Bilkent University

Department of Computer Engineering

Senior Design Project

Project Short Name: Clerk

High-Level Design Report

Ahmet Malal, Ensar Kaya, Faruk Şimşekli, Muhammed Salih Altun, Samet Demir

Supervisor: Prof. Uğur Doğrusöz

Jury Members: Prof. Varol Akman and Prof. Çiğdem Gündüz Demir

Innovation Expert: Mehmet Surav

High-Level Design Report
Dec 31, 2019
This report is submitted to the Department of Computer Engineering of Bilkent University in partial fulfillment of the requirements of the Senior Design Project course CS491/2.

Contents

1 Introduction

Microsoft Word is one of the most popular word-processing programs around the world. It is used primarily to create various types of documents that you can print and publish, such as books, papers and reports. When you open a document in Microsoft Word, it can be edited using various features that Word provides. Currently, these features are accessible with use of some input devices, such as mouse and keyboard.

In this way, it is assumed that people who are going to use Microsoft Word, should able to use these devices in a conventional way. However, some people can't use these devices effectively or at all. Some people might not want to use them. Some people can't use their hands or are visually impaired. There should be a reliable way for these people who also may want to create documents and write reports, poems, and maybe a book.

Let us consider a visually impaired person, for instance. They should be able to create, delete, and save files; type and delete sentences; change the font and type of the text; change the position of the cursor and the alignment of the paragraph; change the color of the words or insert images into the document. Briefly, they should be able to use basic functionalities of a program like Microsoft Word. There is no tool currently available that provides use of major functionalities of Microsoft Word without the use of mouse and keyboard, or similar input devices.

Microsoft allows developers from around the world to develop applications that will extend the functionality of Word. These applications are called "Word add-ins". A Word add-in is an application which is essentially embedded inside of Word and can be launched from within a running Word instance. We want to build our application as an add-in for Word because it is the most popular text editor there is and it fits best with the nature of the work we want to do, which is to extend the conventional text-editor functionalities and make text-editing more accessible for people.

We will build an add-in for Word that will allow users to utilize Word features with voice commands.

1.1 Purpose of the system

Clerk is a Microsoft Word add-in based on speech recognition technology and designed for users who are not able to or do not want to edit documents using keyboard and mouse. The main purpose of the Clerk is to allow the users to type in Word, use several functionalities of Word by speaking. Clerk aims to recognize, and digitize speech to perform typing words, punctuation marks and perform several functionalities of Word such as changing font, aligning margin, find and replace, adding page numbers, adding header and footer, select, read and delete text, cut, copy and paste text, and reviewing grammar.

1.2 Design goals

The aim of Clerk is to obtain the transcript from user's speech and execute commands based on that to edit documents in Word. We want to design a system that is easily extensible and maintainable that performs at an acceptable level and provides users with a good experience. To this end, we define our design goals as follows.

1.2.1 Extensibility and Modularity

Clerk will be sufficiently modular and additional functionality will be added easily. The application will be implemented such that replacing existing modules and adding new ones like different speech recognition API's and libraries will be easy. The additions and replacements in modules, if they are straightforward, will not affect other modules.

1.2.2 Maintainability

Clerk will be an easily maintainable system. This is going to be achieved by the modularity of the system previously mentioned. Changes in one subsystem will not affect others. Maintainability of the system also affects ease of pinpointing the causes for any errors and makes it easier to fix those errors.

1.2.3 Performance

Performance of Clerk depends highly on the response time for the API calls that will be made for speech recognition. The user should not be waiting for longer than 3 seconds for the system to have acceptable usability, and by extension, the speech recognition service that will be used should not have any delays longer than this period. Another aspect that may be a bottleneck for performance is understanding and executing the commands. The user should not be waiting for longer than 5 seconds from the moment they start talking to see the effect of their speech on the document whether they have given a command or they are just in typing mode. Correctness is another aspect of speech recognition that should be considered as part of the performance of the system. User's speech should be converted to text with 90% accuracy. To improve accuracy, a speech adaptation feature which gets used to its user and give better results than 90% when the user spent time with Clerk should be considered if possible.

1.2.4 Robustness

Clerk will not randomly crash. It will be able to recover from most errors. It will be able to communicate the errors and the reasons they occurred to the user as much as possible.

1.2.5 Usability

The users will be able to perform all functionalities of Clerk using voice commands from an audio input device. There will not be a need to use any other input devices except a designated key which distinguishes commands from normal inputs. This is especially important since part of our target users are people who aren't able to use these devices due to visual impairment. Despite working with speech recognition, Clerk should still provide a user interface to its users. Several functionalities such as setting the mode and start and stop voice input options can be used from the UI.

1.2.6 Supportability

The system will be adaptable with the platforms like Word and API's like Google Speech-to-Text since it is highly dependent on them. This is crucial for the system to be flexible against the changes in these platforms or API's.

1.3 Definitions, acronyms, and abbreviations

API: Application Programming Interface
UI: User Interface
Speech Recognition: identifying the content in a particular speech segment(e.g. recognizing the punctuation marks said in a speech).

1.4 Overview

Clerk will be Microsoft Word add-in designed for people who are visually impaired or people who does not feel comfortable with keyboards. The user will be able use the common functionalities of Microsoft Word by dictating commands or by giving a speech from an input device. The Clerk will recognize the speech of user act accordingly such as when the user says, "Dear John comma", the program will insert the words "Dear John," to where the cursor currently is. It also takes commands from the user's speech and apply the related function in Word such as when user presses a designated key and says "select sentence", the program will select the sentence which the cursor is in. It allows users to execute several common functionality except select like copy, paste, delete as well as save the file, make selected text bold or italic. Additionally, the Clerk works multi platform since add-ins can work on any platform that can run Word.

Clerk will also have features to detect errors and misunderstandings, and confirm spelling of words. These features will work by reading the specified text/paragraph or spell a specific word if the user commands the application to do so. If Word detects a spelling error or encounters a word that is not in the dictionary, the application will alert the user. Also, for visually impaired people it is not possible to use the mouse to navigate inside the document. To ease the use of Word for them, we will have commands to set the position of the cursor by specifying the location where they want the cursor to be. For example, "Second Paragraph First Sentence" will take the cursor to the first sentence of the second paragraph.

We are going to use Word JavaScript API to develop the Microsoft Word add-in [1]. HTML, JavaScript and CSS will be used for implementation of our add-in, which is the standard way add-ins are built for Microsoft Word.

To conclude, Clerk will allow users to use Microsoft Word with speech recognition without using a keyboard or mouse.

2 Proposed software architecture

2.1 Overview

The proposed software architecture of Clerk describes details of how different subsystems of the system will interact with each other in the application, the strategies to be utilized for building the system such as the hardware/software platforms, persistent data management, access control and security, global software control and boundary conditions.

2.2 Subsystem decomposition

The subsystem decomposition for Clerk consists of five main subsystems that interact with each other; a voice input management subsystem, a speech recognition subsystem, a command management subsystem, a view subsystem and a control subsystem to bring other subsystems together.

The voice input management subsystem will manage everything related to voice input in the application as suggested by the name. The functionalities related to voice input are: Receiving input from different devices or sound files, setting voice input threshold, reporting any problems with input voice and any kind of signal analysis that may be done on the voice input.

The speech recognition subsystem will use the input signal (the voice) from the voice input management subsystem to generate the transcript of the speech input. To this end, external services such as Google Cloud Speech-to-Text[2] or IBM Cloud Speech-to-Text[3] will be utilized. This subsystem will handle all communications, which in most cases will happen over a network connection, with said services. The typical communication with a speech recognition service consists of sending over the voice data to a remote server, where it will be processed based on various speech recognition and language processing models abstracted from the users of the service, and waiting for a reply. This process should be handled asynchronously, alerting the main application thread only when the reply is received. The subsystem will also handle different parameters such as grammar used and encoding options which may be required by different speech recognition services. This subsystem may require some natural language processing components as well in the future for better user experience.

The command management subsystem will use the transcript obtained by the speech recognition subsystem to apply the changes to the document or execute the command given by the user, depending on the application state. To work with the document and the Word program itself, the Word JavaScript API service[1] will be utilized. This subsystem will handle all interactions with Word and the document. A typical interaction with the document may consist of inserting some text over some range or deleting text in some range. This process, like the speech recognition process, should be handled asynchronously and not block the main application thread.

The view subsystem will mainly consist of a few views presenting information to the user about the state of the program and voice management settings such as the input threshold and voice input device. Clerk will have a relatively simple user interface as most of the work to be done is on the Word document itself and the add-in interfaces aren't expected to be very extensive, most of the focus should be on the document.

The control subsystem will bring all other subsystems together and manage the communications between them. The subsystems should be independent modules that can operate by themselves so that they can be developed separately and be changed easily and effectively. The control subsystem will manage the exchanges between subsystems such as directing voice input from the voice input subsystem to be transcribed by the speech recognition subsystem.

2.3 Hardware and Software mapping

Our back-end side will be implemented in JavaScript and our front-end side will be implemented in HTML and CSS. Since Clerk will run on top of MS Word, it will work on every platform that Microsoft Word runs such as Windows operating system and it is going to be supported by the all versions of Word that support add-ins. The back-end side will run on the user's computer and it will use Word API service.

2.4 Persistent data management

In our application, we will not have an external database because of two reasons. First is that we don't need to store the document ourselves since Word has its own database management system and Clerk just an add-in on top of it. The other is, since Clerk will only change the content of document, we do not need to store any additional data for our application. The data that Clerk needs will be stored by Word, which is basically the document itself. Therefore, we do not need a database and the persistence of the data will be managed by Word. If a decision towards using improved models through the speech data of the user is made then addition of a database storing personalized information may need to be considered.

2.5 Access control and security

Users can use Word with or without their accounts. The process of registration is completely done by Word since Clerk is only a Word add-in. If the user enters Word with his/her account, s/he will be able to reach his/her previous files that are saved before on the database. Personal data of each user, who are registered, are kept in the database of Word. The authentication and authorization process of the progress being made by Word. Users do not need any other account for Clerk.

2.6 Global software control

Clerk uses event-driven software control. When the speech of the user is recognized by Clerk, it is converted into the text by model. Then, the text will be parsed and analyzed by Clerk to detect the reserved words and commands. After detecting commands, the corresponding feature of Word will be triggered by Clerk in the back-end.

2.7 Boundary conditions

Boundary conditions for Clerk are explained as follows.

2.7.1 Initialization

Since we assume that Word is already open, the initialization of Clerk is as easy as clicking to the logo of Clerk at the corner of MS Word window. It does not require any authentication.

2.7.2 Termination

The user can exit from Clerk at any desired time, since the user will not wait for any ongoing process to complete. Those processes will take place immediately. This termination can happen by saying the command for it and if there is an unsaved work, Clerk will terminate after it is saved.

2.7.3 Failure

Since Clerk will be built on top of Word, any failure in Word will affect us and unsaved work will be lost. In case of any failure in Clerk, related data is tried to be saved before termination. This can be achieved by automatic save every five minutes. Also, internet connection is required to use Clerk, lack of internet connection will cause failures in Clerk. A lack of internet connection should be communicated to the user appropriately.

3 Subsystem services

Clerk consists of a total of four subsystems that come together under a control subsystem that manages the general flow of the application.

3.1 Voice Input Management Subsystem

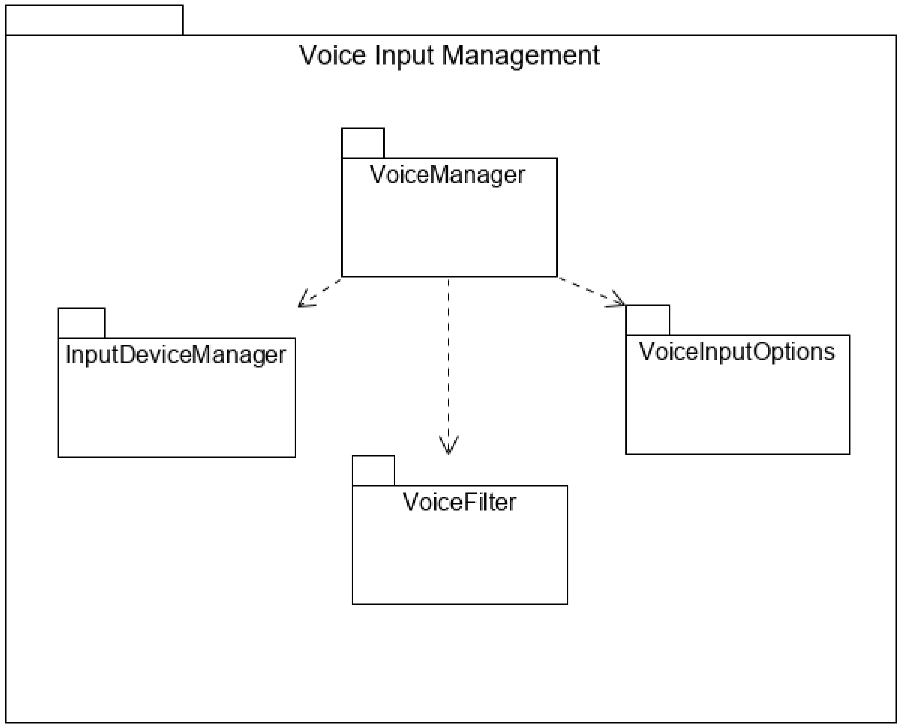

Figure 1: Voice Input Management Subsystem

Voice input management subsystem consists of utilities that are concerned with voice inputs.

InputDeviceManager: Responsible for finding and managing input devices.

VoiceFilter: Responsible for different kinds of signal processing that may be done on the voice input before being sent to the speech recognition subsystem.

VoiceInputOptions: Responsible for possible voice input settings that the user may want to change such as the input threshold.

3.2 Speech Recognition Subsystem

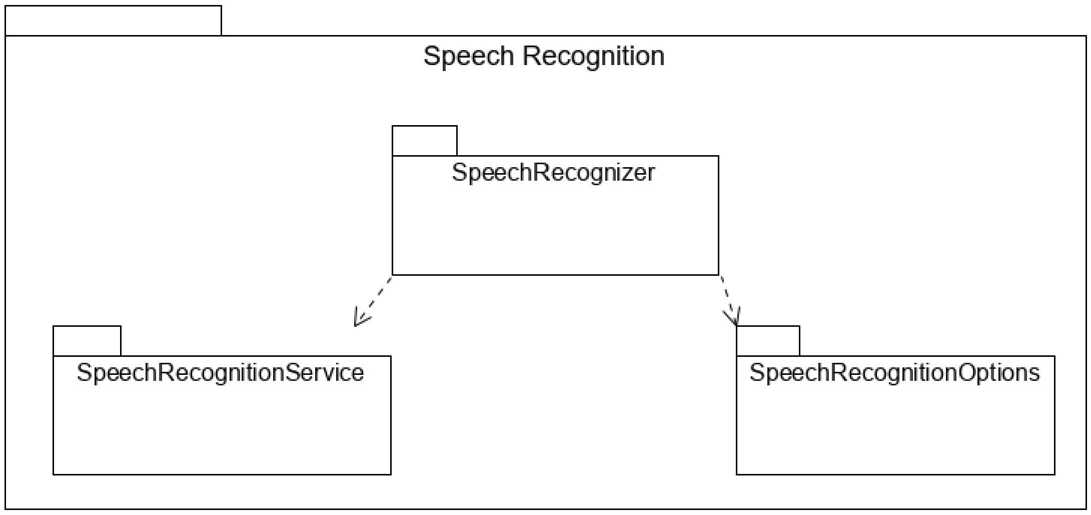

Figure 2: Speech Recognition Subsystem

Speech recognition subsystem consists of utilities that are concerned with interacting with speech recognition services.

SpeechRecognizer: A controller interface for the speech recognition service and speech recognition options.

SpeechRecognitionService: Responsible for connecting to, sending requests to and receiving replies from the specific speech recognition service used, such as Google Cloud Speech-to-Text or IBM Watson Speech-to-Text.

SpeechRecognitionOptions: Responsible for maintaining preferred options for speech recognition such as the language setting or the grammar to be used.

3.3 Command Management Subsystem

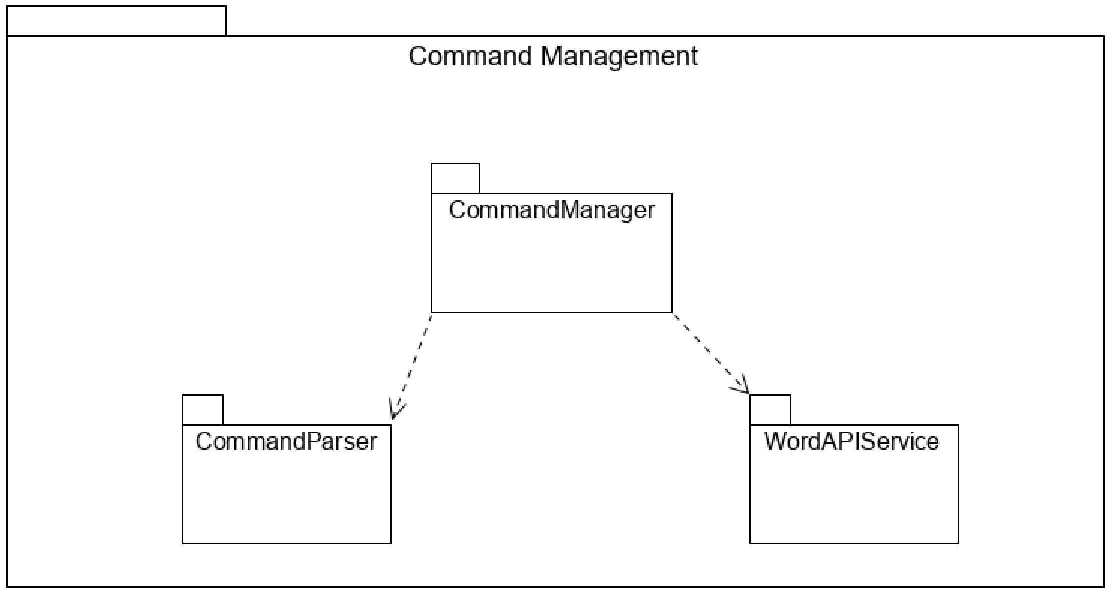

Figure 3: Command Management Subsystem

Command management subsystem consists of utilities to process the text received from the speech recognition service and apply the necessary changes to the Word document based on the input and the state.

CommandManager: A controller interface for the command parser and the Word API service.

CommandParser: Responsible for parsing commands sent in command mode. Decision of which command was sent is made by the parser so that the command can be executed by the WordAPIService.

WordAPIService: Responsible for interacting with the Word document at hand. The details of how to execute the different commands are abstracted by this class.

3.4 View Subsystem

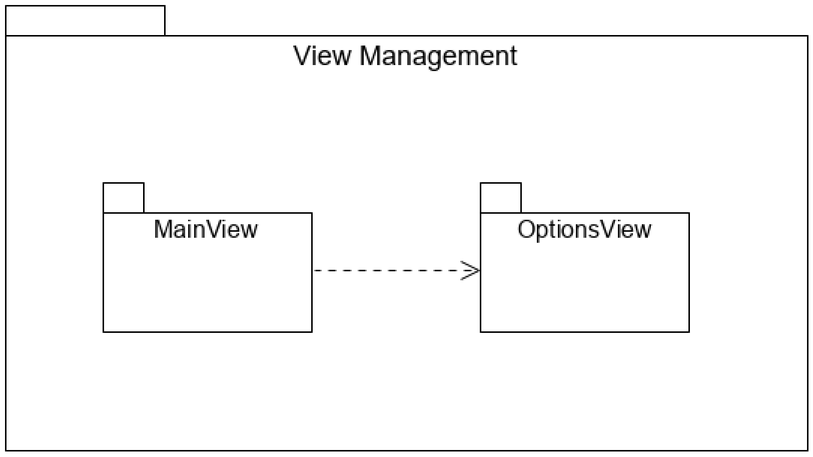

Figure 4: View Subsystem

View subsystem consists of the main view of the application and the options view. Describes the relatively simple user interface of Clerk.

Main View: Main view of the application.

Options View: Options view where the user can change values for some provided settings.

3.5 Control Subsystem

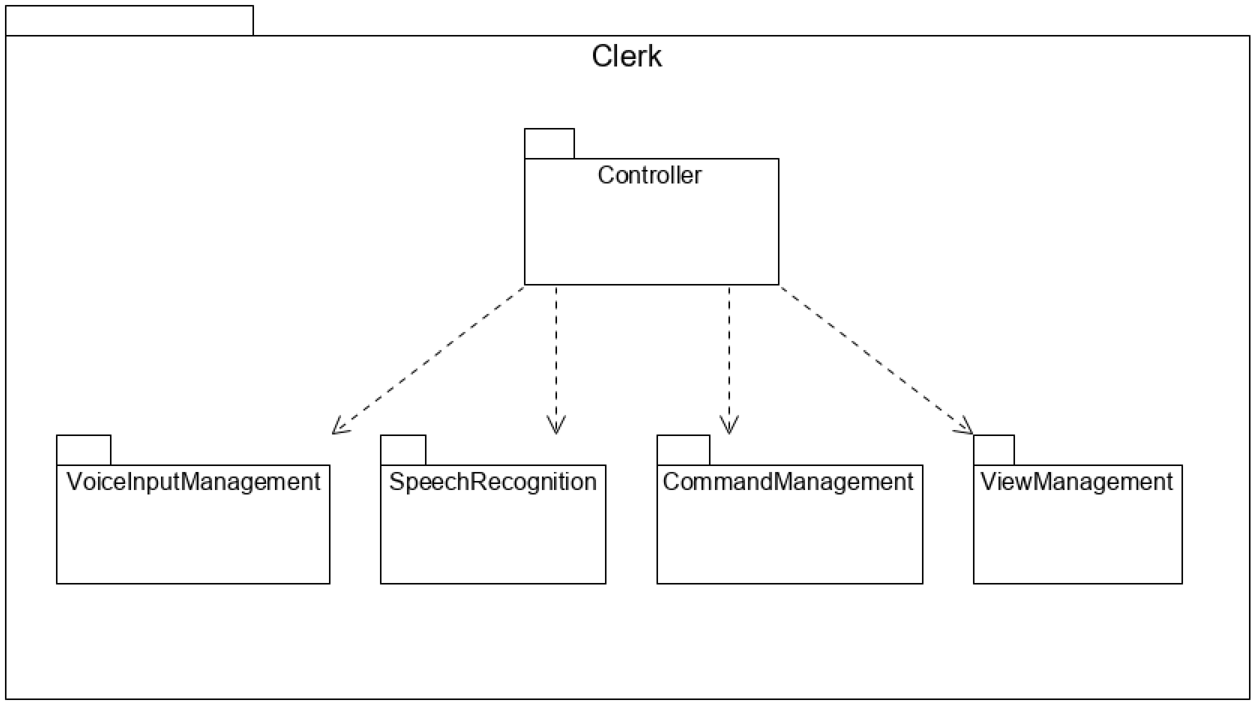

Figure 5: Control Subsystem

Control subsystem consists of all other subsystems of the application. It establishes protocols of how all other subsystems should interact with each other.

4 New Knowledge Acquired and Learning Strategies Used

One of the first goal that we have in work-package 1 was to evaluate the software libraries. Another one was to get familiar with JavaScript, HTML, and CSS. Last one was to create the blue-print for add-in.

To accomplish these goals we need to acquire new knowledge using some learning strategies. All of the group members have used online learning and learning by doing as the strategies to learn JavaScript, HTML, and CSS. During the evaluation of software libraries online learning and hands-on-experience have been used.

Each work-package leader is supposed to learn the responsibilities of a software project manager. So far only one group member has been the leader and rest will be one as well accordingly. These members will use online learning, reading a book, and learning by doing.

5 Glossary

Word add-in: An application that can be built and embedded inside Microsoft Word and work inside a Word instance that can extend the available functionality.

References

[1] "Word JavaScript API Overview - Office Add-ins | Microsoft Docs," https://docs.microsoft.com/en-us/office/dev/add-ins/reference/overview/word-add-ins-reference-overview?view=word-js-preview, Accessed: 2019-12-22.

[2] "Cloud Speech-to-Text - Speech Recognition | Cloud Speech-to-Text | Google Cloud," https://cloud.google.com/speech-to-text/, Accessed: 2019-12-22.

[3] "Watson Speech to Text - Overview | IBM Cloud," https://www.ibm.com/cloud/watson-speech-to-text, Accessed: 2019-12-22.