Infotron PerfectXL Ribbon
Final report

T.I. Molenaar
N. Radenovic

Technische Universiteit Delft

Infotron PerfectXL Ribbon

Final report

by

T.I. Molenaar
N. Radenovic

Bachelor's Thesis

Faculty of Electrical Engineering, Mathematics and Computer Science

Delft University of Technology

July, 2016

Presentation date:	July 1, 2016	
TU coach:	Sohon Roy	
Coordinators:	Dr. ir. Felienne Hermans	Bachelor Coordinator
	Otto Visser	Bachelor Coordinator
	Martha Larson	Former Bachelor Coordinator
Client:	Mateo Mol	Infotron B.V.
	Joeri van Veen	Infotron B.V.

Preface

In this report, the results of the Bachelor project of Computer Science students Tim Molenaar and Natasa Radenovic are presented. Over the course of two months we have developed a Ribbon in Excel that is able to communicate with an existing web application called PerfectXL.

We would like to thank Sohon Roy, our supervisor, for his patience and dedication to the project. Without him, we would have been left with many uncertainties.

We also want to thank Mateo Mol and Joeri van Veen of Infotron, who have commissioned this project, for allowing us to create their product. Their support and flexibility allowed us to do the project in what way we felt would achieve the best results.

T.I. Molenaar
N. Radenovic
Delft, July 2016

Summary

The focus of the project was to create an Excel Ribbon for the spreadsheet analysis tool PerfectXL. The tool allows users to upload Excel workbooks, have them analyzed for risks, and download the analysis results. The Ribbon would allow users to upload Excel workbooks and retrieve the analysis results from within the Excel environment.

First, a two week research phase was conducted. During this research phase, it was established that Visual Studio 2015 would be used for the programming environment, and to use Windows Installer for the deployment of the Ribbon.

The design is based on a separation between the functionality of the Ribbon and its presentation. This design decision was made so that the functionality code could easily be reused for new versions of Excel. Only the presentation would have to be modified.

The implementation conforms to the design, and relies on background processes. These background processes allow the Ribbon to run in the background without blocking the Excel environment.

The testing of the product is based on unit tests, where all classes are tested individually after every new functionality. The performance of the add-in was also tested with Excel workbooks of various sizes, where the upload time and the analysis time were separated. The findings of these performance tests was that the PerfectXL web application is the biggest bottleneck in the performance of the Ribbon.

The final product conforms to the client's needs and requirements.

Contents

1 Introduction

Infotron B.V. is a start-up closely associated with the Delft University of Technology. Their aim is to facilitate the use of Microsoft Excel, which is used by 750 million people worldwide. Their focus is on offering an automated spreadsheet analysis to Excel users in order to better understand problems in their spreadsheets.

Infotron developed a web tool called PerfectXL. It is a spreadsheet analysis and validation tool, and it checks the quality of uploaded workbooks. It helps in visualizing spreadsheets and detects risks, like fixed numbers in formulas, incorrect ranges, and overly complex formulas. It works from within a browser, where it allows users to upload workbooks and see the analysis results.

Infotron, however, was not satisfied with the amount of steps a user must take to analyze their workbook. To upload a workbook, the user would first have to save the file in Excel, open a browser, navigate to PerfectXL, log in, navigate to the folder where the file is located, and select it. Infotron wanted to reduce the amount of steps a user must take to upload a file and obtain the result of the analysis. This motivation led to the conception of this project.

The purpose of this project is to design and build an Excel Ribbon which will enable the user to automatically upload the opened workbook and retrieve the analysis results from within the Excel environment. The Excel Ribbon must be user-friendly, and must not generate a large amount of overhead.

This report describes the process of designing and implementing the Excel Ribbon over the course of 10 weeks. In Chapter 2, we discuss the two week research phase, including the requirements where we analyzed the various facets of the problem and explored possible solutions. Next we describe the use cases in Chapter 3. In Chapter 4, we describe the project plan and the tools that were used during the project. In Chapter 5 and 6, we discuss the design and implementation of the Excel Ribbon respectively. The testing is described in Chapter 7, followed by discussion of limitations and possible future work in Chapter 8. Finally, in Chapter 9 we conclude this report.

2 Research report

2.1. Problem description

The aim of this project is to develop an Excel add-in for Infotron's subscription-based web application, PerfectXL. PerfectXL is able to visualize spreadsheets and analyze them for risks. To analyze Excel worbooks, customers are able to create an account for PerfectXL and access its functionality in the browser. In the web application environment, they can upload an Excel document, which is analyzed by PerfectXL and saved for the user to view later. PerfectXL renders feedback on demand in PDF, Excel, or image format. By looking at these feedback documents, the user can, for example, find out if his Excel sheets contain inconsistencies. The user can also apply filters on information they wish to receive. The user can, for example, select image format as return type, containing the following filters:

- The amount of risks per sheet
- The amount of data per sheet

To reduce the number of steps in the process of analyzing an Excel workbook, users should be able to upload Excel workbooks to PerfectXL from within Excel, without using the browser detour. Excel contains functionality that is able to communicate with web applications. The main problem during this project is: what is the best way to communicate with PerfectXL through the Excel interface, taking into account that the user experience in this application should be similar to the user experience in the browser?

2.2. Requirements analysis

To illustrate the main problem further, this section will present the requirements of the final product. These requirements describe what features the product must, should, can, and won't have. The next section will use these requirements to divide the main problem into manageable subproblems.

Must haves

1. The add-in must have a login system
   (a) The add-in must require a login every time Excel starts
   (b) After installing the add-in and starting the add-in the first time, the login information must be stored on the user's computer
   (c) The username and password must be safely stored
2. The add-in must be able to send Excel workbooks to PerfectXL
3. The add-in must be able to receive images, Excel workbooks, and PDF files from the web application and present it in a comprehensive way:
   (a) PDF files must be opened with a PDF viewer (if present)
   (b) Images must be opened in the current Excel interface
   (c) Sheets must be opened in the Excel interface
4. The add-in must not require the user to install additional software
5. The add-in must not be developed in an open-source environment
6. The add-in must be able to wait 10 minutes for the application to respond
7. The add-in must work for Excel 2013/2016
8. The add-in must be able to make a distinction between paying and non-paying users and show the buttons accordingly
   (a) Free users can generate a basic visualization, a list of all formulas, and use the 'Help' feature
   (b) Paid users can use all functionalities of free users, and they can also generate a spreadsheet analysis, a list of risks, cel formats, and a PDF report.
9. PerfectXL must be able to receive requests from the add-in

Should haves

1. Results received from PerfectXL in Excel workbook format should be opened in the current Excel workbook

Could haves

1. The add-in could work for both installation types of PerfectXL
2. The add-in could be installed in multiple languages
3. The add-in could work for Excel 2010

Won't haves

1. The add-in won't be implemented for Excel 2007 and earlier versions
2. The add-in won't be able to set filter options for the analysis results

2.3. Subproblems

The main problem can be divided into several subproblems, which all contribute towards solving the main problem. The subproblems will be explored in the following order: The best way to establish a connection between Excel and PerfectXL, the best environment to develop the code, the best way to make the code compatible with different versions of Excel, the best way to send confidential information from the client to PerfectXL, the best way to store login information on the client's computer, the best way to retrieve analysis results from PerfectXL, the best way to visualize the results on the client's computer, and the best way to install the product on the client's computer.

2.3.1. How to connect Excel with PerfectXL?

The first question that should be investigated is: what is the best way to establish a connection between Excel and PerfectXL? This can be done in multiple ways.

One of the possibilities of extending Excel with custom functions, is to create a script with Visual Basic for Applications (VBA) [4]. These VBA scripts can be implemented directly in an Excel programming environment contained in the interface. The advantages of this method are:

- It is easy to communicate with the data in the Excel workbook, because the script is running in the Excel environment.
- VBA works for older versions of Excel.
- It is a favored language for creating Excel extensions and therefore a lot of examples can be found.

The disadvantages of this method are:

- Bigger solutions become complex, since Visual Basic is not a high-level language and VBA is usually organized in separate script files.
- The solution must be developed with a limited visual interface.
- The programming environment shows a limited amount of information and is relatively dated compared to other environments (next section).

An alternative solution is to build an additional option bar in Excel, called a Ribbon [7]. A Ribbon contains the possibility to visualize functionality in a user-friendly way and is programmable for the latest versions of Excel. A Ribbon can be developed in several programming environments. In the next section we will go through the possible options.

2.3.2. What is the best environment to develop the solution?

The second issue is to determine the best environment for the development of Excel Ribbons. There are multiple ways of creating a Ribbon in Excel, but many available solutions are created in Visual Studio [5]. Visual Studio is a development environment created by Microsoft to develop programs for Microsoft Windows, websites, web applications, and web services. Visual Studio has the following advantages:

1. Visual Studio provides comprehensive walkthroughs for the development of Excel add-ins [6]
2. It allows the programmer to easily create Excel add-ins in C#. A new project for Microsoft Office already has modules that are activated upon opening and closing a Microsoft Office program. What these modules do is determined by the programmer. Visual Studio also simplifies compilation, debugging, and deployment of the solution.

2.3.3. How to make the ribbon compatible with different Excel versions?

Visual Studio offers several types of Excel add-ins, namely one for each version of Excel 2007 and newer versions. Custom Ribbons in Excel were not available before the 2007 edition, so versions older than 2007 are not supported by the solution that will be developed for Infotron.

It is probable to assume that a significant amount of Infotron's customers do not use the newest version of Excel. They most likely use older versions, such as Excel 2010 or Excel 2013. To make the Ribbon compatible with these versions of Excel, a separate project should be created for each version that Infotron wants the Ribbon for. To prevent duplicate code, functionality that can be shared among different versions of Excel should be in only one location. A separate project can then be created for every new version of Excel. This project will operate as a 'shell' that uses the shared functionality and links it to the right version of Excel.

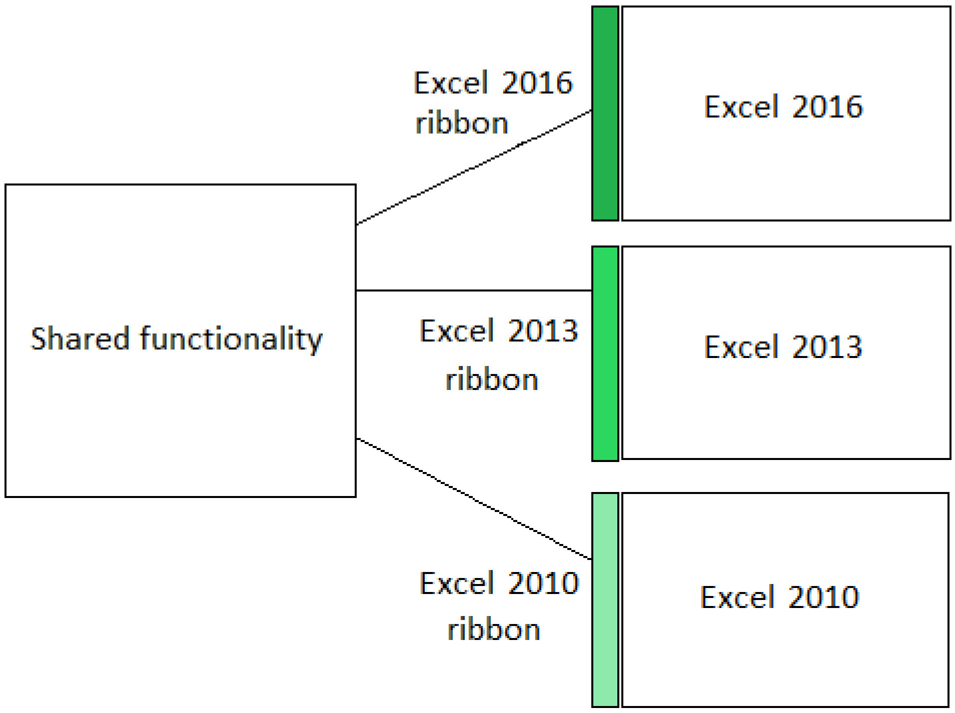


Figure 2.1: Functionality shared by Excel ribbons of different Excel versions should be reused to minimize future workload for ribbons built for newer Excel versions

2.3.4. How to establish a secure connection?

Users of PerfectXL might upload Excel workbooks which contain confidential data, that must not be obtained by third parties. To maintain this confidentiality, files should be sent in a safe way, ether over a secure connection or encrypted over a less secure connection. What is the best option for sending Excel workbooks from the user to PerfectXL?

Communication with the web application PerfectXL is currently done via a HTTPS-connection, which means that the data is automatically encrypted and is therefore not or less vulnerable to eavesdropping and man-in-the-middle attacks. [13]

An alternative is to zip Excel workbooks and password-protect the ZIP file before sending them over the web [12]. However, this option is less safe than sending the Excel workbooks over the HTTPS-connection, because the data is not encrypted.

2.3.5. How to store sensitive information?

PerfectXL is a subscription-based web application, which means that those using the Ribbon will have to log in. PerfectXL requests a email and password combination for authentication each time a new session is started before it gives a token that is used as authentication during the rest of the session. However, the add-in can make this process easier to the user. To increase the Ribbon's usability, the user should not have to enter the same login information during every session. Therefore, this login information will have to be stored safely on the user's computer after it is first entered. This sensitive information cannot be stored in plaintext, so it must be stored in a way that is only accessible to the add-in. It must also be modifiable, in case the user changes their password on PerfectXL and is required to enter their new password in the add-in.

There are two main methods of saving login information:

1. Store in local database
2. Store in encrypted file

A local database has many drawbacks:

- A local database is more difficult to set up than a secured file [3]. It requires creating the database, creating tables, columns, primary keys, and foreign keys, and populating the tables with data.
- A local database would require a security measure so the database cannot easily be accessed. This security measure is usually a password which must only be known by the add-in. This creates another issue on top of the complexity of setting up a database.
- All information presented in the add-in will be obtained through requests to PerfectXL, to always present the most recent information to the user. No information, except for the login information, will be stored on the user's computer. Creating a database to hold the login information of one user only adds unnecessary complexity to the add-in.

Whereas an encrypted file is easier to set up. Visual Studio has various tools that aid in encrypting and decrypting a file [2]. It allows programmers to easily generate keys and write encryption/decryption methods. For simplicity, an encrypted file should be used.

Storing the encryption key is another issue. There are two ways to safely store an encryption key [9]:

1. Split the key and put the parts in different locations.
2. Password-based encryption

Password-based encryption converts passwords into an encryption key. This is useful for applications that require users to enter their passwords at the start of every session. This, however, is not possible for the Ribbon, as the user's password should be remembered after the first session. Therefore, splitting the key and putting the parts in different locations is the better option.

2.3.6. How to retrieve data?

When an Excel document is sent to PerfectXL, it must first be analyzed before a result is generated. Depending on the document size and the complexity of the Excel sheets, this may take a few seconds or 10 minutes. If the sheets aren't processed within 10 minutes, the analysis will automatically be marked as failed and the user will be allowed to try again. After the analysis, the add-in must request the results of the analysis. But how will the add-in know when the analysis is done?

When an Excel document is sent to PerfectXL, it must first be analyzed before a result is generated. Depending on the document size and the complexity of the Excel sheets, this may take a few seconds or 10 minutes. If the sheets aren't processed within 10 minutes, the analysis will automatically be marked as failed and the user will be allowed to try again. After the analysis, the add-in must request the results of the analysis. But how will the add-in know when the analysis is done?

There are two methods:

1. The add-in can actively check in regular intervals if the analysis is done
2. PerfectXL can automatically send a confirmation when the analysis is done while the add-in waits.

To have the server send information without a request from the client would require web-sockets to be available on both sides. Web-sockets allow free two way communication between the client and server [11]. There is no guarantee that the client will support Web-sockets, so this method is less than ideal. Having the client request information is the preferred method, as opposed to having the server send information automatically, for simplicity.

Having the client request information is done through GET requests. There are multiple ways of doing GET requests in C# [8], among others:

1. HttpWebRequest
2. WebClient
3. HttpClient

Out of the three, HttpClient has better syntax for new threading features and allows for easier testing. However, HttpClient requires .NET Framework 4.5, which older machines may not have. This problem can be solved by automatically installing .NET Framework 4.5 along with the add-in.

2.3.7. How to present data?

Once the Excel workbook is sent and analyzed by PerfectXL, it will be retrieved by the add-in. The question remains: how should the system present the data to the user?

In case of a retrieved PDF file, the way to present it to the user is as follows: save the PDF file on the user's hard disk and open it (with a PDF reader).

For an incoming Excel workbook, it is harder to determine how to present it to the user. There are at least two ways of visualising the incoming Excel workbook:

1. Open the Excel workbook in a new Excel window
2. Open the sheets of the incoming Excel workbook in the current Excel window

We have to take into account that the incoming Excel workbook contains extra sheets with information added by PerfectXL. Therefore, the number of sheets in the analyzed document does not map with the currently opened workbook. This is relevant, because if we want to build option 2 where the current workbook will import the analysis sheets, the workbook is 'contaminated' with the data of PerfectXL. If the user wants to analyze a newer version of the current workbook after doing some changes suggested by the first analysis, the add-in must make sure that only the sheets that were in the original document are sent for analysis.

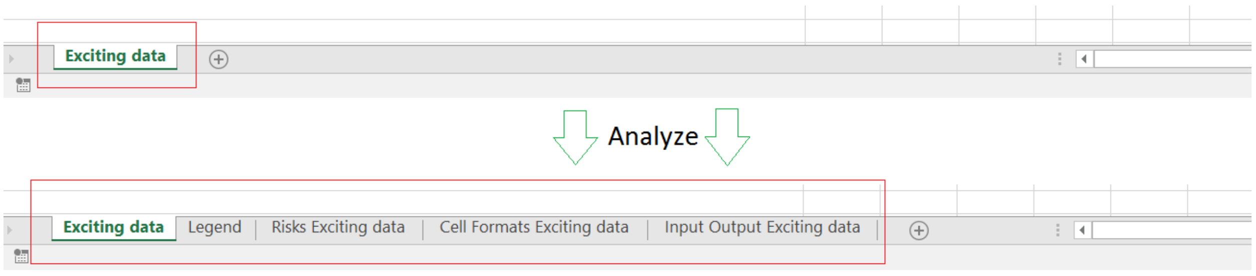


Figure 2.2: Loading the tabs created by PerfectXL into the current workbook 'contaminates' it.

Because the original Excel document might be analyzed more than once, it is best to store the original document and PerfectXL's analysis of the original document separately, to be able to differentiate between the two documents.

2.3.8. How to install the solution?

Once the add-in is ready for deployment, it must be installed on the computers of PerfectXL's users. During the installation process, users must be able to adjust some settings according to their preferences. The add-in will also be deployed in companies, where it will be installed on multiple computers. The installation process must be simple, quick, and comprehensive.

Visual Studio provides tools and tutorials to deploy Excel add-ins. It offers two methods [1]:

1. Using ClickOnce
2. Using Windows Installer

To determine which is better for the add-in's purposes, we compare the two methods:

Windows installer

Advantages	**Disadvantages**
Can be installed for all users of a computer	Updates require a new installer
Can install solution that requires administrative access on the user's computer	
Simple to setup custom installation process	

Table 2.1: Advantages and disadvantages of Windows installer.

ClickOnce

Advantages	**Disadvantages**
Published updates will automatically install	Must be installed separately for every user
	Difficult to setup custom installation

Table 2.2: Advantages and disadvantages of ClickOnce.

ClickOnce has advantages for private users. Its strength lies in its simplicity and ease of use. However, it is not suitable for large companies with private networks, because it cannot install solutions that require administrative access. With a Windows installer, users will be able to select custom options during the installation process. This would allow companies which run PerfectXL on their own server to use the add-in as well.

2.4. Conclusions of the research phase

Infotron wants to offer an option for their clients to communicate with the web application PerfectXL from within the Excel environment. Therefore, we will build a Ribbon which is an extension of the Excel menu bar where custom options can be built in. These options include analyzing an Excel workbook, and generating a visual representation in PDF, image, or Excel format.

The add-in will be developed in Visual Studio, which is an integrated development environment from Microsoft. We decided to choose Visual Studio as development environment because it is:

1. Often used for creating Excel add-ins
2. Provides well-organized and efficient structuring for complex projects
3. Well-documented

Visual Studio offers add-ins for different versions of Excel. We will develop a solution for Excel 2016 and Excel 2013, where the functionality that can be shared among different versions is programmed in a reusable way, so it can easily to extended to work for other Excel versions.

Data sent to and received from PerfectXL should not be intercepted by third parties and if it is intercepted, that party must not be able to read the data. To ensure the safety of the exchanged messages, we will use a HTTPS-connection that will encrypt the data and send it to PerfectXL in a secure way.

Login information should be stored on the user's computer in a safe way. The user should enter his credentials only during the first use and should merely confirm his username and password during every subsequent session. Therefore, the login information will be encrypted and the encryption key will be divided in parts. Every part will be stored on a different location.

After sending an Excel document, the add-in will check in regular intervals whether the analysis is done. When it's finished, the results will be retrieved with HttpClient.

Visualising the data received from PerfectXL is as follows:

- PDF files will be stored on the user's computer and opened with a PDF-reader
- Image files will be opened within the Excel interface (inline or popup)
- Excel files will be stored as a separate document

Installing the application will be done with Windows installer, as it allows users to select custom options during the installation process and this would allow companies that use PerfectXL to customize the add-in in a way that is optimal for them.

3 Use cases

To facilitate in the development of the add-in, this section will describe the add-in's use cases. Use cases are written descriptions of how users will perform on the final product. They are written from the user's point of view and also describe how the system will respond to the user's actions.

The final product, after installation, will be limited to the actions described in the use cases, with the exception of the *Help* functionality, which shows the user a brief description of the available functionalities.

The add-in will differentiate between two types of users: free users and users with a PerfectXL subscription. Functionalities which can only be used by paying users are marked with *PRO function*.

USE CASE 1	**Login**
Summary	The user clicks the *Login* button in order to use the add-in
Actors	Ribbon user
Precondition	The user is not logged in and all add-in functionality, except for the *Login* button and the *Help* button, are locked
Postcondition	The user is logged in and all add-in functionality is unlocked
Flow of activities	1. The user clicks the *Login* button 2. If the user's credentials aren't saved, the user is presented with an empty form 3. The user enters a correct email and password combination into the form 4. The user clicks *Submit* on the form 5. The form closes 6. The add-in sends the credentials to PerfectXL 7. PerfectXL sends back a token 8. The user's credentials are saved on his computer 9. All add-in functionalities are unlocked
Alternatives	2.1 If the user's credentials are saved on his computer, the user is presented with a filled in form containing the saved credentials 3.2.1 The user does not enter an email or a password 3.2.2 The user enters an incorrect email and password combination 7.2 PerfectXL sends back an error 8.2 The user's credentials aren't saved 9.2 The user is presented with a form to login again 4.3 The user closes the form without logging in 6.3 The user's credentials aren't saved

Table 3.1: Use case 1

USE CASE 2	**Logout**
Summary	The user clicks the *Logout* button to delete his credentials
Actors	Ribbon user
Precondition	The user is logged in, his credentials are saved on his computer, and all add-in functionality is unlocked
Postcondition	The user's credentials are deleted from his computer and all add-in functionality, except for the *Login* button and the *Help* button, is locked.
Flow of activities	1. The user clicks the *Logout* button 2. The add-in erases the user's credentials from his computer 3. All add-in functionality is locked

Table 3.2: Use case 2

USE CASE 3	**Visualize spreadsheet**
Summary	The user clicks the *Visualize spreadsheet* button to get visual feedback from PerfectXL
Actors	Ribbon user
Precondition	The user is logged in
Flow of activities	1. The user clicks the *Visualize spreadsheet* button 2. All add-in functionalities are locked 3. The add-in sends the current workbook to PerfectXL 4. The add-in waits until PerfectXL analyzes the workbook 5. The add-in sends a request to PerfectXL 6. PerfectXL sends back an image 7. The image is saved on the user's computer 8. The image is displayed in a separate window in Excel 9. All add-in functionalities are unlocked

Table 3.3: Use case 3

USE CASE 4	**Generate formula overview**
Summary	The user clicks the *Generate formula overview* button to get feedback from PerfectXL
Actors	Ribbon user
Precondition	The user is logged in
Flow of activities	1. The user clicks the *Generate formula overview* button 2. All add-in functionalities are locked 3. The add-in sends the current workbook to PerfectXL 4. The add-in waits until PerfectXL analyzes the workbook 5. The user is presented with a save prompt 6. The user chooses a location on his computer 7. The user clicks *Save* on the save prompt 8. The save prompt closes 9. The add-in sends a request to PerfectXL 10. PerfectXL sends back an Excel workbook 11. The workbook is saved on the user's computer on the selected location 12. The workbook is displayed in a separate window in Excel 13. All add-in functionalities are unlocked
Alternatives	7.1 The user closes the save prompt 9.1 All add-in functionalities are unlocked

Table 3.4: Use case 4

USE CASE 5	**Analyze spreadsheet (PRO function)**
Summary	The user clicks the *Analyze spreadsheet* button to get feedback from PerfectXL
Actors	Ribbon user
Precondition	The user is logged in The user has a PerfectXL subscription
Flow of activities	1. The user clicks the *Analyze spreadsheet* button 2. All add-in functionalities are locked 3. The add-in sends the current workbook to PerfectXL 4. The add-in waits until PerfectXL analyzes the workbook 5. The add-in sends a request to PerfectXL 6. PerfectXL sends back a web page 7. The web page is displayed in a separate browser window 8. All add-in functionalities are unlocked
Alternatives	3.1 If the user is a free user, the add-in will display a prompt urging the user to purchase a PerfectXL subscription 4.1 All add-in functionalities are unlocked

Table 3.5: Use case 5

USE CASE 6	**Generate risk overview (PRO function)**
Summary	The user clicks the *Generate risk overview* button to get feedback from PerfectXL
Actors	Ribbon user
Precondition	The user is logged in The user has a PerfectXL subscription
Flow of activities	1. The user clicks the *Generate risk overview* button 2. All add-in functionalities are locked 3. The add-in sends the current workbook to PerfectXL 4. The add-in waits until PerfectXL analyzes the workbook 5. The user is presented with a save prompt 6. The user chooses a location on his computer 7. The user clicks *Save* on the save prompt 8. The save prompt closes 9. The add-in sends a request to PerfectXL 10. PerfectXL sends back an Excel workbook 11. The workbook is saved on the user's computer on the selected location 12. The workbook is displayed in a separate window in Excel 13. All add-in functionalities are unlocked
Alternatives	3.1 If the user is a free user, the add-in will display a prompt urging the user to purchase a PerfectXL subscription 4.1 All add-in functionalities are unlocked 7.2 The user closes the save prompt 9.2 All add-in functionalities are unlocked

Table 3.6: Use case 6

USE CASE 7	**Generate overview cell formats (PRO function)**
Summary	The user clicks the *Generate overview cell formats* button to get feedback from PerfectXL
Actors	Ribbon user
Precondition	The user is logged in The user has a PerfectXL subscription
Flow of activities	1. The user clicks the *Generate overview cell formats* button 2. All add-in functionalities are locked 3. The add-in sends the current workbook to PerfectXL 4. The add-in waits until PerfectXL analyzes the workbook 5. The user is presented with a save prompt 6. The user chooses a location on his computer 7. The user clicks *Save* on the save prompt 8. The save prompt closes 9. The add-in sends a request to PerfectXL 10. PerfectXL sends back an Excel workbook 11. The workbook is saved on the user's computer on the selected location 12. The workbook is displayed in a separate window in Excel 13. All add-in functionalities are unlocked
Alternatives	3.1 If the user is a free user, the add-in will display a prompt urging the user to purchase a PerfectXL subscription 4.1 All add-in functionalities are unlocked 7.2 The user closes the save prompt 9.2 All add-in functionalities are unlocked

Table 3.7: Use case 7

USE CASE 8	**Generate PDF (PRO function)**
Summary	The user clicks the *Generate PDF* button to get feedback from PerfectXL
Actors	Ribbon user
Precondition	The user is logged in The user has a PerfectXL subscription
Flow of activities	1. The user clicks the *Generate PDF* button 2. All add-in functionalities are locked 3. The add-in sends the current workbook to PerfectXL 4. The add-in waits until PerfectXL analyzes the workbook 5. The user is presented with a form which allows the user to choose between two report types 6. The user chooses a report type by clicking on a button 7. The form closes 8. The user is presented with a save prompt 9. The user chooses a location on his computer 10. The user clicks *Save* on the save prompt 11. The save prompt closes 12. The add-in sends a request to PerfectXL 13. PerfectXL sends back a PDF file 14. The PDF file is saved on the user's computer on the selected location 15. The PDF is displayed in a separate window in Excel 16. All add-in functionalities are unlocked
Alternatives	3.1 If the user is a free user, the add-in will display a prompt urging the user to purchase a PerfectXL subscription 4.1 All add-in functionalities are unlocked 6.2 The user closes the form 8.2 All add-in functionalities are unlocked 10.3 The user closes the save prompt 12.3 All add-in functionalities are unlocked

Table 3.8: Use case 8

4 Project plan

To complete the PerfectXL Ribbon project within the given timespan for the Bachelor project, problems should be identified as early as possible. To achieve this goal, modern software development methods and tools must be used. This chapter contains an overview of all project deliverables, the development process, and the tools used in the development process.

Deliverable	**Deadline**
Project plan	29-04-2016
Research report	03-05-2016
Solution architecture	04-05-2016
First SIG upload	27-05-2016
Final report	17-06-2016
Final product	17-06-2016
Final SIG upload	17-06-2016

Table 4.1: Project deliverables

4.1. Development process

A widely used software development method is Agile. Several aspects of the Agile methodology will be used:

- Frequent meetings with the client and coach
- Frequent showcase and delivery of working products

Frequent meeting with the client for product showcases will ensure that the product conforms to the client's needs. The client's input will also help uncover mistakes or design errors in the product.

The Bachelor project takes place over the course of 10 weeks. Using short iterations makes sure the customer's requirements are taken into account. Frequently meeting with the coach improves the planning and helps to keep an overview of what has to be done. Continuous testing reduces the risk of detecting major errors at the end of the project.

4.2. Development Tools

In the software development process it should be possible to undo mistakes made by the developer. Furthermore, the PerfectXL Ribbon project should not be affected by harddisk crashes or other forms of data loss. Therefore, a version control tool should be used. For this project, GitHub will be used. An added benefit of Git is that it can be integrated into the development environment, Visual Studio.

5 Design

In this chapter, the PerfectXL Ribbon design is presented and explained. All functionalities will be discussed and sometimes visualised for clarification. Finally, a graphical overview is given at the end of this chapter.

5.1. Sessions

The PerfectXL Ribbon will make use of sessions. A session is a certain period of time in which the user is able to communicate with PerfectXL through the Ribbon. The add-in will allow functionalities to be used only within the limits of a session.

A new session is started when the user authenticates himself via the login system of the add-in. A session will end when one of the following happens:

1. The user actively logs out.
2. Excel is shut down.
3. The user remains inactive for a certain period of time.

The strict timespan in which the functionalities are available ensures safe use of the add-in, as only the time limits of the session have to be monitored: entering a session (logging in) and leaving a session (logging out).

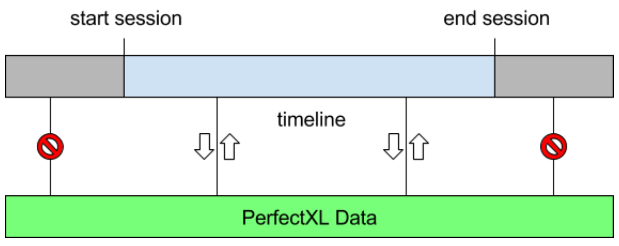


Figure 5.1: Session timespan

5.2. Authentication

Multiple factors should be taken into account when creating a login system:

1. Credentials used to authenticate for the Excel add-in should be equal to the credentials used for PerfectXL
2. Credentials should be stored safely on the local machine and inaccessible outside the application
3. If a machine contains multiple users, every user should not be able to see credentials other than his own

A user should be able to authenticate with PerfectXL credentials in the Excel add-in, therefore the user's credentials must be sent over the web to PerfectXL. This requires a secure connection. Secure connections will be elaborated upon in the Implementation section.

The process of authentication should be easy and safe. To meet the easy criterion, the user should experience as little annoyance as possible in the authentication process. The user experience is improved by storing the user's credentials on the user's machine. If the credentials are not available on the user's machine, the user should enter these manually at the beginning of a session. Saving the credentials increases the convenience of use.

However, the safe criterion should also be met. Credentials should not be accessible for every user on the machine, only for the user(s) that registered at PerfectXL. They should also not be available when the application is not running.

5.3. Ribbon functionalities

The following communication options between the add-in and PerfectXL should be available within a valid session:

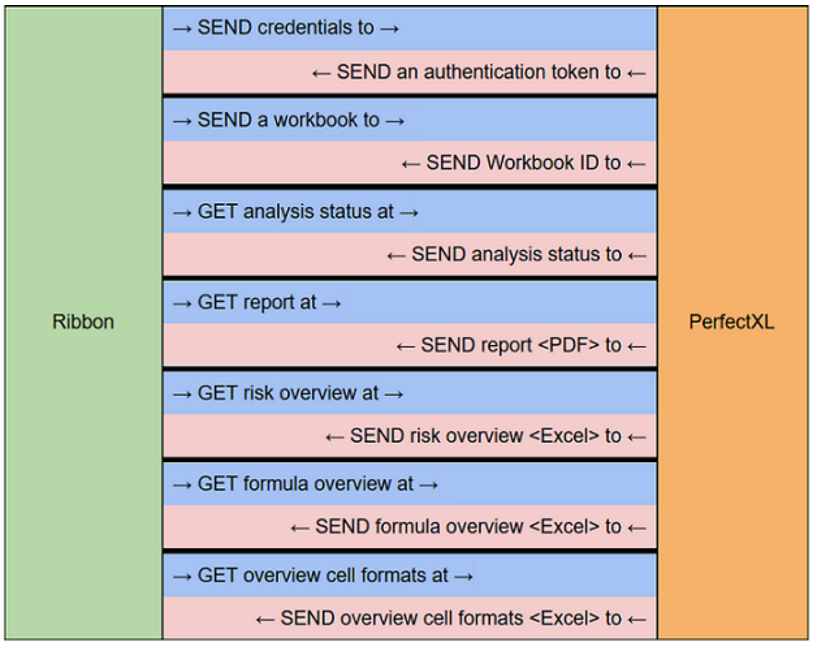

Ribbon
→ SEND credentials to →
← SEND an authentication token to ←
→ SEND a workbook to →
← SEND Workbook ID to ←
→ GET analysis status at →
← SEND analysis status to ←
→ GET report at →
← SEND report <PDF> to ←
→ GET risk overview at →
← SEND risk overview <Excel> to ←
→ GET formula overview at →
← SEND formula overview <Excel> to ←
→ GET overview cell formats at →
← SEND overview cell formats <Excel> to ←
PerfectXL

Figure 5.2: API calls and responses

Every button functionality of the Ribbon that sends API calls to PerfectXL will follow a standard procedure, as illustrated in Figure 5.3:

- The add-in will verify that there is an available internet connection. If there is no internet connection, the add-in is not able to start.
- Secondly, there will be a check for the validity of the session. If there is no valid session, the user should authenticate to start a new session.
- Thirdly, the current workbook is sent to PerfectXL. If uploading the workbook fails, the process will exit and pass the reason of the upload failure to the user.
- Lastly, the workbook will be analyzed by PerfectXL, which can take up to 10 minutes. The add-in should actively wait for the workbook to be ready by requesting the workbook status. If the workbook is not analyzed, the add-in should time out for a certain amount of time and again send a request, until the workbook has been analyzed. If the workbook is analyzed, the procedure is finished. If the analysis fails, the process will exit and inform the user of the failure.

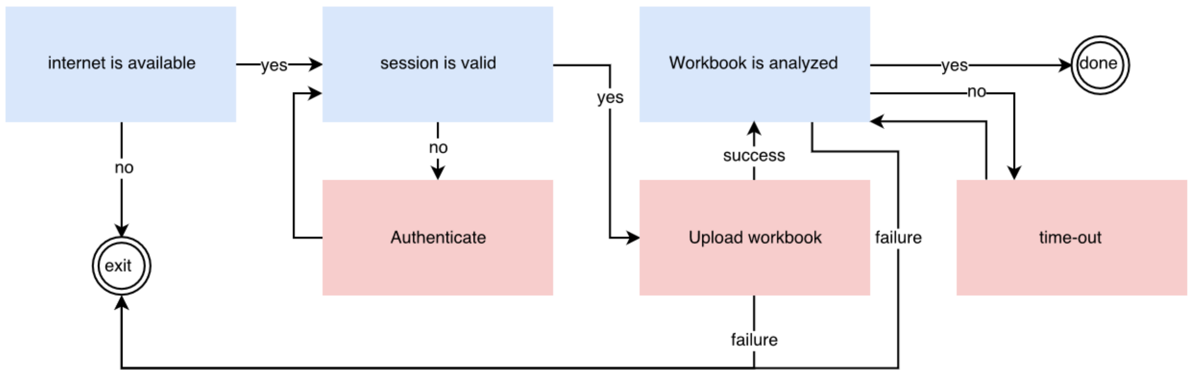


Figure 5.3: Flowchart of authentication, uploading, and analysis

This process is shared by all functionalities that require a connection with PerfectXL. After going through the authentication and uploading process, the functionalities will go through their respective unique functions. The add-in will have the following unique functionalities of PerfectXL:

- Generate formula overview
- Analyze workbook
- Generate risk overview
- Generate overview cell formats
- Generate PDF report

5.3.1. Generate formula overview

The web application PerfectXL has an option to generate a list of all unique formulas; this functionality returns an Excel workbook containing the original workbook, with an added tab containing all unique formulas in the original workbook. In the Ribbon, the button *Generate formula overview* prompts the add-in to retrieve the result and open it in a new Excel window.

5.3.2. Analyze workbook

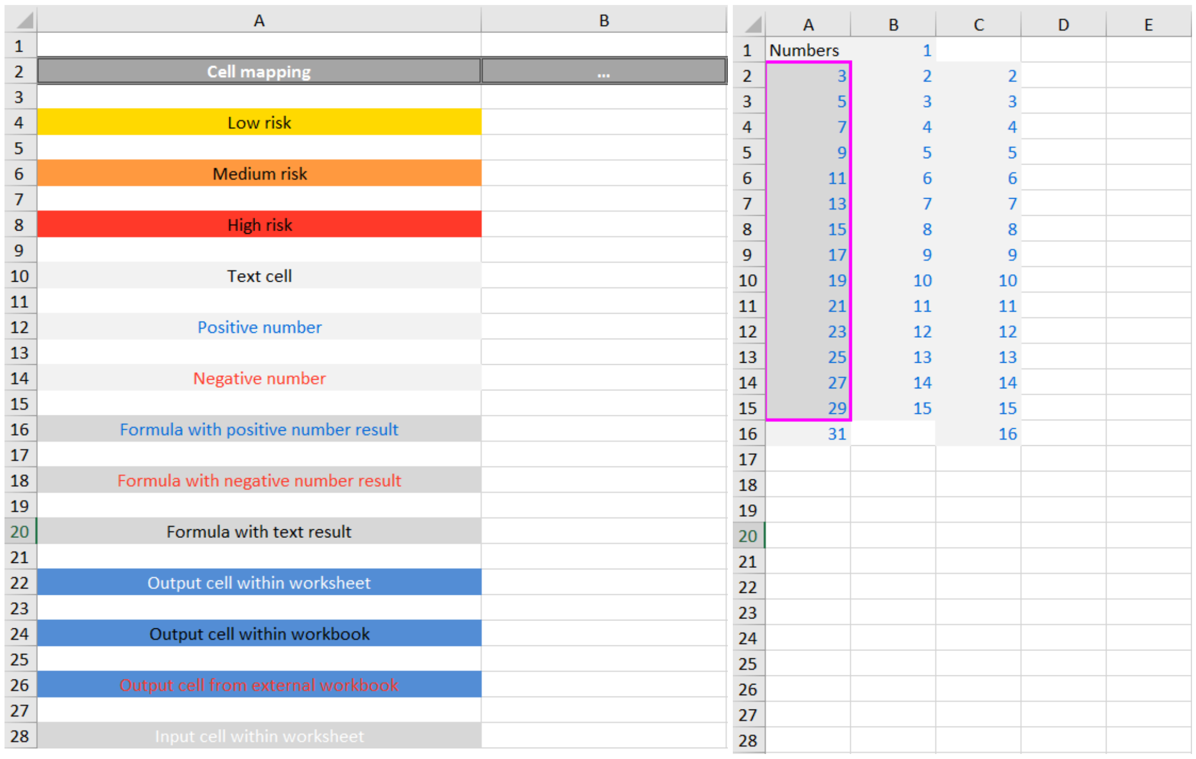
After authentication, uploading, and analyzing the current workbook, the analysis is saved in the database of PerfectXL and can be viewed in the browser. The web environment contains aspects of this analysis that cannot be sent to the Excel add-in because it is presented in HTML. This can be solved by creating a portal to the web application within Excel by using an embedded browser. The button *Analyze workbook* prompts the add-in to set up an interface where the aspects of the analysis written in HTML can be viewed.

5.3.3. Generate risk overview

The risk overview is returned in the same format as the formula overview: in an Excel workbook containing an overview of all risks in the current workbook. The button *Generate risk overview* prompts the add-in to retrieve the result and present it in a new Excel window.

5.3.4. Generate overview cell formats

The overview of cell formats is presented in the form of an Excel workbook. Cells that match with a certain criterion are marked with the same color as presented in the example presented in Figure 5.4.



	A	B
1		
2	Cell mapping	...
3		
4	Low risk	
5		
6	Medium risk	
7		
8	High risk	
9		
10	Text cell	
11		
12	Positive number	
13		
14	Negative number	
15		
16	Formula with positive number result	
17		
18	Formula with negative number result	
19		
20	Formula with text result	
21		
22	Output cell within worksheet	
23		
24	Output cell within workbook	
25		
26	Output cell from external workbook	
27		
28	Input cell within worksheet	

	A	B	C	D	E
1	Numbers	1			
2	3	2	2		
3	5	3	3		
4	7	4	4		
5	9	5	5		
6	11	6	6		
7	13	7	7		
8	15	8	8		
9	17	9	9		
10	19	10	10		
11	21	11	11		
12	23	12	12		
13	25	13	13		
14	27	14	14		
15	29	15	15		
16	31		16		
17					
18					
19					
20					
21					
22					
23					
24					
25					
26					
27					
28					

Figure 5.4: Cell formats legenda and an example of analyzed cells

5.3.5. Generate PDF report

PerfectXL contains two kinds of PDF reports:

1. Summary report
2. Detailed report

The button Generate PDF prompts the add-in to download the PDF report and open it in a PDF reader.

5.3.6. Overview

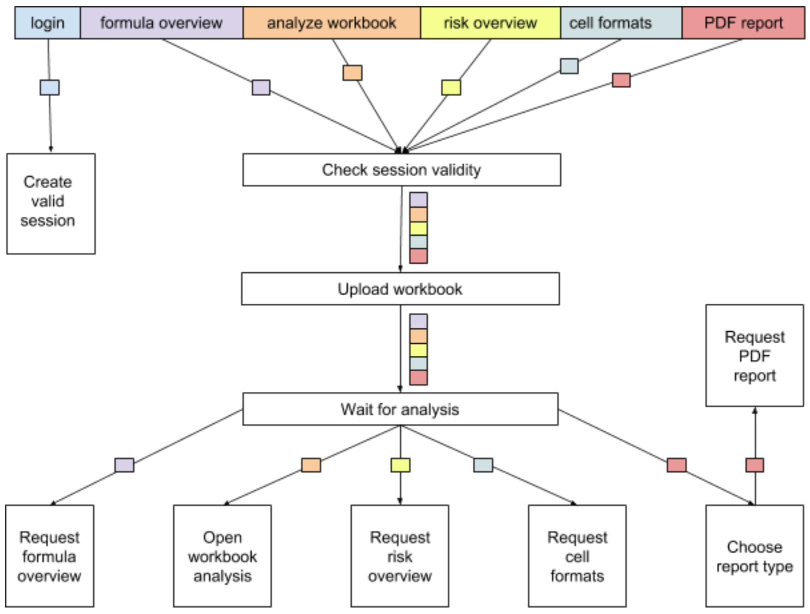


Figure 5.5: Overview of Ribbon functionalities

5.4. Presentation

An important aspect of the final product is the visual presentation and the presentation of the content received from PerfectXL, as the user experience largely depends on it. To clarify this statement, here are possible examples of (visual) requirements:

- The buttons of the Ribbon should be self-explanatory. This can be achieved by, for example, using images and labels.
- Whenever the add-in is sending or loading data, the Ribbon should indicate the presence of background activity.

Furthermore, the data which is retrieved by the add-in should be presented in an understandable way, e.g.:

- Files downloaded from PerfectXL should be stored on a location chosen by the user and opened when the user explicitly instructs the Excel add-in to open these files.
- Spreadsheets containing analysis results should be opened in a new window, to avoid corruption of the original spreadsheet.

6 Implementation

The implementation chapter explains in detail how the Ribbon design is implemented in Visual Studio. Three components of the PerfectXL Ribbon will be discussed and we will elaborate on the login system and upload functionality.

6.1. Code structure

As discussed in the research report, the PerfectXL Ribbon is developed in the Visual Studio environment, where the solution is divided in four different projects:

1. Presentation of the add-in (Ribbon)
2. Functionality of the add-in
3. Unit tests
4. Installation files

The unit tests (3) will be elaborated upon in the next chapter. This section will discuss the Ribbon (1) and the functionality of the add-in (2). The relationship between the Ribbon and its functionality is illustrated in Figure 6.1.

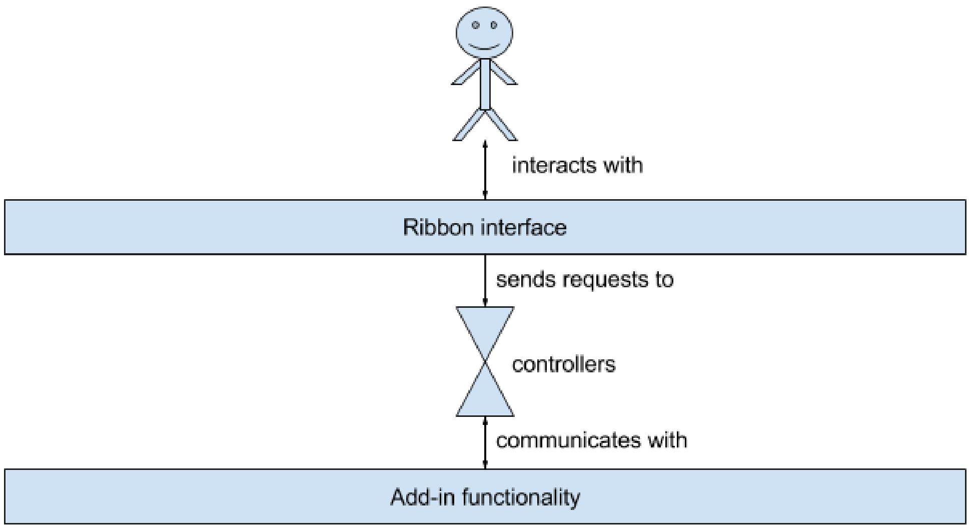


Figure 6.1: Interactions between components of the Ribbon and the user

6.2. Presentation of the add-in

The Ribbon is the part of the add-in a user can interact with. Part of the Ribbon is the buttons. The user is able to click the buttons to make the add-in perform certain actions. The buttons are self-explanatory. By adding a PDF picture to the *generate PDF* button, it is clear to the user that clicking this button will generate a PDF file. When the notification button lights up, the user knows it is possible to trigger a subsequent action by clicking the notifications button.

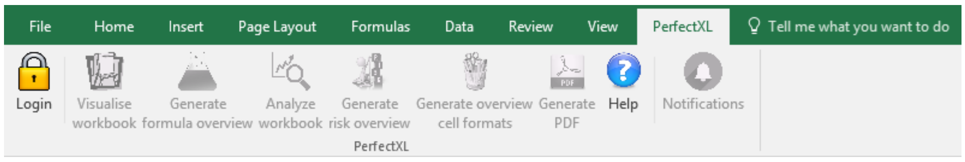


Figure 6.2: The Ribbon before logging in

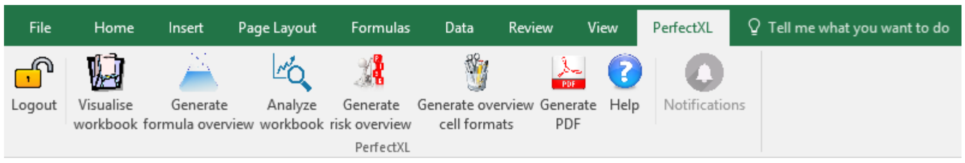


Figure 6.3: The Ribbon after logging in

The main goal of the Ribbon is to provide a layer that translates the wishes of the user (e.g.: I want an Excel file containing a risk overview of my spreadsheet) into commands for the add-in functionalities (validate the current session, upload the active workbook, wait for the PerfectXL analysis, request an Excel file with a risk overview of the uploaded workbook, and present it to the user). Therefore, the interface cannot communicate directly with PerfectXL. For this purpose, a layer is built in that translates the aforementioned wishes of the user into PerfectXL commands.

The Visual Studio project that provides the user interface is slightly dependent on the Visual Studio project containing the actual functionality. Some elements of the Ribbon interface can be reused for every new add-in for Excel and therefore we decided to go for convenience and re-usability instead of a perfect separation of the interface and the underlying functionality. By moving as much visual functions (e.g. Windows Forms) to the Visual Studio project with the actual functionality, making the add-in compatible with never versions of Excel is easier. More can be read about this topic in Chapter 2.

6.3. Functionality of the add-in

6.3.1. Controllers

All requests from the Ribbon pass through the 'controller' classes. The controllers are a medium between the Ribbon and the classes that contain the actual functionality of the add-in. For example, if the user presses a button, the controller will receive a request from the Ribbon interface, the controller will check the preconditions for a PerfectXL request, and pass it to the corresponding class that will handle the request.

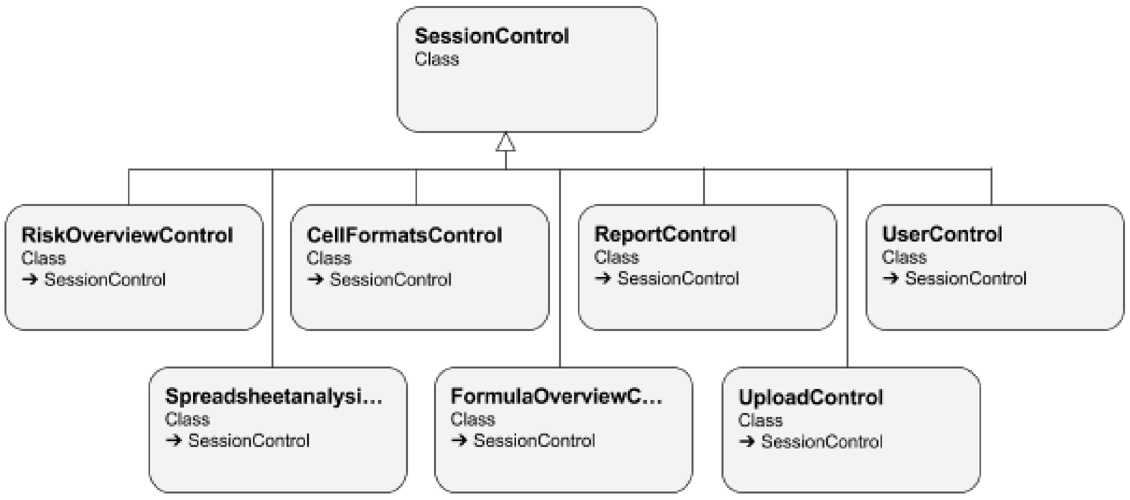


Figure 6.4: The controller classes

Most buttons in the Ribbon have a corresponding controller class that handles the unique functionality of that button. The following controller classes are available:

1. SessionControl
2. UserControl
3. UploadControl
4. FormulaOverviewControl (Generate formula overview)
5. SpreadsheetAnalysisControl (Open spreadsheet analysis)
6. RiskOverviewControl (Generate risk overview)
7. MappingsControl (Generate cell formats overview)
8. ReportControl (Generate PDF report)

SessionControl.cs contains variables that can be shared among all controllers, for example the ID of the current workbook, the time to wait for web requests, whether the workbook has been altered, and locations of documents stored in temporary files.

6.3.2. Forms

The add-in contains the following forms:

- LoginForm
- SaveForm
- PDFForm

LoginForm is used in the authentication process. It allows the user to enter an email and password combination. It is used in the Login class, as seen in Figure 6.6.

SaveForm is used in the downloading process. It allows the user to select a location on the disk where the analysis result will be saved. It is used in the FormulaOverviewControl, RiskOverviewControl, MappingsControl, and ReportControl classes.

PDFForm is used in the downloading process of the Generate PDF report functionality. It allows the user to select a report type, as described in Section 5.3.5. It is used in the ReportControl class.

6.3.3. Background processes

To improve the user experience, processes that should not require explicit actions of the user, such as waiting for the workbook analysis to complete, run in the background. An advantage of using background processes, is that the user is still able to use the Excel workbook while another process runs in the background.

Visual Studio offers a component that handles background processes, called the BackgroundWorker. The BackgroundWorker operates on a separate thread. Communication with the Ribbon is therefore asynchronous. Whenever the BackgroundWorker is finished with a given task, the background thread invokes the main thread and highlights the notification button, which means a user action is required.

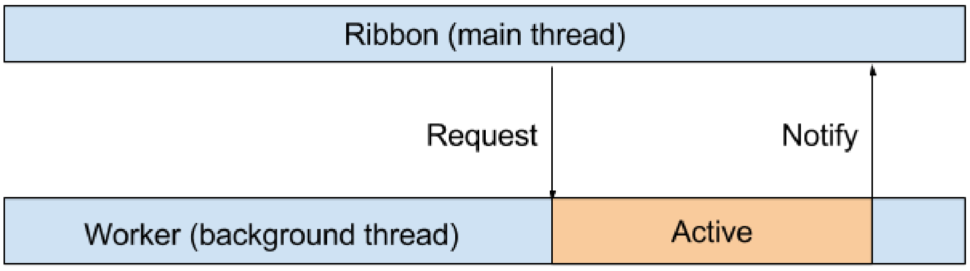


Figure 6.5: The BackgroundWorker

The BackgroundWorker cannot start a new thread. As a result, the user can only execute one Ribbon command at a time. This is not just a limitation, but a deliberate design choice. The following facts were taken into account:

- The PerfectXL web application can only analyze one workbook at a time
- Running multiple processes at the same time might result in data and processes being associated with the wrong workbook.

6.3.4. Login system

This section elaborates on the *authenticate* process, as seen in Figure 5.3. The Ribbon first makes a call to UserControl, where the authentication process takes place as illustrated in Figure 6.6.

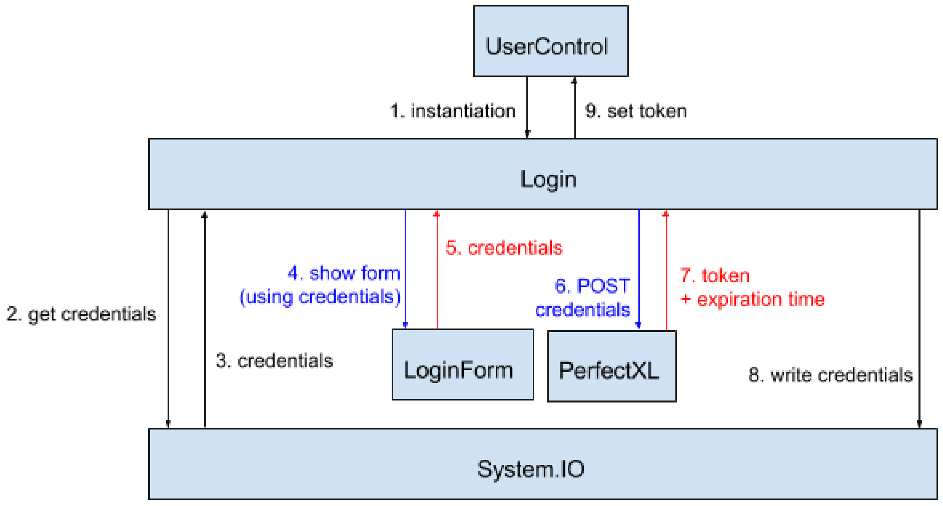


Figure 6.6: Steps in authentication

Steps 2. and 3. read from a file containing the user's saved credentials. If the file doesn't exist, the credentials default to empty strings. In step 4. these credentials are displayed on a LoginForm, which allows the user to confirm the credentials or enter new ones. In step 6. the credentials are sent in an API call to PerfectXL. If the credentials are correct, PerfectXL sends back a token and the remaining time for which the token is valid in step 7. If the credentials are validated by PerfectXL, they are saved in a file in step 8. If they are not validated, a new LoginForm is shown. In step 9. the response from PerfectXL, containing the token and expiration time, is given to UserControl, where the token is stored. This token is used in future API calls.

In step 8. the credentials are encrypted with a hardcoded string before being written to a JSON file. In step 3. the credentials are decrypted using the same string.

6.3.5. Uploading workbooks

The uploading process starts when the UploadControl class is called from the Ribbon. The UploadControl class then calls the Upload class, which contains the actual functionality.

Before the process starts, it first checks whether changes have been made in the opened workbook since the last upload. The purpose of this is to avoid redundant uploads and shorten the time between the click of a button and the presentation of the analysis result.

The SessionControl class keeps track of these changes in a single variable. It contains methods that change the variable to signify that a new workbook must be uploaded. These methods are attached to events that fire when the workbook gets altered. If no changes have been made, the analysis result can be fetched right away. However, if changes have been made, a new workbook must first be uploaded.

Before uploading, a copy of the opened workbook is saved in the user's temporary folder. This is a necessary step, because opened workbooks that have not yet been saved cannot be uploaded. The file to uploaded must exist on the user's disk. This copy is then sent in a POST request to PerfectXL. The response contains an ID which is associated with the uploaded workbook. This initiates the await analysis process.

6.3.6. Await analysis

The analysis is done by the PerfectXL web application. This can take from a few seconds up to 10 minutes. The await analysis process actively checks whether the analysis of the workbook is done by making API calls to PerfectXL every 3 seconds. It stops checking after 10 minutes and automatically assumes failure.

When the response to the API call indicates the workbook is done, the await analysis process exits to the Ribbon interface which starts the specific functionality associated with the button that was clicked.

6.4. Installation files

In order to deploy and install the Excel add-in, an installer is required. The PerfectXL Ribbon installer project requires the Microsoft Visual Studio 2015 Installer Projects extension. This extension adds Visual Studio Installer projects to Visual Studio 2015, which enable programmers to create Windows Installers.

6.4.1. Project output

In order to deploy the add-in, the project output must be added to the installer. The Microsoft Visual Studio 2015 Installer Projects extension allows the programmer to create an installer project using a Setup Wizard, which has the option to add project output from the PerfectXL Ribbon project. Any dependencies in the project are automatically detected and added.

6.4.2. Project files

Some files must be added separately. The installer project requires the VSTO deployment manifest and application manifest, which are added separately. These files are located in the PerfectXL Ribbon project bin/Debug folder.

The embedded browser, as mentioned in Section 5.3.2, requires xulrunner 33.0.1, which is a Mozilla runtime package containing the browser. This package must be downloaded separately and put in the File System of the installer project.

6.4.3. Prerequisites

The installer project for the PerfectXL Ribbon has the following prerequisites:

- Microsoft .NET Framework 4.5.2
- Microsoft Visual Studio 2010 Tools for Office Runtime
- Windows Installer 4.5

Prerequisites are software components that must be installed on the user's computer before the PerfectXL Ribbon can be installed and used. The installer must be shipped with the prerequisites' installation files. If the prerequisites are not installed on the user's computer, those installation files will be used to install the prerequisites before installing the PerfectXL Ribbon.

6.4.4. Registry keys

In order for the add-in to be detected by Microsoft Excel, the registry keys must be configured by adding the PerfectXL.RibbonAddIn key to the registry. The key contains the location of the add-in. After installation, the PerfectXL.RibbonAddIn key is added to the Microsoft Addins folder on the user's computer, and the add-in shows up in the Excel Ribbon.

6.4.5. Installation folder

By default, Microsoft Office add-ins are intended to be installed for a single user. So the installer does not have the option to install for all users on the computer. The default folder for the add-in will be the user's Application folder. The added benefit of this location is that it does not require administrative privileges for the installer.

7 Testing

In order to ensure product quality, the code must be evaluated by the client and tested. Biweekly meetings and product showcases with the client allow the client to evaluate the product's functionality. The client also has access to the code on GitHub.

Because of the novelty of this product, there was no set structure at the beginning of the project. Creating tests for unknown code structure was not possible. Therefore, the testing code was written later, when some functionality was already developed. The tests served to validate, rather than dictate the code.

Most of the testing is done through use case testing, after every change in the code. To test the internal functionality of the add-in, unit tests will be used. Every class will be tested separately.

7.1. Testing request-response

The Ribbon's sole purpose is to facilitate the use of PerfectXL. It passes user input to PerfectXL and displays the results. Therefore, it is based on API calls and responses.

As described in Section 6.1, most of the code is in the add-in functionality, which is divided into authentication and functionalities. Authentication enables the user to log into PerfectXL, while functionalities enable the Ribbon to send and receive files. Both of these parts perform API calls, using HttpClient.

7.1.1. Testing authentication

Authentication deals with sensitive information: the login credentials of the user. The Login class sends these credentials to PerfectXL and receives the access token in return. If the credentials are incorrect, the response will not contain an access token.

Testing the code with a default HttpClient would require correct credentials to be stored in the code for testing, which poses a security threat. Or it would require the tester to enter credentials during testing. Based on the scenario to be tested, these credentials would have to be correct or incorrect, which gives less control over the way the test is executed. In order to test this code, there must be a way to control the response that is received after sending the credentials.

HttpClient has two types of constructors: an empty constructor, and one that takes a HttpMessageHandler. In order to control the response and allow for easier testing, the Login class will take a HttpClient in its constructor.

Classes that create a Login instance during runtime pass a default HttpClient instance to the Login constructor. However, during testing, a new HttpClient instance is created with a custom HttpMessageHandler.

There are three classes that inherit HttpMessageHandler: LoginSuccessTestHandler, LoginFailTestHandler, and LoginFailAndSuccessTestHandler. They send a response that emulates a successful login attempt, a failed login attempt, and a failed login attempt followed by a successful login attempt, respectively. This allows the tests to simulate three different login scenarios.

7.1.2. Testing data transfer

The functionalities deal with the uploading of Excel workbooks, and retrieve Excel workbooks and PDF files once the analysis from PerfectXL is completed. If the upload or workbook analysis fails, no files will be retrieved and the user will be able to initiate the upload again. Before uploading a workbook, the user must log in with correct credentials.

Unlike authentication, functionality classes are tested with a default HttpClient. Functionality classes use long URLs, MIME (Multipurpose Internet Mail Extensions) requests, file writers, and file readers. Therefore, it is more useful to test those classes with real responses from PerfectXL.

Functionality classes require access tokens in order to upload and download files. Therefore, the tester must log in once with correct credentials before the tests for these classes are run. This is done through test helpers. These helpers create a new Login instance and present the tester with a login form.

In order to retrieve files from PerfectXL, a new workbook must first be uploaded. The test helpers create an empty Excel workbook instance, which is then uploaded using the Upload class.

7.2. Testing user interface

The Excel add-in relies heavily on the user interface, the Ribbon. Every action is prompted by the user and the user must give intermediate input, e.g. where to save the results from PerfectXL. During automated unit tests, these user interface elements form an obstacle, as they require user interaction before continuing. This section will discuss how unit tests of the add-in functionality circumvent the user interface.

7.2.1. Avoiding message boxes

When errors occur during runtime, they are handled by the Ribbon, and the user is informed of the exact nature of the error through message boxes. During testing, however, when errors are created intentionally for the purpose of testing, message boxes pose as nuisance. They interrupt the testing process until they are closed manually. To circumvent this issue, all message boxes are replaced by DialogService. The DialogService class is a class created to enable automated testing. The DialogService class inherits from IDialogService, and shows the error on a message box. Every class that uses message boxes takes in its constructor an object that inherits IDialogService. During runtime, DialogService instances are used.

During testing, a DialogServiceTest instance, which also inherits from IDialogService, is used as an argument in the constructor of the class to be tested. The DialogServiceTest instance does not show a message box, but instead writes to the debug console.

7.2.2. Avoiding opening files

After retrieving the results of the PerfectXL analysis, the results are saved as files and automatically opened. Like the message boxes, this poses as nuisance during testing when the tested code opens a file after every iteration.

Similarly to message boxes, this issue is circumvented by using IStartService, StartService, and StartServiceTest. Every class that opens files takes in its constructor an instance that inherits from IStartService. StartService and StartServiceTest both inherit IStartService. StartService is used during runtime to open files. StartServiceTest is used during testing and simply writes to the debug console.

7.2.3. Testing Windows forms

As described in Section 6.3.2, the Ribbon displays multiple forms to the user, which allow the user to enter information and state their preferences.

The most common approach is to design the code in an MVC (Model–view–controller) pattern [10], which separates the part that is presented to the user from the part that contains the actual logic. The form represents the view, which merely shows information generated by the controller to the user and sends user input to the controller. It doesn't do anything else with the information, it passes it from the controller to the user and vice versa.

Furthermore, forms have a limited amount of actions the user can perform on them. The user can either click on a button or close the form. Thus, forms are a controlled environment for which the responses are predictable and all responses can be taken into account in the controller. Thus, forms will not be subjected to unit tests.

To test methods that use forms, a mocked form is used with predetermined responses that play out a test scenario. The responses from forms are limited, therefore it is easy to test different scenarios in which the forms may be used.

7.3. Testing Control classes

As discussed in Section 6.3.1, the code has Control classes. These control classes pass user input to classes which communicate with PerfectXL. The classes themselves contain no actual functionality. Similarly to Windows Forms, testing Control classes will not help improve the code. Therefore, Control classes are not subjected to unit tests.

7.4. Performance testing

The PerfectXL Ribbon will mainly be used to upload workbooks. The complexity of these workbooks can vary from one simple worksheet to several large worksheets with complex formulas.

The Ribbon's main design goal is user friendliness. Uploading large and complex workbooks must be possible, and it must be finished within a reasonable timespan.

When considering the performance of the Ribbon there are two distinct time elements that must be taken into account:

- The time required for uploading files
- The time required for PerfectXL to analyze the file

The waiting time for analysis includes the polling time of the Ribbon, which contributes no more than 3 seconds to the total waiting time for the analysis, as explained in Section 6.3.6. The total waiting time therefore, from the perspective of the user, is the upload time plus the analysis time. Between the upload time and the analysis time, only the upload time is an indicator of Ribbon performance, but the analysis time is not.

The upload and analysis time was tested with workbooks provided by Infotron, ranging from 10 KB to 30 MB. Workbooks of 0.01 MB, 4 MB, 5 MB, 9 MB, 11 MB, 18 MB, and 32 MB were uploaded five times each through the PerfectXL Ribbon, and the averages of these times were noted.

The average waiting times for 11 MB, 18 MB, 32 MB are shown in Figure 7.1

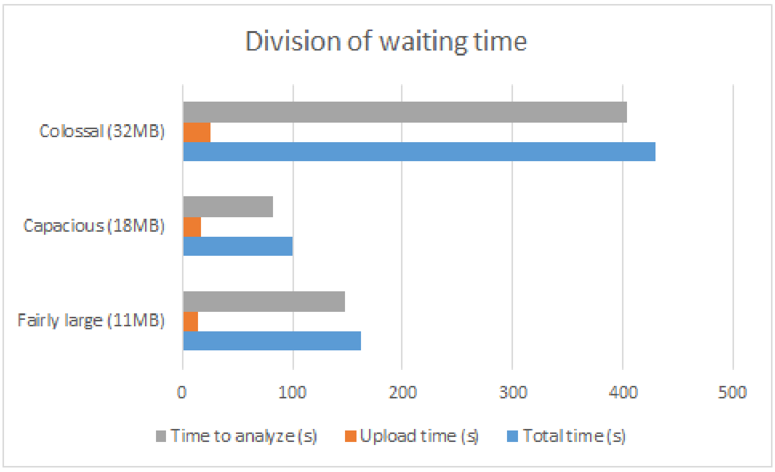


Figure 7.1: The average waiting time for the three largest files

In Figure 7.1, we observe that the upload time increases with increasing file size, which is expected. The analysis time, however, does not increase with increasing file size, as the dip for the 18 MB file shows. This may be due to the fact that the analysis time possibly depends on complexity rather than on the file size. However, confirmation of that is not within the scope of this report, as it is independent of the PerfectXL Ribbon project.

The following graph shows the waiting time for the analysis in more detail:

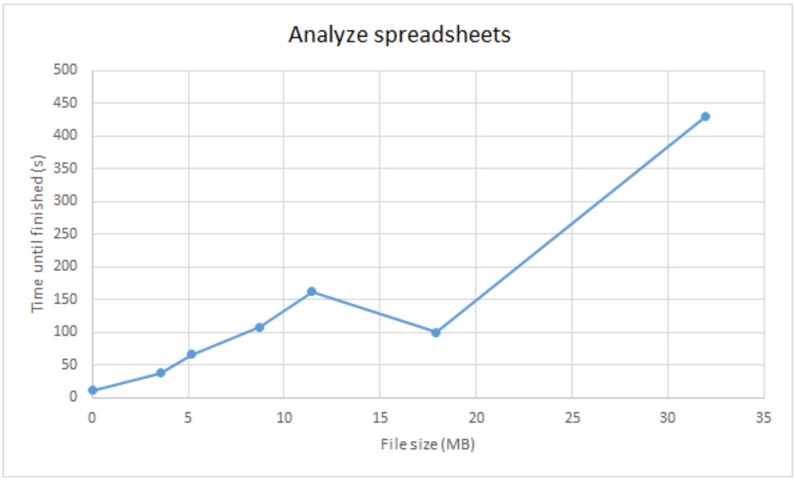


Figure 7.2: The average analysis time for increasing file sizes

The waiting time for the analysis increases with increasing size, except for the 18 MB file, where there is a dip, as discussed above.

Overall, we can conclude that the Ribbon is performing within acceptable limits, as only a fraction of the waiting time consists of the upload time. This is even demonstrated in the case of the 32 MB file, the largest file size tested. Thus, the principal bottleneck in the overall performance is due to the time required on the PerfectXL web application for analysis of the workbooks.

8 Discussion

This section will discuss the limitations of the final product and possible improvements that were not implemented due to the limited time allocated for the Bachelor project.

8.1. Limitations

While the application works in normal situations, it has demonstrable limitations. An example of such a limitation is the use of the PerfectXL Ribbon with multiple Excel windows. The add-in is running a synchronous session for all open workbooks. When an action is performed in one workbook, all the other workbooks are affected by this action, e.g. when a user logs in in one Excel windows, he is automatically authenticated in all open workbooks. This might be convenient in some cases, but there are certain limitations and inconveniences, e.g. when a user wants to run multiple analyses asynchronously in different workbooks.

In addition, Excel workbooks are often in protected view mode, which means the Excel file is read-only and its resources are not readily available for the Excel Ribbon. The user must disable the protected view mode manually before uploading the workbook. A possible solution is to prompt the user to disable protected view mode.

Another limitation worth mentioning is the time it takes to upload and analyze the workbooks. The time taken to analyze workbooks may be a few seconds, up to 10 minutes at most. It is important to take the user experience into consideration, i.e. is the user willing to wait several minutes for the analysis to complete? As pictured in Figure 7.1, the analysis done by the PerfectXL web application causes the biggest delay in the entire process. Therefore, the biggest bottleneck in the process of uploading a workbook and retrieving the results, is the PerfectXL web application.

8.2. Future investigation

Besides the limitations, there is potential to expand the functionality of the PerfectXL Ribbon. There are two ways of expanding the functionality:

- Improving existing functionality
- Implementing new functionalities

Improving existing functionality: A functionality that is worth improving, is the installation module. Infotron would like their customers to be registered before they install the PerfectXL Ribbon. Therefore, an elaborated installation module would be helpful, because it would allow the user to register during the installation process. Such an installation module requires a connection with PerfectXL.

Furthermore, some clients of Infotron run the PerfectXL application in an intranet environment, which causes the web application URL to differ from the standard URL. Therefore, this URL should be dynamically editable, either in the installation module or in the PerfectXL Ribbon. Clients of Infotron either use the intranet application or the web application, which means the URL should not necessarily change later on. Implementing this functionality in the installation module is the more preferable option.

Implementing new functionalities: The PerfectXL Ribbon could contain options that make correcting Excel workbooks easier in real-time. One option is to highlight formulas in Excel spreadsheets. Even though this is not part of the Ribbon, clients of Infotron may like to see this feature implemented.
A new feature that may be of interest is to present analysis results in a user-friendly way, without interfering with the workbook the user is currently working in. Currently, the analysis results are opened in a new Excel window or in a PDF reader. Preferably, the Ribbon should be able to integrate the analysis results into the original workbook or present it in a tab in the same Excel window, instead of presenting them in separate windows, so that the user does not have to close all the windows manually.

9 Conclusion

In the Bachelor project, we developed an Excel Ribbon for the company Infotron, to analyze spreadsheet data from within the Excel environment. This Ribbon communicates with an existing web application, PerfectXL. Excel workbooks are sent to the web application to be analyzed, and a visual representation of this analysis can be downloaded to the user's local machine and presented to the user.

Before attempting the design and implementation, we spent two weeks on the research phase, where we made design choices for the final product. This was followed by design and implementation. Testing was done both during and after the implementation phase.

In the process of building an Excel add-in, we consulted regularly with the client about the functionality of the Ribbon, and the expectations and needs of the user. Most of the initial requirements were implemented into the Ribbon and the client expressed their satisfaction with the final product delivered.

The biggest challenge during the project was the novelty of the product. Many aspects of Excel add-ins were not well documented. Most of the project consisted of finding and combining information from various sources, some of which were outdated, to create an optimal solution.

PerfectXL aims to aid Excel users in visualizing their spreadsheets and recognize problem areas in them. Our PerfectXL Ribbon project helps the users even further by bringing within their reach the PerfectXL functionalities from inside the Excel environment itself. As such, this project is a valuable addition to the services offered by Infotron to their customers. While in all other respect, the Ribbon serves to enhance the user experience of PerfectXL clients, the aspect of waiting time can be further improved if the loss due to the performance bottleneck occurring in the PerfectXL application side could be reduced. We hope this will be done in the future and the PerfectXL Ribbon will then be able to offer optimum user-friendliness and ease of use to Infotron clients.

A
Original project description

A.1. Project description

Infotron, tightly associated with the Delft Technical University, has developed the tool PerfectXL in cooperation with the research group SERG. PerfectXL visualises spreadsheets and analyzes them for risks. Years of effort went into its making, with version 1.0 launching in May of 2016. Big companies like PwC, the Port of Rotterdam, Nationale Nederlanden, and Connexxion are amongst our clients. Each of them use big, complex spreadsheets for critical company decisions. Their subscription with PerfectXL allows them to validate their spreadsheets more easily than ever before.

PerfectXL is a web application. Clients upload their spreadheet and can view analysis results directly in their browser (more on `www.infotron.nl`). PerfectXL has been written in C# and the AngularJS framework.

An interesting question emerged from our clients: could we make it so that PerfectXL can be controlled directly from Excel? Wouldn't that make it even easier to use? We think so. That's why we made it into a 10-week internship project for 2 students. Who can build an Excel Ribbon that will fluently and quickly communicate with the web tool and databases of PerfectXL? The project will encompass a wide variety of topics: from C# to VBA programming, from backend work to client-friendly navigation and design. We ask you to develop the utility from beginning to end. Our clients are ready to test what you create.

We work from both Amsterdam and Delft.

A.2. Company description

`http://www.infotron.nl`

B

Feedback Software Improvement Group

B.1. First feedback

The score from the first round of feedback was four stars out of five for maintainability, which is above average. The score was lower than five starts because the code scored lower for Unit Size and Module Coupling.

Unit Size rates code that is above average in size. The long methods in the embedded browser, as mentioned in Section 5.3.2, could be split into several separate methods. This was solved by splitting up the methods per functionality.

Module Couples rates the code based on how often the same class is called from other classes. The issue was the Session class, which was used instead of the controller classes mentioned in Section 6.3.1. The Session class was called 38 times by different classes. This was solved by splitting up the Session class into the controller classes, per functionality.

The rest of the code was not altered following the SIG feedback.

B.2. Second feedback

The second round of feedback showed that, despite the increase in code size, the score for maintainability had stayed roughly the same.

The score for Unit Size and Module Coupling had increased for both the old and the new code. However, there was no significant increase because we did not do structural refactoring. No further explanation was given for this.

The ratio between new code and new tests increased 1:1, which is viewed as optimal.

From these observations, it was concluded that we properly took into account the feedback from the first round.

C
Infosheet

General information

Project title: Infotron PerfectXL Ribbon
Name of the client organization: Infotron
Date of the final presentation: July 1, 2016

Description of the project

The purpose of this project was to build an Excel Ribbon for a web based spreadsheet analysis and validation service, called PerfectXL. The PerfectXL Ribbon would enable users to upload the opened Excel workbook to the PerfectXL web application and receive the analysis results back from it, from within the Excel environment, thereby removing the need for using a web browser.

The most challenging part of the project was the novelty of the product. Many aspects of Excel add-ins were not well documented. Most of the project consisted of finding and combining information from various sources, some of which were outdated, to create an optimal solution.

During the research phase, we learned much about Visual Studio's tools and features. These tools helped us setup the initial project and create a good basis for the code with little trouble, allowing us to make sure the internal processes of the product work properly.

During the project, parts of the Scrum methodology were used, i.e. frequent meetings with the client and coach, frequent deliveries of working products, and continuous testing. No unexpected challenges arose during the project, and expected challenges were quickly resolved because of the adaptability of the team.

The final product is an Excel Ribbon that can be run on Excel 2013 and Excel 2016. Every new functionality was unit tested and the user interface was tested with use cases.

The final product does not greatly differ from the initial design made by the client, barring minor adjustments to the user interface. It can be used in its current state, but its user interface and installer will most likely undergo some changes before deployment.

Members of the project team

Name: Tim Molenaar (tim.molenaar@kpnmail.nl)
Interests: Web Application Development, Data Visualisation
Contribution and role: Environment Setup, Ribbon Design

Name: Natasa Radenovic (nradenovic93@gmail.com)
Interests: Software Engineering, Algorithm Design, Gaming
Contribution and role: Functionality Developer, LaTeX Master

All team members contributed to the research report, the design, the implementation, the testing, and preparing the final report and the final project presentation.

Client
Name: Mateo Mol
Affiliation: Infotron B.V.

Name: Joeri van Veen
Affiliation: Infotron B.V.

Coach
Name: Sohon Roy
Affiliation: Department of Software and Computer Technology, TU Delft

The final report for this project can be found at: `http://repository.tudelft.nl`.

Bibliography

[1] Microsoft Corporation. Deploying an office solution, . URL https://msdn.microsoft.com/en-us/library/bb386179.aspx.

[2] Microsoft Corporation. How to encrypt and decrypt a file by using visual c#, . URL https://support.microsoft.com/en-us/kb/307010.

[3] Microsoft Corporation. Walkthrough: Creating a local database file, . URL https://msdn.microsoft.com/en-us/library/ms233763(v=VS.110).aspx.

[4] Microsoft Corporation. Creating vba add-ins to extend and automate microsoft office documents, . URL https://msdn.microsoft.com/en-us/library/office/gg597509(v=office.14).aspx.

[5] Microsoft Corporation. Welcome to visual studio, . URL https://msdn.microsoft.com/en-us/library/dd831853.aspx.

[6] Microsoft Corporation. Walkthrough: Creating your first vsto add-in for excel, . URL https://msdn.microsoft.com/en-us/library/cc668205.aspx.

[7] Excel Easy. Ribbon in excel - easy excel tutorial. URL http://www.excel-easy.com/basics/ribbon.html.

[8] Diogo Nunes. Webclient vs httpclient vs httpwebrequest. URL http://www.diogonunes.com/blog/webclient-vs-httpclient-vs-httpwebrequest/.

[9] Stack Overflow. Storing encryption keys – best practices, . URL http://stackoverflow.com/questions/723653/storing-encryption-keys-best-practices.

[10] Stack Overflow. Advice on unit testing a windows forms application, . URL http://stackoverflow.com/questions/9097044/advice-on-unit-testing-a-windows-forms-application.

[11] Quora. How a server can send update without a client request? URL https://www.quora.com/How-a-server-can-send-update-without-a-client-request.

[12] Telerik. Zip content of a folder into an encrypted zip file, 2014. URL http://www.telerik.com/forums/zip-content-of-a-folder-into-an-encrypted-zip-file.

[13] Wikipedia the free encyclopedia. Https. URL https://en.wikipedia.org/wiki/HTTPS.