William Smith

Process for Migrating Help System to HTML5

Metropolia University of Applied Sciences

Bachelor of Engineering

Information Technology

Bachelor's Thesis

10 May 2021

Abstract

Author:	William Smith
Title:	Process for Migrating Help System to HTML5
Number of Pages:	24 pages + 5 appendices
Date:	10 May 2021
Degree:	Bachelor of Engineering
Degree Programme:	Information Technology
Professional Major:	Software Engineering
Supervisors:	Janne Salonen, Head of the Major

This thesis provides a process for software providers to migrate legacy help systems to support HTML5 content with aim of improving assistance and experience for end users. Issues with legacy help systems are discussed, namely the merging of different help files and projects, support for displaying and searching content in multiple languages and different character sets. Compatibility of online help formats with HTML, CSS, and JavaScript versions is mentioned followed by a review on the current limitations of Microsoft HTML Help published in CHM or compiled HTML format. The review describes how the content in a CHM file can be accessed and presented using different WPF controls such as WebBrowser and WebView2 which is Chromium-based. An approach to decompile and recompile CHM files to create a modular help system is presented. In comparison, a separate approach is described using a content authoring tool that can publish HTML5 outputs. Both approaches are applied to a case company currently using Microsoft HTML Help with localized support for Asia-Pacific (APAC) and Europe, the Middle East and Africa (EMEA) regions. The proposed process defines phases or decision points to enable the migration of online help to support modern web technologies.

Keywords: online help, Microsoft HTML Help, CHM, HTML5, software documentation

Contents

1 Introduction

This thesis evaluates the current help system of a case company to provide a process for migrating the system to deliver web-based content that supports HTML5. The process aims to facilitate technical writers, developers, and software companies to deliver information that leverages current web technologies and supports the locale of their customers. Additionally, the process addresses the need to deliver and update information in a timely manner, thereby decoupling the delivery of information from software releases. The tool or control in which information is accessed by a user is criteria for measuring the value of the process and its adoption by the case company.

Information or end user documentation in the context of this thesis is set of HTML files and resources packaged in a file format or directory structure to serve as a web app. Information and the tool for viewing it is known as online help. Online help enables a user to find information on how to use a particular feature of software or learn more about it. Limitations with online help formats combined with the needs of the case company are contributing factors to identify a process to modernize online help and the system that creates it. The key areas to address are the display, navigation, and search functionality of online help formats.

The case company is a provider of software used worldwide that supports multiple languages. The current help system delivers Microsoft HTML Help or compiled HTML (CHM) files which are shipped with each software release and installed locally on the computer of the end user. The case company currently releases about two to three times per year, which limits the ability to add or change content in the help files.

In addition, the case company is a platform provider with a network of original equipment manufacturers (OEM) and value-added resellers (VAR) who redistribute the software made by the case company. The VAR and OEM of the case company face the difficulty in localizing and merging their own online help with the current format. Generally, an OEM of the case company develops additional features for the software, and the online help for these features needs to be integrated and suitable to their own end users, such as different display language for the content.

This thesis is structured in the following manner. First, there is an overview of the current state and limitations of the help system and delivery processes used by the case company. This is followed by an explanation on the research method used to evaluate online help of the case company, particularly the development tools needed for the research. Next, a software review of the online help used by the case company is examined, which includes tests on how the online help can be viewed in its current state and environment. Finally, a process that provides short-term and long-term recommendations is described with mention of additional topics for further research.

2 Current Help System

This section provides an overview of the app developed by the case company and its help system. Processes used by the case company for building its online help are outlined including the occurrence of handoffs. The limitations of the online help format used by the case company and its impact on supporting VAR and OEM requests are defined to indicate the need for a solution that is modular, adaptive, and scalable.

2.1 App and its Help System

The software developed by the case company is a WPF desktop app that runs on Windows. The app supports eight languages, four of which use the Western Latin character set. To localize its user interface, the app encodes characters in UTF-8. Online help for the app is built for each supported language, all of which are installed with the app on its host machine. This means eight CHM files are built and packaged with the app to provide online help. When a user changes the display language of the app, it also changes the version of online help shown to the user. One limitation is the app must be restarted to properly load language resource files and display the correct CHM file. The app also packages three other CHM files of online help for its Application Programming Interface (API). Note that these three files are only available in English. In total, the app comes with 11 CHM files, ten of which are created and maintained by the case company. The case company makes four variants of the app with different features, so four variants of each help file, excluding the API files, are generated with each release of the app. Issues with any version of online help found after a release cannot be dealt with until the next release or by communicating a workaround to end users, such as a post on the company forum or support ticket.

2.2 Process for Building Online Help

Figure 1 illustrates the process of creating online help at the case company for its app and Python API. A team of application engineers uses a help authoring tool, RoboHelp, to manage the help project and create its content. Authoring of content is done with the aid of subject-matter experts, such as developers who designed and implemented features of the app. The team compiles the help project into CHM files, and then delivers them to the development team. This is also known as a handoff. A developer then integrates the CHM files, meaning a new version of the files are added to a repository, and then a build pipeline generates a version of the app with those files. The online help of the app is reviewed by a quality assurance (QA) team of testers before it is approved and deployed with the app.

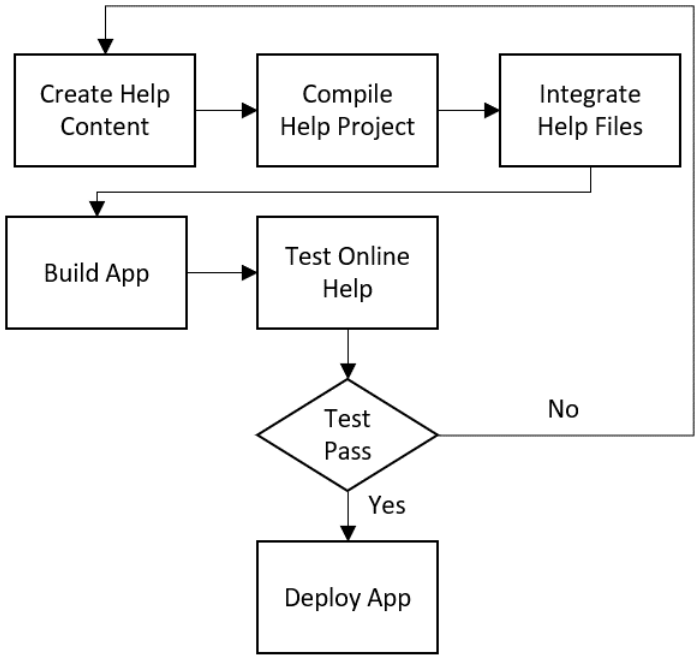

Figure 1. Process for delivering main help file

The process outlined in figure 1 highlights that several handoffs occur in the process. An unseen handoff is the trigger of the process itself in which new features need to be documented or a revision is needed. Testing all CHM files for each supported language is performed manually and time consuming.

The process for creating .NET API is depicted in figure 2. Developers insert XML comments into the source code of the app to document public API that is available to end users, for instance OEM developers. These comments are used by Sandcastle, a tool for generating documentation, to build a CHM file. The Sandcastle project is managed by the development team, and the CHM file is created using the build pipeline. This means the version of the app made by the build pipeline contains the most recent version of .NET API. The online help for the .NET API is briefly tested by a team of testers before it is approved and deployed with the app.

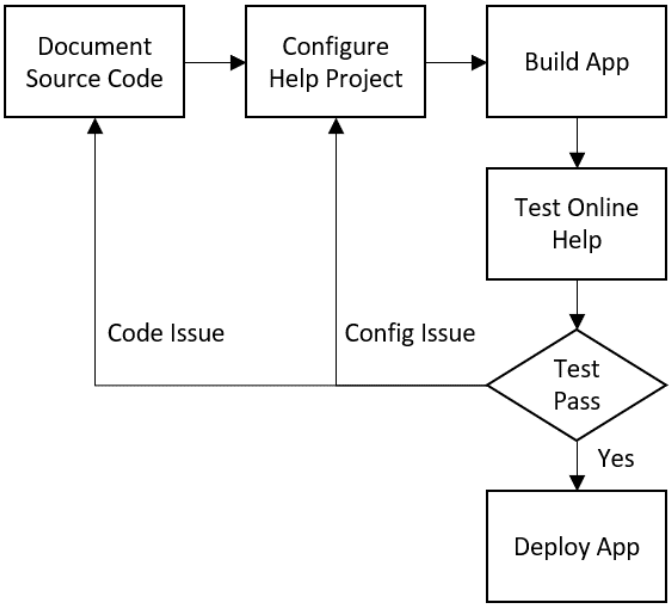

Figure 2. Process for delivering .NET API

The main handoff shown in figure 2 is when a test fails, and this may occur when building the app or during QA testing. Config issues are less likely to occur since the Sandcastle project has minimal change frequency. The risk of code issues or human error is possible since developers manually insert XML comments for their code as well others. Generally, errors are caught during the build process since the CHM file is compiled in a build pipeline. Testing the CHM file is limited to checking if the file and its content are visible and searchable. Any error requires a developer, application engineer or IT admin to fix the issue.

The outputs of both processes are CHM files shipped with the app. The CHM files for the API must be accessible and searchable in the online help. The help project managed by the team of application engineers must reference and merge those files before compiling the main help project. This requires inserting references to the API files in the online help's table of contents. Furthermore, the referenced files must be stored in the same directory as the online help file, which is why the installer of the app installs all help files in one directory.

2.3 Process for VAR and OEM Online Help

VARs and OEMs of the case company take one to several approaches regarding the help system of the app. One approach is to use the same online help as the case company. For branding reasons, the name of the app in the online help might be changed by either party.

Another approach is the case company creating online help for the OEM. Generally, an agreement is made to distinguish what features the case company documents for the OEM, for example features specific to the OEM variant of the app. The opposite of this approach is for the OEM to make its own online help, thereby the case company ensures the app can support it.

The last approach to mention is the merging of online help made by both parties. Therein lies one of the challenges of using CHM files for online help. The online help made by a VAR or OEM and case company need to be in the same format to have a one, integrated system for end users. If the formats are different, the online help of the app would need to create a link to the other file, for instance a packaged PDF file or remote source such as a web site. These external sources would also not be searchable in the online help of the app. In either case, the help project would need to be modified and recompiled to support this approach.

2.4 Challenges of CHM format

The use of CHM files creates challenges for the case company and its stakeholders. For example, CHM is old technology first released with Windows 98. It does not fully support UTF-8 and uses the Trident engine of Internet Explorer (IE), which has limited support for web technologies and security risks. By default, a CHM file will use IE7 to display its content on a machine running Windows 7 and later. This means technical writers and content creators cannot use HTML5, CSS3 nor JavaScript/ECMAScript 2018 to author HTML content for a CHM without there being potential rendering issues. A workaround to make a CHM use a version of IE that supports these technologies is known. It requires editing registry items related to the feature browser emulation of IE. Note that viewing a CHM file on a mobile phone would require a third-party tool.

The compiling of a CHM file presents another challenge because the project language and locale of the host computer affects the help file's table of contents and search engine. For example, a CHM file meant for Japanese end users compiled with an English locale and language would not display Japanese characters correctly in the table of contents nor support those characters in any search query.

Merging and localization are other factors to highlight. Working with CHM files requires technical knowledge of Microsoft HTML Help and its project files to properly merge them. Localizing a CHM file requires it to be decompiled or unzipped, translated, and then recompiled with the added risk of making a new project for the localized version.

3 Research Method

This section describes the research method for analysing the help system of the case company. It details how the research was conducted, software and services needed to perform tests, and the data used in the tests.

3.1 Type of Research

The type of research conducted was qualitative and focused on a software review of the case company. This required learning about the help system of the case company, its processes, and stakeholders. It also involved development work to create and test different approaches for generating and viewing online help. Working at and using the software of the case company helped with data collection. That is, background and technical knowledge of the case company and its services reduced time and effort in collecting data. This should, however, be remarked as a potential bias in any proposed solution.

3.2 Tools Needed

The following tools were used to analyze the help system, test WPF controls, and collect data:

- Laptop with 32GB of RAM and user admin rights
- Windows 10 Pro
- Visual Studio 2019 Enterprise
- Visual Components Premium 4.3.0
- HTML Help Workshop
- Notepad++
- ShareX
- Command Prompt
- Adobe RoboHelp 2020

Visual Studio was used to develop plugins for Visual Components Premium with the aid of its API. The CHM files of Visual Components Premium were decompiled using HTML Help Workshop then made into new projects. The command prompt was used to quickly compile projects. Notepad++ was used to edit the project files of decompiled CHM files. RoboHelp was used to generate HTML5 content and test processes. For example, application engineers at the case company use RoboHelp, while some of its VAR and OEM stakeholders use a similar help authoring tool that can create HTML5 content.

3.3 Procedure

The first step in the research focused on the rendering limitations of a CHM file by exploring different ways to view its content. This is also served the matter of online help being limited to one display language per app session. The next step focused on merging online help into one system or project. Finally, a process was tested in which online help of the case company was converted to HTML5 output, and then accessed and viewed from the app. Afterwards, modifications to the HTML5 web help were made to test risks, time, and effort for making changes.

4 Software Review

This section details a software review for the online help of the case company. First, there is an overview on implementing a plugin for the app using its API. Next, tests are described for accessing and viewing online help with WPF controls. Methods for decompiling and merging online help are explained. The process for converting online help to HTML5 output is covered with a breakdown of the work required to implement it.

4.1 Developing a Test Plugin

The app of the case company is a WPF desktop application that uses the .NET Framework. The Model-View-ViewModel (MVVM) design pattern is used for controls and their logic to be worked on separately and (binded) bound together to perform the desired results [1]. The Managed Extensibility Framework or MEF is used to support plugins. This enables the ability to import items needed to develop a plugin, and then export the plugin as a type recognized and used by the app [2]. WPF resources are used for localization as well as styling of the app. That is, a plugin can embed a WPF user control with the same look and feel as the app. Note that the option to create and call a client application using a plugin is available, but take note the client app is a separate process from the main app.

API for the app is available for .NET languages as well as Python. The .NET API is meant for development of add-ons known as plugins. Python API is meant for scripting but can also be used for add-on development. The .NET API was chosen for developing a test plugin using C# language, thereby a .NET Class Library project was created in Visual Studio. The target .NET framework was version 4.7.2, the DLL was built in the same directory as the app and its name used a prefix of “Plugin.” to help identify the DLL as a plugin.

4.1.1 View Model

The ViewModel (Appendix 1) for the test plugin was exported as both a plugin and action item. The action item is a reusable command that can be executed independently of the plugin, for example called directly by another command or reused in other areas of the app. In this case, the action item is listed as a UI element on the Ribbon of the app under a Help tab as depicted in figure 3.

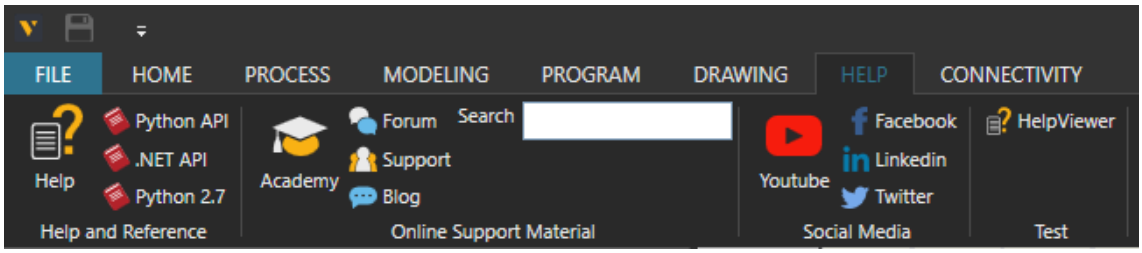

Figure 3. Help tab for online help

Note the amount of online help controls in figure 3. Each UI element in the Help and Reference group is implemented as a separate action item. That is, one action item with passable arguments is not used to service a call for specific online help, and each action starts a new process for every opened help file.

4.1.2 View

The first control tested in the review was a WebBrowser control. The View (Appendix 2) of the test plugin was a WPF user control containing a WebBrowser. It is important to note that a WebBrowser control and the one used by Windows to display a CHM file are similar. Both use IE features and an ActiveX control [3-6]. When a CHM file is opened, a program named “hh.exe” is used to open the file in a HTML Help ActiveX control. Security issues with ActiveX controls and IE are common in both. The WebBrowser control is subject to WPF security and navigation issues as well [6].

4.2 Displaying CHM Content in a User Control

Content in a CHM file can be accessed using a URI scheme supported by IE technology [7]. The URI schemes “mk@MSITStore:”, “ms-its:” and “its:” are supported by a WebBrowser control as shown in figure 4 and can open the table of contents and HTML files of a CHM. These URI schemes do work in web browsers other than IE, for example Chrome, but a browser that does not use IE technology may prompt to open IE to access the file.

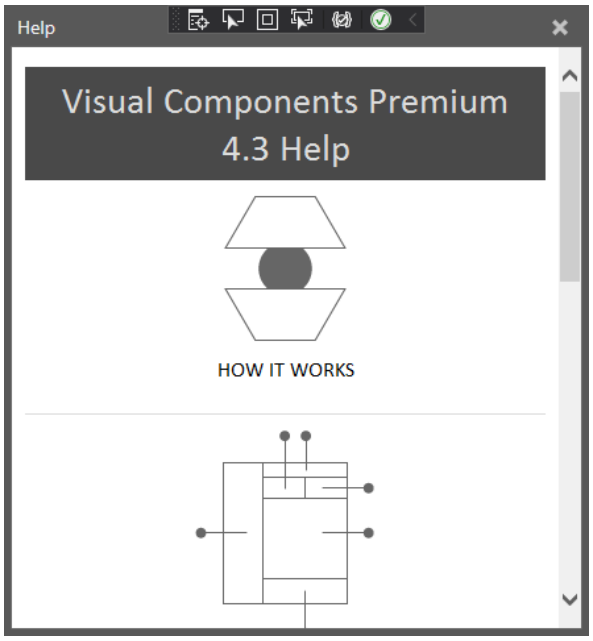

Figure 4. Landing page of online help

The online help in figure 4 indicates design can play a factor in a help system. For example, accessing or displaying the table of contents of a CHM file becomes unnecessary if the HTML page itself has site navigation. The CHM format can still be used in a help system based on this knowledge.

4.2.1 Limitation of Frame Control

The URI schemes for accessing CHM content are not supported by a Frame control. Tests resulted in URI prefix errors and exceptions in the app. This is a result of how a Frame control, which creates a WebBrowser ActiveX control instance to navigate to a URI, handles an invalid URI [8]. A WebBrowser control, however, is a wrapper for a WebBrowser ActiveX control, so the control and wrapper can handle this type of URI issue [4,8].

4.3 Frame and Web Browser Comparison

Both Frame and WebBrowser controls can access online help in other formats, such as content hosted on a web server. One observable difference was the handling of HTTP requests and responses by the controls out of box as illustrated in figures 5.

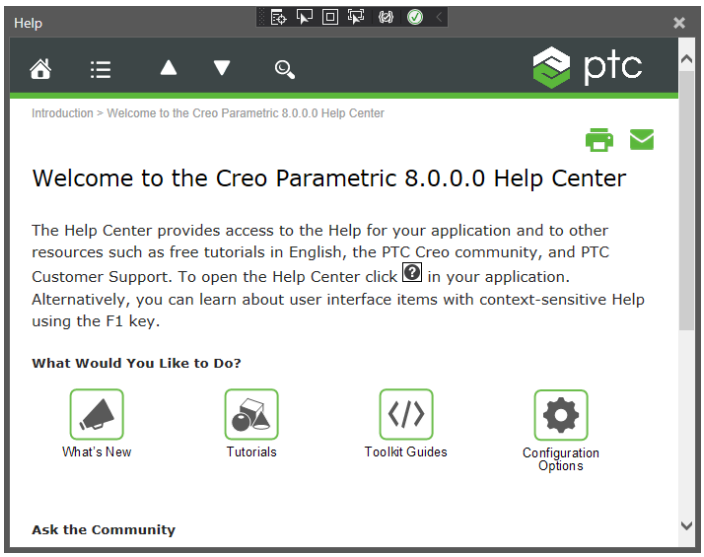

Figure 5. Online help retrieved from server using WebBrowser

The WebBrowser control in figure 5 was given a URI to an online help system using HTTPS. Navigation to the site and handling of the web requests were successful with no additional coding. A Frame control given the same URI to its Source property would require additional handling of HTTP requests and responses [9]. Figure 6 displays the exception thrown by the app when the Frame tried to display the online help system.

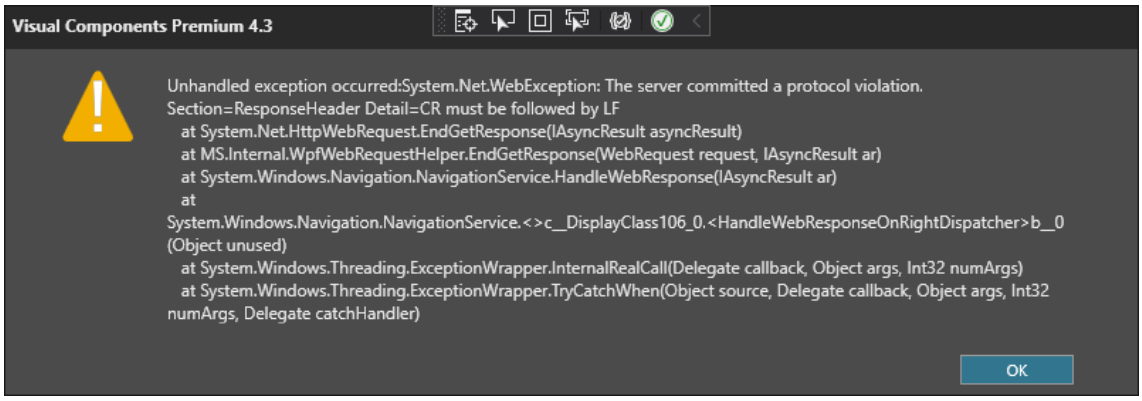

Figure 6. Exception in trying to access help from web server

The exception in figure 6 highlights an additional layer of development when trying to access web sources. No notable differences were observed in accessing local HTML files in tests. For example, the Source property of both controls supports site of origin pack URIs, which can be used to online help files installed with the app. One discrepancy is a pack URI with the URI schemes supported by a CHM did not work and failed to load the content.

4.4 Evaluation of WebView2 Control and Client Apps

A test was done to evaluate the WebView2 control in displaying online help with the app [9-10]. WebView2 uses a different rendering engine than WebBrowser, Frame and other controls that use IE technology. The test proved worthwhile in that WebView2 is not suitable for the app.

The setup for using a WebView2 control required installing the WebView2 Runtime on the machine hosting the test plugin. An alternative was to install a preview version of Microsoft Edge based on Chromium to the machine. Afterward the NuGet package for the control was added to the test project.

An initial test was attempted to use the WebView2 control in a WPF user control. This test did not work. At the time of test there was a known limitation of WebView2 control. If the app runs in admin mode, the WebView2 does not load properly and data director errors might occur. Note that the app of the case company requires user admin rights to install it. One workaround for this issue would be to have a backup View using a WebBrowser control or to run a WebView2 control in a client application.

4.4.1 Client App Test

A WPF application was developed to function as a client app hosting online help. Setup for the client app to use a WebView2 control was straightforward (Appendix 3). Displaying local HTML files worked, but the WebView2 control could not display CHM content and instead displayed an empty page. Errors in accessing data directories also occurred in some tests. In comparison, a WebBrowser control in the client app worked in displaying different formats of online help, but software security issues did occur based on the IE settings of the test computer [5].

4.5 Merging CHM Files

It is possible to create a help system with merged CHM files from different sources. HTML Help Workshop can be used to create a project file. The project file defines the configuration for building the project, what window to use for displaying content, and other CHM files to merge when compiling the project (Appendix 4).

Merged files are referenced in a compiled CHM file and work in searches. The compiler, however, does not create copies of merged CHM files, so they need to be added to the same directory as the compiled CHM file [12]. To display the table of contents of merged files, the project needs to reference those files in its table of contents file as object elements with a sitemap in an unordered list (Appendix 5).

This approach was tested using every CHM file of the app. One project merged all CHM files, thereby making them accessible to a user as depicted in figure 7.

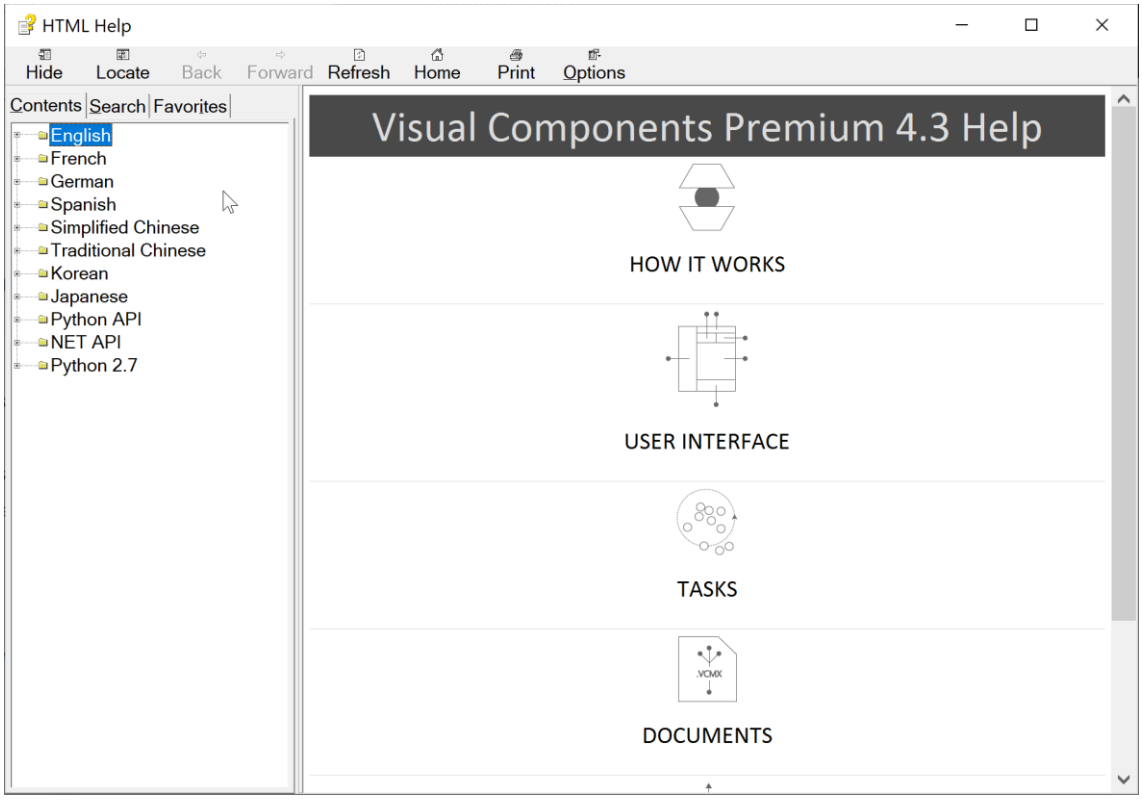

Figure 7. Integrated help system for supported languages

On the surface, the online help illustrated in figure 7 is a simple solution with inherit design flaws. First, the amount of information presented in the table of contents can be overwhelming. One CHM file used by the case company has over 900 files. However, help files for a plugin should not be extensive as online help for the app.

One workaround to limit merged files is to reference an external source of help content, for example a website, in a merged HTML page. Content from the external source, however, would not be searchable in the CHM file.

The locale issue of building a CHM file is also inherit when browsing files as shown in figure 8. The HTML Help ActiveX control can display Chinese, Japanese, and Korean character sets in the main view. However, the locale and region of the computer building the CHM is used for searches and displaying table of contents entries. This means characters in the table of contents may not show correctly and searches using these character sets will not return results.

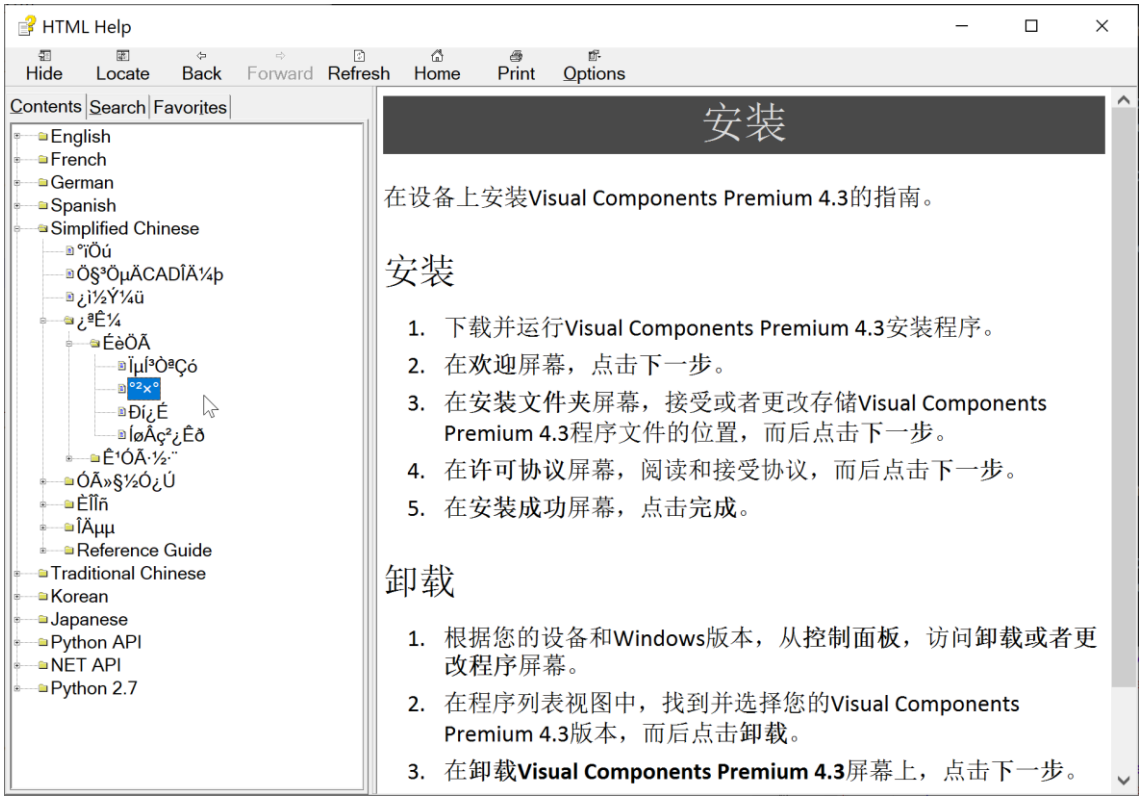

Figure 8. HTML Help built using English - US locale and region

The encoding issues illustrated in figure 8 highlights a limitation of using the CHM format for online help, especially for an app being used worldwide.

If a help system needs to create a CHM file with content in multiple languages which require extended character sets for ASCII support, workarounds are limited. One approach is to redesign the HTML pages to provide site navigation and perhaps search functionality. This approach does not fix the search and table of contents issues with the HTML Help ActiveX control. Making different versions of online help for each language is another option. This requires building the CHM files with the correct language and locale for the intended end users, specifically to support searching. Changing the region and language of a computer and compiling the files can be automated using a batch file. This is currently done by the case company. Using another control to display CHM content is possible (see section 4.2 Displaying CHM Content in a User Control). That approach could take advantage of HTML Help API to display content or support the development of a client app for hosting online help [13].

4.6 Generating HTML5 Web Help

The last test in involved taking the existing CHM files of the app and converting them to HTML5 content. This format allows the online help to be viewed by a web-friendly device such as a mobile phone and web browser. The content was created using RoboHelp 2020.

The workflow for creating a working solution required a few steps. First, a CHM file of the app was decompiled to access its HTML files. Next, a project was created in RoboHelp. The directories and HTML files from the decompiled CHM were imported to this project. The imported HTML files had markup issues, for example unclosed HTML tags. The AutoFix feature of RoboHelp resolved these errors. A table of contents was automatically generated from the imported files, and then the project was compiled into HTML5 content, in this case a frameless, responsive web app. The entire workflow from start to finish took a few minutes to complete.

Online help in this format can be displayed using a web browser. It can also be displayed in a WPF control as depicted in figure 9.

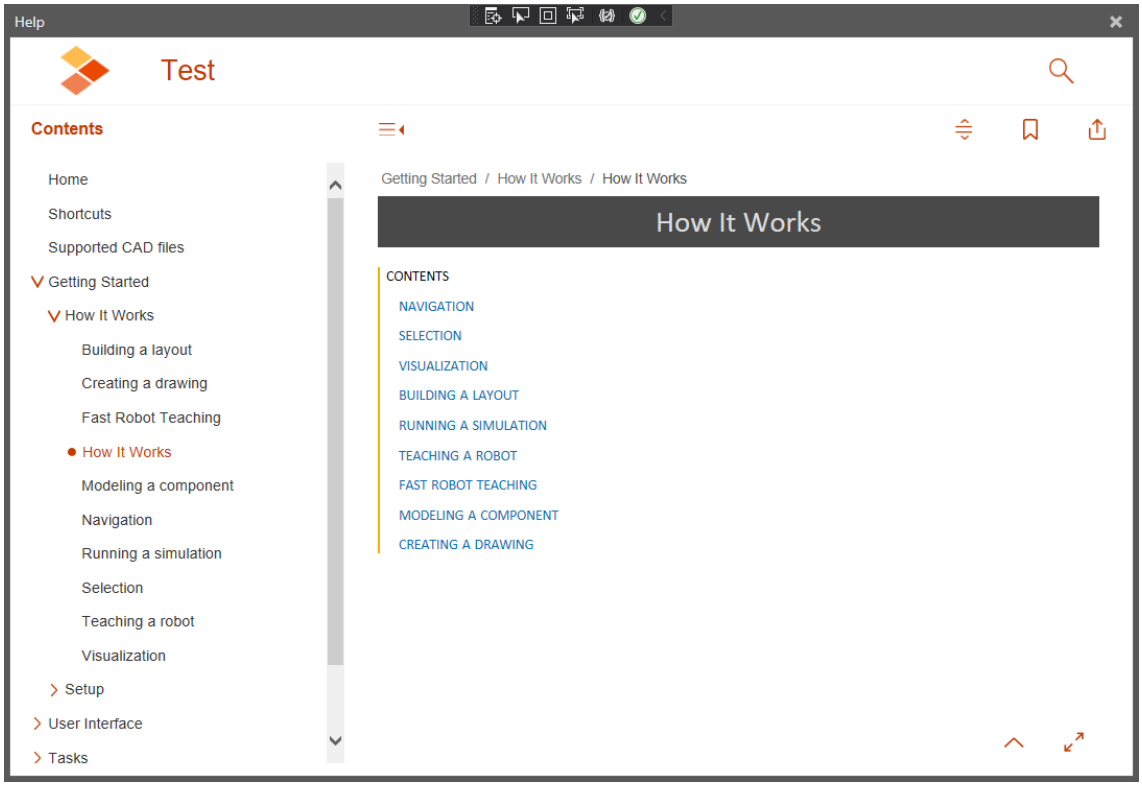

Figure 9. HTML5 help content in WebBrowser control

The structure of the online help shown in figure 9 had issues. First, the breakpoint for the table of contents element was too large, requiring the window to be increased in size to display it. The user experience of viewing the HTML5 content in an embedded control versus a web browser such as Chrome was also noticeably different. Importing HTML files decompiled from a CHM also generated additional entries in the table of contents. This issue can be fixed, however, by removing the entries and specifying bookmarks in HTML pages not be considered topics/sections in a table of contents.

4.6.1 Merging Additional Files

The process for merging additional content to the HTML5 output has setbacks. First, merging other RoboHelp projects can be done, but different content authoring tools are used by the VARs and OEMs of the case company. Merging CHM files is supported, but the project itself would need to be compiled as a CHM file. Manually inserting files requires technical knowledge and possibly reverse engineering of the output generated by RoboHelp. For example, the HTML5 web app is heavily reliant on scripts, which would require modification to integrate new content manually. Furthermore, there is no public documentation on these scripts, so the process of manually editing the files would take time. These setbacks can be mitigated if the content creators for online help require only the HTML files, which can then be used to generate a new version of online help.

5 Proposed Process

This section describes a process for migrating a help system to generate content that supports HTML5. Although personas are not explicitly stated, the process considers different types of users associated with help systems. It also factors in the needs of different organizations that want to integrate their content into a common help system.

5.1 Identify the Need

The first phase of the process involves identifying the need to migrate. At the case company content creators are limited by the technology of their help system. Testing is time consuming. Several handoffs are required to complete an update process. Online help is shipped with the app, but revisions cannot be done until the next release and content is not accessible on the Internet. The online help might not display content properly nor support searching, which impacts users who need assistance. The needs in this case are to avoid limitations and optimize services for end users.

5.2 Strategy for Migration

The second phase defines a strategy for addressing the needs. The work and costs for migrating can be estimated. For example, evaluating the current help system and development needed to address legacy issues such as action items that open help files. Proposals can be evaluated to purchase a new tool to generate help files or use an online repository for managing the files. The strategy should also consider how the help system is used by others. The VARs and OEMs of the case company need the online help for their own products and the ability to merge new content. A design review should be done before the next phase to ensure the strategy makes sense both for the company and its stakeholders. This is an important decision since the case company is also a platform provider.

5.3 Delivery of Help System

The third phase implements the strategy as a project. There is a clear start to the project and planned end date. During the project, sanity and other forms of testing can be done. This helps review the strategy, its design, and the scheduled work. The goal being to ensure the needed outcome is delivered.

During this phase, challenges might occur and require change. A developer, for example, might find issues with implementing a WPF control or process that displays the online help. An application engineer might discover that a new tool is difficult to use and suggest an alternative to making the HTML files. Others might not like the branding and styling of content. These changes would have to be managed.

The project can be considered done when feedback and testing are completed. For example, different teams in the company and stakeholders have accepted the new help system and it can be put into production.

5.4 Adoption

The last phase is the adoption of the new help system and online help in the app. The help system should generate content that can support HTML5. It should also support different languages and be viewable in different devices. The amount of work to maintain the system and make corrections should be stable at this point. For example, an application engineer at the case company could add new content for a feature, commit the changes to a repository, and then handover the changes to the build system. From there, the build system generates a new version of online help for the app. Additional work could entail some form of automatic testing for the online help, such as checking hyperlinks, site navigation to different topics, content is being displayed correctly, and search functionality is working.

6 Conclusion

This thesis addressed the need of software companies to upgrade or migrate their help system by providing a set of tests and blueprint on how to plan and implement the change. Many tools can generate online help from a project of HTML files. Containing the project in a compiled HTML format or CHM enables the content to be viewed locally with the added benefits of site navigation and search functionality. The format and its control, however, are limited by its technology and present challenges for apps that need to provide online and offline assistance to users worldwide and on demand. Displaying CHM content in other controls is possible and one option to consider. Another option is to generate online help as a web application or site that supports HTML5. This format can be viewed offline and online, support the latest web technologies, service any region and locale, and be revised quickly to correct errors and limit the impact to users. The tool, control and format of online help is less important, however, to the processes that govern the development and management of a help system.

The process of migrating online help to a format other than CHM has been ongoing at the case company for many years. During that time, the need to migrate to another format was known, and several pilot projects were done. The migration was not successful due to strategy and adoption. That is, there was no approval for the migration and other stakeholders were not included in the project. The recommendation for the case company is to formulate a strategy for migrating their help system, include stakeholders in planning and implementing the project, and have criteria for adopting a new version of online help.

In retrospective, this thesis raised some questions for future research. First, it is unknown why the help system for Office 365 apps is not packaged as a tool or developer kit for others to use in their own apps. The viewer of a CHM file and a WebBrowser control can render HTML5 content. However, why is the default in Windows for these controls to not use the latest version of IE that supports them such as IE11? What are the foreseeable benefits of the WebView2 control and roadblocks for widespread adoption of it? Lastly, what are the preferences of users and companies with regards to online help in the modern era going forward: online content driven by community of users, predictive and suggested tips, automated troubleshooting, and modular content management systems?

References

1 davidbritch. The model-view-ViewModel pattern [Internet]. Microsoft.com. [cited 2021 Apr 28]. Available from: https://docs.microsoft.com/en-us/xamarin/xamarin-forms/enterprise-application-patterns/mvvm

2 gewarren. Managed Extensibility Framework (MEF) [Internet]. Microsoft.com. [cited 2021 Apr 28]. Available from: https://docs.microsoft.com/en-us/dotnet/framework/mef/

3 HTML help ActiveX control overview [Internet]. Microsoft.com. [cited 2021 Apr 28]. Available from: https://docs.microsoft.com/en-us/previous-versions/windows/desktop/htmlhelp/html-help-activex-control-overview

4 adegeo. WebBrowser Control Overview [Internet]. Microsoft.com. [cited 2021 Apr 28]. Available from: https://docs.microsoft.com/en-us/dotnet/desktop/winforms/controls/webbrowser-control-overview?view=netframeworkdesktop-4.8

5 adegeo. Security - WPF .NET Framework [Internet]. Microsoft.com. [cited 2021 Apr 28]. Available from: https://docs.microsoft.com/en-us/dotnet/desktop/wpf/security-wpf?view=netframeworkdesktop-4.8

6 About the HTML help executable program [Internet]. Microsoft.com. [cited 2021 Apr 28]. Available from: https://docs.microsoft.com/en-us/previous-versions/windows/desktop/htmlhelp/about-the-html-help-executable-program

7 About HTML Help URLs [Internet]. Microsoft.com. [cited 2021 Apr 28]. Available from: https://docs.microsoft.com/en-us/previous-versions/windows/desktop/htmlhelp/about-html-help-urls

8 dotnet-bot. Frame Class [Internet]. Microsoft.com. [cited 2021 Apr 28]. Available from: https://docs.microsoft.com/en-us/dotnet/api/system.windows.controls.frame?view=net-5.0

9 karelz. HttpClient Class [Internet]. Microsoft.com. [cited 2021 Apr 28]. Available from: https://docs.microsoft.com/en-us/dotnet/api/system.net.http.httpclient?view=net-5.0

10 MSEdgeTeam. Microsoft Edge WebView2 control - Microsoft Edge development [Internet]. Microsoft.com. [cited 2021 Apr 28]. Available from: https://docs.microsoft.com/en-us/microsoft-edge/webview2/

11 MSEdgeTeam. Getting started with WebView2 for WPF apps - Microsoft Edge Development [Internet]. Microsoft.com. [cited 2021 Apr 28]. Available from: https://docs.microsoft.com/en-us/microsoft-edge/webview2/gettingstarted/wpf

12 Merging Help Files at run time [Internet]. Microsoft.com. [cited 2021 Apr 28]. Available from: https://docs.microsoft.com/en-us/previous-versions/windows/desktop/htmlhelp/merging-help-files-at-run-time

13 HTML Help API Overview [Internet]. Microsoft.com. [cited 2021 Apr 28]. Available from: https://docs.microsoft.com/en-us/previous-versions/windows/desktop/htmlhelp/html-help-api-overview

ViewModel for Test Plugin

```
using System;
using Caliburn.Micro;
using System.ComponentModel.Composition;
using VisualComponents.UX.Shared;

namespace Plugin.HelpViewer
{
    [Export(typeof(IPlugin))]
    [Export(typeof(IActionItem))]
    public class HelpViewAction : ActionItem, IPlugin
    {
        //associate icon with action
        public HelpViewAction():base("HelpViewer","HelpViewer","Help/rHelp")
        {
        }

        public override void Execute()
        {
            IDockAwareWindowManager windowManager =
IoC.Get<IDockAwareWindowManager>();
            windowManager.ShowFloatingWindow(String.Empty, new
HelpViewModel());
        }

        public void Exit()
        {
        }

        public void Initialize()
        {
            AddActionItemToUI();
        }

        private void AddActionItemToUI()
        {
            //register command and its site to ribbon group
            var cmdReg = IoC.Get<ICommandRegistry>();
            var actionItem = cmdReg.FindItem("HelpViewer");
            var item = new UXSiteSetup
            {
                UXSiteIdPath = "VcTabHelp/Test",
                UXSiteCaption = "Help",
                EntryCaption = "HelpViewer",
                EntryId = actionItem.Id,
            };
            cmdReg.RegisterActionItem(actionItem, item);
        }
    }

    //simple floating window used by action to display viewer
    class HelpViewModel : DockableScreen
    {
        public HelpViewModel()
        {
            this.PanelId = "VcHelpPanel";
            this.DisplayName = "Help";
        }
    }
}
```

View for Test Plugin using WebBrowser Control

```
<UserControl x:Class="Plugin.HelpViewer.HelpView"
             xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
             xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
             xmlns:mc="http://schemas.openxmlformats.org/markup-
compatibility/2006"
             xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
             xmlns:local="clr-namespace:Plugin.HelpViewer"
             mc:Ignorable="d"
             d:DesignHeight="450" d:DesignWidth="800">
    <WebBrowser

Source="mk:@MSITStore:C:\Program%20Files\Visual%20Components\Visual%20Componen
ts%20Premium%204.3\Help\Help_English.chm::/English/Help.htm" />
</UserControl>
```

View for WPF Application using WebView2 Control

```
<Window x:Class="MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
        xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
        xmlns:local="clr-namespace:TestHelpViewer"
        xmlns:wv2="clr-
namespace:Microsoft.Web.WebView2.Wpf;assembly=Microsoft.Web.WebView2.Wpf"
        mc:Ignorable="d"
        Title="MainWindow" Height="450" Width="800">
    <Grid>
        <wv2:WebView2 Name="webView"
                      Source="C:\Program Files\Visual Components\Visual Components
Premium 4.3\test.html"
    />
    </Grid>
</Window>
```

Project File for a Merged HTML Help Project

```
[OPTIONS]
Contents file=ToC.hhc
Compiled File=Help.chm
Title=Help
Default topic=Help_English.chm::/English/Help.htm
Full-text search=Yes
Auto Index=No
Binary Index=Yes
Enhanced decompilation=No
Binary TOC=No
Flat=No
Default Window=Test

[WINDOWS]
Test="Help","ToC.hhc",,"Help_English.chm::/English/Help.htm","Help_English.chm
::/English/Help.htm",,,,,0x63520,222,0x386e,[618,334,1218,736],,,,,,,0

[MERGE FILES]
Help_Chinese.chm
Help_English.chm
Help_French.chm
Help_German.chm
Help_Japanese.chm
Help_Korean.chm
Help_Spanish.chm
Help_TraditionalChinese.chm
PublicAPI.chm
python271.chm
Python_API.chm
```

Table of Contents File for Merged HTML Help Project

```
<html>
<!-- Sitemap 1.0 -->
<object type="text/site properties">
  <param name="SiteType" value="toc">
  <param name="Image Width" value="16">
  <param name="Window Styles" value="0x800027">
  <param name="ExWindow Styles" value="0x100">
</object>
<ul>
      <!-- ENGLISH -->
    <li><object type="text/sitemap">
      <param name="Name" value="English">
    </object>
    <ul>
    </ul>
    <object type="text/sitemap">
      <param name="Name" value="Help_English.chm::/Help_English.hhc">
      <param name="Merge" value="Help_English.chm::/Help_English.hhc">
    </object>
    <ul>
      </ul>
     <!-- FRENCH -->
     <li><object type="text/sitemap">
      <param name="Name" value="French">
    </object>
    <ul>
    </ul>
    <object type="text/sitemap">
      <param name="Name" value="Help_French.chm::/Help_French.hhc">
      <param name="Merge" value="Help_French.chm::/Help_French.hhc">
    </object>
    <ul>
      </ul>
</ul>
</html>
```