AdvancedText2SpeechEditor

Sprint Report

VRAZITOULIS ARISTEIDIS 4034

PAPADOPOULOS NIKOLAOS 4140

VERSIONS HISTORY

Date	Version	Description	Authors
27/03/2021	v0	Use cases definition and sprints organisation. After taking a close look at the user stories and the project requirements in general, we produced the use cases that would guide or implementation of the project and also set out the sprint plan in realistic, balanced timelines. We also made an initial CRC cards model that is open to change during the development process. This helped us get an initial understanding of the classes we will later develop.	Aristeidis Vrazitoulis, Papadopoulos Nikolaos
28/05/2021	v1	Filled out the report with the package/class diagrams and the architecture/design description. Changed some things about the CRC cards.	Aristeidis Vrazitoulis, Papadopoulos Nikolaos

1 Introduction

This document provides information concerning the sum of all project sprints.

TextToSpeechEditor is an interactive application, the main purpose of which is to allow the user to convert text from documents into audio. The application's interface allows the user to open .txt, .xlsx and .docx files, view their contents, edit them and play part of the text or even the whole text as audio. The user can save the edited contents, with the option of encoding before flushing to the disk. Of course, a decoding option appears when the user opens a file of his preference. In the current version, only Atbash and Rot-13 are supported. There is also the option to change audio parameters through the application's interface. Last but not least, we give the user the opportunity to record each text-to-speech operation, in order to be able to replay them all later.

1.1 Purpose

The purpose of this document is to present the way we went about the implementation of the project, according to the Scrum approach. This approach means that the team organizes a number of sprints during which it shall implement user stories from the project backlog and their acceptance tests.

1.2 Document Structure

The rest of this document is structured as follows. Section 2 describes our Scrum team and specifies the this Sprint's backlog. Section 3 specifies the main design concepts for this release of the project.

2 Scrum team and Sprint Backlog

2.1 Scrum team

Product Owner	Apostolos Zarras
Scrum Master	Aristeidis Vrazitoulis
Development Team	Nikolaos Papadopoulos, Aristeidis Vrazitoulis

2.2 Sprints

<List below the sprints that you performed and the user stories that have been realized in each Sprint>

Sprint No	Begin Date	End Date	Number of weeks	User stories
1.	**01/04/2021**	**15/04/2021**	**2 weeks**	**US1**
2.	**16/04/2021**	**30/04/2021**	**2 weeks**	**US2, US3**
3.	**01/05/2021**	**15/05/2021**	**2 weeks**	**US4, US5, US6**
4.	**16/05/2021**	**23/05/2021**	**1 week**	**US7, US8, US9**

2.3 Tests for each user story

User story ID	User story description	Corresponding test method
US1	Open a file stored on disk and view its contents.	testOpenDocument
US2	Edit the contents of the file.	testEditDocument
US3	Save the contents of the file on disk.	testSaveDocument
US4	Convert text to audio.	testDocumentToSpeech
US5	Convert **part of the text** to audio.	testDocumentToSpeechPartially
US6	Tune audio parameters (volume, speech rate and pitch).	testParametersSpeech
US7	Activate recording operation that keeps track of a sequence of text to audio transformation actions.	testRecordingCommand
US8	Replay the recorded sequence of actions.	testReplayCommand
US9	De-activate the recording operation.	testEndRecording

3 Use Cases

<Specify the concrete Use Cases that describe the interaction of the user with the applications, as derived from the abstract user stories. Give a **UML Use Case diagram** and the **detailed use case descriptions**.>

3.1 <Use Case 1>

Use case ID	OpenDocument
Actors	The user
Pre conditions	There needs to be a file with either a .docx, .xlsx or .txt extension.
Main flow of events	1 The use case starts when the user presses the "File" option. 2 The user selects the "Open" option. 3 A file chooser pops up. 4 The user selects the document. 5 The user selects decoding option
Post conditions	The content appears in the empty space.
Alternative flow	If the user closes the file chooser then he is allowed to do it again

3.2 <Use Case 2>

Use case ID	EditDocument
Actors	The user
Pre conditions	There needs to be a document opened in our stage.
Main flow of events	[Main flow of events that describes the interaction between the user and the application]

	1 The use case starts when the user makes a change in the editor. 2 The document object's data is updated.

3.3 <Use Case 3>

Use case ID	Save
Actors	The user
Pre conditions	There needs to be a document opened in our stage.
Main flow of events	1 The user presses the "save" button. 2 We store the changes to the disk, with the same decoding as it was opened with.

3.4 <Use Case 4>

Use case ID	SaveAs
Actors	The user
Pre conditions	There needs to be a document opened in our stage.
Main flow of events	1 The user presses the "SaveAs" button. 2 A File chooser opens up 3 The user is prompted to choose the file name and the extension 4 The user selects a encoding option
Post conditions	A new file is saved to disk

3.5 <Use Case 5>

Use case ID	DocumentToSpeech
Actors	The user
Pre conditions	There needs to be a document opened in our stage.
Main flow of events	1 The user presses the button “Convert Text”. 2 The system converts text to speech. 3.1 If the user has not selected a part of the text, then we convert the whole text. 3.2 If the user has selected a part of the text, then we convert the selected part.
Post conditions	The audio is played

3.6 <Use Case 6>

Use case ID	TuneAudioParameters
Actors	The user
Pre conditions	-
Main flow of events	1 The system shows a layout with tuning option on the bottom. 2 The user adjusts the parameters with a slider ● The user can adjust the volume. ● The user can adjust the pitch. ● The user can adjust the speech rate incrementally with slow or fast buttons.
Post conditions	The system applies the audio parameter changes.

3.7 <Use Case 7>

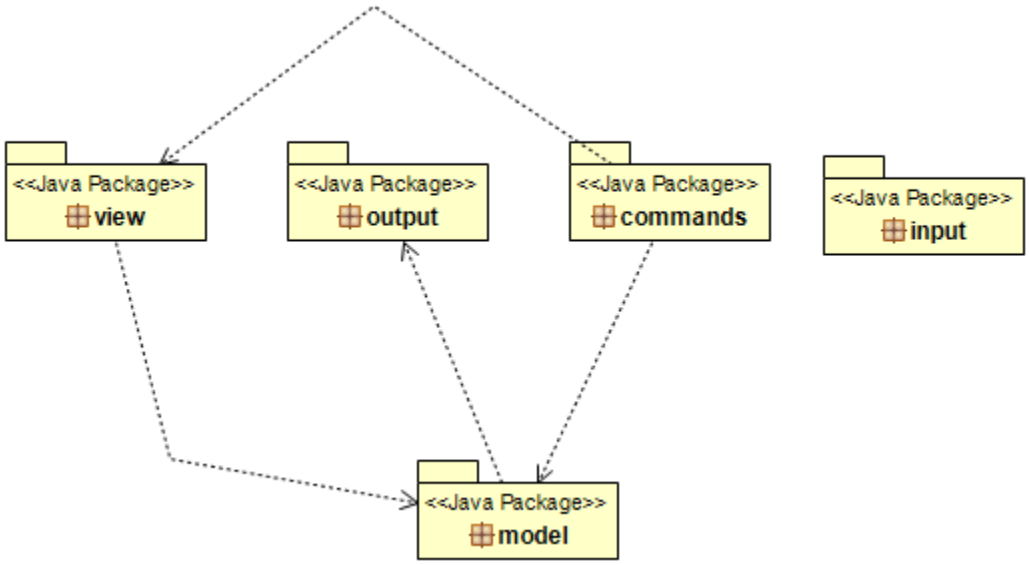
Use case ID	Record
Actors	The user
Pre conditions	There needs to be an opened file.
Main flow of events	[Main flow of events that describes the interaction between the user and the application] 1 The user presses the “recording” button. 2 The system for every conversion saves the DocumentToSpeech object to a list. 3 The replay button is enabled
Alternative flow 1	If the user presses the same button then the recording is deactivated and the sequence is flushed, disabling the replay button.
Post conditions	The recording boolean parameter is updated to true

3.8 <Use Case 8>

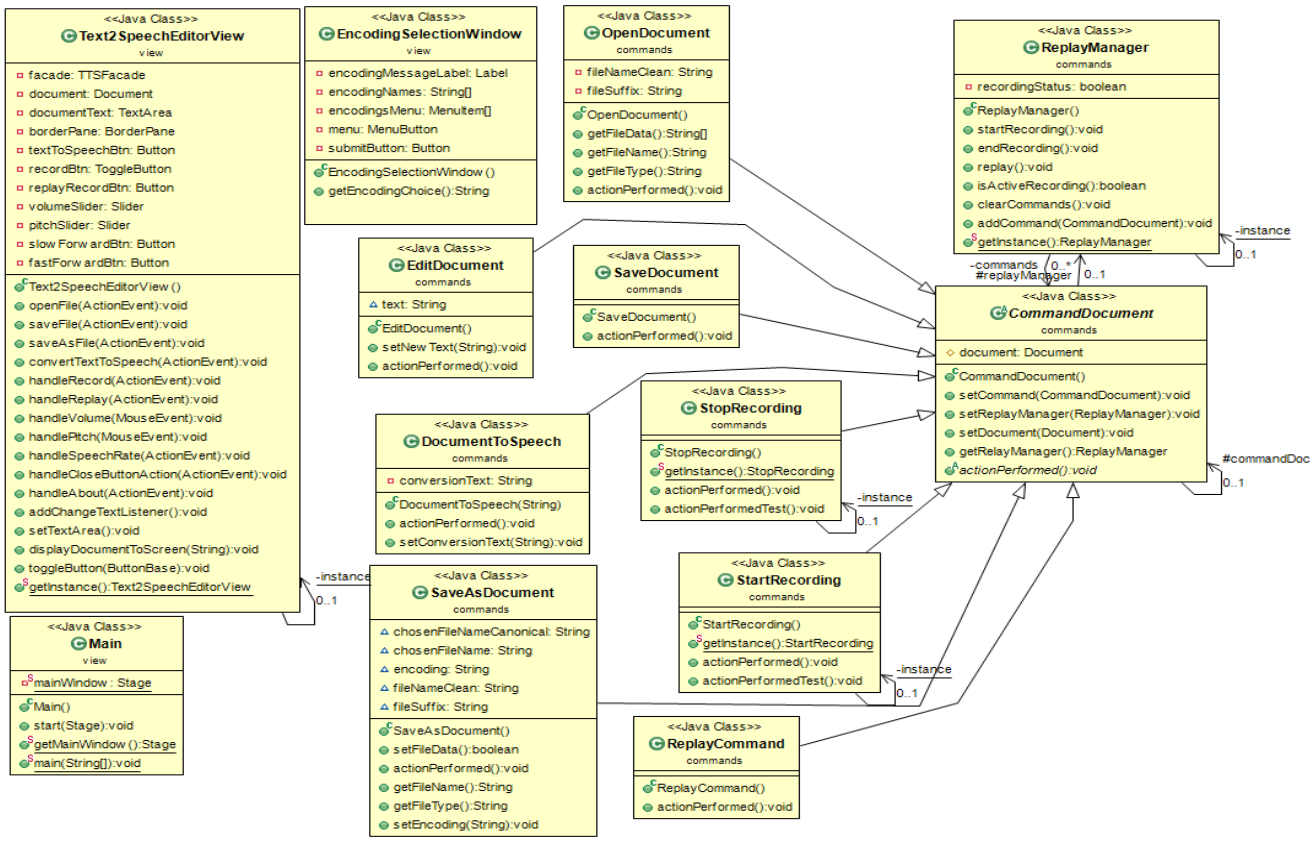
Use case ID	Replay
Actors	The user
Pre conditions	There needs to be an opened file.
Main flow of events	[Main flow of events that describes the interaction between the user and the application] 1 The user presses the “replay” button. 2 The system iterates the list with DocumentToSpeech objects and executes the audio sequence
Post conditions	The system applies the audio parameter changes.

4. Design

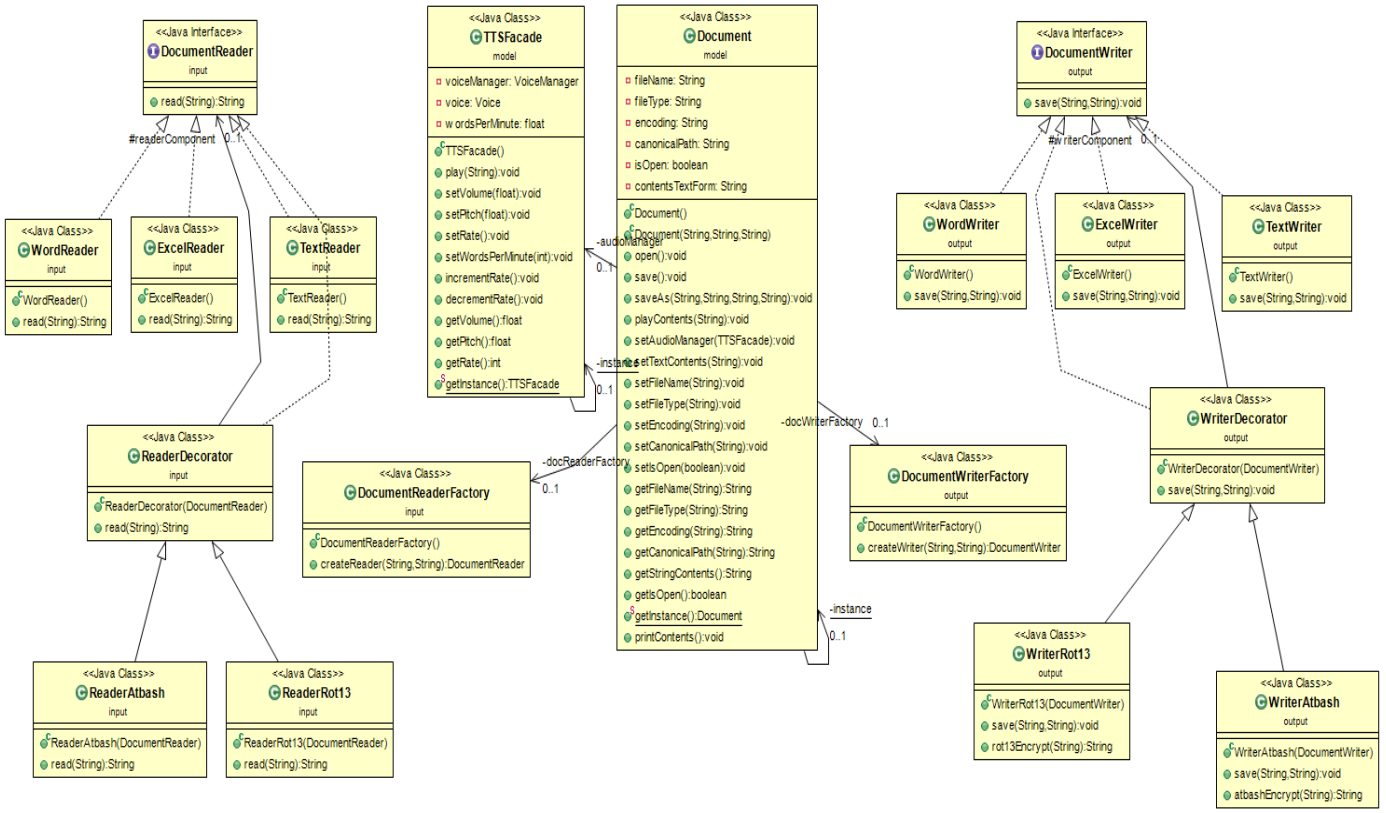
4.1 Architecture



4.2 Design



FRONT-END DIAGRAM



BACK-END DIAGRAM

Class Name: WordWriter	
Responsibilities: ▪ Save data in a word document on disk	**Collaborations:** ▪ DocumentWriter ▪ Document

Class Name: ExcelWriter	
Responsibilities:	**Collaborations:** ▪ DocumentWriter

▪ Save data in an excel document on disk	▪ Document

Class Name: TextWriter	
Responsibilities: ▪ Save data in a text document on disk	**Collaborations:** ▪ DocumentWriter ▪ Document

Class Name: DocumentWriter	
Responsibilities: ▪ Java interface that defines methods for writers	**Collaborations:** ▪ DocumentWriterFactory

Class Name: WriterDecorator	
Responsibilities: ▪ Used for easier addition of new encoding methods to the application	**Collaborations:** ▪ DocumentWriter

Class Name: WriterRot13	
Responsibilities: ▪ Encrypt file data with the Rot-13 encoding method ▪ Save encoded data on disk	**Collaborations:** ▪ WriterDecorator

Class Name: WriterAtbash	
Responsibilities: ▪ Encrypt file data with the Atbash encoding method	**Collaborations:** ▪ WriterDecorator

▪ Save encoded data on disk	

Class Name: DocumentWriterFactory	
Responsibilities: ▪ Create an instance of a writer for a particular file format, with a certain canonical path	**Collaborations:** ▪ Document

Class Name: Document	
Responsibilities: ▪ Store information about the current opened document in the application ▪ Get information about the current opened document in the application ▪ Front-end – back-end connector	**Collaborations:** ▪ DocumentWriterFactory ▪ DocumentReaderFactory ▪ TTS Façade ▪ Text2SpeechEditorView

Class Name: DocumentReaderFactory	
Responsibilities: ▪ Create an instance of a reader for a particular file format, with a certain canonical path	**Collaborations:** ▪ Document

Class Name: ReaderDecorator	
Responsibilities: ▪ Used for easier addition of new decoding methods to the application	**Collaborations:** ▪ DocumentReader

Class Name: ReaderRot13	
Responsibilities:	**Collaborations:**

▪ Decrypt file data using the Rot-13 decoding method ▪ Feed the text box with the file data	▪ ReaderDecorator

Class Name: ReaderAtbash	
Responsibilities: ▪ Decrypt file data using the Atbash decoding method ▪ Feed the text box with the file data	**Collaborations:** ▪ ReaderDecorator

Class Name: WordReader	
Responsibilities: ▪ Read a word document from the disk	**Collaborations:** ▪ DocumentReader ▪ Document

Class Name: ExcelReader	
Responsibilities: ▪ Read an excel document from the disk	**Collaborations:** ▪ DocumentReader ▪ Document

Class Name: TextReader	
Responsibilities: ▪ Read a text document from the disk	**Collaborations:** ▪ DocumentReader ▪ Document

Class Name: DocumentReader

Responsibilities:	**Collaborations:**
▪ Java interface that defines methods for readers	▪ DocumentReaderFactory

Class Name: EditDocument	
Responsibilities:	**Collaborations:**
▪ Change the file data, according to the text box changes made by the user during application runtime with a document opened	▪ CommandDocument ▪ Text2SpeechEditorView ▪ Document

Class Name: SaveDocument	
Responsibilities:	**Collaborations:**
▪ Save the current data of the file to a file on disk with the same encoding and format as the initial file	▪ CommandDocument ▪ Text2SpeechEditorView ▪ Document

Class Name: SaveAsDocument	
Responsibilities:	**Collaborations:**
▪ Save the current data of the file to a file on disk with encoding and format specified by the user (may be new)	▪ CommandDocument ▪ Text2SpeechEditorView ▪ EncodingSelectionWindow

	▪ Document

Class Name: ReplayManager	
Responsibilities: ▪ Start recording operation ▪ End recording operation ▪ Add text-to-speech command ▪ Replay recorded text-to-speech commands	**Collaborations:** ▪ CommandDocument ▪ Text2SpeechEditorView

Class Name: StartRecording	
Responsibilities: ▪ Initialise a recording operation (change the recording status to "ON")	**Collaborations:** ▪ CommandDocument ▪ Text2SpeechEditorView

Class Name: StopRecording	
Responsibilities: ▪ End a recording operation (change the recording status to "OFF")	**Collaborations:** ▪ CommandDocument ▪ Text2SpeechEditorView

Class Name: ReplayCommand

Responsibilities:	Collaborations:
▪ Initialise replay operation of recorded text-to-speech commands	▪ CommandDocument ▪ Text2SpeechEditorView ▪ ReplayManager

Class Name: CommandDocument	
Responsibilities:	**Collaborations:**
▪ Set/get the replay manager ▪ Add command to the replay manager ▪ Give instructions to the replay manager	▪ ReplayManager ▪ StartRecording ▪ StopRecording ▪ ReplayCommand ▪ SaveDocument ▪ EditDocument ▪ DocumentToSpeech ▪ SaveAsDocument

Class Name: DocumentToSpeech	
Responsibilities:	**Collaborations:**
▪ Set the text to be converted to speech	▪ CommandDocument ▪ Text2SpeechEditorView

Class Name: OpenDocument

Responsibilities:	**Collaborations:**
▪ Open a file from disk	▪ CommandDocument ▪ Text2SpeechEditorView ▪ Document

Class Name: EncodingSelectionWindow	
Responsibilities:	**Collaborations:**
▪ Handle the process of showing a window where the user can select encoding ▪ Save the choice	▪ Text2SpeechEditorView

Class Name: Text2SpeechEditorView	
Responsibilities:	**Collaborations:**
▪ Show the application's graphical interface ▪ React to buttons pressed/sliders changed with the corresponding system methods ▪ Toggle buttons ▪ Change audio parameters ▪ Store audio parameters ▪ Display document data to the screen	▪ Main