Software Requirements Specification

Smart Undo Capability

(Added to a java text editor)

COMP 5541

Tools and Techniques for Software Engineering

Assignment 1

Due date: February 16, 2021

Team members

First name, Last name	Student ID	Contribution
Eric Ting-Kuei, CHOU	40070424	Introduction (1.1 to 1.4), Overall Description (2.1 to 2.6) Report Formatting
Adnan, ALI	40181614	User Cases, User Diagram, Class Diagram
Seyed Homamedin, HOSSEINI	40137467	Review and edits, formatting
Mohammed, ZAFAR	40152488	Design Approach/Algorithms, Contribution to the definitions, functional reqs and use cases

Contents

1 Introduction

1.1 Purpose

This document outlines the specifications for the Smart Undo Capability added to a Text Editor in order to make sure that the project requirements and expectations are clear to both the client and the development team.

This report loosely follows SRD structure found in the subsection “reference”. This software project will be managed and developed by Team #3. This document will serve to communicate the project progression to the client as well as acts as a documentation for future references.

1.2 Overview

Most text editing software provides users with standard functions such as the ability to create, to open, to edit (write), and to save a text file. In this project, we aim to implement a smart undo feature into an existing text editor. In other words, the users would be able to undo edits in any order they wish, in one or multiple steps, and create different groups of edits.

1.3 Definitions

This section provides clear and precise definitions of key terminology and concepts. This is essential as it allows the client and those involved to better understand the project requirements and helps prevent any ambiguity.

Edit

Edit is a function that is executed by the user of the text editor. It consists of any user's activities in the editor. For example, when a user enters a space, or a character, each of those is considered as an edit. If a user enters a new line, or backspace (delete a character), it is an edit as well. Essentially every single input from the user's input devices including keyboard and mouse inside the text editor is considered as an edit. e.g. User enters the word `hello`. This consists of 5 edits, 1 for each character insert.

Undo

Undo is a function that reverses a user's edit. In the previous example, the user enters the word `hello`. Since that's 5 edits, it would take 5 undos to remove the word `hello`. If the user decides to delete the word entirely by selecting the word and pressing backspace, that is considered as a single edit as well. So an undo will result in reversing the deletion of the word.

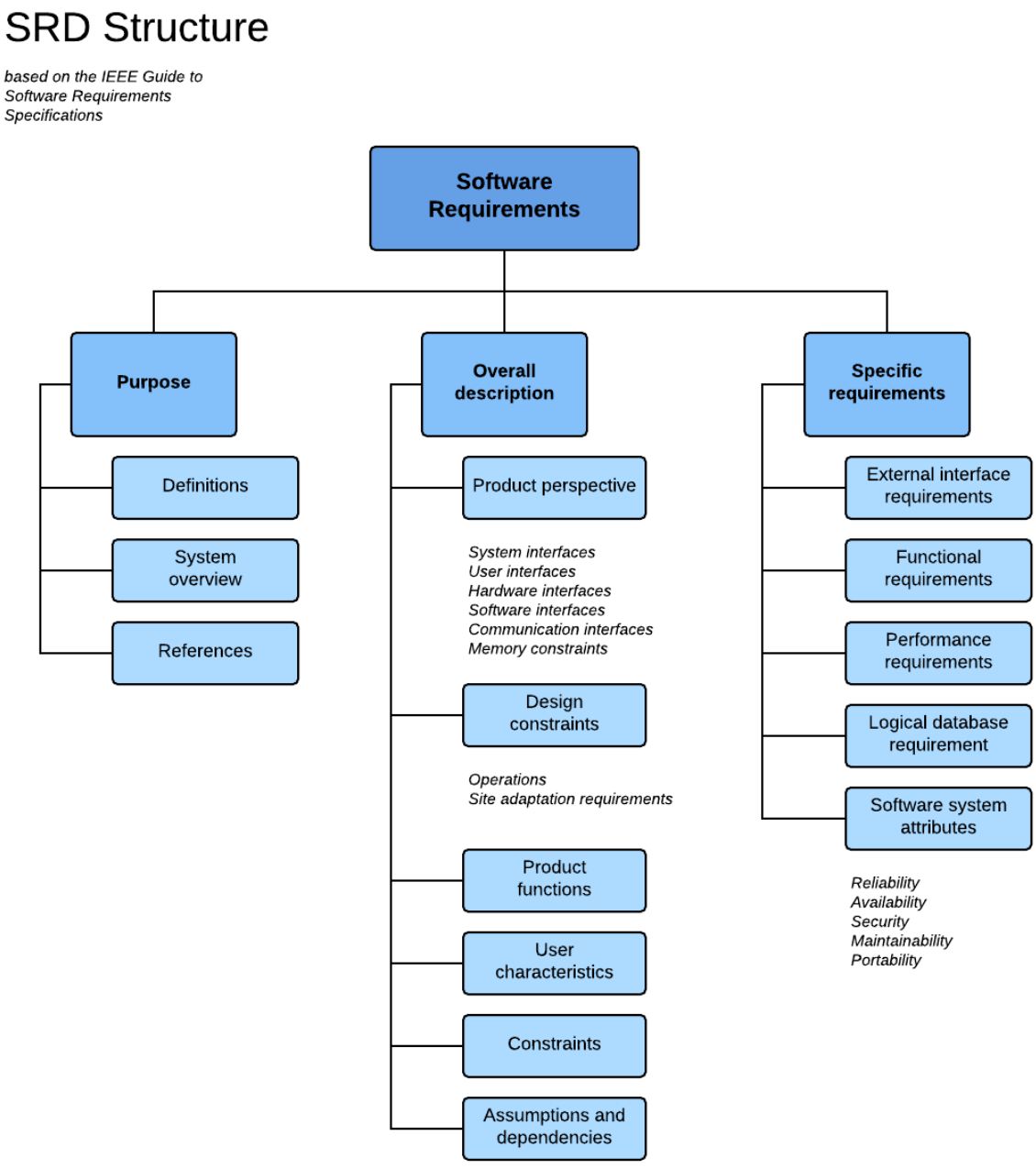
Group of edits

As the name suggests, a group of edits contains a series of edits that are associated together as a group of edits by the user. For example, the user can decide to start a group edit at the beginning of line x and start typing. At the end of the edits at line y (where x <= y), the user decides to close this group of edits here and later down the text document, the user can create additional groups of edits.

1.4 Reference

[1] Assignment 1 outline on moodle. **Filename: Asg1.5541w21.pdf**

[2] SRD diagram based on the IEEE Guide to Software Requirements Specifications.



2 Overall Description

2.1 Product Perspective

The Smart Undo capability is not a standalone product and will be added to an already existing java-based text editor. This product aims to implement the smart undo function.

2.2 Product functions

The user will be able to select specific edits in any order from a list and perform them in one step. In the example below, the user can decide to select to undo the input of values B and e, in that order.

Example: The user wrote `Bye`. In the undo memory we get the following:

Group of Edits #	Line	Column	Function	Value
0	1	1	input	B
0	1	2	input	y
0	1	3	input	e

1. The editor will allow the user to end and create new groups of edits.
2. The user will be able to select edits in any order from within different groups, and undo those edits in the same selection order, and in one step.
3. The user can arbitrarily select a list of edits that they no longer wish to have, and delete them. Once this action is performed, the user won't be able to undo these edits.
4. The user can also delete a group of edits so that it's no longer possible to undo the edit within this group.
5. Users should be able to select which group to undo via user input.

2.3 Design constraints

I. The text editor must be written in Java code.
II. The open source code for the text editor should be the following: *https://opensource.com/article/20/12/write-your-own-text-editor*

2.4 Functional requirements

I. Groups of edits always begin from the beginning of a line.
II. Groups of edits always complete at the end of a line.
III. There can be no overlaps between groups.
IV. Allow users to set a limit on how far back edits are saved.

Create/End group of edits

- This function ends the group of edits on the current line and starts a new group of edits on the next line.

Delete Edits or Group of Edits

- This function prevents an edit or a group of edits to be undone.

Clear memory

- This function clears all edits from the memory.

2.5 Non-Functional requirements

- The editor should be designed to be intuitive and user-friendly.
- Performance should not be affected by the Smart Undo of large quantities of edits.

2.6 Design approach / Algorithms

Key User Events	**Algorithm for the action taken by the "Smart Undo Capability" module of the text editor system**

User creates a group	Invoke group_creation_method ( )
	{
	Declare variables/arrays for storing the group name
	Read the group name from the user input
	Create a dropdown list for the created group, inside the "Undo Group menu"
	}

User performs an edit action	Invoke edit_method () {
	● Declare variables /arrays of dynamic size for storing the following information of an edit action
	○ Edited character value
	○ Character Position (in the editor)
	○ Edit Type ("Add" or "Delete")
	● The character value is read and added to the array
	○ If Edited character value = "backspace", Then Edit type = "Delete", else " Add"
	○ Character position = (Row #, Col #)
	● Display the edit information in the group dropdown
	}

User selects multiple edits from a group and performs an “Undo” action	Invoke undo_method ()
	{
	● Declare arrays of dynamic size for storing the below information pertaining for each of the selected edits
	○ Edited character value
	○ Character Position (in the editor)
	○ Edit Type (“Add” or “Delete”)
	● The information of selected edits is read and added to the arrays
	● Access the position of each edit from the array in the same order the edits were selected by the user and perform the following actions
	
	If the “Edit Type” info of the edit = “Delete”, then add the “Edited Character Value” to the accessed position of the editor,
	Else if the “Edit Type” info= “Add”, then remove the “Edited character Value” from the accessed position of the editor.
	}

3. Use Cases

In this section we will describe some of the case scenarios for this Smart Undo feature.

3.1 Create a group, edit the text, and undo some of the edits

Events	
Input	User enters a series of 50 characters, where each input is an edit.
Decision	User decides to remove the characters that were entered on the 35th and the 16th input, and in that order.
Action	In the undo menu, the user selects the 35th and the 16th input, and press undo.
Result	Character 35th is first removed, followed by the 16th.

3.2 Delete characters and undo the deletion of some characters

Events	
Input	User enters a series of 10 characters, where each input is an edit.
Input	User deletes the last 5 characters starting from the end. Each delete is an edit as well.
Decision	User decides to undo the first 2 deleted characters.
Action	In the undo menu, the user selects the 11th and the 12th edits, and press undo.
Result	The 10th character entered initially will re-appear, followed by the 9th.

3.3 Create a new group of edits

Events	
Decision	User decides to create a new group of edits
Action	User clicks on Create a new Group
Result	The previous Group ends on the current line, while the new Group starts on the next line.

3.4 Select and undo edits from different groups

Events	
Decision	User decides to undo edits from 2 groups of edits
Action	In the undo menu, the user selects the edits in the order they wish the undo to be performed in any of the groups in the Undo List. The user presses Undo.
Result	All selected edits in those groups are undone in the same order of selection.

3.5 Delete edits so that they can never be undone

Events	
Decision	User decides to prevent accidental undo for some edits.
Action	In the undo menu, the user selects the edits in the order they wish to never undo. The user presses Clear.
Result	All selected edits are now cleared from the undo menu / list.

3.6 Delete all edits within a Group

Events	
Decision	User decides to delete all edits of a group
Action	In the undo menu, the user selects a group and presses “Delete Group” button
Result	The entire group is deleted from the undo menu.

4. User Stories

In order to create a software requirement document, our team was given a basic set of instructions by our stakeholders, Babita Bashal and R. Jayakumar. However, to make sure our features were realistic, we decided to interview potential future users and implement new ideas into our project, while keeping the time constraint in mind.

Based on the answers of our interviewees, we created some personas that were related to potential user problems and implemented related tasks that our software could solve. These personas, along with the team’s initial discussion were used to form the use-case scenarios for our smart text editor. A standard model was implemented so that a high-level design of our smart undo-capability could be understood at a glance. A summary of each use-case is explained thereof.

4.1 Initial Personas

Personas	Use Cases	Potential Implementation
Entrepreneur, Business Owner	Use on multiple screens, simplicity, and shortcuts.	Responsive design for PC and phone, button tooltip.
Engineering Student	Use in different environments, Add/remove ideas on a whim.	Theming engine for dark mode, visualize edit by groups, Add notes to each edit.
Computer Science student	Reverting back to original statements for debugs, keep track of changes per code snippet.	Hotkey commands for certain functions, Sort edits from most recent, Color code each group of edit, easily accessible user manual.
International Researcher	Removing outliers permanently, keep track of edits per thesis section, prefers user-friendly interface.	Button tooltip for common functions, localization for language support.
Lawyer, Intern	Common edits for quick amendments, create a list of edits to transfer it to another document.	Basic functions, ability to save groups across accounts.

4.2 Detailed Personas

Persona Name: Jeff Bezos

Specifics: Mr. Bezos is a 57-year-old business owner for a large e-commerce website. He is decently comfortable with the software of his phone and his desktop, but still not as comfortable as his younger counterparts. He prefers to keep his meeting notes in a text file and make amendments whenever he gets free time from his busy schedule. With Android being the most accessible piece of hardware, he prefers to brainstorm during his travels. He needs a text editor that works on his work computer as well as his Android with simple shortcuts for common functions. Even though he is comfortable with some key functions, he likes the ease of use on his phone. He also likes to transfer his work from one platform to the other, just by logging in.

Persona Name: Thomas Edison

Specifics: Thomas is a 23-year-old engineering student with a passion for design. Like every Undergrad, Thomas thinks about his projects on a daily basis while being occupied with basic chores. He likes to keep an easy and accessible diary, of his ideas, where he can keep track of his thought process without having to write them separately. He appreciates the idea of editing his train of thought, while keeping track of his original design without creating multiple copies by adding notes to any small changes.

Persona Name: Bill Gates

Specifics: Bill is a 3rd-year-computer science student that prioritizes efficient programming over theoretical knowledge. Even though most of his work is designated to IDE, he sometimes uses a text editor to compare various codes and make amendments, if needed. Even when outside the GUI interface, he likes to be meticulous by keeping track of each small change he makes just so it is easier to debug later. He thinks that this project would help him tremendously, if a group of edits can be color coded to match the change in a particular method of a class. This would not only help find the source of the problem later, but also be easy to share his code by pointing fellow engineers to visual aid, rather than the specific line of code. He also likes to switch to hotkeys in a given setting, and therefore prefers using applications with an in-depth manual.

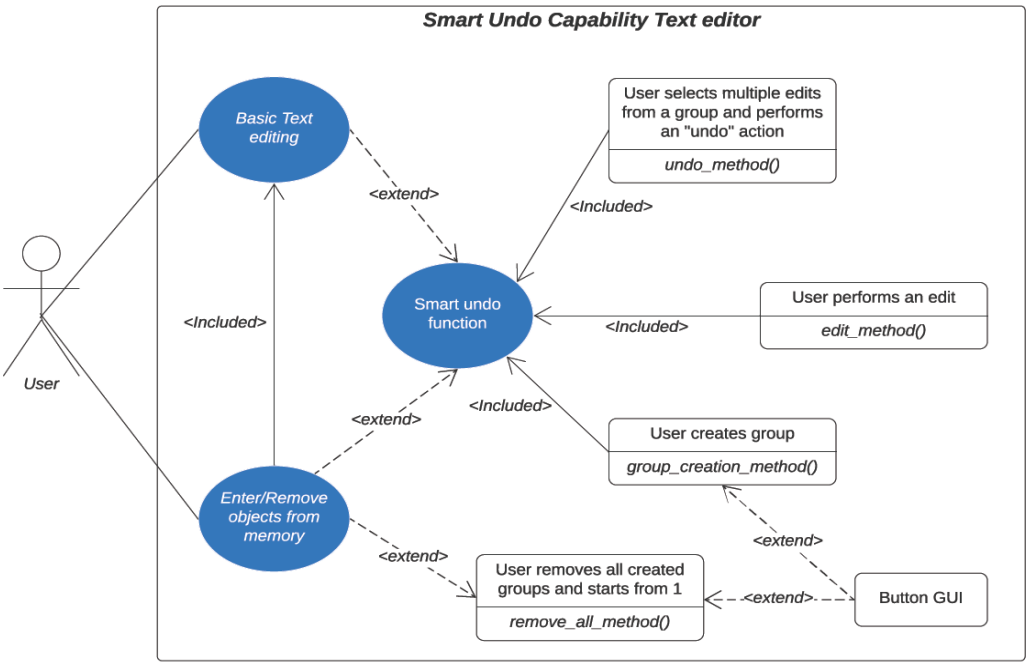
Persona Name: C. Ronaldo

Specifics: Ronaldo is a 35-year-old graduate research student, with a Portuguese descent. Even though he conducts his research in English, he is not a native speaker and prefers to communicate in his mother tongue. He uses the windows localization settings generously during his stay in Canada and prefers choosing his software based on the same feature. As a biology student, he is not familiar with all of the commonly used functions and prefers to have a mouse accessible list.

Persona Name: Sarah Wells

Specifics: Sarah is first-year law student, who is interning at Smart & Biggar for the winter co-op program. Her aim is to become a contract lawyer and open her own drafting firm. A normal day-to-day routine for Sarah requires a lot of revision of court documents i.e., amending case proceedings in a very efficient, yet timely manner. She was thrilled at the prospect of undo/redoing edits in one go, however, she mentioned that she would definitely appreciate the idea of carrying forward those groups of edits to upcoming cases, as some of these drafts are interlinked. Sarah admitted that she was very comfortable with using either the button tooltip or keyboard, as long as her work would not be time constrained.

During this initial requirement gathering phase, we decided to keep our set of use-cases simple due to a small team. These would form the base of the functionality we wish to offer our stakeholders and users.



User Diagram

4.3 Structure and Relationship class diagram