Abschlussprüfung Herbst 2019

Fachinformatiker für Anwendungsentwicklung

Dokumentation zur betrieblichen Projektarbeit

Erstellung eines Keyloggers zur Schulung von Assistenzsoftware für blinde und sehbehinderte Anwender

Prüfungsbewerber:

Alexander Stahl

Anne-Frank-Straße 13

35037 Marburg

Ausbildungsbetrieb:

iad GmbH

Neue Kasseler Straße 62e

35039 Marburg

Ausbildungsleiter:

Silke Würtz

Inhalt

Abbildungsverzeichnis

Tabellenverzeichnis

1. Einleitung

1.1. Projektumfeld

Dräger & Lienert Informationstechnologie GbR (im Folgenden DL) ist ein Kleinunternehmen mit Sitz in Marburg (Hessen). Das Unternehmen entwickelt und vertreibt deutschlandweit Software für ergonomisch optimierte Arbeitsplätze für blinde und sehbehinderte Arbeitnehmer.

Zu den Eigenentwicklungen gehört ein Customer Relationship Management System, ein elektronisches Telefonbuch, sowie eine modularisierte Suite, die es dem Anwender ermöglicht auf Dateien, Verzeichnisse, Programme und Internetdienste mit Hilfe von Direktzugriffen zuzugreifen.

Außerdem werden herstellerunabhängig Hard- und Software wie Screenreader und Braillezeilen vertrieben.

Für die oben genannte Hard- und Software werden auch Schulungen am jeweiligen Arbeitsplatz des Kunden angeboten.

1.2. Ausgangsituation

Um auf spezielle Kundenbedürfnisse eingehen zu können, finden Schulungen in der Benutzung von Assistenzsoftware grundsätzlich am Einsatzort, in der Regel dem Arbeitsplatz und PC der Kunden, statt.

Im Moment stützen sich die Schulungsinhalte lediglich auf die Aussagen des Kunden, ohne eine Möglichkeit der Analyse des Arbeitsverhaltens seitens des Schulenden. Dadurch ergibt sich oft die Situation, dass nachträgliche Schulungstage nötig werden, da die Kunden häufig nicht überblicken können, wie sich ihr Arbeitsverhalten mit neuer Assistenztechnologie verändert, bzw. welche Möglichkeiten die eingesetzten Assistenzprogramme bieten.

1.3. DL EasyTask

DL EasyTask ist ein von Dräger & Lienert entwickeltes Programm, welches dazu dient, die Produktivität von blinden und sehbehinderten Anwendern zu steigern. Es läuft zunächst im Hintergrund und kann durch eine einfache Tastenkombination aktiviert werden. Diese Tastenkombination öffnet eine Eingabemaske, in der der Anwender eine von ihm gewählte Abkürzung eingibt, welche eine von ihm vorher definierte Aktion (Job) ausführt. Diese Jobs können zum Beispiel Direktzugriffe auf Programme oder Dateien, Aufrufen und navigieren von Webseiten, oder einfügen von Textbausteinen sein.

So kann ein Anwender mit einer Abkürzung den Browser starten, zu einem bestimmten Eingabefeld navigieren und dort etwas eingeben. Dies ist besonders nützlich, bei Webseiten, wo Anwenderinformationen wie Benutzername und Passwort eingegeben werden müssen.

1.4. Projektziel

Den Schulenden soll die Möglichkeit gegeben werden, das Arbeitsverhalten der Kunden hinsichtlich des Gebrauchs von Assistenztechnologien zu analysieren. Dafür soll ein Programm entwickelt werden, welches Tastatureingaben im Zusammenhang mit verschiedenen Assistenzprogrammen aufzeichnet. Das Programm soll folgende Funktionen haben:

- Manuelles Starten und Stoppen des Logvorgangs
- Nur Tastenkürzel für die jeweilig relevanten Assistenzprogramme werden geloggt
- Es wird eine Logdatei angelegt
- Daten aus der Logdatei werden grafisch aufbereitet
- Einfach zu bedienende grafische Oberfläche
- Assistenzprogramme sollen ausgewählt werden können
- Programme in denen geschult wird sollen ausgewählt werden können
- Logdatei soll bei Beendung des Programms gelöscht werden können
- Übersichtliche Auflistung der Tastatureingaben
- (Wo sinnvoll) Mögliche EasyTask Jobs werden angezeigt
- Kann von blinden und sehenden Schulenden genutzt werden

Die Entscheidung, einen eigenen Keylogger zu programmieren, liegt darin begründet, dass man volle Kontrolle darüber hat, was für Eingaben aufgezeichnet werden und was mit diesen dann passiert. Die Logdatei soll aus Sicherheitsgründen lediglich lokal auf dem Computer gespeichert werden.

1.5. Projektabgrenzung

Da der Umfang des Projekts lediglich 70 Stunden beträgt, wird sich die Logfunktion auf einige wenige, wichtige Programme beschränken. Die Auswahl wurde danach getroffen, wie häufig die Programme von Anwendern genutzt werden. Die Wahl fiel hierbei auf die Browser Mozilla Firefox, Google Chrome und Internet Explorer, die Microsoft Office Anwendungen Word, Outlook und Excel, sowie die Assistenztechnologien JAWS, NVDA und ZoomText.

2. Projektplanung

2.1. Vorgehensmodell

Als Vorgehensmodell wurde sich für das V-Modell entschieden. Agile Methoden konnten von vornherein ausgeschlossen werden, da lediglich ein Entwickler regelmäßig an dem Projekt gearbeitet hat. Das V-Modell erschien daher als das praktikabelste Vorgehensmodell. Im Rahmen dessen wurden zunächst die Nutzungsanforderungen mit dem Geschäftsführer von Dräger & Lienert festgelegt. Dann wurden die Systemanforderungen wie im Punkt 2.4 beschrieben geplant. Dazu wurde ein Anwendungsfalldiagramm erstellt. Danach wurde ein Klassendiagramm entworfen und die Klassen im Detail geplant. Auf dessen Basis wurde programmiert. Die Komponenten- und Integrationstest sollen als White-Box-Tests vom Programmierer durchgeführt werden, der Systemtest als Black-Box-Test von einem Testnutzer.

Die einzelnen Phasen waren zeitlich wie folgt eingeteilt:

Projektphase	Stunden
Planung der Nutzungsanforderungen	3
Funktionaler Systementwurf	5
Technischer Systementwurf	8
Implementierung	40
Komponententests	3
Integrationstest	3
Erstellen der Dokumentation	8

Tabelle 1: Planung der Projektphasen

2.2. Ressourcen

2.2.1. Hardware

Für die Projektdauer stand ein Büroarbeitsplatz mit Microsoft Windows PC zur Verfügung.

2.2.2. Software

Da sich für ein Windows Presentation Foundation (WPF) User Interface entschieden wurde, wurde zum Programmieren des Keyloggers Microsoft Visual Studio 2019 verwendet. Die Entwicklungsumgebung ist ideal zum Entwickeln von .NET Anwendungen in C#.

Des Weiteren wurden die Programme Google Chrome, Mozilla Firefox und Microsoft Internet Explorer, die MS Office Programme Word, Excel und Outlook, sowie die Assistenzprogramme JAWS, NVDA und ZoomText benutzt.

Außerdem werden in der Auswertung Hinweise zur Benutzung des Programms DL EasyTask gegeben.

2.2.3. Personal

Die Anforderungen und Rahmenbedingungen des Projektes wurden vom Geschäftsführer festgelegt.

Für Fragen zur Entwicklung stand der Chefprogrammierer zur Verfügung.

2.3. Projektkosten

Die Projektkosten setzten sich aus den Stundenlöhnen des Praktikanten, Geschäftsführers und Chefprogrammierers zusammen und berechnen sich wie folgt:

	Zeit	Berechnung (Zeit x Stundenlohn)	Ressourcen	Gesamt
Praktikant	70h	70h x 12€ = 840€	2500€[1]	3340€
Geschäftsführer	2h	2h x 30€ = 60€		60€
Chefprogrammierer	4h	4h x 28€ = 112€		112€
Gesamt	76h	1012€	2500€	3512€

Tabelle 2: Projektkostenaufstellung

[1] Kosten für Softwarelizenzen

2.4. Oberfläche

Die Oberfläche soll aus zwei Fenstern bestehen, einem Hauptfenster und einem Auswertungsfenster. Auf dem Hauptfenster soll der Anwender die Möglichkeit haben auszuwählen, welche Programme ausgewertet werden sollen und welche Assistenztechnologie er benutzt. Außerdem sollen sich dort je ein Button zum Beenden des Programms, zum Öffnen des Auswertungsfensters und zum Starten des Keyloggers befinden.

Die Auswahl der einzelnen Programme soll über Checkboxen erfolgen, damit auch mehrere Programme ausgewählt werden können. Diese werden zur besseren Übersichtlichkeit auf Groupboxen positioniert.

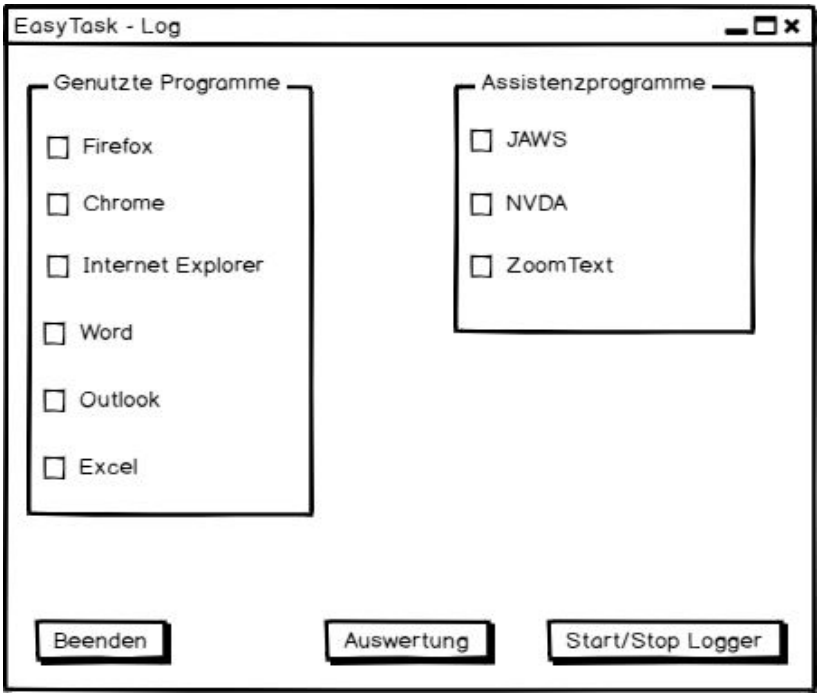

Abbildung 1: Mock-up des Hauptfensters

Auf dem Auswertungsfenster soll sich ein DataGrid mir DataTable befinden, welche anzeigt, für welches Programm ein Tastenkürzel benutzt wird, welches Tastenkürzel benutzt wurde und, wenn möglich, welcher EasyTask Job die Arbeitsabläufe vereinfachen würde. Zusätzlich sollen sich noch drei Buttons auf dem Fenster befinden, einer der das Programm schließt, einer der

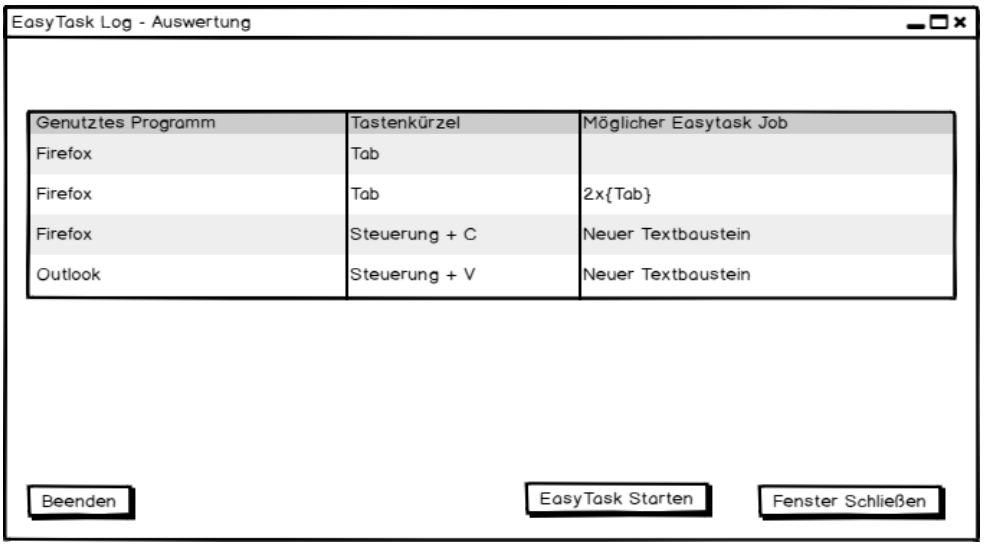

Abbildung 2: Mock-up des Auswertungsfensters

EasyTask öffnet, um eventuell Jobs zu kreieren und einer, um das Fenster zu schließen.

2.5. Klassen

Geplant sind drei Klassen, zwei Klassen für die beiden Fenster der Benutzeroberfläche, und eine Klasse für den Keylogger.

2.5.1. wndMain

Diese Klasse soll die Programmierlogik für das Hauptfenster des Programms enthalten. Es soll überprüft werden, welche der Checkboxen auf dem Hauptfenster ausgewählt sind und ob die entsprechenden Programme laufen. Ist eine Checkbox ausgewählt, deren Programm nicht läuft, soll es die Möglichkeit geben, dieses zu starten. Außerdem soll hier auch über einen Button der Keylogger gestartet werden.

2.5.2. wndAnalysis

Diese Klasse soll die Programmierlogik für das Analysefenster enthalten. Die Daten aus der Logdatei sollen ausgelesen und tabellarisch dargestellt werden. Die Tabelle soll drei Spalten enthalten, die anzeigt, welche Taste gedrückt wurde, welches das aktive Fenster zum Zeitpunkt des Tastendrucks war und ob eventuell ein EasyTask Job kreiert werden kann, um die Arbeitsabläufe zu vereinfachen.

2.5.3. Logger

In dieser Klasse befindet sich die Programmierlogik für den Keylogger. Ein Objekt der Klasse Logger wird angelegt, wenn im Hauptfenster der Button Start/Stop Logger geklickt wird. Im Konstruktor soll eine Methode aufgerufen werden, die hier einen Anfangslog schreibt. Diese Methode schreibt in eine Textdatei, welches Assistenzprogramm, welcher Browser und welches Microsoft Office Programm läuft. Dies geschieht mithilfe von Windows API Funktionen. (Process.GetProcesses Method, kein Datum)

Die vom Benutzer gedrückten Tasten sollen mit Keydownereignissen registriert werden. Um sicherzugehen, dass nur für das Assistenzprogramm relevante Tasten aufgezeichnet werden, wird eine Methode mit Kontrollstrukturen programmiert. So wird jede Eingabe darauf geprüft, ob sie in die Logdatei geschrieben wird. Dadurch wird verhindert, dass sensible Daten geloggt werden.

3. Implementierung

3.1. Oberfläche

3.1.1. Das Hauptfenster (wndMain)

Die Anwendung besteht aus einem Hauptfenster, welches sich öffnet, sobald das Programm gestartet wird. Auf dem Hauptfenster befinden sich zwei GroupBoxen, eine für *Relevante Programme* und eine für *Assistenzprogramme*. Auf beiden GroupBoxen befindet sich je ein nicht sichtbares StackPanel. Dies wird benötigt, da es in WPF GroupBoxen nicht möglich ist, mehr als ein Element auf GroupBoxen zu platzieren und somit ein Kindelement, welches mehrere Elemente aufnehmen kann, erfordert. Auf dem StackPanel werden dann die eigentlich benötigten CheckBoxen platziert. Die erste GroupBox fasst alle Programme, die mit den Assistenzprogrammen benutzt werden, zusammen. Es wurde sich für CheckBoxen entschieden, damit der Anwender mehr als ein Programm angeben kann. Die zweite GroupBox enthält die Assistenzprogramme. Auch hier kann mehr als eins ausgewählt werden. Anders, als bei *Relevante Programme* muss hier mindestens eine CheckBox ausgewählt sein, damit der Keylogger die entsprechenden Tastenkombinationen aufzeichnen kann.

Außerdem befinden sich drei Buttons auf dem Fenster. Der *Beenden* Button schließt das Programm, der *Auswertung* Button öffnet das Auswertungsfenster und der *Start/Stop Log*

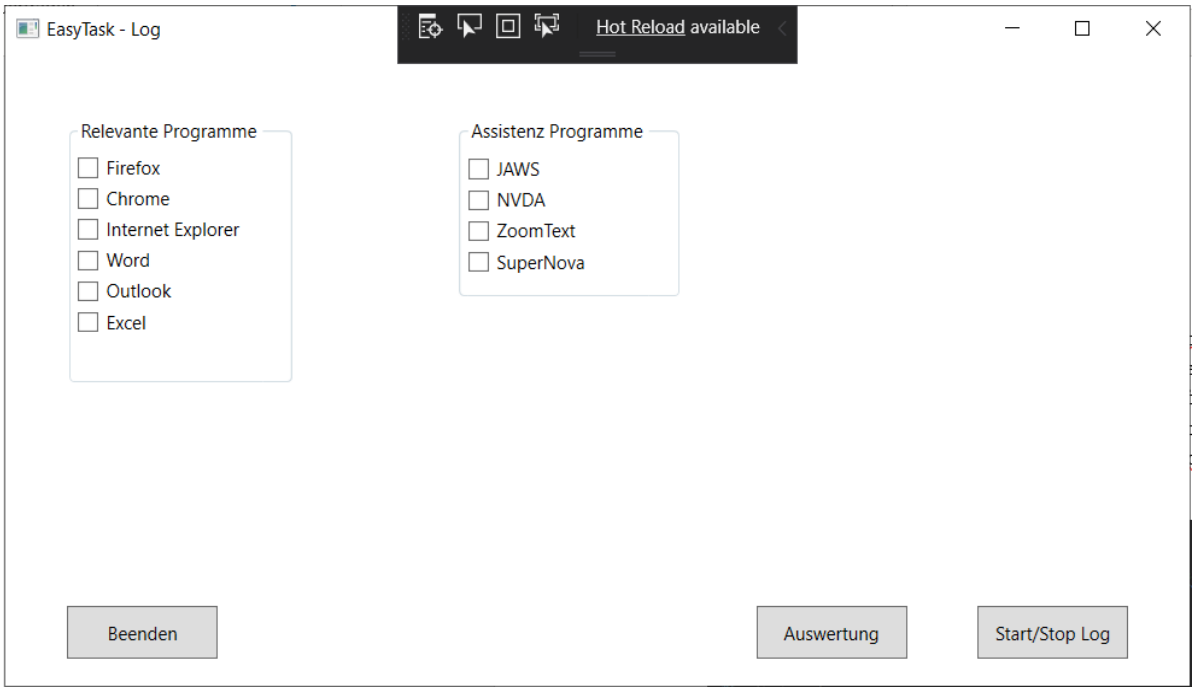

Abbildung 3: Das Hauptfenster

Button dient dazu, den eigentlichen Logvorgang zu starten.

3.1.2. Das Auswertungsfenster (wndAnalysis)

In diesem Fenster werden die Daten aus den Logdateien in eine DataGrid übertragen, dort wird auch angezeigt welches Fenster das aktive Fenster ist. Zusätzlich werden die Daten aus der Logdatei analysiert und darauf geprüft, ob die benutzten Tastenkürzel eventuell für einen Vorgang benutzt werden, der durch ein EasyTask Job leichter realisiert werden kann.

Es befinden sich drei Buttons auf diesem Fenster. Der *Beenden* Button schließt das Programm, der *Schließen* Button schließt lediglich das Fenster und der *EasyTask Starten* Button öffnet das Programm EasyTask.

3.1.3. Message Boxen

Zusätzlich besteht die Oberfläche noch aus zwei Fenstern, welche als MessageBoxen fungieren. Es konnten hier nicht die von WPF zur Verfügung gestellten MessageBoxen verwendet werden, da die MessageBox wndMessageBoxAsstTech die Möglichkeit bieten soll, ein Assistenzprogramm aus einer Liste von CheckBoxen auszuwählen und zu starten. Außerdem befinden sich auf diesem Fenster zwei Buttons. Der Button Starten startet die ausgewählten Assistenzprogramme und der Button Logger Schließen beendet das gesamte Programm.

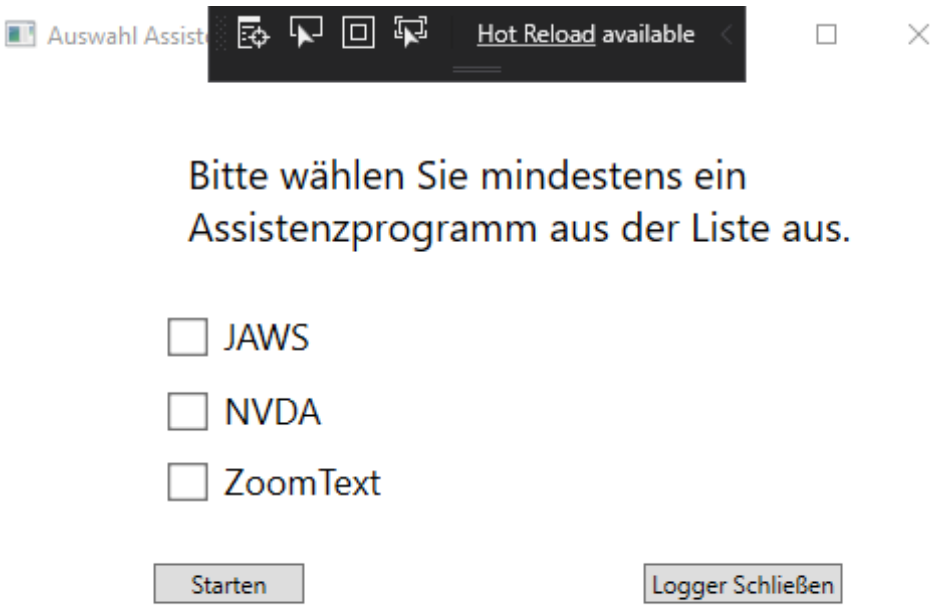

Abbildung 4: wndMessageBoxAsstTech

Die MessageBox wndMessageBoxStartLog wird angezeigt, wenn der Anwender im Hauptfenster den Button Start/Stop Log auslöst und alle Bedingungen zum Starten des Loggers erfüllt sind. Es beinhaltet lediglich den Text "Der Logger wurde gestartet". Es konnte

keine WPF MessageBox verwendet werden, da sie nur für eine Sekunde angezeigt werden soll.

3.2. Klassen

3.2.1. wndMain

Die Klasse wndMain enthält Variablen und Methoden des Hauptfensters des Programms. Zunächst wird ein Objekt des Typs Logger mit Namen logFile angelegt.

Als nächstes werden boolsche Variablen deklariert, die zum Überprüfen der Checkboxen für die gewünschten Programme, unter denen der Keylogger laufen soll, nötig sind.

Außerdem werden boolsche Variablen deklariert, welche zum Überprüfen der laufenden Prozesse nötig sind.

In der Methode BtnLog_Click werden alle boolschen Variablen zunächst auf false gesetzt. Dann werden die Methoden CheckCheckBox, CheckProcesses und isCheckedAndRunning aufgerufen. Diese Methoden werden im Folgenden noch näher erläutert. Es wird mit einer if-Abfrage überprüft, ob mindestens ein Assistenzprogramm läuft, ist dies nicht der Fall, wird das Fenster wndMessageBoxAsstText aufgerufen. (siehe Kapitel 3.2.3) Läuft mindestens ein Assistenzprogramm, erscheint zunächst das Fenster, um dem Anwender anzuzeigen, dass der Logger gestartet wurde. Dann wird ein KeystrokeAPI Objekt angelegt und in der Variablen ki gespeichert. Es werden dann zwei Logdateien angelegt, zum einen die Datei KeyLog.txt und zum anderen die Datei KeysOnly.txt.

```
95
96              // wenn dieser Button gedrückt wird, soll eine Überprüfung stattfinden, ob der Logvorgang überhaupt gestartet werden soll, wenn nicht
97              // soll die Möglichkeit bestehen, JAWS/ZoomText zu starten... zur Überprüfung soll zunächst nur auf JAWS und ZoomText geprüft werden,
98              // Browser /Word können nachher noch gestartet werden
99              checkCheckBox();
100             checkProcesses();
101             isCheckedAndRunning();
102
103             // überprüft, ob Assistenzprogramme laufen, wenn nicht, wird ein Fenster geöffnet, in dem man ein Assistenzprogramm starten kann.
104             if (procJaws == false && procNvda == false && procZoomtext == false && jaws == false && nvda == false && zoomtext == false)
105             {
106                 wndMessageBoxAsstTech wndMessageBox = new wndMessageBoxAsstTech();
107                 wndMessageBox.Show();
108             }
109             else
110             {
111                 wndMessageBoxStartLog wndMessageBoxStartedLog = new wndMessageBoxStartLog();
112                 wndMessageBoxStartedLog.Show();
113                 App.IsLoggingStarted = true;
114
115             }
116
117
118         }
```

Abbildung 5: Ausschnitt von BtnLog_Click

Die Methode CheckCheckBox überprüft, ob die CheckBoxen für die verwendeten Programme angehakt sind über die Eigenschaft checkBox.IsChecked. Dies geschieht mit Hilfe von if-Abfragen für jede Checkbox, die sich auf dem Fenster befindet. Ist diese Eigenschaft wahr, ist die Checkbox angehakt. Dann wird die zuvor deklarierte boolsche Variable für das jeweilige Programm (z.B. bool firefox) auf true gesetzt.

Die Methode CheckProcesses überprüft, welche der möglichen logbaren Programme laufen. Dies geschieht, indem ein Array processList angelegt wird, in dem alle aktiven Prozesse des Computers gespeichert werden. Mit Hilfe einer foreach-Schleife wird dieses Array durchlaufen und auf relevante Prozesse mittels ihres Namens geprüft.

Die Methode isCheckedAndRunning vergleicht die boolschen Variablen für die Checkboxen, mit denen für die Prozesse. Wenn eine Checkbox angehakt ist, das Programm aber nicht läuft, wird eine MessageBox gezeigt, die einem die Möglichkeit gibt, das entsprechende Programm zu starten. Entschließt sich der Anwender dazu, das Programm nicht zu starten, wird der Wert für IsChecked der entsprechenden CheckBox auf false gesetzt und somit der Haken in dieser Checkbox entfernt.

```
1 reference
private void isCheckedAndRunning()
{
    if (firefox == true && procFirefox == false)
    {
        // Zeigt eine MessageBox wenn Firefox angehakt ist, aber nicht läuft. Ermöglicht es Firefox zu starten
        MessageBoxResult result = MessageBox.Show("Firefox läuft nicht, soll das Programm geöffnet werden?", "My Title", MessageBoxButton.YesNo, MessageBoxImage.Question);
        if (result == MessageBoxResult.Yes)
        {
            Process.Start(@"C:\Program Files\Mozilla Firefox\firefox.exe");
        }
        else
        {
            cbFirefox.IsChecked = false;
        }
    }
```

Abbildung 6: Ausschnitt isCheckedAndRunning

3.2.2. wndAnalysis

Die Klasse wndAnalysis enthält Methoden für das Auswertungsfenster. Auf diesem Fenster befindet sich ein DataGrid mit 3 Spalten. Im Wesentlichen besteht die Klasse wndAnalysis aus der Methode fillDataGrid(), welche zunächst die Tastenkürzel aus der Logdatei in die erste Spalte des DataGrids schreibt. In der zweiten Spalte steht der Titel des Fensters, das aktiv war, als das Tastenkürzel benutzt wurde und in der dritten Spalte steht ein Vorschlag für einen Job für EasyTask, sollte dies möglich sein.

3.2.3. wndMessageBoxAsstTech

Die Klasse wndMessageBoxAsstTech enthält Methoden für das Fenster Auswahl Assistenzprogramm.

Hier wird überprüft, welche der Checkboxen für die auswählbaren Assistenzprogramme ausgewählt sind. Das Programm der entsprechenden ausgewählten Checkboxen wird dann gestartet. Anschließend wird der Keylogger gestartet und das Fenster wndMessageBoxStartLog gezeigt.

3.2.4. wndMessageBoxStartLog

Die Klasse wndMessageBoxStartLog enthält Methoden für das Fenster *Logger Gestartet*.

Die Methode Timer ermöglicht es, dass das Fenster lediglich für eine Sekunde gezeigt wird und der Anwender es nicht manuell schließen muss.

3.2.5. Logger

Die Klasse Logger enthält Methoden, welche Informationen über die zu loggenden Daten geben.

Die Methode GetTitleOfActiveWindow gibt den Titel des aktiven Fensters als String zurück.

Die Methode GetBrowser liefert den Namen des verwendeten Browsers als String zurück. Hierzu werden boolsche Variablen für die einzelnen Browser (Firefox, Chrome, Internet

```
1reference
public string GetBrowser()
{
    string BrowserName = string.Empty;
    bool firefox = false;
    bool chrome = false;
    bool intex = false;

    // holt alle Prozesse, die laufen
    Process[] processlist = Process.GetProcesses();
    // läuft durch die Prozesse
    foreach (Process theprocess in processlist)
    {
        //schreibt nur Browser in die Logdatei
        if ((theprocess.ProcessName == "firefox" && firefox == false) || (theprocess.ProcessName == "chrome" && chrome == false) || (theprocess.ProcessName == "iexplore" && intex == false))
        {
            BrowserName = theprocess.ProcessName;

            switch (BrowserName)
            {
                case "firefox":
                    firefox = true;
                    break;
                case "chrome":
                    chrome = true;
                    break;
                case "iexplore":
                    intex = true;
                    break;
                default:
                    // hier muss ich mir noch was ausdenken
                    break;
            }
        }
    }

    return BrowserName;

}
```

Abbildung 7: Ausschnitt GetBrowser

Explorer) deklariert und deren Werte werden auf false gesetzt.

Dann werden ähnlich wie bei der Methode CheckProcesses aus wndMain ein Array der laufenden Prozesse durchlaufen und die boolschen Variablen werden auf true gesetzt, wenn einer der Prozesse der ausgewählte Browser ist.

Die Methode GetAssistTech funktioniert auf exakt die gleiche Weise, wie die Funktion GetBrowser. Hier wird lediglich auf die Prozesse für die zu verwendenden Assistenzprogramme geprüft, und der Name dieser wird auch hier als String zurückgeliefert.

Auf ähnliche Weise funktioniert auch die Methode GetOffice, welche auf Prozesse der Microsoft Office Programme Word, Excel und Outlook prüft. Statt diese als String zurückzuliefern, werden hier Stringvariablen mit den Namen der Office Programme belegt, wenn diese in den Prozessen auftauchen.

Die Methode Get FocusedControl liefert das aktuell fokussierte Kontrollelement zurück.

Die Methode WriteInitialLog schreibt die Textdatei log.txt. Sie wird im Konstruktor der Logger Klasse aufgerufen, so dass sie gleich beim Start des Keyloggers erstellt wird. Zweck ist es zunächst einen Überblick über die verwendeten Programme zu erhalten. In der Datei erscheinen vier Zeilen. Die erste Zeile ist eine Überschrift, die mit der Methode WriteAllText erstellt wird. Die nächsten drei Zeilen werden mit der Methode AppendAllText erstellt. Die zweite Zeile enthält den Browsernamen, es wird GetBrowser aufgerufen. Die dritte Zeile enthält durch den Aufruf von GetAssistTech den Namen der verwendeten Assistenzprogramme. Die letzte Zeile enthält die Namen der verwendeten Office Programme,

```
191         public void WriteInitialLog()
192         {
193
194             // Überschrift für die Log Datei
195             File.WriteAllText(@"D:\log.txt", "Active Processes: \r\n\r\n");
196
197             // Browser
198             File.AppendAllText(@"D:\log.txt", "Browser: " + GetBrowser() + "\r\n");
199             // Assistive Technology
200             File.AppendAllText(@"D:\log.txt", "Assistive Tech: " + GetAssistTech() + "\r\n");
201             // Office Programme
202             GetOffice();
203             File.AppendAllText(@"D:\log.txt", "Office Programme: " + word + " " + outlook + " " + excel + "\r\n");
204         }
205     }
206 }
```

Abbildung 8: WriteInitialLog

welche über die String Variablen für die Office Programme dort hineingeschrieben werden.

3.3. Keystroke API

Da sich herausstellte, dass ein Keylogger mit Hilfe von Keydownereignissen nicht möglich ist, da diese verlangen, dass sich das Logprogramm stets im Vordergrund befindet, wurde eine API eingebunden, die die Bezeichnung der gedrückten Taste als String zurückgibt. Die Informationen über die gedrückten Tasten bekommt die Keystroke API von der Windows API. (Rissetto, 2018)

3.4. Die Logdateien

3.4.1. KeyLog.txt

Die Datei KeyLog.txt wird angelegt, um im Auswertungsfenster anzeigen zu können, welche Tasten gedrückt wurden, welches Fenster dabei aktiv war. Dadurch lässt sich auslesen, wie der Anwender in bestimmten Programmen navigiert. Dies wird grafisch im Fenster wndAnalysis aufbereitet, um es besser lesbar und verständlich zu machen.

3.4.2. KeysOnly.txt

Die Datei KeysOnly.txt speichert lediglich die vom Anwender gedrückten Tasten. Dies macht es einfacher die Daten weiterzuverarbeiten und zu analysieren, wo dem Anwender geholfen werden kann.

3.5. Was passiert mit den Daten?

Bei diesem Projekt war es wichtig sicherzustellen, dass der Anwender sicher sein kann, dass seine Daten nicht in fremde Hände gelangen, da die Schulungen oft an Arbeitsplätzen stattfinden, wo mit sensiblen Daten gearbeitet wird. Daher besteht die Möglichkeit über eine Schaltfläche beim Beenden des Programms die Logdateien zu löschen.

3.6. Tests

Die angewandten Testverfahren waren White-Box-Tests und Black-Box-Tests. Diese Testverfahren erschienen am sinnvollsten, wenn man die Art und den Umfang des Projekts bedenkt.

Die White-Box-Tests wurden angewandt, um die einzelnen Methoden zu testen. Ausführlichst wurde die Methode isCheckedAndRunning() getestet, um sicherzustellen, dass kein Programm angehakt sein kann, welches nicht läuft. Aus diesem Test ergab sich die Notwendigkeit der Implementierung einer MessageBox, welche den Anwender fragt, ob ein angehaktes Programm gestartet werden soll. Wird dies bestätigt geschieht dies, wird dies abgelehnt, wird der Haken in der entsprechenden TextBox entfernt.

Außerdem wurden White-Box-Tests zur Überprüfung der Logvorgänge verwendet. Hierzu wurden Testdateien angelegt und manuell ausgelesen.

Die Black-Box-Tests wurden von Testanwendern durchgeführt. Hier lag das Hauptaugenmerk auf Bedienerfreundlichkeit und Navigierbarkeit. Dies führte zu einer geringfügigen Veränderung der Tabreihenfolge.

4. Fazit

Noch läuft der Keylogger autonom, ohne in andere Programme eingebunden zu sein. Perspektivisch soll der Keylogger jedoch als Modul in das Programm EasyTask von Dräger & Lienert implementiert werden und dort als Schulungstool dienen. Ein Keylogger mit Auswertung kann dazu dienen Schulungen zu vereinfachen, oder gar unnötig zu machen, da der Keylogger selbst das Bedienverhalten analysiert und auswertet.

Es sind noch etliche weitere Möglichkeiten vorstellbar, das Schulungstool um Funktionen zu erweitern. So könnten zum Beispiel weitere Assistenzsoftwares hinzugefügt werden. Es wäre auch vorstellbar bei der Auswertung eine Möglichkeit zu schaffen zu überprüfen, welche Kontrollelemente (TextBox, CheckBox etc.) angesteuert werden.

Nach Möglichkeit soll das Schulungstool in Zukunft auch unabhängig von der benutzten Software Tastenkürzel aufzeichnen, damit auch individuelle Programme, wie spezielle Branchensoftware ausgewertet werden können.

5. Glossar

Customer Relationship Management (CRM) System:

Software zur Verwaltung von Kundendaten.

Screenreader

Software, die grafische Bildschirminhalte entweder mittels Sprachsynthese über Lautsprecher oder auf einer Braillezeile ausgibt.

Braillezeile

Taktiles Ausgabegerät von grafischen Inhalten für blinde und sehbehinderte Menschen. Die Ausgabe erfolgt in Punktschrift (Brailleschrift) und wird in der Regel von einem Screenreader angesteuert.

Windows Presentation Foundation

Grafikgerüst und Fenstersystem des .NET Frameworks. (Chapell, 2006)

API

Anwendungsschnittstelle, die es Anwendungen erlaubt miteinander zu kommunizieren.

.NET Framework

".NET Framework ist eine Entwicklungsplattform zum Erstellen von Apps für Web, Windows, Windows Phone, Windows Server und Microsoft Azure." (.NET Framework – Leitfaden, 2019)

Keylogger

Eine Anwendung, die es erlaubt Tastatureingaben aufzuzeichnen und zu speichern.

JAWS

Eine Screenreader-Software

NVDA

Eine Screenreader-Software

ZoomText

Eine Vergrößerungssoftware

Literaturverzeichnis

.*NET Framework – Leitfaden*. (02. April 2019). Von https://docs.microsoft.com/de-de/dotnet/framework/ abgerufen

Chapell, D. (September 2006). *Introducing Windows Presentation Foundation*. Von https://docs.microsoft.com/en-us/previous-versions/dotnet/articles/aa663364(v=msdn.10)?redirectedfrom=MSDN#introducingwpf_topic13 abgerufen

Entwicklungshandbuch für .NET Framework. (30. März 2017). Von https://docs.microsoft.com/de-de/dotnet/framework/development-guide abgerufen

Hooks. (31. Mai 2018). Von https://docs.microsoft.com/en-us/windows/win32/winmsg/hooks#:~:targetText=A%20hook%20is%20a%20point,reach%20the%20target%20window%20procedure. abgerufen

JAWS 2020 Documentation. (kein Datum). Von https://support.freedomscientific.com/Products/Blindness/JAWSdocumentation abgerufen

NVDA 2019.2.1 User Guide. (kein Datum). Von https://www.nvaccess.org/files/nvda/documentation/userGuide.html abgerufen

Process.GetProcesses Method. (kein Datum). Von https://docs.microsoft.com/en-us/dotnet/api/system.diagnostics.process.getprocesses?view=netframework-4.8#System_Diagnostics_Process_GetProcesses abgerufen

Rissetto, F. (25. September 2018). Von https://github.com/fabriciorissetto/KeystrokeAPI abgerufen

ZoomText Benutzerhandbuch. (2019). Von http://www.aisquared.com/docs/zt11/ZoomText_11_User_Guide_German.pdf abgerufen

Anhang

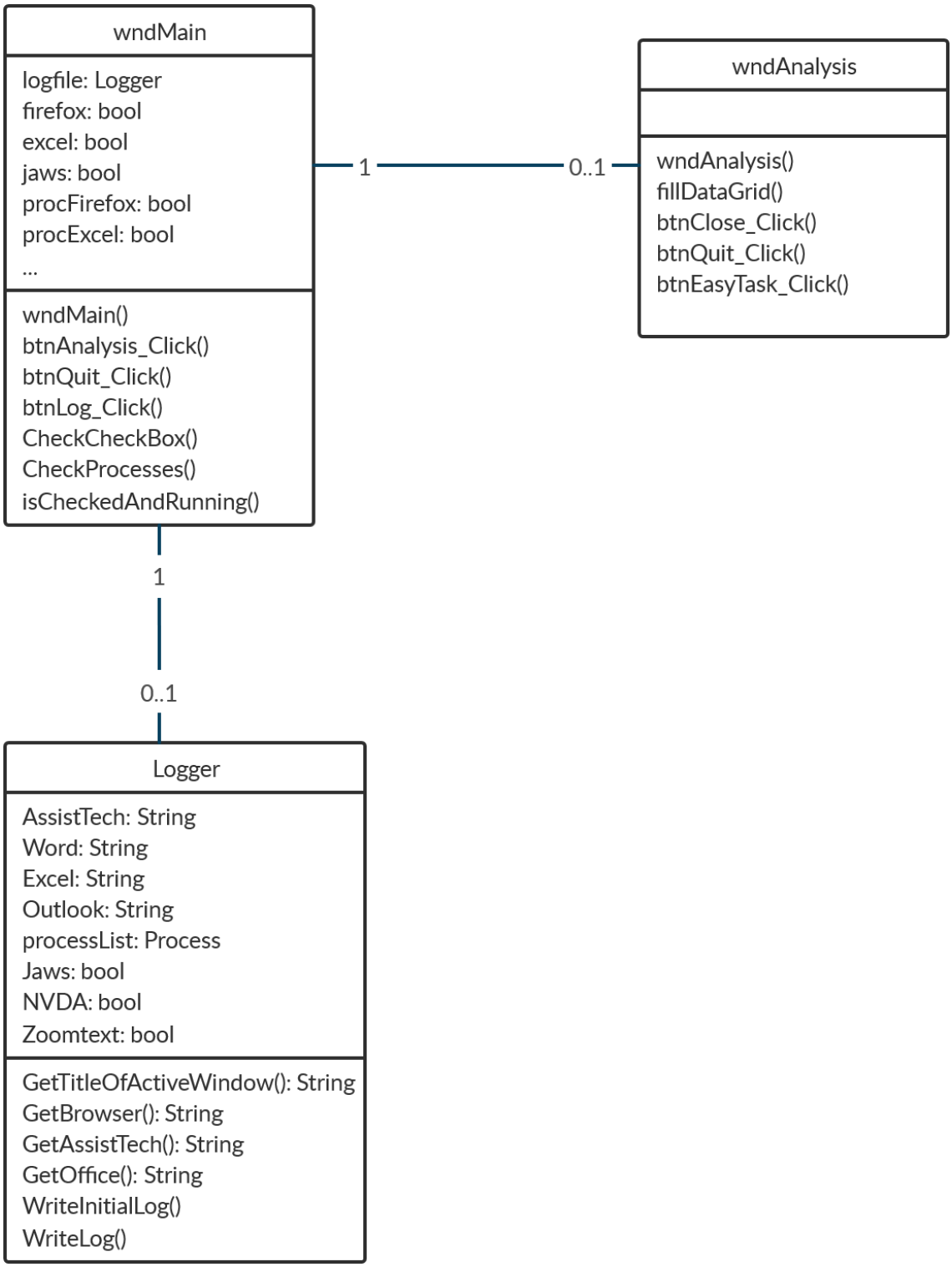

Abbildung 9: Klassendiagramm

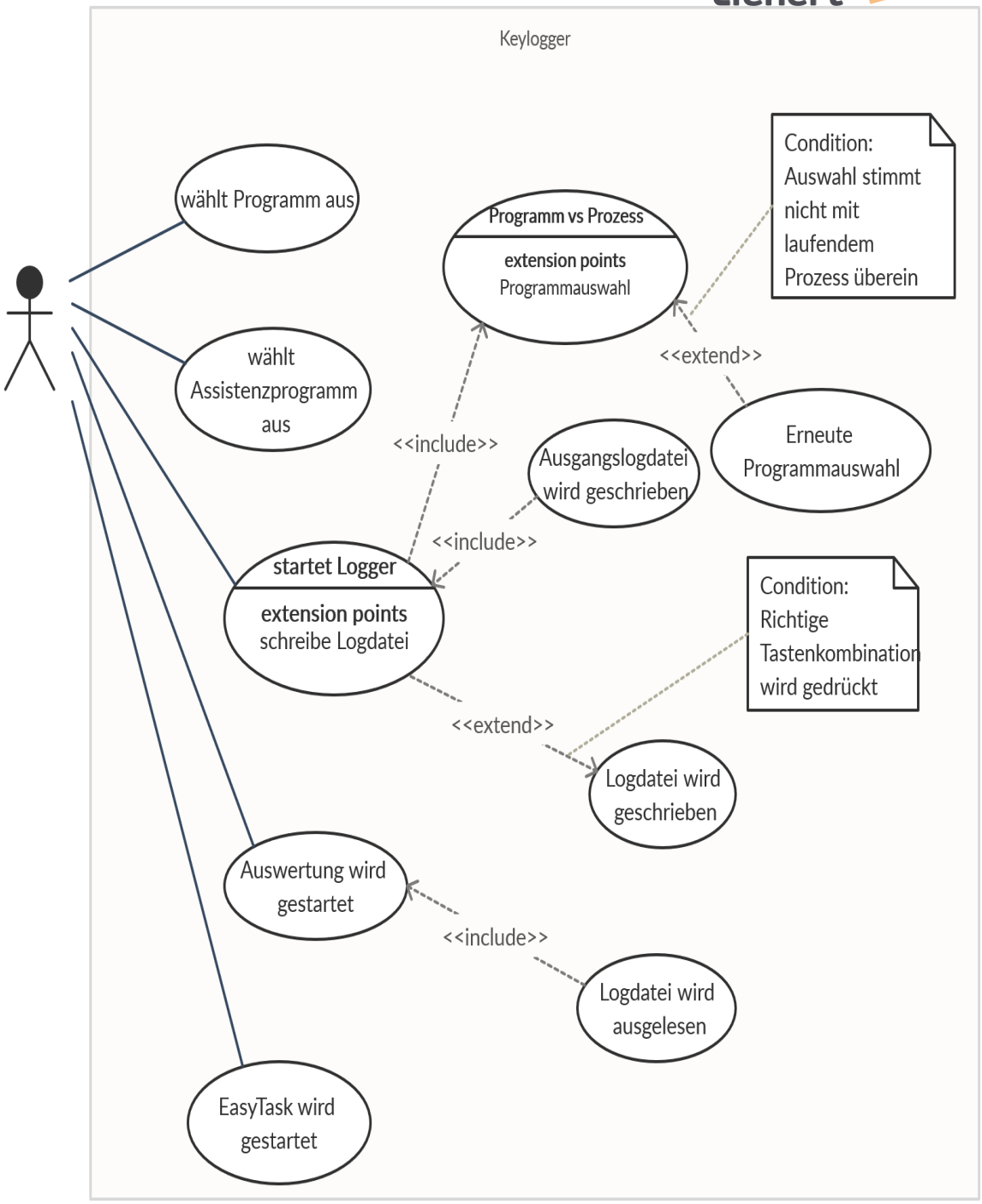

Abbildung 10: Anwendungsfalldiagramm