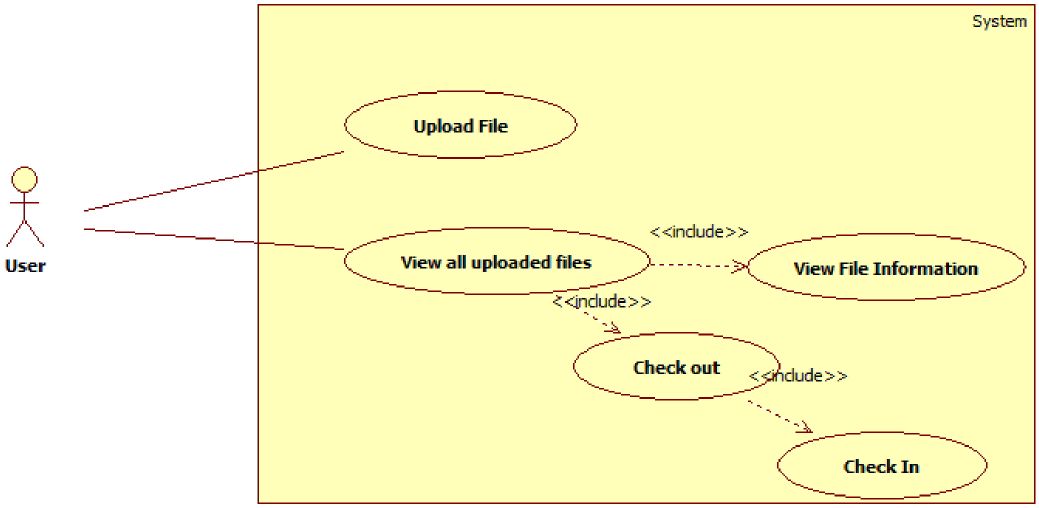
## 2. Problem Statement

The client wishes to have a change counter tool built that would compare two versions of a file and reports the added, deleted, modified & total lines back to the user.

## 3. Team Project Information

| | |
|---|---|
| **Project** | GangOf4 |
| **System** | Change Counter |
| **TSP Team#** | Team 4 |
| **Roles (Cycle1)** | Instructor/Client – Mel Rosso Llopart<br>Team Lead & Support Manager – Abishek Minde<br>Planning Manager – Nisha Narayan<br>Development Manager – Krishnan Ganesan<br>Quality Manager – Joao Carlos Almeida |
| **Project Start Date** | March 16th, 2009 |
| **Project End Date** | May 3rd, 2009 |
| **Cycle 1 Start Date** | March 20, 2009 |
| **Cycle 1 End Date** | April 17th, 2009 |
| **Cycle 2 Start Date** | April 18th, 2009 |
| **Cycle 2 End Date** | April 29th, 2009 |

# 2 Functional Requirements



|  |
|---|
| **UC 1(Upload File):** The user wishes to upload a file into the system. This means, the user wishes to upload a new file that is not present in the system. |
| **Precondition:**<br>1. The user is provided a menu of options in the command line interface and the user chooses the option to Upload a file.<br>2. There are no revisions of that file in the system i.e. the file is new. |
| **Basic Flow** |

| **Line** | **System Actor Action** | **System Response** |
|---|---|---|
| 1 | The user chooses the option pertaining to uploading a file from the menu shown. | The system prompts the user to enter the name of the file along with the complete physical path. |
| 2 | The user enters the file name along with the complete path information and confirms the operation | The system creates an initial version of the file in the data store. |

**Post condition:** The system creates the initial version of the file and displays a success notification for the operation performed.

**UC 2(View all files):** The user wishes to see the list of all files that are version controlled by the system. This operation will not list file information and revisions.

**Precondition:**

The user is provided a menu of options in the command line interface and the user chooses the option to view the list of uploaded files.

**Basic Flow**

| **Line** | **System Actor Action** | **System Response** |
|---|---|---|
| 1 | The user chooses the option pertaining to viewing the list of files uploaded. | The system displays the list of uploaded files and options to check out and view file information. |

**Post condition:** The system displays the list of files along with the options to check out and view file information.

**UC 3(Check Out a File):** The user wishes to check out a file to perform modification.

**Precondition:**

1. The user has provided a file name and chosen “checkout” operation.
2. The file is present in the system.

**Basic Flow**

| **Line** | **System Actor Action** | **System Response** |
|---|---|---|
| 1 | The user chooses the check out option | The system asks the user to enter the filename |
| 2 | The user enters the filename | If the file is present, the system returns the latest revision of the file. |

**Post condition:** The system returns the latest revision of the file.

<table>
<tr><td colspan="3"><b>UC 4(Check In a File):</b> The user wishes to check in a file after modification.</td></tr>
<tr><td colspan="3"><b>Precondition:</b><br>• The user has already checked out a particular file.<br>• The file user wants to check-in has a filename that matches with one of the filenames in the system.</td></tr>
<tr><td colspan="3"><b>Basic Flow</b></td></tr>
<tr><td><b>Line</b></td><td><b>System Actor Action</b></td><td><b>System Response</b></td></tr>
<tr><td>1</td><td>The user inputs the following information during check-in operation:<br>- Filename<br>- Author name<br>- Purpose of change</td><td>The system saves it as a new revision and it compares this new revision with the prior revision to identify the added, deleted and total lines of code and stores this change summary in the data store.</td></tr>
<tr><td colspan="3"><b>Post condition:</b> The system stores the latest revision and change summary in the system data store.</td></tr>
</table>

<table>
<tr><td colspan="3"><b>UC 5(View File Information):</b> The user wishes to view file information.</td></tr>
<tr><td colspan="3"><b>Precondition:</b> The user is in the view uploaded files screen and user enters a filename that is present in the system.</td></tr>
<tr><td colspan="3"><b>Basic Flow</b></td></tr>
<tr><td><b>Line</b></td><td><b>System Actor Action</b></td><td><b>System Response</b></td></tr>
<tr><td>1</td><td>The user chooses to view the information for a particular file by choosing the view file information option on the command line interface and the filename.</td><td>The systems displays summary of changes to that file. The summary includes information that is described in the chapter 5, section 2 "Header labelling specification”.</td></tr>
<tr><td colspan="3"><b>Post condition:</b> The system displays the change summary of the selected file.</td></tr>
</table>

# 3 Requirements Classification

## Functional Requirements :( Must Have)

| Requirement Id | Description |
|---|---|
| REQ001 | Identify the added and deleted lines of code in a modified program. This requirement is described in the use cases, UC4 and UC5; however, we are considering only added and deleted lines while comparing two versions of the file. |
| REQ002 | Count the added and deleted lines of code in a modified program. This requirement is described in the use case UC5; however, we are considering only added and deleted lines while comparing two versions of the file. |
| REQ003 | Count the total lines of code in the modified program. |

## Functional Requirements :( Nice to have)

- Attach line label reference for added and deleted lines of code
- Provide the change label in a program header comment indicating the change number, date, who made it ,why it was made , the added , deleted and total lines of code.
- When a previously modified program is further modified all the prior modifications records must be maintained.
- On demand, print program listing with change label and line label information
- On demand, print a program listing with line number of modified program inserted at the head of each line of code.
- On demand, print a program-change report with statistics for the modified program and all prior changes

## Quality Requirements :( Must Have)

Usability (Immediate Understanding):- Needs to be elicited from the client.

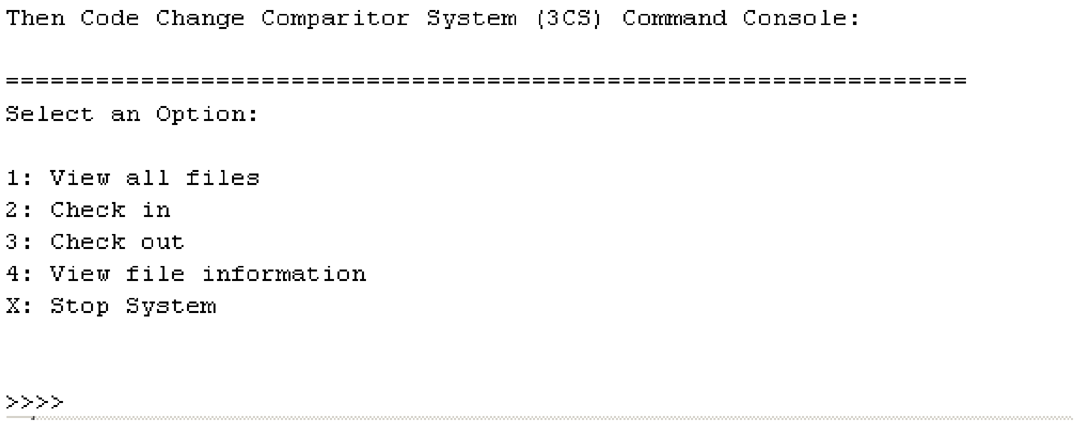
## Quality Requirements :( Nice to Have)

Extensibility (Multiple Language Support):- The software will initially be built to work with any one of the languages (Example Ada, C++, C, Pascal) but logic can be extended for other languages.

# 4 External Interfaces

## 1. User Interface

The user interface for the change counter tool will be a command line interface displaying options to the user. Figure below shows an example of such an interface.



```
Then Code Change Comparitor System (3CS) Command Console:

================================================================
Select an Option:

1: View all files
2: Check in
3: Check out
4: View file information
X: Stop System


>>>>
```

Option 1 – If a user selects this option the system shall present a list containing al the files under system change control (UC1).

Option 2 – The option should be used when a user wants to add a new file into system control (UC2).

Option 3 – The option should be used when a user wants check in the file as described in the use case (UC3).

Option 4 – This option should be selected by a user when he needs to change a file that is present in the change control system (UC4).

Option 5 – This option is used if a user wants to see a file information summary change. After selecting this option, a user should select the file that he wants to see the information (UC5).

Option X – After selecting this option, the system closes.

## 2. Data Store Interface

All the information (change information) pertaining to a file in the system will be maintained in a persistent data store.

Following information shall be maintained by the system.

1) List of programs that are version controlled in the system

2) For every program file there can be multiple revisions.

3) Change summary for every revision of a file

4) Change header for every file

# 5 Design/Implementation Constraints

## 1. Counting Specifications

1.1 Have the LOC counter count text lines with the coding standards that requires one logical LOC to be placed on a single text line.

1.2 In comparing a program with its prior version the comparisons are made as follows:

1.2.1 If a line is contained in both original version 3 and modified version 4, it is an original and an unchanged line.

1.2.2 If a line is in version 4 only and not in version 3, it is an added line.

1.2.3 If a line is in version 3 only and not in version 4, it is a deleted line.

1.3 Count each changed line as an added and deleted line.

1.4 Analyze added-deleted line pairs to identify truly added-deleted lines and modified lines.

1.5 Moving or copying a section of code would attribute to deleting the section of lines from the original location and adding them to a new location.

1.6 All differences between two program version to same change number.

## 2. Header Labeling Specification

2.1 Every (.java) file in that is being version controlled by the system shall have one program header. The program header shall contain following things,

2.1.1 Program Name – String

2.1.2 Program creation date: MM/DD/YYYY format

2.1.3 Number of changes done since the initial version: Integer

2.1.4 Number of deleted lines since the initial version

2.1.5 Number of added lines since the initial version

2.1.6 For every change following fields shall be present, we call this change summary

2.1.6.1 Changed by whom

2.1.6.2 Purpose of the change

2.1.6.3 Timestamp when the file revision was checked in

2.1.6.4 Number of lines added (when compared to the previous revision)

2.1.6.5 Number of lines deleted (when compared to the previous revision).

2.1.7 Other requirements:

2.1.7.1 Change summary will have display all lines as added lines. There will be no deleted lines.

2.1.7.2 Change header is placed at the beginning in the program file.

## 3. Formatting Specification

In order to develop this system some standard must be followed by development team. In this section some of these standards are described and might be applied on team deliverables.

### Requirements

All the requirements should be identified according to the following criteria:

***REQNNN***

REQ – characters used to easily identify a requirement;

NNN – three digits number for requirement unique identification;

Example: REQ001 – The system shall count the number of added code lines.

### Design

The development team will use UML diagrams to represent the system architecture. The following diagram can be used to make this representation

- **Class diagram** – this diagram must be used to represent the classes to be created to build the system.
- **Activity diagram** – this diagram can be used in complex classes to detail the activities to be performed by classes' methods. This diagram is optional and will be under developer criteria the need to create or not this diagram.

### Language

In order to gain system platform flexibility the programming language to be used on the project will be Java.

### Coding

In order to our system be able to correctly evaluate a code file, programmers might follow the standards describe in this document. Otherwise is not guaranteed that system evaluation about a code file will represent correct information regarding added, deleted and changed lines of code.

Here are some standards that should be respected on the files to be controlled by the system:

1. Avoid more than one command per line – programmer should avoid using more than one method or function per line of code;

2. Comments will not be counted as line of code;

3. Avoid using multi-line comment tags like "/*  */", instead use single line comments, like "//";

4. Empty lines will not be counted as line of code;

## 4. General Specification

**Language Support:** The Change Code Counter will work only for programs written in Java.

**User interface:** The product will have a command line user interface.

**Platform:** The product will run on operating systems that can run the Java Runtime Environment.

# 6 References

- **Chapter 6 – Defining the Requirements, Introduction to Team Software Process, Watts Humphrey.**