INITIAL SOFTWARE DESIGN DOCUMENT

FOR

PPTX TO HTML5 CONTENT CONVERTER

PREPARED BY Limon

Shamil Farajullayev

Rustam Hashimov

Nahid Hamidli

Ömer Baykal

this page is intentionally left blank

Table of Contents

1. Introduction

This is the Initial Software Design Document of “PPTX to HTML5 Content Converter” project. This document provides how the software system will be designed to satisfy the requirements and functionalities that were stated in the Software Requirements Analysis Document of the project [1]. It is the crucial document for developers of the project, since it includes details that ease the implementation of the project. This document is written based on standard of SDD writing that are stated in IEEE Std 1016-1998: IEEE Recommended Practice for Software Design Descriptions [2].

1.1.Problem Definition

Educational institutions that have their lectures online and departments of companies that are responsible from education of employees are faced with the problem that they are unable to keep track of the development of their students. To solve this problem, for example, companies are making lecturers to go to the towns of employees and educate them by arranging instructive meetings. Some companies even bringing their employees to headquarters of the company and arranging meetings in there. These methods are seemed to solve the problem but they also have some problems. They are not much efficient and they cost the companies a lot of time, effort and money. Hence, they are started to look at for alternative solutions like cooperating with consulting companies. Since most of the documents prepared by lecturers are presented as PowerPoint slides and it is impossible to follow the process of learning, these consulting companies convert the PowerPoint files to e-learning packages manually. We intend to solve this difficulty by developing a Microsoft PowerPoint plug-in which will automatically convert PPTX documents to HTML5 formatted documents without almost any human effort while keeping the structure of slides. The software we intended to create will have two main advantages.

Firstly, the created product from PPTX file will obey the SCORM [3] standard. The SCORM standard defines communication between client side content and a host system (generally referred as learning management system) and how content of may be packaged into a transferable ZIP file “Package Interchange Format”. So, obeying SCORM standard will provide the product ability of being tracked.

Secondly, the created product will keep its content in HTML5 [4] format. HTML5 files are accessible from any platform and its only requirement is a web browser. HTML5 will provide the product independency. Users of the product will be able to use the system from any environment; any operating system (like MAC OS, Windows, iOS, Android and Linux distributions), any device (like MACs, PCs, Smart Phones, Tablets); and any time they demand.

1.2.Purpose

In the Software Requirements Analysis Document of the PPTX to HTML5 Content Converter project [1]; desired features, functionalities and requirements were stated. This Software Design Document is intended to create a software design which will satisfy the stated functionalities and requirements based on the constraints and assumptions made in SRS of the project. This document is intended to be viewed by development team of the project because design issues are mainly related to development phase of the project and therefore they are related to developers of the project.

1.3.Scope

Scope of this Software Design Document is to explain design related issues of the project. This document is an Initial Software Design Document. Therefore, it does not include a detailed design. It includes the overall system architecture and data architecture. Design considerations like assumptions, dependencies, and constraints; design of user interfaces; development schedule and development tools are also covered in this document.

1.4.Overview

This first chapter of this ISDD is an introduction to the project. It includes information about this document (its purpose, scope and contents), the clear statement of problem that the project is intending to solve, definitions and abbreviations used and references referred throughout this document. The rest of the document covers the sections, respectively:

System Overview; provides general description of the software system. It explains major parts of the software to be created.

Design Considerations; addresses special design issues which need to be resolved before dividing into design. Some examples are time and performance constraints, hardware or software related limitations.

Data Design; explains how data is stored, processed and organized in the product.

System Architecture; provides description of the program architecture.

User Interface Design; covers the information of how the users of the product will be able to use the system. Also there are some screen images of designed user interfaces.

Libraries and Tools; lists the libraries and tools that are planned to be used throughout the development phase of the project.

Time Planning; demonstrates time planning and scheduling issues by representing a Gantt Chart.

Conclusion; sums up this Initial Software Design Document.

1.5.Definitions, Acronyms and Abbreviations

Definitions, acronyms and abbreviations that are used throughout this document are listed in the following table.

The Project	PPTX to HTML5 Content Converter Project
SRS	Software Requirements Specification
SDD	Software Design Document
ISDD	Initial Software Design Document
SCORM	Sharable Content Object Reference Module
IDE	Integrated Development Environment
ECMA	An international standards organization for Information Communication Technology and Consumer Electronics
ISO	International Organization for Standardization
JSON	Java Script Object Notation

1.6.References

[1] Software Requirements Specification of PPTX to HTML5 Content Converter Project

The SRS document of PPTX to HTML5 Content Converter Project, written according to IEEE Std 830-1998: IEEE Recommended Practice for Software Requirements Specifications.

[2] IEEE Std 1016-1998: IEEE Recommended Practice for Software Design Descriptions

A common software engineering standard to provide some guidance and recommended approaches for specifying software design descriptions.

[3] SCORM Standard,

http://en.wikipedia.org/wiki/Sharable_Content_Object_Reference_Model

[4] Specifications of HTML5 in W3 Consortium Page for Developers,

http://dev.w3.org/html5/spec/Overview.html

[5] Microsoft Visual C#,

http://en.wikipedia.org/wiki/Microsoft_Visual_C_Sharp

[6] John Sharp, “Microsoft Visual C# 2010 Step by Step”, Microsoft Press, 2010

[7] Microsoft .NET Framework,

http://en.wikipedia.org/wiki/.NET_Framework

[8] Visual Studio in Microsoft Developer Network Page (MSDN),

http://msdn.microsoft.com/en-us/library/fx6bk1f4.aspx

[9] Getting Started with the Open XML SDK 2.0 in Microsoft Developer Network Page (MSDN),

http://msdn.microsoft.com/en-us/library/bb456488.aspx

[10] JSON in Wikipedia,

http://en.wikipedia.org/wiki/JSON

[11] User Control Class Documentation in MSDN

http://msdn.microsoft.com/en-us/library/system.web.ui.usercontrol.aspx

[12] Ribbon Class Documentation in MSDN

http://msdn.microsoft.com/en-us/library/microsoft.office.tools.ribbon.aspx

2. System Overview

As explained in detail at the Problem Definition section (1.1), the aim of our product is to make it easier for the educators to keep the track of the employee training programs, so that they can know how much of the presentation slides they have learnt. But how this can be achieved? What should be concerned while implementing the application in terms of design? User interface had to be very easy, since the consumer might have almost no PowerPoint knowledge. That's why we created an add-in for PowerPoint application of Microsoft Office so that every operation happens inside a single program. In addition, we have decided to implement object oriented programming to treat every element as a separate object.

As the means of interface; Template, Interface and general Add-in (Ribbon) windows will sit inside the PowerPoint. By a few mouse clicks only, output will be ready. Open XML SDK will be used to parse the XML files to ParsedSlides format (which will be described in Detailed Design Report) taken from PPTX files. Hierarchy window will let the user see and edit hierarchy of slides in real time. Template window will let the users to select the best fitting template for each slide, so that our static HTML generator creates the appropriate HTML5 code. By pressing publish button in Add-in tab interface, user will be asked to choose the save directory, and after HTML5 files are crated for each slide, a SCORM based compressed file will be published.

3. Design Considerations

In this chapter of SDD, some design considerations about the project PPTX to HTML5 Content Converter such as basic assumptions, constraints and dependencies as well as goals and guidelines will be described.

3.1.Design Assumptions, Dependencies and Constraints

General assumptions, dependencies and constraints are discussed respectively in the following subsections.

3.1.1. Assumptions

- First goal is to implement a PowerPoint add-in as a final product. However, depending on the circumstances (based on the permissions to developers given by Microsoft Office PowerPoint) a windows application can also be developed separately.
- It is assumed that users of system will be aware that video and audio contents in slides will not be transferred to the final outcome of conversion process.

3.1.2. Dependencies

- Microsoft Visual Studio 2010.
- Open XML SDK v2.0.
- Microsoft Office PowerPoint 2007 or a later version.
- Microsoft Office running on machines with Microsoft Windows.

3.1.3. Constraints

- **Time Constraint:** Project will be finalized by the mid of June 2012.
- **Performance Constraint:** The conversion duration will be depends on both device and input PPTX file's size. On the other hand, it is desired and planned that the conversion process will finish in a few seconds.

3.2.Design Goals and Guidelines

3.2.1. KISS Principle

KISS is the abbreviation for "Keep it simple, Stupid". As the name says it all, we tried to keep our product as simple as possible to make it easy for both developer to develop, and the user to use.

3.2.2. Usability

Since our product will be implemented as an add-in to Microsoft Office PowerPoint, it will provide the user ability to see the add-in specific windows and original PowerPoint windows at the same time. Therefore, user sees every change made at real time without leaving

PowerPoint application. If the product was an application, it would be weaker in the means of interface, consequently making the application difficult to use.

4. Data Design

4.1.Data Description

Several files are processed during the process of conversion and running of the system. In addition some files are created and stored by the system that keeps the changes made by users before the conversion that makes the conversion and final product more effective. In following subsections, detailed information can be found.

4.1.1. External Data Objects

4.1.1.1. PresentationML

Using the Open XML SDK 2.0, we can create document structure and content that uses strongly-typed classes that correspond to PresentationML elements. The following table lists the class names of the classes that correspond to some of the important presentation elements.

Package Part:	Top Level PresentationML Element:	Open XML SDK 2.0 Class:	Description:
Presentation	<presentation>	Presentation	The root element for the Presentation part. This element specifies within it fundamental presentation-wide properties.
Presentation Properties	<presentationPr>	PresentationProperties	The root element for the Presentation Properties part. This element functions as a parent element within which additional presentation-wide document properties are contained.
Slide Master	<sldMaster>	SlideMaster	The root element for the Slide Master part. Within a slide master slide are contained all elements that describe the

			objects and their corresponding formatting for within a presentation slide.
Slide Layout	<sldLayout>	SlideLayout	The root element for the Slide Layout part. This element specifies the relationship information for each slide layout that is used within the slide master.
Theme	<officeStyleSheet>	Theme	The root element for the Theme part. This element holds all the different formatting options available to a document through a theme and defines the overall look and feel of the document when themed objects are used within the document.
Slide	<sld>	Slide	The root element for the Slide part. This element specifies a slide within a slide list.

Presentation Class:

This element specifies within it fundamental presentation-wide properties.

Parent Elements:
Root element of PresentationML Presentation part

Child elements:
custDataLst (Customer Data List)
defaultTextStyle (Presentation Default Text Style)
embeddedFontLst (Embedded Font List)
extLst (Extension List)
handoutMasterIdLst (List of Handout Master IDs)
kinsoku (Kinsoku Settings)
modifyVerifier (Modification Verifier)

notesMasterIdLst (List of Notes Master IDs)
notesSz (Notes Slide Size)
photoAlbum (Photo Album Information)
sldIdLst (List of Slide IDs)
sldMasterIdLst (List of Slide Master IDs)
sldSz (Presentation Slide Size)
smartTags (Smart Tags)

Attributes:	Description:
autoCompressPictures (Automatically Compress Pictures)	Specifies whether the generating application should automatically compress all pictures for this presentation. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
bookmarkIdSeed (Bookmark ID Seed)	Specifies a seed for generating bookmark IDs to ensure IDs remain unique across the document. This value specifies the number to be used as the ID for the next new bookmark created.
compatMode (Compatibility Mode)	Specifies whether the generating application is to be in a compatibility mode which serves to inform the user of any loss of content or functionality when working with older formats.
conformance (Document Conformance Class)	Specifies the conformance class to which the PresentationML document conforms. If this attribute is omitted, its default value is transitional. [Example: Consider the following PresentationML Presentation part markup: XML <p:presentation conformance="strict"> _ </p:presentation> This document has a conformance attribute value of strict, therefore it conforms to the PML Strict conformance class end example] The possible values for this attribute are defined by the ST_ConformanceClass simple type.
embedTrueTypeFonts (Embed True Type Fonts)	Specifies whether the generating application should automatically embed true type fonts or not.

	The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
firstSlideNum (First Slide Number)	Specifies the first slide number in the presentation. The possible values for this attribute are defined by the W3C XML Schema int datatype.
removePersonalInfoOnSave (Remove Personal Information on Save)	Specifies whether to automatically remove personal information when the presentation document is saved. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
rtl (Right-To-Left Views)	Specifies if the current view of the user interface is oriented right-to-left or left-to-right. The view is right-to-left is this value is set to true, and left-to-right otherwise. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
saveSubsetFonts (Save Subset Fonts)	Specifies to save only the subset of characters used in the presentation when a font is embedded. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
serverZoom (Server Zoom)	Specifies the scaling to be used when the presentation is embedded in another document. The embedded slides are to be scaled by this percentage. The possible values for this attribute are defined by the ST_Percentage simple type.
showSpecialPlsOnTitleSld (Show Header and Footer Placeholders on Titles)	Specifies whether to show the header and footer placeholders on the title slides. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
strictFirstAndLastChars (Strict First and Last Characters)	Specifies whether to use strict characters for starting and ending lines of Japanese text. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.

PresentationProperties Class:

This element functions as a parent element within which additional presentation-wide document properties are contained. All properties and their corresponding settings are defined within the child elements.

Parent Elements:
Root element of PresentationML Presentation Properties part

Child elements:
clrMru (Color MRU)
extLst (Extension List)
prnPr (Printing Properties)
showPr (Presentation-wide Show Properties)

SlideMaster Class:

This element specifies an instance of a slide master slide. Within a slide master slide are contained all elements that describe the objects and their corresponding formatting for within a presentation slide. Within a slide master slide are two main elements. The cSld element specifies the common slide elements such as shapes and their attached text bodies. Then the txStyles element specifies the formatting for the text within each of these shapes. The other properties within a slide master slide specify other properties for within a presentation slide such as color information, headers and footers, as well as timing and transition information for all corresponding presentation slides.

Parent Elements:
Root element of PresentationML Slide Master part

Child elements:
clrMap (Color Scheme Map)
cSld (Common Slide Data)
extLst (Extension List with Modification Flag)
hf (Header/Footer information for a slide master)
sldLayoutIdLst (List of Slide Layouts)
timing (Slide Timing Information for a Slide Layout)
transition (Slide Transition for a Slide Layout)
txStyles (Slide Master Text Styles)

Attributes:	Description:
preserve (Preserve Slide	Specifies whether the corresponding slide layout is deleted

Master)	when all the slides that follow that layout are deleted. If this attribute is not specified then a value of false should be assumed by the generating application. This would mean that the slide would in fact be deleted if no slides within the presentation were related to it. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.

SlideLayout Class:

This element specifies an instance of a slide layout. The slide layout contains in essence a template slide design that can be applied to any existing slide. When applied to an existing slide all corresponding content should be mapped to the new slide layout.

Parent Elements:
Root element of PresentationML Slide Layout part

Child elements:
clrMapOvr (Color Scheme Map Override)
cSld (Common Slide Data)
extLst (Extension List with Modification Flag)
hf (Header/Footer information for a slide master)
timing (Slide Timing Information for a Slide Layout)
transition (Slide Transition for a Slide Layout)

Attributes:	Description:
matchingName (Matching Name)	Specifies a name to be used in place of the name attribute within the cSld element. This is used for layout matching in response to layout changes and template applications. The possible values for this attribute are defined by the W3C XML Schema string datatype.
preserve (Preserve Slide Layout)	Specifies whether the corresponding slide layout is deleted when all the slides that follow that layout are deleted. If this attribute is not specified then a value of false should be assumed by the generating application. This would mean that the slide would in fact be deleted if no slides within the presentation were related to it. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.

showMasterPhAnim (Show Master Placeholder Animations)	Specifies whether or not to display animations on placeholders from the master slide. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
showMasterSp (Show Master Shapes)	Specifies if shapes on the master slide should be shown on slides or not. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
type (Slide Layout Type)	Specifies the slide layout type that is used by this slide. The possible values for this attribute are defined by the ST_SlideLayoutType simple type.
userDrawn (Is User Drawn)	Specifies if the corresponding object has been drawn by the user and should thus not be deleted. This allows for the flagging of slides that contain user drawn data. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.

Theme Class:

This element defines the root level complex type associated with a shared style sheet (or theme). This element holds all the different formatting options available to a document through a theme and defines the overall look and feel of the document when themed objects are used within the document.

Parent Elements:
Root element of DrawingML Theme part

Child elements:
custClrLst (Custom Color List)
extLst (Extension List)
extraClrSchemeLst (Extra Color Scheme List)
objectDefaults (Object Defaults)
themeElements (Theme Elements)

Attributes:	Description:
name (Name)	Specifies the name given to the theme.

	The possible values for this attribute are defined by the W3C XML Schema string datatype.

Slide Class:

This element specifies a slide within a slide list. The slide list is used to specify an ordering of slides

Parent Elements:
Root element of PresentationML Slide part

Child elements:
clrMapOvr (Color Scheme Map Override)
cSld (Common Slide Data)
extLst (Extension List with Modification Flag)
timing (Slide Timing Information for a Slide Layout)
transition (Slide Transition for a Slide Layout)

Attributes:	Description:
show (Show Slide in Slide Show)	Specifies that the current slide should be shown in slide show. If this attribute is omitted then a value of true is assumed. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
showMasterPhAnim (Show Master Placeholder Animations)	Specifies whether or not to display animations on placeholders from the master slide. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
showMasterSp (Show Master Shapes)	Specifies if shapes on the master slide should be shown on slides or not. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.

4.1.1.2. HTML5

As stated before, HTML5 is the last version of HTML. So, all the tags, attributes, properties supplied by the HTML including the new ones and the APIs provided that comes with HTML5 will be used. Since the output data will be stored as HTML formatted files, the structure of the files determined by HTML standards.

4.1.2. Internal Data Objects

4.1.2.1. Hierarchy Tree Storage

Since users of the software product will add some hierarchy information to the process and document, that information has to be kept. It is decided that it will be stored in XML files; so that users will be able to reuse the hierarchy they have structured before and also will be able to modify it. Detailed information will be provided in Detailed Design Report.

4.1.2.2. Template Storage

The software will have built-in templates stored in it. Those templates will be represented in Template Editor Window Interface. Just like with Hierarchy Editor, users will be able to set templates for slides manually. Again to make those choices reusable by the users, they should be stored. Template choice information is also going to be kept as XML files. Detailed information will be provided in Detailed Design Report.

4.2.Data Dictionary

Function	Reference
int index(void)	5.2.2.3
int parent(void)	5.2.2.3
Vector<int> children(void)	5.2.2.3
int which_child(void)	5.2.2.3
int set_parent(int)	5.2.2.3
int add_child(int)	5.2.2.3
int remove_child(int)	5.2.2.3
int ShowTree(int)	5.2.2.3
int convert(string)	5.2.4.3
int createHtml(void)	5.2.5.3
int savefile(void)	5.2.5.3
void Add-In_Ribbon_Load (object sender, RibbonUIEventArgs e)	5.2.1.3
void publish_button_Click (object sender, RibbonControlEventArgs e)	5.2.1.3
void hierarchy_button_Click (object sender, RibbonControlEventArgs e)	5.2.1.3

void template_button_Click (object sender, RibbonControlEventArgs e)	5.2.1.3
void TemplateEditor_Load (object sender, EventArgs e)	5.2.3.3.1
void ShowTemplates(Template[])	5.2.3.3.1
void Preview(Slide)	5.2.3.3.1
void apply_button_Click(object sender, EventArgs e)	5.2.3.3.1
void preview_button_Click(object sender, EventArgs e)	5.2.3.3.1
Slide Convert(Slide activeSlide, Template template)	5.2.3.3.2
void Save ()	5.2.3.3.2

5. System Architecture

The converter application will be highly object oriented. Architecture will contain a Ribbon class for the add-in tab, 2 interface classes namely Hierarchy class and Template class. However, those classes will not be doing any conversion or calculation. These will be handled by Hierarchy editor class, Template editor class, Node class, XmlParser class and Publish class which will be described in detail in section 5.2.

5.1.Architectural Design

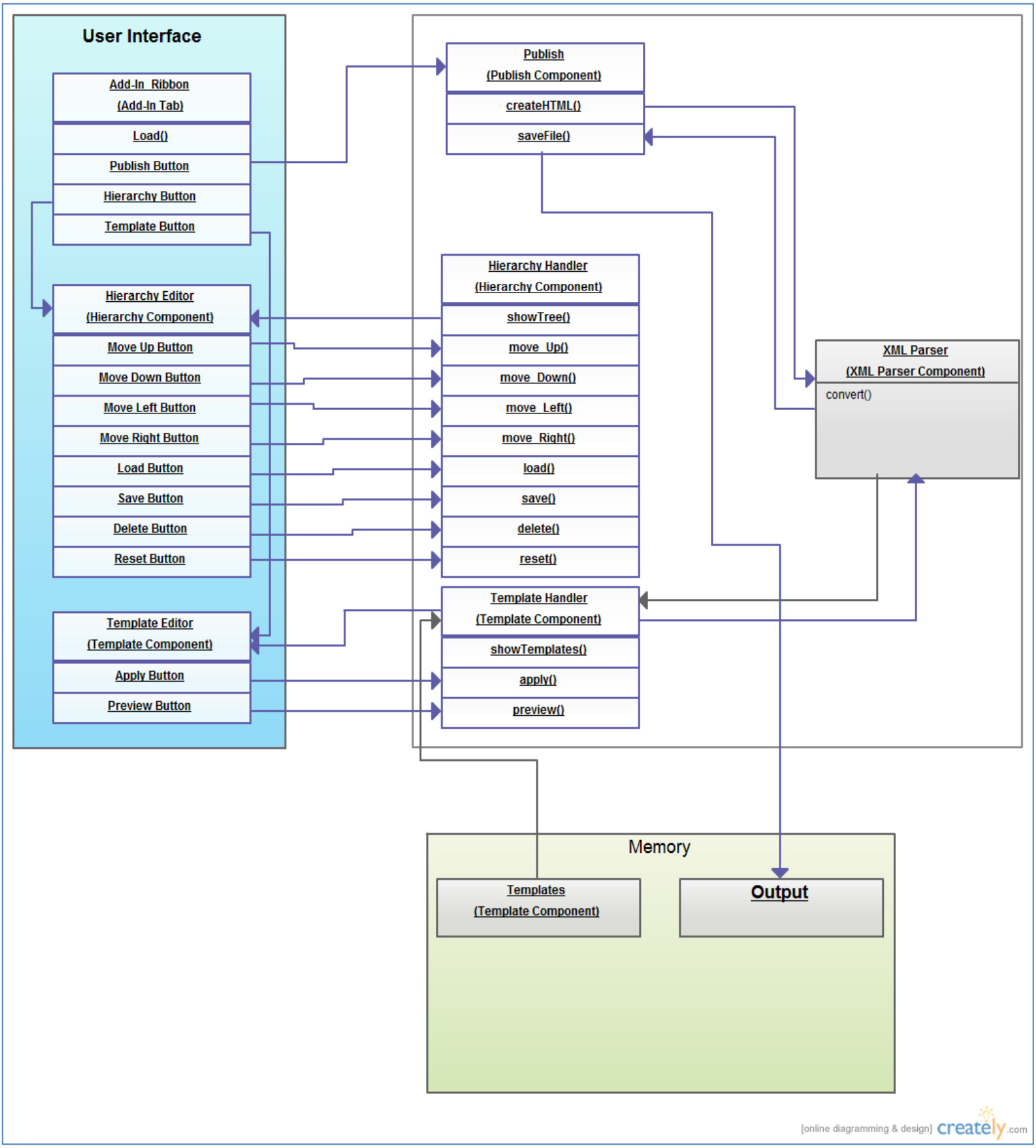


Figure 5.1: General Architectural Design

5.2.Description of Components

5.2.1. Add-In Tab Component

Add-In Tab Component is main user interface part of software. This component creates a tab in Microsoft PowerPoint. The tab contains buttons: "Publish", "Hierarchy" and "Template". By pressing these three buttons, user starts to use the three main functionalities: Publish, Hierarchy Editor, Template Editor. The component has a class, named Add-In Ribbon. This class is derived from OfficeRibbon Interface [11].

5.2.1.1. Processing Narrative of Add-In Tab Component

The Add-In Tab component is designed to serve a tab in Microsoft PowerPoint. Its responsibility is to handle button clicks and to activate Template, Hierarchy and Publish components.

5.2.1.2. Interface Description of Add-In Tab Component

This component does not have any input-output relation with other components. Add-In Ribbon only has Event Handler methods for each buttons.

5.2.1.3. Processing Details of Add-In Tab Component

Class Add-In Ribbon : OfficeRibbon		
Element	Type	Description
void Add-In_Ribbon _Load (object sender, RibbonUIEventArgs e)	Method	Loader method, called automatically, when Microsoft PowerPoint is opened.
void publish_button_Click (object sender, RibbonControlEventArgs e)	Method	EventHandler, called when the user pushes "Publish" button. This function calls Publish() function in Publish class.
void hierarchy_button_Click (object sender, RibbonControlEventArgs e)	Method	EventHandler, called when the user pushes "Hierarchy" button. This function makes visible the HierarchyEditor taskpane.
void template_button_Click (object sender, RibbonControlEventArgs e)	Method	EventHandler, called when the user pushes "Template" button. This function makes visible the TemplateEditor taskpane.

5.2.1.4. Dynamic Behavior of Add-In Tab Component

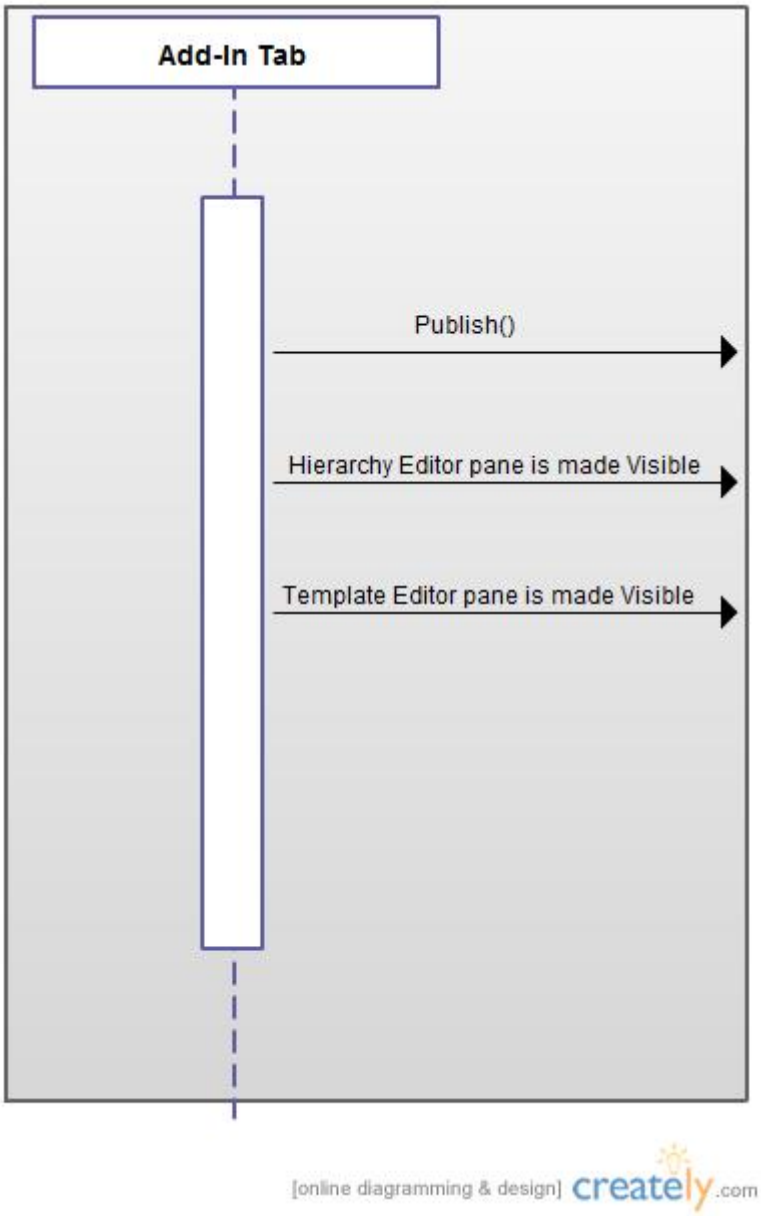


Figure 5.2 : Add-In Tab Component Sequential Diagram

5.2.2. Hierarchy Component

Hierarchy is one of the main components of the application. Since the aim of the product is to keep the trace of the training program by SCORM standard, hierarchical structure is the most important necessity. Application will create a tree structure out of the slides included in PowerPoint file, briefly explained, each subchapter being the child node of the chapter above it. This hierarchy, or in other words, tree structure, will be saved in a temporary file so that SCORM modifications are applied depending on this user-created hierarchy of slides.

5.2.2.1. Processing Narrative of Hierarchy Component

The main responsibility of the component is to create a file with a pre-defined structure which will contain the hierarchical relationship between slides. User will use “Hierarchy” window lying on the right side panel of the MS PowerPoint program to create/edit/reset the structure. There will be “up”, “down”, “left”, “right”, “load”, “save”, “delete” and “reset” buttons. They will be explained later in following subsections 5.2.1.2 and 5.2.1.3.

5.2.2.2. Interface Description of Hierarchy Component

Hierarchy component will be not visible by default. When the user presses “hierarchy” button in add-in tab of the application, a new side pane will be opened at the right. Buttons will be at the bottom side of the panel, and the panel will let user maximize it as a new window to see the structure in detail in case it doesn’t fit in the pane because of the number of the slides. Detailed interface description will be given at section 6.2. The Hierarchy interface class description is given in the Figure 5.3

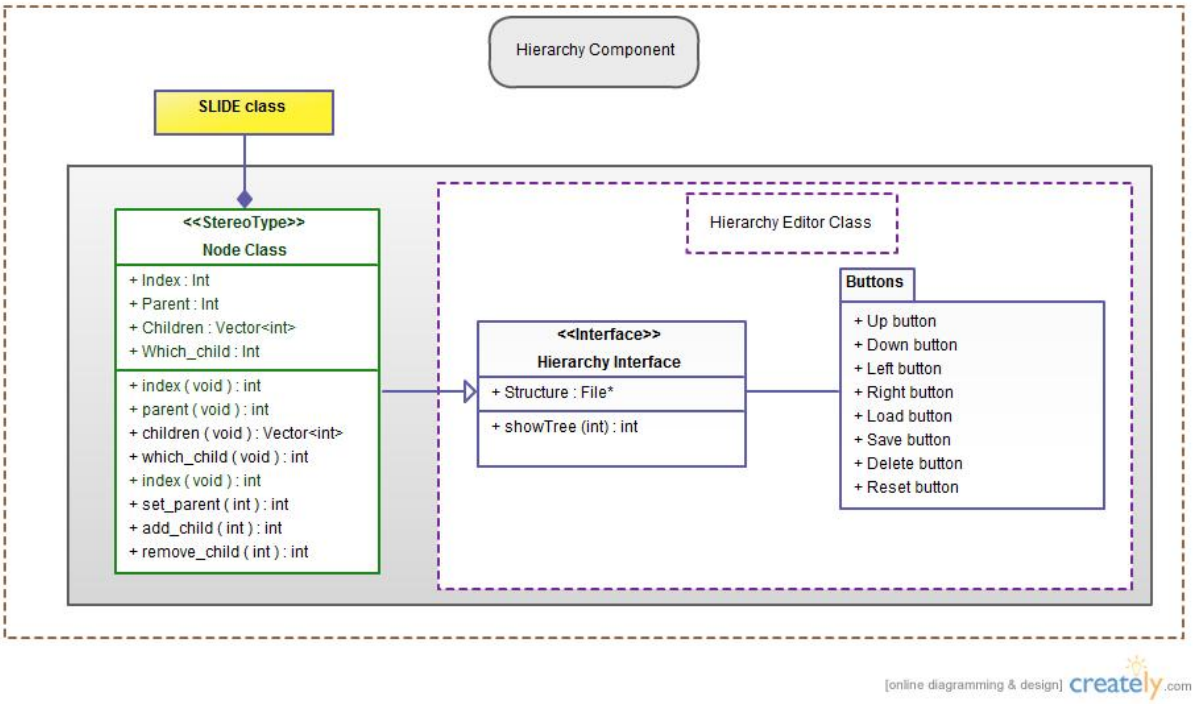


Figure 5.3: Hierarchy Component Interface Diagram

5.2.2.3. Processing Details of Hierarchy Component

In this subsection, all the details and algorithms of the Hierarchy component will be explained. To being with, the Node class is a class similar to the Slide class of the PresentationML given in chapter 4 while explaining data models. We decided not to use Slide class directly in order to get rid of massive temporary memory; therefore

implementation will use arrays of nodes and node indexes only pointing to the real Slide class objects.

Node Class		
Element	Type	Description
Index	Attribute (int)	Holds which slide this node corresponds to.
Parent	Attribute (int)	Holds index of parent node
Children	Attribute(vector<int>)	Vector of children indices
Which_child	Attribute(int)	Holds which child of the parent the node is
int index(void)	Method	Returns index
int parent(void)	Method	Returns parent index
Vector<int> children(void)	Method	Returns pointer to the children array
int which_child(void)	Method	Returns Which_child
int set_parent(int)	Method	Sets argument to Node::parent Returns 1 if successful; 0 otherwise
int add_child(int)	Method	Adds arg. to Node::children Returns 1 if successful; 0 otherwise
int remove_child(int)	Method	Removes arg. indexed child from Node::Children Returns 1 if successful; 0 otherwise

Buttons will be a “package” of functions used by Hierarchy Editor Class.

Button name	Description
Int move_up(int)	Moves the node with arg. index up in hierarchy meaning to decrease the Node::which_child Returns 1 if successful; 0 otherwise
Int move_down(int)	Moves the node with arg. index down in hierarchy meaning to increase the Node::which_child Returns 1 if successful;

	0 otherwise
Int move_left(int)	Decreases the depth of node in hierarchy making the node its parent's sibling Returns 1 if successful; 0 otherwise
Int move_right(int)	Increases the depth of node in hierarchy making the node the child of its left sibling Returns 1 if successful; 0 otherwise
File * load(string)	Loads the file with directory of the arg. and applies hierarchical structure in that file to current structure. Returns the file's pointer
File * save(string)	Saves the structure to the file with directory of the arg. Returns the file's pointer.
Int delete(int)	Removes the slide with arg. index from the hierarchy. Returns 1 is successful, 0 otherwise.
Int reset(void)	Resets the structure to its default state, which is all nodes having same depth=1

Hierarchy interface will be displayed by a function named "int ShowTree(int)" called for every change taken from EventHandler() function. This can be saved to a file which is hold by an attribute "Structure" and loaded later.

Hierarchy Editor Class		
Element	Type	Description
Structure	Attribute (File *)	It is the pointer to the file where the structure is saved
Int ShowTree(int)	Method	It displays the current hierarchy of slides inside the hierarchy window.

We will keep the subchapters of a chapter as a vector of children slide indices, and ShowTree function will do a depth-first-search in this tree-like structure. ShowTree will be called each time a change is made in hierarchy window.

5.2.2.4. Dynamic Behavior of Hierarchy Component

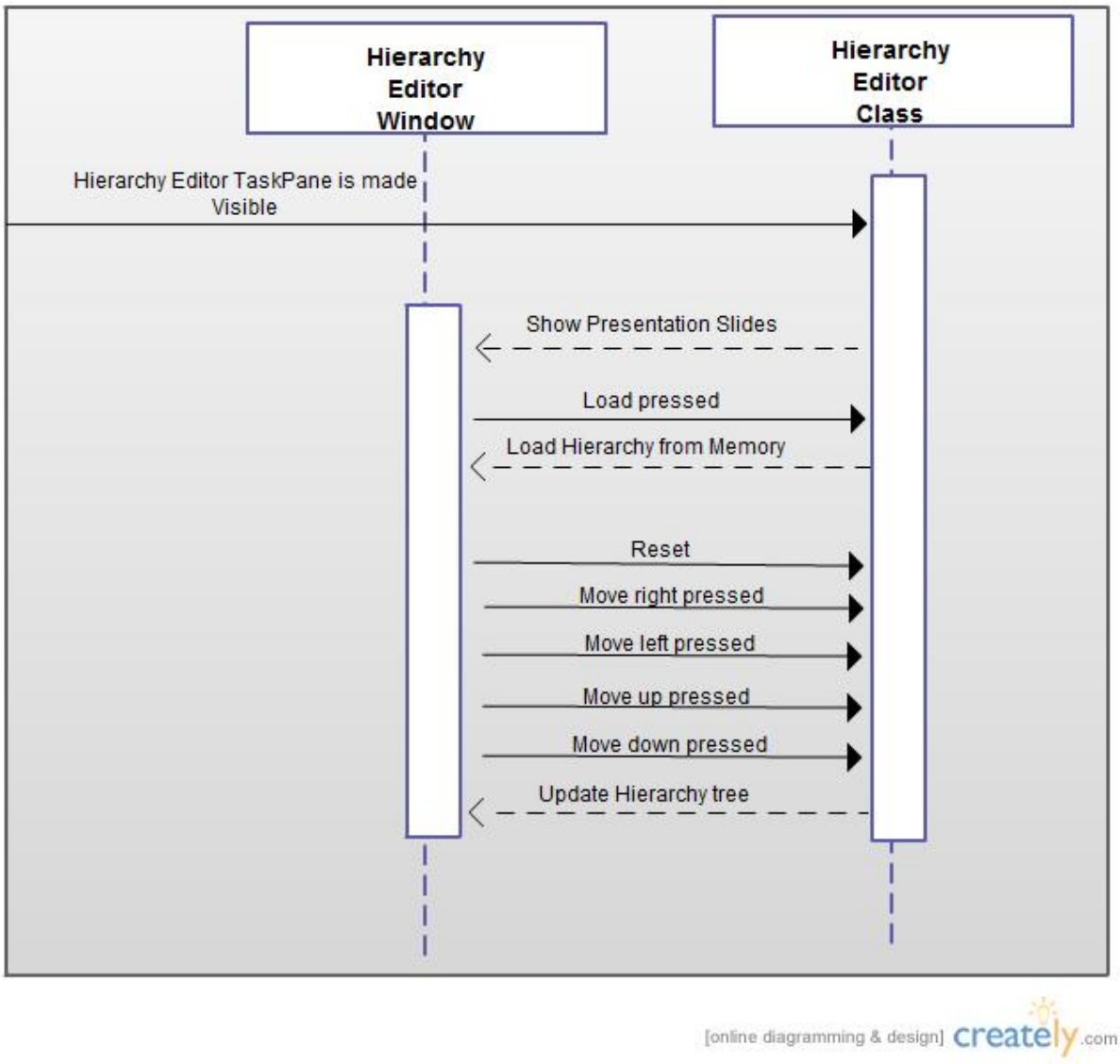


Figure 5.4 : Hierarchy Component Sequential Diagram

5.2.3. Template Component

Template Component is designed to support Template Editor functionality of Software. It has three parts: UI part, named TemplateEditor class, Control part, named TemplateHandler class and Template class. Firstly, TemplateEditor is a User Control class [12]. It is a panel, in which software (our plug-in) templates are shown . Secondly, Template Handler class is in interaction with UI part in order to convert the active presentation slide to chosen template in Template Editor panel. The converted result is represented in "Preview" part of Template Editor panel. Lastly, Template class is a data structure, used to create templates for Template Component.

5.2.3.1. Processing Narrative of Template Component

Template Component has four responsibilities:

- Representation of all templates that our software suggests to its user.
- Auto-Conversion of active presentation slide to template, in case a template selected.
- Preview the result of conversion.
- Save the result in memory for Publish Component.

5.2.3.2. Interface Description of Template Component

The basic description of input and output of the classes are showed in Figure 5.5. The description for each method is given in section 5.2.3.3.

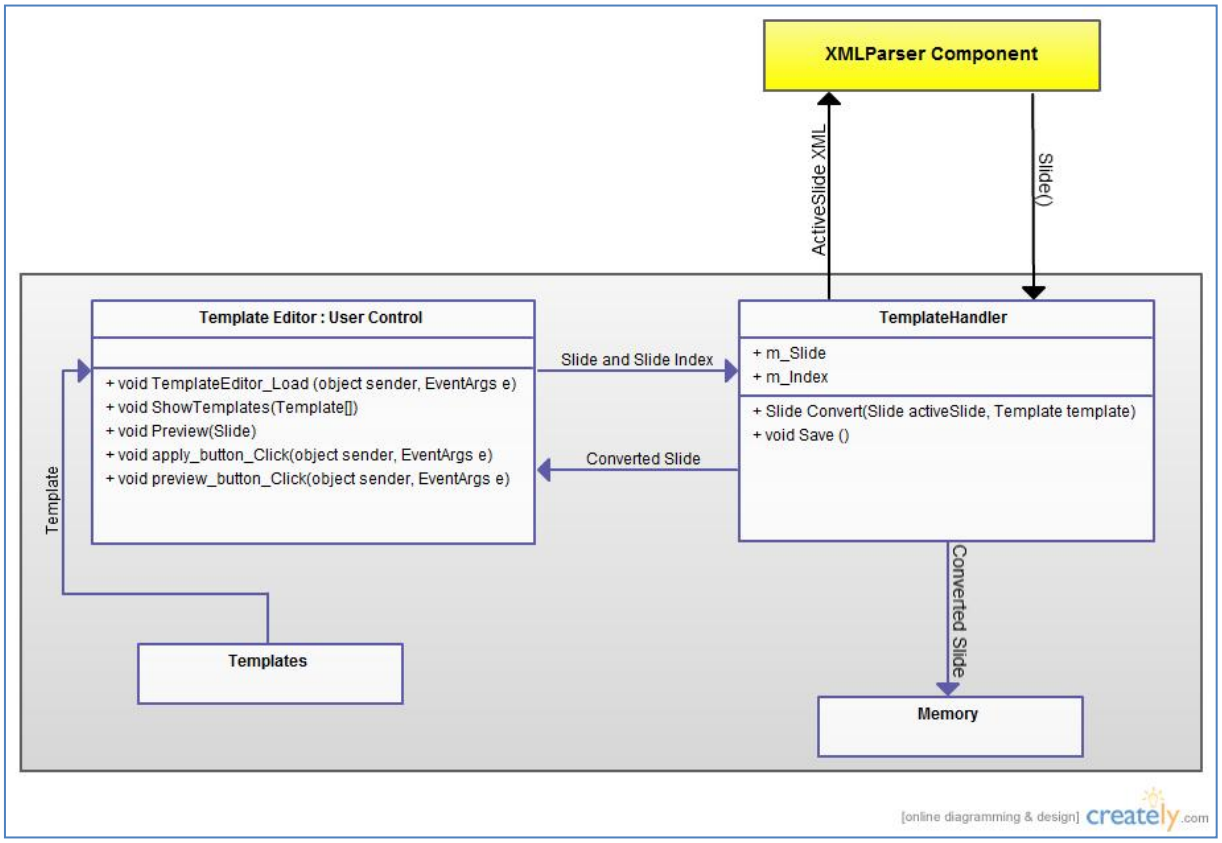


Figure 5.5: Template Component Interface Diagram

5.2.3.3. Processing Details of Template Component

Template Component has 3 classes:

5.2.3.3.1. Template Editor Class

Class TemplateEditor : UserControl		
Element	Type	Description
void TemplateEditor_Load (object sender, EventArgs e)	Method	Loader method, called automatically, when TemplateEditor.Visible() is changed to true. This is basic method for creation of TemplateEditor.
void ShowTemplates(Template[])	Method	Fills the part, templates represented in panel, with the templates that software suggests to user. Templates are saved in memory.
void Preview(Slide)	Method	Updates the part, converted slide is showed.
void apply_button_Click(object sender, EventArgs e)	Method	EventHandler, called when the user pushes "Apply" button. This function calls Save(Slide) function in Template Handler class in order to save new version of slide in memory.
void preview_button_Click(object sender, EventArgs e)	Method	EventHandler, called when the user pushes "Preview" button. This function creates Preview class instance

5.2.3.3.2. Template Handler Class

Class TemplateHandler		
Element	Type	Description
m_Slide	Attribute(Type Slide)	Slide type parameter, is used to keep the converted version of the latest slide.

m_Index	Attribute(Type int)	Integer type parameter, is used to keep the index of latest converted object.
Slide Convert(Slide activeSlide, Template template)	Method	Called by Template Editor class. It converts the form of the slide to template form. Before conversion, by using XMLParser component, it gets content of slide.
void Save ()	Method	Called by TemplateEditor :: apply_button_Click function. This function saves the latest form of the slide in memory for the use of Publish functionality.

5.2.3.3.3. Template Class

The detailed information about this class will be given in Detailed Software Design Document, Section 7.

5.2.3.4. Dynamic Behavior of Template Component

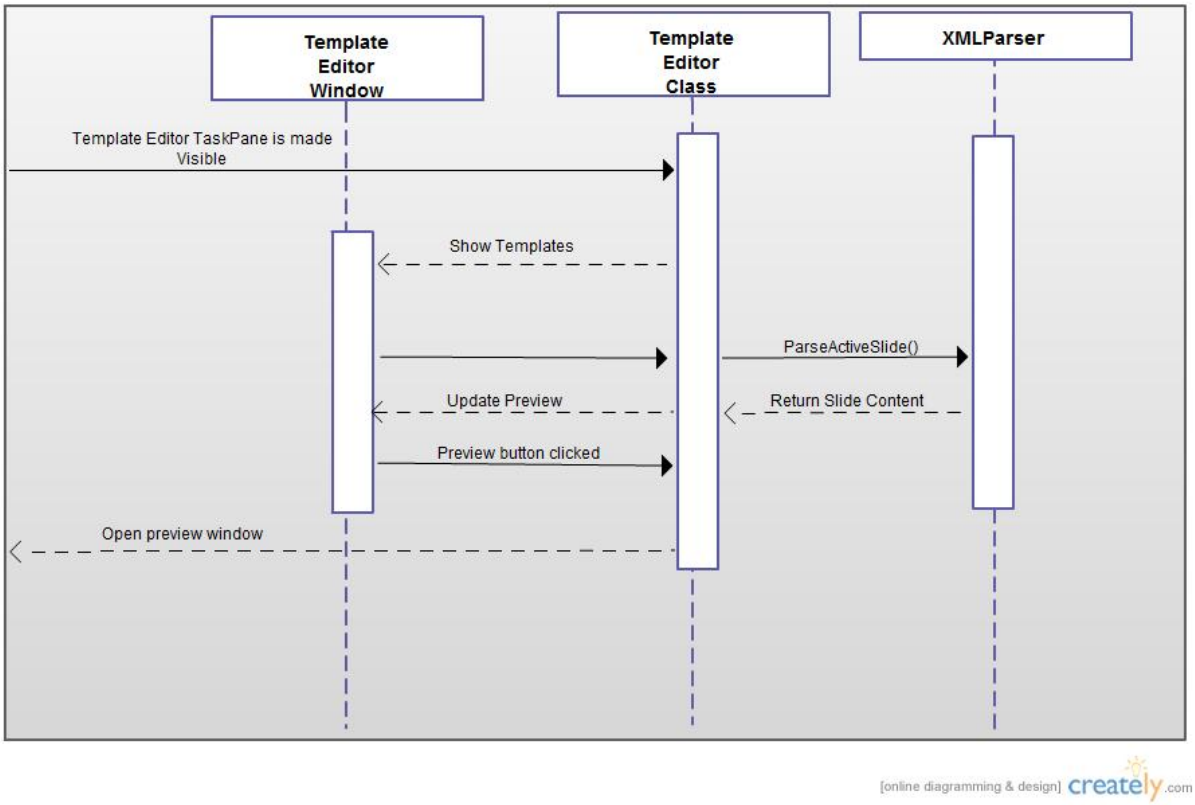


Figure 5.6: Template Component Sequential Diagram

5.2.4. XMLParser Component

This component of the product, in brief, is going to parse the xml contents of the PPTX file and save it for OpenXML classes to use it. The output of the parsing operation and the way it is going to be handed will be discussed in Detailed Design report on chapter 7. Component contains only XmlParser class.

5.2.4.1. Processing Narrative of XMLParser Component

XmlParser class has a simple duty: taking a PPTX file exact directory, it sends the contents to its parser (converter) function and saves it in ParsedSlides class object created after parsing.

5.2.4.2. Interface Description of XMLParser Component

XmlParser component takes a slide as an input and outputs an object of type ParsedSlides class. This component will be used by Publish component - section 5.2.5 and Template component – 5.2.3.

5.2.4.3. Processing Details of XMLParser Component

XmlParser Class		
Element	Type	Description
M_slide	Attribute (ParsedSlides)	It is a data structure type containing parsed content
Int convert(string)	Method	Taking PPTX file directory as an argument, it converts PPTX to ParsedSlides type object and assigns it to M_slide. Returns 1 is successful, 0 otherwise.

Int convert(string) function will use OpenXML SDK functions in general to retrieve each element one by one (such as titles, subtitles, standard text, image, bulleted lists etc.)

5.2.4.4. Dynamic Behavior of XMLParser Component

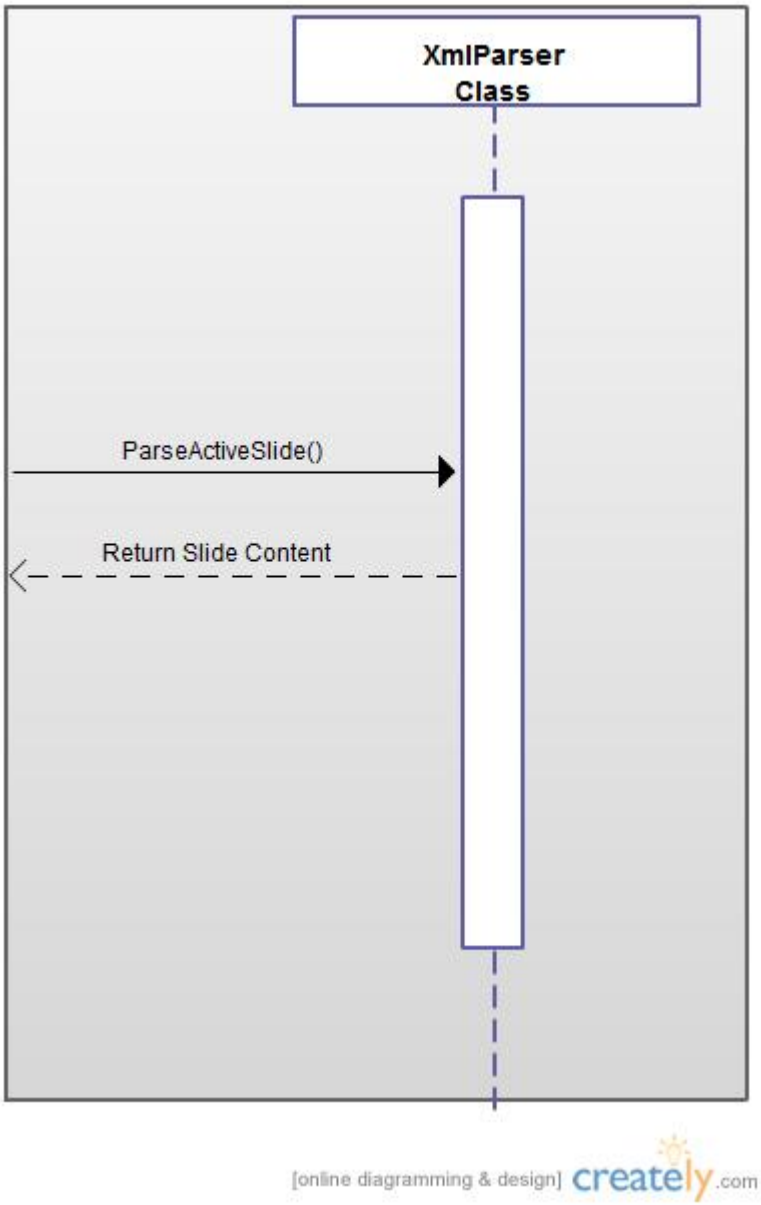


Figure 5.7 : XMLParser Component Sequential Diagram

5.2.5. Publish Component

Publish component, in brief, is the class creating the output HTML files in a compressed zip folder. It contains one class only as in XmlParser component. Publish Class does the main conversion operation from ParsedSlides (which are in templates already) to HTML5 format with a given directory address.

5.2.5.1. Processing Narrative of Publish Component

The scenario of the publishing process is quite straightforward. User does any necessary operation with Hierarchy editor and Template editor before clicking Publish button in Add-in window. This button will trigger a standard “Windows Directory Selection” pop-up window

to select the directory for the output compressed zip file. The directory will be kept in Publish class and saving operation will be done.

5.2.5.2. Interface Description of Publish Component

Publish component takes the directory input from user interface, through a Windows Directory Selection pop-up and saves it as a string to argument Publish::outputDirectory. In addition, Publish component will receive ParsedSlides type object and convert it to HTML5 using dynamic or static html generator. Product will provide static conversion, meaning that converting from a premade template to html5 for any document. However dynamic conversion, which implies conversion from PPTX slide directly to HTML5 without using any templates, will not be available for all document types depending on their designs.

5.2.5.3. Processing Details of Publish Component

Publish Class		
Element	Type	Description
outputDirectory	Attribute (String)	Keeps the directory selected from Directory Selection Pop-up Window
Int createHtml(void)	Method	This function does all the main html5 generation from either a template or a slide directly. Returns 1 is successful, 0 otherwise.
Int savefile(void)	Method	Creates a zip compressed file with HTML files inside and saves it the directory Publish::outputDirectory Returns 1 is successful, 0 otherwise.

Int createHtml(void) function will use the advantages and some methods of ASP.NET framework to be handle calculations and controls of data objects. Those ASP codes will be embedded inside HTML5 code in this function. ASP.NET framework will enable us to create html codes as simple as inserting arguments to some methods. To illustrate, let's take a Template to hand. Let's assume that this template will contain a title in the middle of the slide, an image at the bottom right corner, and a simple text box at the bottom left corner without any backgrounds. In this case, we know approximately how the html code will look like, except the exact title, text and image contents. Therefore, using 3 functions for each, we can generate proper HTML codes using this framework such that, for the title "HTML create_title($title)", "HTML create_image($image_directory)" and "HTML create_text($text_body)" for title, image and text respectively.

5.2.5.4. Dynamic Behavior of Publish Component

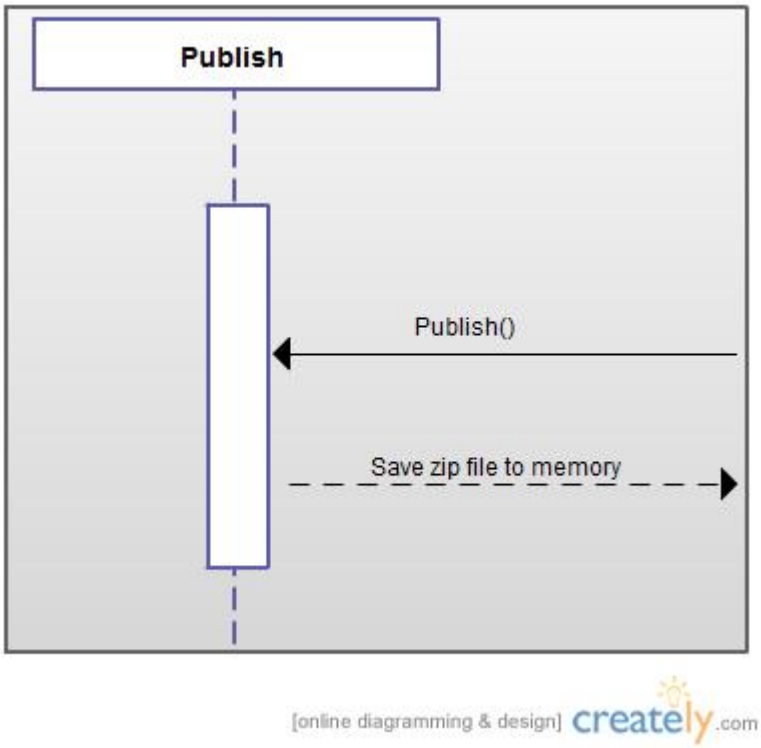


Figure 5.8 : Publish Component Sequential Diagram

6. User Interface Design

This section describes the Graphical User Interfaces (GUI) offered by the software product for PPTX to HTML Content Converter project. Both information about the planned interfaces and some screen images of them are given in the following subsections.

6.1.Overview of User Interface

As stated in previous sections, the targeted software product for PPTX to HTML Content Converter project is a Microsoft Office PowerPoint add-in. Therefore, the interfaces supported by this software product will be visible to the users only through the Microsoft Office PowerPoint program, assuming that this add-in is installed. So, the interfaces that will be provided by the system are going to be windows that are belong and embedded to the Microsoft Office PowerPoint program.

There will be only three graphical user interfaces. The details of those interfaces and the facilities they provide will be listed in following subsections. In addition, there are also some screen images of those interfaces in the next subsection (6.2).

6.1.1. Main Window Interface (Publish Interface)

The main window of the add-in will contain only three buttons and nothing more. The names and functions of the buttons are as followed;

Publish Button: This is the button that will start the conversion process. After this button is pressed, user will not be able to interfere in the process. All the modifications that are done by the user (they will be explained), are occurred before this button have been clicked.

Hierarchy Editor Button: This button has the effect of toggling the Hierarchy Editor Window, which is explained in next section (6.1.2), on and off.

Template Editor Button: Just like the hierarchy editor button, this button closes and opens the Template Editor Window as it is clicked. Template Editor Window is also explained in section 6.1.3.

In section 6.2, there is a screen image of the main window interface design (Figure 6.1).

6.1.2. Hierarchy Editor Window Interface

Microsoft Office PowerPoint does not provide any facilities that allows categorization; crating a tree-like hierarchy; of the slides. It just keeps them sequentially. This is not a problem for presentations, but it is against the nature of e-learning contents, which are targets of this project. Dividing the whole content into subgroups that builds sense of chapters is so common for this kind of contents.

Since Microsoft Office PowerPoint do not provide hierarchy facilities, it has to be done by the add-in. The Hierarchy Editor accomplishes this mission. The system fetches all the slides and lists them sequentially in the editor window. There are 7 buttons that let the user to alter the hierarchy of the slides. Their missions are explained below using the tree notation.

Move Up Button: This button is to change the order of the slide that is checked with the sibling slide that comes before it.

Move Down Button: This button is the opposite of move up button. Order switch occurs between checked slide and sibling slide after the checked one.

Move Right Button: This button achieves the real categorization. It makes the slide that is selected, child of the sibling slide that comes before it.

Move Left Button: This is the button that undoes the hierarchy that is created before, only one step. It makes the slide that is selected, sibling of the slide that is the parent slide of it.

Load Hierarchy Button: This button pop-ups a window to make user to select a hierarchy XML file that is created before for the PPTX that is open.

Save Hierarchy Button: This button is to save the Hierarchy that is created (or modified).

Reset Hierarchy Button: This button have the function of resetting the currently loaded hierarchy so that slides are again in sequential order.

In section 6.2, there is a screen image of the hierarchy editor window interface design (Figure 6.2).

6.1.3. Template Editor Window Interface

During the conversion of PPTX documents to HTML5 it is crucial to keep the structure of the slides. For example, if there is a bulleted list in the slide, there should be one in corresponding HTML5 file. Presenting that bulleted list as a newline separated sequence of texts is not enough. Moreover, locations of the elements in slides also have to be transferred similarly to the newly created HTMLs. If a text is the title of the slide, or there is an image at the bottom-right corner of the slide, they should also be in corresponding HTML in same positions.

Template editor is to solve this issue. The PPTX to HTML5 Content Converter will solve this problem automatically. It will recognize the content and structure of the slide that is wanted to be converted and try to conserve them. However, it will not be 100% precise especially for abnormally designed slides. So, using the template editor users will be able to increase the correctness performance of the conversion process.

There will be built-in templates that can be selected by the user. User will be able to determine a template for each of the slides. Choosing a template for a slide will not be mandatory and initially all slides will be marked to be their templates selected automatically by the system.

The Template Editor Window will be divided into two parts. In the first part all slides will be listed. In the second part, all the built-in templates will be listed. User will choose the slide that he/she wants to choose its template, and then from second menu, will choose the desired template.

There are 3 buttons that let the user to affect the template choices of the slides. Their missions are explained below.

Load Template Button: This button pop-ups a window to make user to select a template XML file that is created before for the PPTX that is open.

Save Template Button: This button is to save the Template that is created (or modified).

Reset Template Button: This button have the function of resetting the currently loaded template selections so that all slides' templates are selected automatically.

In section 6.2, there is a screen image of the template editor window interface design (Figure 6.3).

6.2.Screen Images

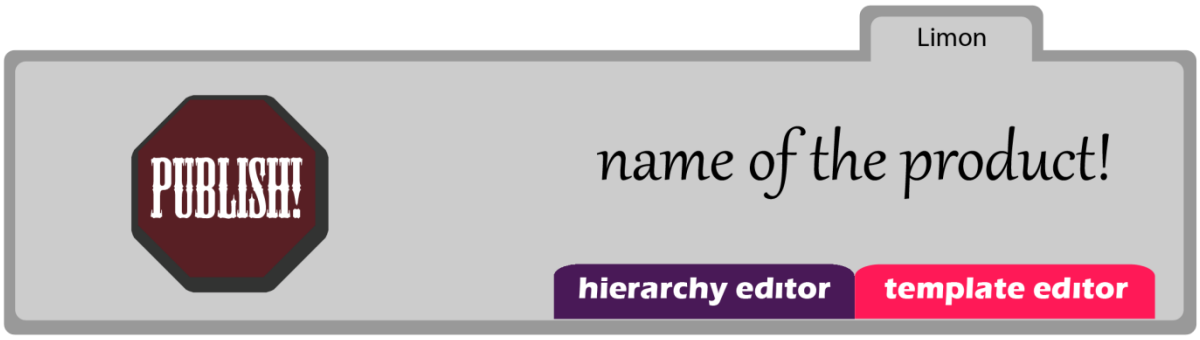


Figure 6.1 : Main Window Interface

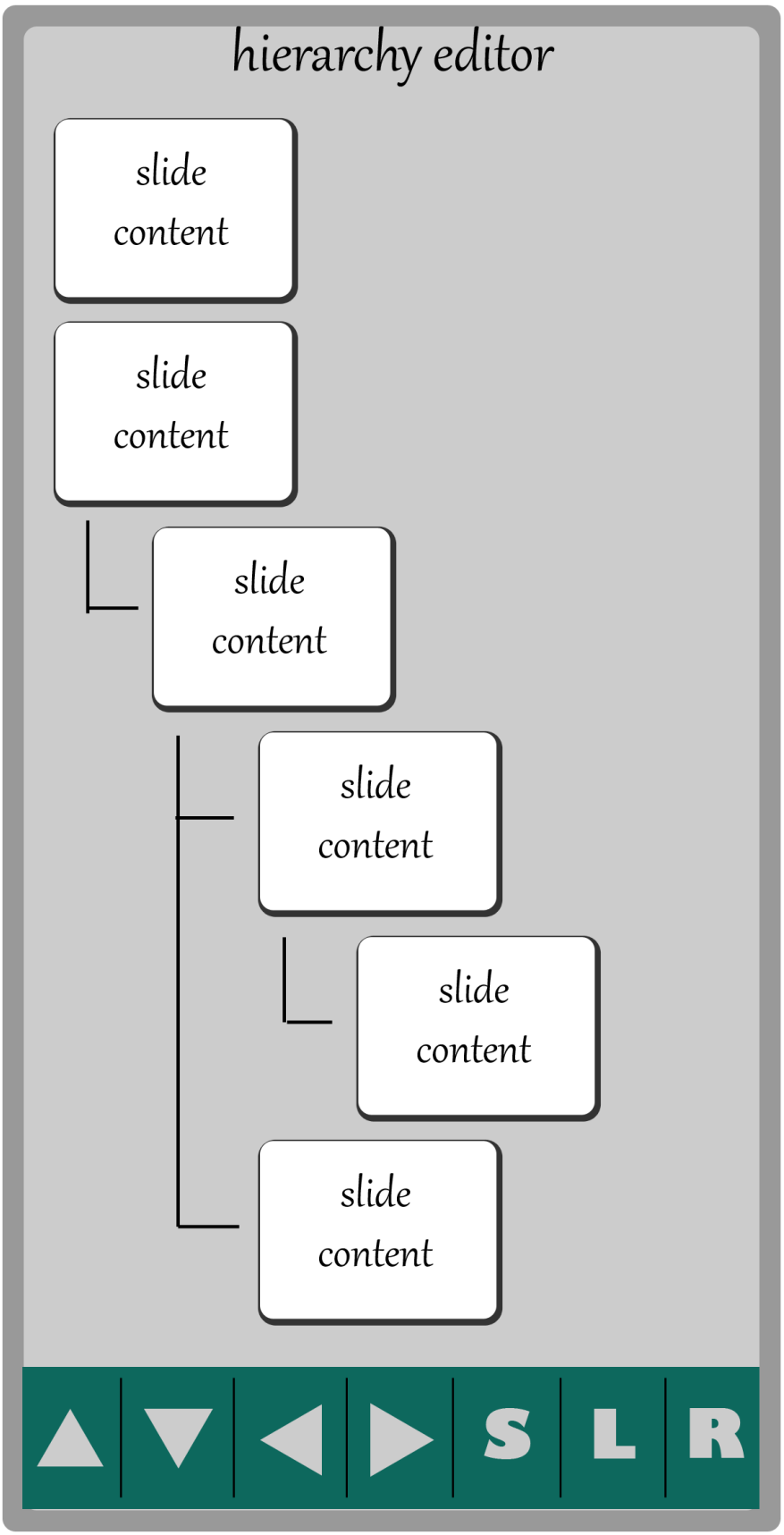


Figure 6.2 : Hierarchy Editor Window Interface

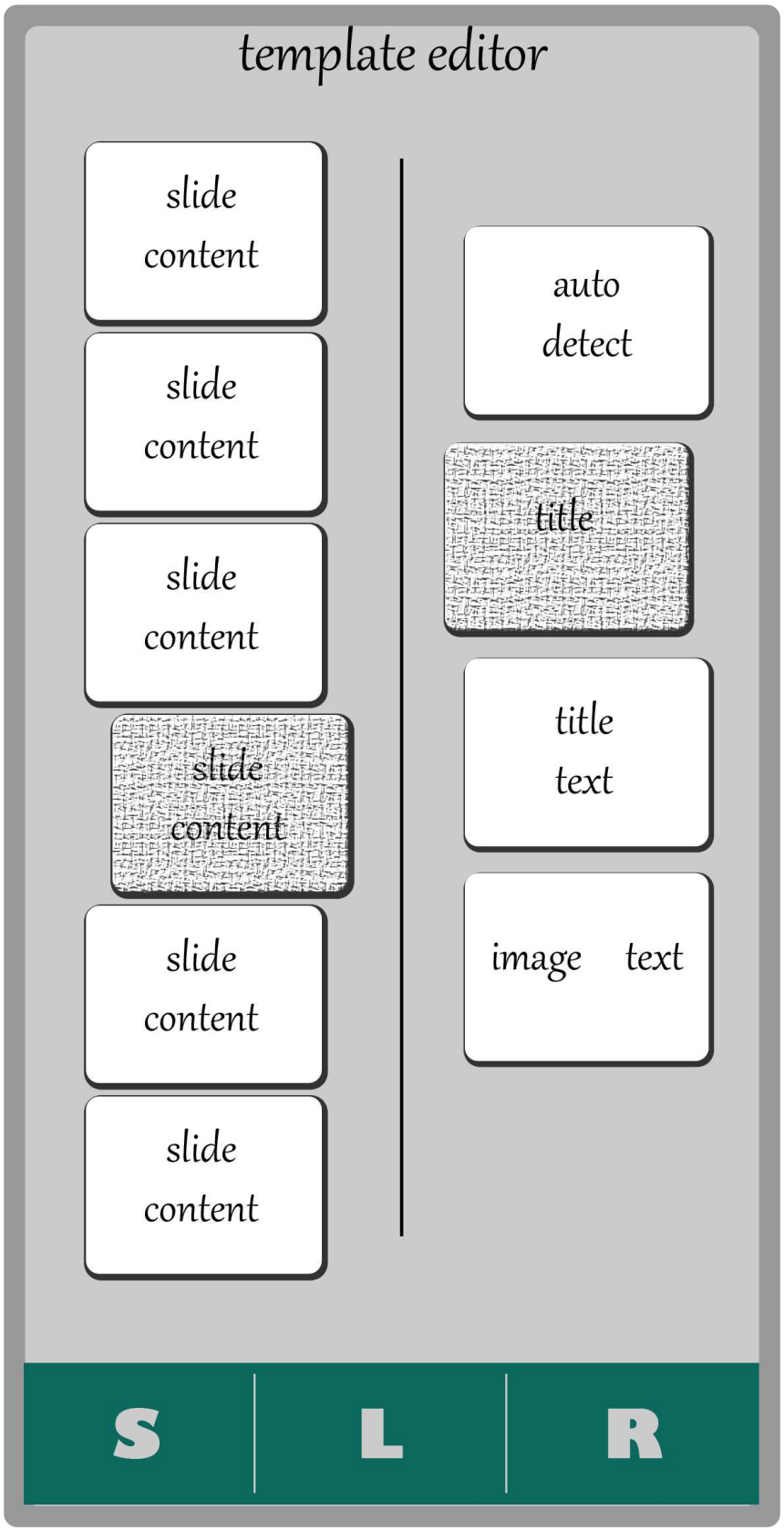


Figure 6.3 : Template Editor Window Interface

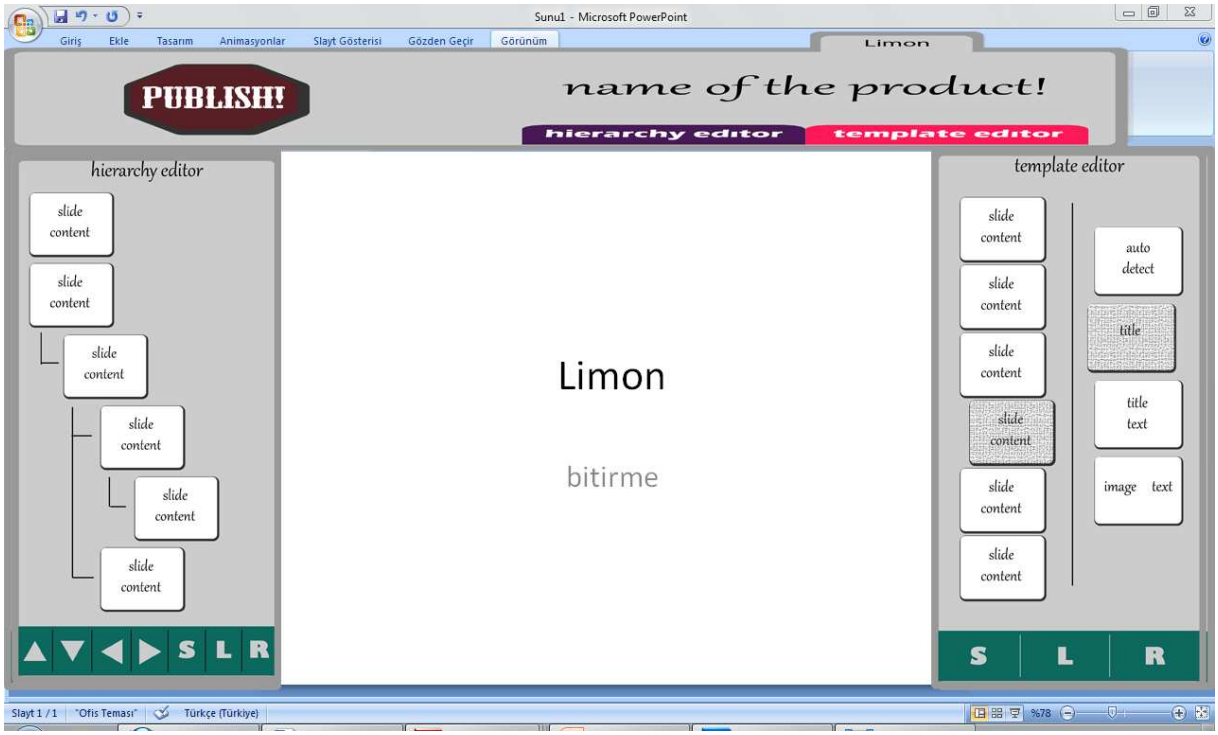


Figure 6.3 : All three interfaces together on MS Office PowerPoint program

7. Libraries and Tools

Many libraries and tools will be used by the PPTX to HTML5 Content Converter project during the development phase. Descriptions of those libraries and tools, and their planned usage in the project are stated in following subsections.

7.1.Microsoft Visual C#

7.1.1. Description

Microsoft Visual C# is Microsoft's implementation of the C# specification. It is based on the ECMA/ISO specification of the C# language, which Microsoft also created. While multiple implementations of the specification exist, Microsoft Visual C# is by far the one most commonly used [5]. It is a powerful but simple language aimed primarily developers creating applications by using the Microsoft .NET Framework [6]. It inherits many of the best features of C++ and Microsoft Visual Basic, but few of the inconsistencies and anachronisms, resulting in a cleaner and more logical language.

7.1.2. Usage in the Project

Microsoft Visual C# programming language will be the primary programming language used during the development of the project. Considering the libraries that needed to be used for easing the development (those libraries are explained in following subsections), Microsoft Visual C# is obviously the best choice for the project.

7.2.Microsoft .NET Framework

7.2.1. Description

The Microsoft .NET Framework [7] is a software framework that runs primarily on Microsoft Windows. It includes a large library and supports several programming languages which allow language interoperability (each language can use code written in other languages). Programs written for the Microsoft .NET Framework execute in a software environment known as the Common Language Runtime (CLR), an application virtual machine that provides important services such as security, memory management, and exception handling. The class library and the CLR together constitute the .NET Framework. The .NET Framework's Base Class Library provides user interface, data access, database connectivity, web application development, and network communications. The .NET Framework is intended to be used by most new applications created for the Windows platform.

7.2.2. Usage in the Project

Microsoft .NET Framework will be widely used in the project. As stated in previous section (7.2.1) this framework can be used with different programming languages. However, again stated in the section 7.1.2, Microsoft Visual C# is chosen and it will be used. The final software product of the project will be a Microsoft PowerPoint plug-in. Microsoft .NET Framework provides capability of creating plug-ins for Microsoft Office programs. In addition, it includes the Open XML library that provides handy ways of parsing XML files that constructs the Microsoft Office formatted documents like .pptx and .docx documents. So, Microsoft .NET Framework is kind of a requirement for the project.

7.3.Microsoft Visual Studio

7.3.1. Description

Microsoft Visual Studio [8] is an integrated development environment (IDE) from Microsoft. Microsoft Visual Studio is a complete set of development tools for building ASP.NET Web applications, XML Web Services, desktop applications, and mobile applications. Visual Basic, Visual C++, Visual C#, and Visual J# all use the same IDE, which allows them to share tools and facilitates in the creation of mixed-language solutions. In addition, these languages leverage the functionality of the .NET Framework, which provides access to key technologies that simplify the development of applications.

7.3.2. Usage in the Project

Microsoft Visual Studio 2010 will be the primary tool that is used on development phase of the project. Considering the libraries planned to be used and content of the project, Visual Studio is the native tool for creating the project.

7.4.Open XML SDK 2.0

7.4.1. Description

Office Open XML (also known as Open XML) is a zipped, XML-based file format developed by Microsoft for representing spreadsheets, charts, presentations and word processing documents. The Office Open XML specification was initially standardized by ECMA and later by ISO. Starting with Microsoft Office 2007, the Office Open XML file formats have become the default target file format of Microsoft Office.

The Open XML SDK 2.0 for Microsoft Office, which is also supplied by Microsoft, simplifies the task of manipulating Open XML packages and the Open XML schema elements within a package. The classes in the Open XML SDK 2.0 encapsulate many common tasks (like getting animations, themes, and elements) that developers perform on Open XML packages, so that they can perform complex operations with just a few lines of code [9].

7.4.2. Usage in the Project

Since input documents of the project are PPTX documents, which are said to be an Office Open XML file format in previous section (7.4.1), managing the contents of these documents is essential to convert them to HTML5 formatted documents. Theoretically, the XMLs contained by these documents can be parsed manually because the documentation of structures of them is online. However, it costs developers lots of effort and time. Therefore, the libraries in Microsoft .NET Framework that are supplied by Open XML SDK 2.0 will be used to handle these issues with a few lines of code.

7.5.JSON

7.5.1. Description

JSON [10] or JavaScript Object Notation is a lightweight text-based open standard designed for human-readable data interchange. It is derived from the JavaScript scripting language for representing simple data structures and associative arrays, called objects. Despite its relationship to JavaScript, it is language-independent, with parsers available for most languages. JSON is a low-overhead alternative to XML.

7.5.2. Usage in the Project

The XMLs contained by PPTX documents are very detailed. There are lots of tags and attributes used in these files. JSON is a good alternative to XML. Because it is more human-

readable, in the debugging phases of development, JSON will be preferred to be used to make it easier to check the validity of the returned results of the codes.

7.6.HTML5

7.6.1. Description

HTML5 is a language for structuring and presenting content for the World Wide Web, and is core technology of the Internet. It is the fifth revision of the HTML standard (created in 1990 and standardized as HTML4 as of 1997) and is still under development. Its core aims have been to improve the language with support for the latest multimedia while keeping it easily readable by humans and consistently understood by computers and devices.

In particular, HTML5 adds many new syntactical features. These include the `<video>`, `<audio>`, `<header>` and `<canvas>` elements. These features are designed to make it easy to include and handle multimedia and graphical content on the web without having to resort to proprietary plug-ins and APIs. Other new elements, such as `<section>`, `<article>`, `<header>` and `<nav>` are designed to enrich the semantic content of documents. New attributes have been introduced for the same purpose, while some elements and attributes have been removed. The APIs and document object model (DOM) are fundamental parts of the HTML5 specification.

7.6.2. Usage in the Project

As stated in Section 1.1 (problem definition), HTML5 will have many advantages compared to its opponents. To benefit from those advantages, the target document format is decided to be HTML5 in project proposition phase. Therefore, the elements from older versions of HTML and new elements and APIs from HTML5 will be widely used in the project.

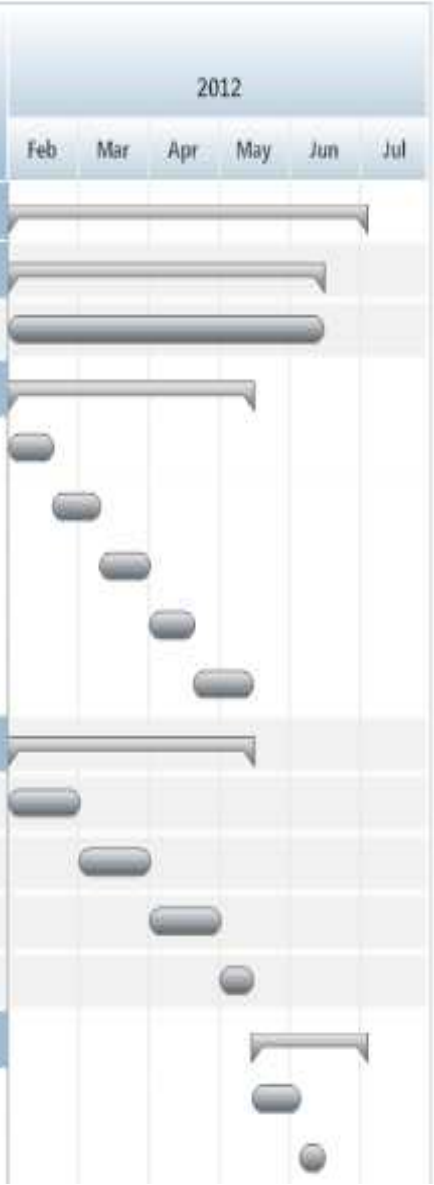
8. Time Planning

8.1.Term 1

	Task	Assigned To	Start	End	Dur	%
	PPTX to HTML5 Content Converter		15/10/11	20/1/12	70	
1	Project Management		15/10/11	20/1/12	70	
1.1	Project Proposal	All Team Members	15/10/11	31/10/11	11	
1.2	Software Requirements Analysis Report	All Team Members	31/10/11	25/11/11	20	
1.3	Initial Software Design Document	All Team Members	25/11/11	19/12/11	17	
1.4	Detailed Software Design Document	All Team Members	19/12/11	5/1/12	14	
1.5	Team Presentation and Prototype Demo	All Team Members	5/1/12	20/1/12	12	
2	Library and Tool Analysis		20/11/11	1/1/12	30	
2.1	Microsoft Visual C# and Microsoft .NET Framework	Ömer Baykal - Shamil Farajullayev	20/11/11	10/12/11	15	
2.2	Microsoft Visual Studio 2010	All Team Members	20/11/11	10/12/11	15	
2.3	Open XML SDK 2.0	Ömer Baykal - Rustam Hashimov	10/12/11	1/1/12	15	
2.4	MS Office Add-in Development with .NET	Rustam Hashimov - Ömer Baykal	20/11/11	1/1/12	30	
2.5	HTML5	Nahid Hamidli - Shamil Farajullayev	10/12/11	1/1/12	15	
3	Demo Implementations of Components		10/12/11	15/1/12	25	
3.1	Parsing XMLs of PPTX with Open XML SDK	Ömer Baykal - Rustam Hashimov	10/12/11	25/12/11	10	
3.2	Creating HTML Files from Parsed Data	Nahid Hamidli - Shamil Farajullayev	25/12/11	5/1/12	9	
3.3	Integrating Parsing and HTML Creating Phases	All Team Members	5/1/12	15/1/12	7	
4	Demo Interface Implementation		20/11/11	5/1/12	34	
4.1	General Add-In Structure Implementation	Rustam Hashimov - Nahid Hamidli	20/11/11	10/12/11	15	
4.2	Hierarchy Editor Implementation	All Team Members	10/12/11	25/12/11	10	
4.3	Template Editor Implementation	All Team Members	25/12/11	5/1/12	9	

8.2. Term 2

	Task	Assigned To	Start	End	Dur	%
	PPTX to HTML5 Content Converter		1/2/12	3/7/12	110	
1	Project Management		1/2/12	15/6/12	98	
1.1	Web Page Development and Updating	Ömer Baykal - Shamil Farajullayev	1/2/12	15/6/12	98	
2	Final Implementations of Components		1/2/12	15/5/12	75	
2.1	Parsing XMLs of PPTX with Open XML SDK	Ömer Baykal - Rustam Hashimov	1/2/12	20/2/12	14	
2.2	Creating HTML Files from Parsed Data	Nahid Hamidli - Shamil Farajullayev	20/2/12	10/3/12	15	
2.3	Integrating Parsing and HTML Creating Phases	All Team Members	10/3/12	1/4/12	15	
2.4	Implementing SCORM Standards	All Team Members	1/4/12	20/4/12	15	
2.5	Final Integration and Enhancement	All Team Members	20/4/12	15/5/12	18	
3	Final Interface Implementations		1/2/12	15/5/12	75	
3.1	General Add-In Structure Implementation	Rustam Hashimov - Nahid Hamidli	1/2/12	1/3/12	22	
3.2	Hierarchy Editor Implementation	Rustam Hashimov - Ömer Baykal	1/3/12	1/4/12	22	
3.3	Template Editor Implementation	Rustam Hashimov - Ömer Baykal	1/4/12	1/5/12	22	
3.4	Final Integration and Enhancement	All Team Members	1/5/12	15/5/12	11	
4	Finalization		15/5/12	3/7/12	36	
4.1	Testing and Evolving the Product	All Team Members	15/5/12	5/6/12	16	
4.2	Final Presentation	All Team Members	5/6/12	15/6/12	9	



9. Conclusion

In this document which is the Software Design Document of PPTX to HTML5 Content Converter project; design considerations, data and system architecture, information about development phase of the project like what to be used during, interfaces that will form the communication between users and the system, and scheduling of the project is discussed.

This document is an Initial Software Design Document, and so, it does not include very detailed information about the project design. This is general design and architecture of the project. There will be a final Software Design Document which will be very detailed, and it will take this document as a reference point.