Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Bachelorarbeit

Nick Stegemann

Entwicklung einer automatischen Konsistenzprüfung für Systemmodelle in Microsoft Visio

Fakultät Technik und Informatik
Department Fahrzeugtechnik und Flugzeugbau

Faculty of Engineering and Computer Science
Department of Automotive and Aeronautical Engineering

Nick Stegemann

Entwicklung einer automatischen Konsistenzprüfung für Systemmodelle in Microsoft Visio

Bachelorarbeit eingereicht im Rahmen der Bachelorprüfung

im Studiengang Mechatronik
am Department Fahrzeug- und Flugzeugbau
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Erstprüferin: Prof. Dr.-Ing. Jutta Abulawi
Zweitprüferin: Prof. Dr. Zhen Ru Dai

Abgabedatum: 18.04.2019

Zusammenfassung

Nick Stegemann

Thema der Bachelorthesis

„Entwicklung einer automatischen Konsistenzprüfung für Systemmodelle in Microsoft Visio“

Stichworte

Microsoft Visio, Konsistenzprüfung, Systemmodelle

Kurzzusammenfassung

Microsoft Visio ist ein häufig eingesetztes Programm zur Modellierung von Systemmodellen. In einem Projekt werden mehrere Systemmodelle erstellt um verschiedene Blickwinkel oder Abläufe des Projektes darstellen zu können. Im Laufe der Modellierung schleichen sich häufig Fehler in die Bezeichnungen der Systemkomponenten ein. Infolgedessen entstehen am Ende der Modellierungen inkonsistente Systemmodelle, da die Systemkomponentennamen nicht mit denen, der anderen Systemmodellen konsistent sind. In dieser Arbeit wird ein Programm vorgestellt, dass Systemmodelle in Microsoft Visio automatisch auf Konsistenz hin prüft.

Abstract

Microsoft Visio is a frequently used program for modeling system models. In one project several system models are created in order to represent different perspectives or processes of the project. In the course of modeling, errors often creep into the designations of the system components. As a result, inconsistent system models are created at the end of the modeling process because the system component names are not consistent with those of the other system models. This paper presents a program that automatically checks system models for consistency in Microsoft Visio.

Inhaltsverzeichnis

1 Einleitung

1.1 Motivation

Die Komplexität und somit auch die Interdisziplinarität von technischen Systemen nehmen immer weiter zu. Im Zuge der Weiterentwicklung der Industrie in Richtung Internet oft Things und Industrie 4.0 hat sich dieser Trend in den letzten Jahren noch verstärkt. Infolgedessen sind an der Entwicklung von neuen Systemen zunehmend mehr Entwickler beteiligt, die aus verschiedenen Fachbereichen stammen. Die damit einhergehende Multidisziplinarität des Projektes darf allerdings nicht zu einer langsameren oder ineffizienteren Systementwicklung führen. Das folgende Beispiel veranschaulicht ein typisches Problem dieser Entwicklung. Einem Entwickler liegen Informationen aus einem ihm nicht bekannten Fachbereich vor, auf dessen Basis er eine Entscheidung für seinen Anteil an dem Projekt treffen muss. Er muss daher die Informationen verstehen, anwenden und anderen Entwicklern aus seinem Fachbereich erklären können. Dies wird einige Zeit in Anspruch nehmen und ihm möglicherweise nicht komplett gelingen. Würden die Informationen in einer ihm bekannten „Sprache“ vorliegen, würden ihm aufkommende Verständnis- und Kommunikationsprobleme, sowie nicht effizient investierte Arbeitszeit erspart bleiben.

Das bedeutet, dass im Idealfall alle an der Entwicklung beteiligten Personen eine gemeinsame „Sprache“ sprechen, die ein tiefgehendes Verständnis des kompletten Systems ermöglicht, um eine reibungsfreie Kommunikation zwischen den einzelnen Disziplinen zu gewährleisten. Aus diesem Grunde nimmt der Einsatz von Modellierungssprachen in der Industrie stetig zu. Mithilfe dieser „Sprachen“ können abstrahierte Diagramme von einem System modelliert werden. Die Entwickler nutzen diese Systemmodelle als disziplinübergreifendes Verständigungsmittel.

Entwickler bekommen in der Systemmodellierung allerdings häufig ein „Konsistenzproblem“.

Bei der Entwicklung eines Systems ist stets auf Konsistenz zwischen den Systemmodellen zu achten. In der Entwicklungsphase geschieht es häufig, dass Namen von Systemkomponenten, beispielsweise wegen einer Nachbearbeitung, leicht geändert werden. Aus „Düsentriebwerk“ wird „Triebwerk“. Da diese ungewollten Namensänderungen nicht automatisch mit allen, das System betreffenden Systemmodellen synchronisiert werden, ergeben sich am Ende der Entwicklung nicht mehr einheitliche Bezeichnungen der einzelnen Komponenten des Systems in den verschiedenen Modellen. Es herrscht dementsprechend eine Inkonsistenz zwischen den Bezeichnungen der Systemkomponenten der verschiedenen Systemmodelle.

Microsoft Visio (Visio) ist ein weit verbreitetes Visualisierungsprogramm, welches unter anderem zur Modellierung von Systemmodellen genutzt werden kann. Visio bietet keine Möglichkeit eine direkte Verbindung zwischen den Systemmodellen herzustellen an, mit der sich auf Konsistenz hin prüfen lies. Dementsprechend tritt das „Konsistenzproblem“ in von Visio erstellten Modellen häufig auf. In dieser Bachelorarbeit wird ein Programm vorgestellt, welches eine Konsistenzprüfung zwischen in Visio erstellten Systemmodellen ermöglicht.

1.2 Abgrenzung der Arbeit

Das Ziel dieser Bachelorarbeit ist es nicht den Visio Basiscode zu verändern umso eine neue Visio Version zu erstellen die eine Verbindung zwischen verschiedenen Modellierungen erlaubt. Stattdessen soll mit der Skriptsprache Visual Basic for Applications (VBA) eine Möglichkeit gefunden werden eine Verbindung zwischen verschiedenen Visio- Modellen herzustellen umso eine permanente Konsistenzprüfung zu ermöglichen.

1.3 Aufbau der Arbeit

Die vorliegende Arbeit ist wie folgt aufgebaut.

Grundlegende Kenntnisse über Microsoft Visio, Visual Basic for Applications und die Systems Modeling Language werden in Kapitel 2 vermittelt.

In Kapitel 3 werden die Problemstellung und die einzelnen Entwicklungsschritte erläutert. Dabei werden die einzelnen Funktionen der verschiedenen Makros vorgestellt und deren Aufgaben in der Konsistenzprüfung beschrieben.

Die Implementierungsanleitung des Programmes befindet sich in Kapitel 4, die Schritt für Schritt erklärt, wie die Konsistenzprüfung zu implementieren ist. In Kapitel 5 werden die Schritte erläutert, die notwendig sind um die Konsistenzprüfung auszuführen.

Die Konsistenzprüfung wird in Kapitel 6 mit verschiedenen SysML- Diagrammen modelliert. Diese werden mit den Visio Schablonen von Dr. Pavel Hruby erstellt.

In Kapitel 7 wird die Konsistenzprüfung anhand eines Praxisbeispiels verifiziert. Dabei werden zwei, aus den Visio Schablonen von Dr. Pavel Hruby erstellte, Diagramme auf Konsistenz hin geprüft und das Ergebnis präsentiert.

Das Kapitel 8 beschäftigt sich mit dem Ausblick auf folgende Arbeiten und weiteren Möglichkeiten die Konsistenzprüfung zu ergänzen um mehre Systemmodelle auf einmal auf Konsistenz hin zu prüfen oder andere Daten der Systemmodelle zu vergleichen.

Eine Zusammenfassung dieser Bachelorarbeit befindet sich in Kapitel 9, in der die wichtigsten Erkenntnisse der Arbeit festgehalten werden.

2 Grundlagen

2.1 Systems Modeling Language

Die Systems Modeling Language ist eine formale Modellierungssprache, die am 1. September 2001 von der Object Management Group (OMG) veröffentlicht worden ist. Seitdem wurde die Modellierungssprache immer wieder von Arbeitsgruppen der OMG überarbeitet und ist aktuell (Stand April 2019) in der Version 1.5 verfügbar. Diese Modellierungssprache baut auf der Unified Modeling Language 2.0 (UML 2.0) auf und erweitert diese um das Zusicherungs- und das Anforderungsdiagramm. Das Interne Blockdiagramm, das Blockdefinitionsdiagramm und das Aktivitätsdiagramm wurden im Gegensatz zur UML 2.0 modifiziert. In Abbildung 1 ist die SysML- Taxonomie mit den Unterschieden zur UML 2.0 dargestellt. [Wik19]

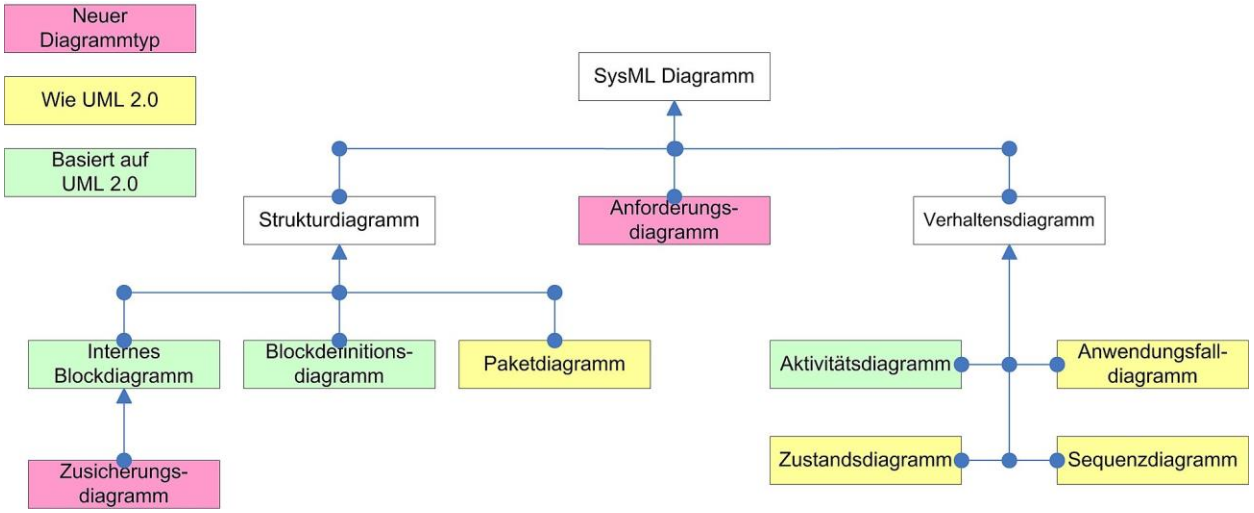

Abbildung 1: SysML- Diagramm- Taxonomie [Wik19]

Wenn aus verschiedenen Fachbereichen stammenden Entwickler gemeinsam ein System modellieren wollen, ist eine gemeinsame Sprache zur disziplinübergreifenden Kommunikation von Nöten. Mit den insgesamt neun verschiedenen Diagrammtypen lässt sich mit der SysML die Struktur, die Anforderungen und das Verhalten eines

Systems modellieren. Abgesehen von den Systemanforderungen sind die Systemstruktur und das Systemverhalten jeweils in verschiedenen Diagrammtypen darstellbar. Dies erleichtert das Verständnis, da dieselbe Information aus mehreren Perspektiven betrachtet werden kann. Die SysML kann in verschiedenen Modellierungstools ausgeführt werden. In dieser Bachelorarbeit wird sie ausschließlich in Microsoft Visio angewendet.

2.2 Microsoft Visio

Microsoft Visio ist ein Visualisierungsprogramm, welches erstmals als Visio 1.0 im Jahr 1992 veröffentlicht worden ist. Im Jahr 2000 ist das Unternehmen Visio von Microsoft für 1,3 Milliarden Dollar gekauft worden und seitdem wird die Software unter dem Namen Microsoft Visio (Visio) vertrieben. Am meisten findet Visio Anwendung in der Visualisierung von Ablaufdiagrammen, Geschäftsprozessen und Modellierungssprachen. Visio ist weit verbreitet, da es in der Microsoft Office- Suite enthalten ist und über vielfältige Anwendungsmöglichkeiten verfügt. Es lassen sich zum Beispiel von Visio erstellte Diagramme auf Computern präsentieren, die kein Visio installiert haben. In dieser Bachelorarbeit wird mit der Version Microsoft Visio 2016 gearbeitet. [Luc19]

2.2.1 Shapes in Microsoft Visio

In Visio sind Shapes die zentralen Elemente einer Zeichnung. Sie werden in Schablonen zusammengefasst. Es sind fertig erstellte Zeichnungskomponenten mit denen die Diagramme erzeugt werden. Microsoft Visio enthält in seiner Datenbank viele vorgefertigte Schablonen mit den entsprechenden Shapes für die gewünschte Modellierung. Es gibt ebenfalls die Möglichkeit Schablonen aus einer externen Quelle in Visio einzufügen. In dieser Bachelorarbeit wird ausschließlich mit den Namen der Shapes gearbeitet, da diese ausreichen, um die Systemmodelle auf Konsistenz hin zu prüfen.

2.3 Visual Basic for Applications

Visual Basic for Applications (VBA) ist eine Skriptsprache die seit Mitte der 1990er Jahre als Makro Sprache für einige Microsoft Office- Anwendungen zur Verfügung gestellt wird. Mit der Einführung von Office 2000 ist die Entwicklungsumgebung der Sprache in den einzelnen Office- Anwendungen vereinheitlicht worden. Heutzutage ist sie in den Microsoft Office- Anwendungen Word, Excel, Access, Project, Powerpoint, Frontpage, Visio und Outlook verfügbar. Mit ihr lassen sich automatisierte Abläufe in den Office- Anwendungen erstellen. Es können zum Beispiel sich wiederholende Aufgaben mit einem VBA- Skript beschleunigt werden. VBA kann ebenfalls zum Erweitern von Office- Anwendungen genutzt werden. Es kann zum Beispiel ein Benutzer beim Öffnen eines bestimmten Dokumentes zu einer Aufgabe aufgefordert werden. VBA ist die am weitesten verbreitete Programmiersprache um Programme für Office- Anwendungen zu erstellen. In dieser Bachelorarbeit wird sie in den Office-Anwendungen Visio 2016, Access 2016 und Excel 2016 eingesetzt. [Rum19] [Wik19a]

3 Entwicklungsschritte

3.1 Problemstellung

Bei der Entwicklung eines Systems werden mehrere Systemmodelle erstellt, mit dem Ziel, allen an dem Projekt beteiligten Entwicklern ein Verständnis des kompletten Systems zu ermöglichen. Daher werden möglichst viele Aspekte des Systems modelliert, wie zum Beispiel die Struktur, die Anforderungen und das Verhalten des Systems. Durch die steigende Komplexität der Systeme, steigt auch der Umfang und die Anzahl der Systemmodelle innerhalb eines Projektes. Infolgedessen kommt es häufig zu einem „Konsistenzproblem“.

Das „Konsistenzproblem“ beschreibt das Auftreten von Inkonsistenzen bei den Namen der einzelnen Systemkomponenten zwischen den verschiedenen Systemmodellen. Im Laufe der Entwicklung führen kleinere Abänderungen der Systemkomponentennamen zur Entstehung von Inkonsistenzen zwischen den Systemmodellen. Zum Beispiel ändert ein Entwickler in einem Systemmodel den Namen eines Schalters unbeabsichtigt von „switch0.15188“ zu „switch0.1588“. Diese kleineren Fehler wiederholen sich und im schlimmsten Falle sind am Ende der Modellierung die einzelnen Systemkomponentennamen zwischen den Systemmodellen in so einem Ausmaß nicht mehr konsistent, dass die Systemmodelle scheinbar nicht mehr zum selben System gehören.

Microsoft Visio ist ein Visualisierungsprogramm mit dem sich Systemmodelle modellieren lassen. Wie viele Visualisierungsprogramme, bietet auch Visio keine Möglichkeit, die erstellten Systemmodelle zu vergleichen und auf Konsistenz hin prüfen zu können. Es muss daher ein Programm entwickelt werden, welches die Systemkomponentennamen aus den Visio- Dateien heraus exportiert und anderweitig verarbeitet um dem Entwickler eine Konsistenzprüfung aufzeigen zu können. Dieses Programm soll sich möglichst benutzerfreundlich und zu jedem gewünschten Zeitpunkt aus Visio heraus ausführen lassen um eine langfristige Lösung des „Konsistenzproblems“ zu gewährleisten.

3.2 Konzeptidee

Die Entwicklung der Konsistenzprüfung bezieht sich auf den Vergleich von zwei Systemmodellen, die in zwei verschiedenen Visio- Zeichenblättern gespeichert sind. Die Namen der Systemkomponenten werden in den Visio- Shapes gespeichert. In Microsoft Visio gibt es keine Möglichkeit die Daten der einzelnen Shapes zu vergleichen. Von daher muss ein Weg gefunden werden die Daten aus den Visio- Shapes zu exportieren und sie in einem anderen Programm vergleichen zu lassen. Die einzige Möglichkeit Daten aus Visio- Shapes zu extrahieren ist der Datenbankexport- Assistenten. Mit ihm lassen sich die Shape-Daten in eine Datenbank exportieren.

Da die in dieser Bachelorarbeit entworfene Konsistenzprüfung auf mehreren Computern installiert werden soll, ist es von Vorteil, wenn keine weiteren Programme installiert werden müssen um die Installation der Software nicht allzu umfangreich zu gestalten. Von daher bietet es sich an als Datenbank Microsoft Access zu verwenden, da diese, genau wie Visio, in dem Microsoft Office- Paket enthalten und dementsprechend ebenfalls auf den Computern installiert ist.

Aus Access lassen sich die Daten wiederum in Microsoft Excel exportieren. In Excel können die Daten mithilfe verschiedene Funktionen verglichen sowie farblich illustriert werden. Die finale Excel Datei enthält dann die Namen der Systemkomponenten der verschiedenen Systemmodelle. Falls ein Systemkomponentenname aus einem Systemmodell mit einem anderem konsistent ist, wird dieser farblich markiert. Ist dieser Name nicht konsistent, wird er nicht markiert.

Die Programmierung findet in der Skriptsprache VBA statt, die sich in jedem der drei beteiligten Office- Anwendungen (Visio, Access, Excel) ausführen lässt. Es wird mit der Version Microsoft Office 2016 gearbeitet.

3.3 Datenexport aus Visio zu Access

In Microsoft Visio wird der Datenbank- Assistent eingerichtet um die Shape- Daten nach Access zu exportieren. Dafür muss zunächst eine Access- Datenbank erstellt werden. Anschließend werden die Shape- Daten gemäß den Schritten in Kapitel 4 exportiert. Ebenfalls ist die Ausführung des Datenbankexport- Assistenten per Rechtsklick sehr benutzerfreundlich. [Mic19]

3.4 Makroprogrammierung in Access

Aus Microsoft Access sollen die Shape- Daten, welche nun als Access- Tabellen vorliegen, nach Microsoft Excel exportiert werden. Dieses lässt sich manuell realisieren, indem man die einzelnen Tabellen direkt exportiert.

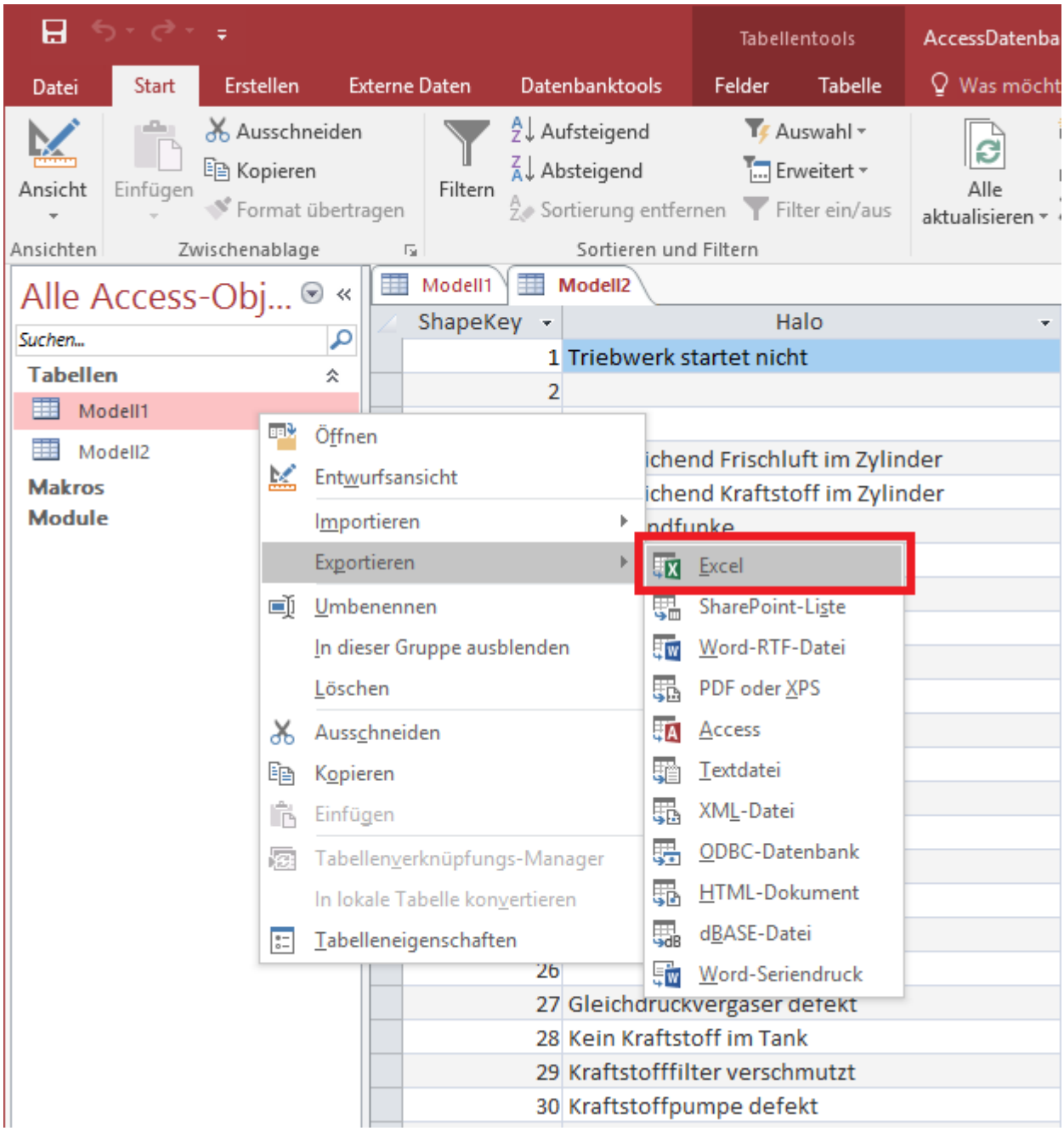

Abbildung 2: Manuelle Exportierung aus Access zu Excel

Diese Lösung ist allerdings nicht benutzerfreundlich und lässt sich auch nicht automatisieren. Von daher wird ein Makro in VBA geschrieben, welches den Export automatisch ausführt.

Die folgende Abbildung zeigt das Access- Makro „Modul2“.

```
Option Compare Database

Public Function ExcelDatenAusführen() As String

Dim dbTable As String
Dim dbTable2 As String


Dim xlWorksheetPath As String
Dim xlWorksheetPath2 As String

'Modell1 in Excel erstellen

xlWorksheetPath = "Z:\Bachelorarbeit\Konsistenzprüfung\"
xlWorksheetPath = xlWorksheetPath & "Modell1.xls"
dbTable = "Modell1"
DoCmd.TransferSpreadsheet transfertype:=acExport, spreadsheettype:=acSpreadsheetTypeExcel9, tablename:=dbTable, FileName:=xlWorksheetPath, hasfieldnames:=True

'Modell2 in Excel erstellen

xlWorksheetPath2 = "Z:\Bachelorarbeit\Konsistenzprüfung\"
xlWorksheetPath2 = xlWorksheetPath2 & "Modell2.xls"
dbTable2 = "Modell2"
DoCmd.TransferSpreadsheet transfertype:=acExport, spreadsheettype:=acSpreadsheetTypeExcel9, tablename:=dbTable2, FileName:=xlWorksheetPath2, hasfieldnames:=True


'ExcelMakroausführen

Dim xlApp As Object
Dim sFile As String

sFile = "Z:\Bachelorarbeit\Konsistenzprüfung\Excel\ExcelDatenbank.xlsm"
Set xlApp = CreateObject("Excel.Application")
xlApp.Workbooks.Open sFile
xlApp.Run "Mehrere_Dateien_einlesen"
xlApp.Workbooks.Close

Set xlApp = Nothing

'Datei öffnen

Dim oAppXL As Excel.Application
   Set oAppXL = CreateObject("Excel.Application")
   oAppXL.Workbooks.Open "Z:\Bachelorarbeit\Konsistenzprüfung\Excel\ExcelDatenbank.xlsm"
   oAppXL.Visible = True
   Set oAppXL = Nothing


End Function
```

Abbildung 3: Access- Makro „Modul2"

In diesem Makro werden zunächst zwei Excel Dateien erstellt die je eine der beiden Access- Tabellen beinhalten. Sie werden im Ordner ***Konsistenzprüfung*** abgelegt und als Typ: *„Excel 97-2003“* gespeichert. Ihre Namen sind *Modell1* und *Modell2*. Die Auswahl des Excel Typs erfolgt durch die Funktion: *“spreadsheettype:=acSpreadsheetTypeExcel19“.* Diese gestattet auch die Excel Dateien im neuerem Typ *„Excel 2016“* zu speichern. Werden die Excel Dateien allerdings in diesem neueren Format gespeichert, treten Schwierigkeiten in der Excel-Makroprogrammierung auf. Daher wird der Typ: *„Excel 97-2003“* ausgewählt.

Nach dem Erstellen der Excel Dateien wird das Excel Makro: *„Mehrere_Dateien_einlesen“* ausgeführt. Dieses Makro wird im Abschnitt 5.4 ausführlich beschrieben. Anschließend wird die Datei *ExcelDatenbank*, aus dem Ordner ***Konsistenzprüfung*** geöffnet.

3.5 Makroprogrammierung in Excel

In Excel wird einzig die Sub: *„Mehrere_Dateien_einlesen“* ausgeführt, die in der Excel Makrodatei „Modul1“ ist. Die zu vergleichenden Shape- Daten sind zum jetzigem Entwicklungszeitpunkt in den Excel Dateien *Modell1* und *Modell2* gespeichert. Um diese in Excel vergleichen zu können, müssen sie in einer Excel Datei in einer Tabelle vorliegen. Dafür werden sie in die Datei *ExcelDatenbank* eingefügt.

```
Sub Mehrere_Dateien_einlesen()


    'Hier werden in der ExcelDatenbank die Tabellen Modell1 und Modell2 erstellt

    Dim strFile As String

    strPath = "Z:\Bachelorarbeit\Konsistenzprüfung\"
    strExt = "*.*"

    If strPath = "" Then
        Exit Sub
    Else

     strFile = Dir(strPath & strExt)

            Do While Len(strFile) > 0
            Workbooks.Open Filename:=strPath & strFile

                letztezelle = Workbooks(strFile).Worksheets(1).Range("B1").SpecialCells(xlCellTypeLastCell).Address
                ThisWorkbook.Worksheets.Add.Name = strFile
                ThisWorkbook.Worksheets(strFile).Range("B1:" & letztezelle).Value = _
                Workbooks(strFile).Worksheets(1).Range("B1:" & letztezelle).Value



            Workbooks(strFile).Close False  ' Arbeitsmappe wird geschlossen
            strFile = Dir()                 ' nächste Datei wird ermittelt
         Loop
    End If
```

Abbildung 4: Excel Makro „Modul1“, Teil 1

Der in Abbildung 18 gezeigt Code kann nicht nur zwei Excel Dateien in die, die das Makro ausführt einfügen, sondern alle die in dem Ordner des Pfades drin sind. Er eignet sich daher gut um in einer möglichen folgenden Arbeite mehrere Modelle miteinander zu vergleichen. Mehr dazu in Kapitel 8.

Anschließend wird eine dritte Tabelle mit dem Namen *Konsistenzprüfung* in *ExcelDatenbank* erstellt. Daraufhin werden die Tabellendaten aus den Tabellen *Modell1* und *Modell2* in die Tabelle *Konsistenzprüfung* eingefügt. *Modell1* wird in die Spalte A, *Modell2* in die Spalte E eingefügt, um längeren Systemkomponentennamen ausreichend Platz zur Verfügung zu stellen.

Die dann nicht mehr benötigten Tabellen *Modell1* und *Modell2* werden anschließend gelöscht.

```
'Dritte Tabelle erstellen: Konsistenzprüfung

Worksheets.Add

ActiveSheet.Name = "Konsistenzprüfung"


'Hier werden die Tabellendaten aus Modell1 und Modell2 in die Tabelle Konsistenzprüfung eingefügt


Worksheets("Modell1.xls").Range("B:B").Copy Destination:=Worksheets("Konsistenzprüfung").Range("A:A")

Worksheets("Modell2.xls").Range("B:B").Copy Destination:=Worksheets("Konsistenzprüfung").Range("E:E")


'Hier werden die Tabellen Modell1 und Modell2 anschließend wieder gelöscht


Application.DisplayAlerts = False

Worksheets("Modell1.xls").Delete
Worksheets("Modell2.xls").Delete

Application.DisplayAlerts = True
```

Abbildung 5: Excel Makro „Modul1", Teil 2

Eigentlich erscheint eine Warnung vor dem Löschen der nicht mehr benötigten Tabellen. Diese werden zwecks automatisierter Ausführung unterbunden.

Die Excel Datei *Konsistenzprüfung* muss die beiden Spalten mit den Shape- Daten der verschiedenen Modelle miteinander vergleichen und diesen Vorgang automatisch ausführen. Dafür wird der Makro- Rekorder verwendet, welcher Excel Ausführungen speichert und als VBA Code im Makro zur Verfügung stellt. Die beiden Spalten werden über die Funktion **Start → Bedingte Formatierung → Regeln zum Hervorheben von Zellen → Gleich…** miteinander verglichen. Die folgende Abbildung zeigt den vom Makro- Rekorder dabei aufgezeichneten und anschließend zur Verfügung gestellten Code. [Exc19]

```
    'Hier werden die unterschiedlichen ShapeNamen farblich markiert

    Range("A1").Select
    ActiveCell.FormulaR1C1 = "Modell1"
    Range("E1").Select
    ActiveCell.FormulaR1C1 = "Modell2"
    Range("A:A,E:E").Select
    Range("E1").Activate
    Selection.FormatConditions.AddUniqueValues
    Selection.FormatConditions(Selection.FormatConditions.Count).SetFirstPriority
    Selection.FormatConditions(1).DupeUnique = xlDuplicate
    With Selection.FormatConditions(1).Font
        .Color = -16752384
        .TintAndShade = 0
    End With
    With Selection.FormatConditions(1).Interior
        .PatternColorIndex = xlAutomatic
        .Color = 13561798
        .TintAndShade = 0
    End With
    Selection.FormatConditions(1).StopIfTrue = False
    Range("I14").Select
    Columns("A:A").ColumnWidth = 40.86
    Columns("E:E").ColumnWidth = 32.43
    Columns("E:E").ColumnWidth = 43.43

    If ThisWorkbook.Saved = False Then
        ThisWorkbook.Save
    End If

End Sub
```

Abbildung 6: Excel Makro „Modul1", Teil 3

Am Ende des Makros wird eine Speicherabfrage unterbunden, um eine automatisierte Ausführung zu gewährleisten.

3.6 Microsoft Access Makro (*MAM*) einbinden

Zum jetzigem Entwicklungsstandpunkt funktioniert die Konsistenzprüfung. Allerdings müssen die Makros in Access und Excel noch manuell ausgeführt werden. Daher wird die Excel Sub: *„Mehrere_Dateien_einlesen“*, die in der Excel Makrodatei „Modul1“ ist, aus dem Access Makro heraus ausgeführt, wie in Kapitel 5.3 beschrieben. Dadurch muss nur noch das Access Makro manuell ausgeführt werden um die

Konsistenzprüfung zu starten. Ziel ist es, dieses Access Makro aus Visio heraus starten zu lassen um eine automatisierte Konsistenzprüfung zu gewährleisten.

In Microsoft Access werden die VBA Makrodateien in Modulen gespeichert, genauso wie es in Excel der Fall ist. Es gibt allerdings auch Microsoft Access Makros, die im folgendem mit *„MAM“* beschrieben werden um eine Verwechslung zu vermeiden.

Zunächst ist versucht worden das Access Makro „Modul2“ dahingehend zu verändern, dass es sich automatisch ausführt, sobald die Access- Tabellen *Modell1* und *Modell2* geändert werden [The19]. Dies funktioniert allerdings nicht, da durch den Datenbankexport- Assistenten zwar die Tabellen in Access erstellt werden, sich das Programm aber nicht startet. Es gibt die Möglichkeit ein *MAM* zu programmieren, welches Funktionen aus Access Makros ausführt. Diese Aktion heißt *„AusführenCode“* [The19a]. Daher ist der auszuführende Code in dem Access Makro „Modul2“ als Funktion *„ExcelDatenAusführen“* gespeichert. (VGL Abbildung 17) *MAMs* führen sich automatisch beim Starten von Microsoft Access aus, wenn sie unter dem Namen *„AutoExec“* gespeichert sind.

Das *MAM* mit dem Namen *„AutoExec“* führt automatisch beim Starten von Microsoft Access die Funktion *„ExceldatenAusführen“*, die in dem Access Makro „Modul2“ gespeichert ist, aus. Es öffnet sich die Excel Datei *ExcelDatenbank* mit der Excel Tabelle *Konsistenzprüfung*. [Pro19]

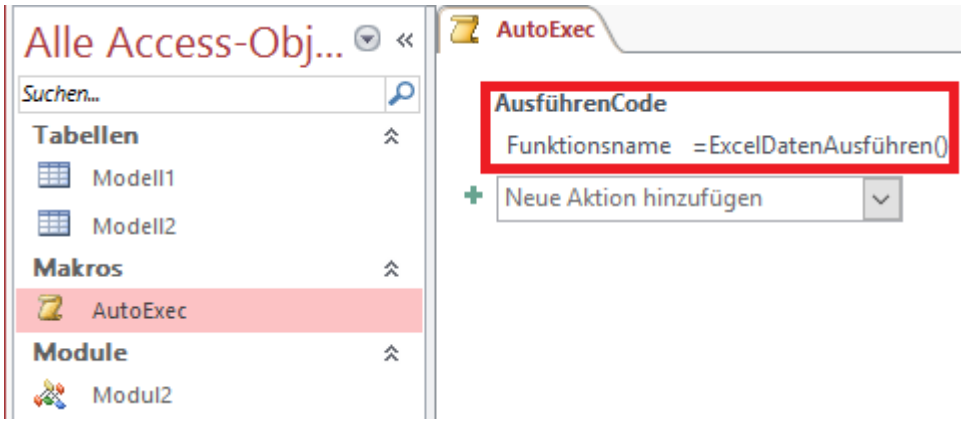

Abbildung 7: Microsoft Access, *MAM*

3.7 Makroprogrammierung in Visio

Das Visio Makro öffnet die Access Datei *AccessDatenbank* und schließt sie wieder. Da das *MAM „AutoExec“* die Funktion *„ExcelDatenAusführen“* ausführt, wird die Konsistenzprüfung durchgeführt und es öffnet sich die Excel Datei *ExcelDatenbank* mit der Excel Tabelle *Konsistenzprüfung*. Der Anwender bemerkt die Öffnung der *AccessDatenbank* nicht, da sie gleich wieder geschlossen wird.

```
Sub Hola()

Dim App As New Access.Application

App.OpenCurrentDatabase "Z:\Bachelorarbeit\Konsistenzprüfung\Access\AccessDatenbank.accdb", True

End Sub
```

Abbildung 8: Visio Makro

4 Implementierungsanleitung

4.1 Dateienanordnung und grundlegende Hinweise

Die hier zur Verfügung gestellte Konsistenzprüfung geht davon aus, dass zwei Visio Zeichenblätter auf Konsistenz hin geprüft werden. Dabei ist es egal, ob sie in einer oder in verschiedenen Visio Dateien sind. Im Rahmen einer Folgearbeit ist es sicherlich möglich mehr als zwei Zeichenblätter zu vergleichen, siehe Kapitel 8. Das ist mit dieser Konsistenzprüfung allerdings nicht möglich. Achten Sie daher unbedingt darauf stets nur zwei Zeichenblätter auf einmal auf Konsistenz hin zu prüfen! Des Weiteren ist die Konsistenzprüfung auf Microsoft Office 2016 unter dem Betriebssystem Windows ausgelegt. Eine problemlose Ausführung der Konsistenzprüfung unter einem anderen Betriebssystem oder einer anderen Microsoft Office Version, kann daher nicht gewährleistet werden. Der Ordner *Konsistenzprüfung* befindet sich auf der beigefügten CD.

Der Ordner ***Konsistenzprüfung*** ist der einzige, den Sie für die Ausführung der Konsistenzprüfung benötigen. Er kann an einem beliebigen Ort auf dem Computer gespeichert werden. Die Dateien in dem Ordner dürfen nicht geändert, umbenannt oder verschoben werden. Falls sie zum Beispiel die Datei *AccessDatenbank* und *ExcelDatenbank* in einen Ordner legen führt das zu einer Fehlermeldung. Ebenso dürfen die Excel- Dateien *Modell1* und *Modell2* nur geändert werden, wenn der Code entsprechend angeglichen wird. Diese beiden Excel Dateien sind vom Typ: *„Excel 97-2003“.* Dies ist gewollt, hat Kompatibilitätsgründe und ist nicht weiter zu beachten. Die Ergebnis Excel Datei öffnet sich in Excel 2016.

Falls Sie in Microsoft Visio, Excel oder Access einen Reiter nicht finden, den Sie für die Ausführung der beschriebenen Schritte brauchen, dann gehen Sie auf: **Datei** → **Optionen** → **Menüband anpassen**. Anschließend wählen Sie in dem Auswahlmenü: **Menüband anpassen** das Feld **Hauptregisterkarten** aus. Jetzt können Sie die entsprechenden Reiter auswählen, die Ihnen fehlen.

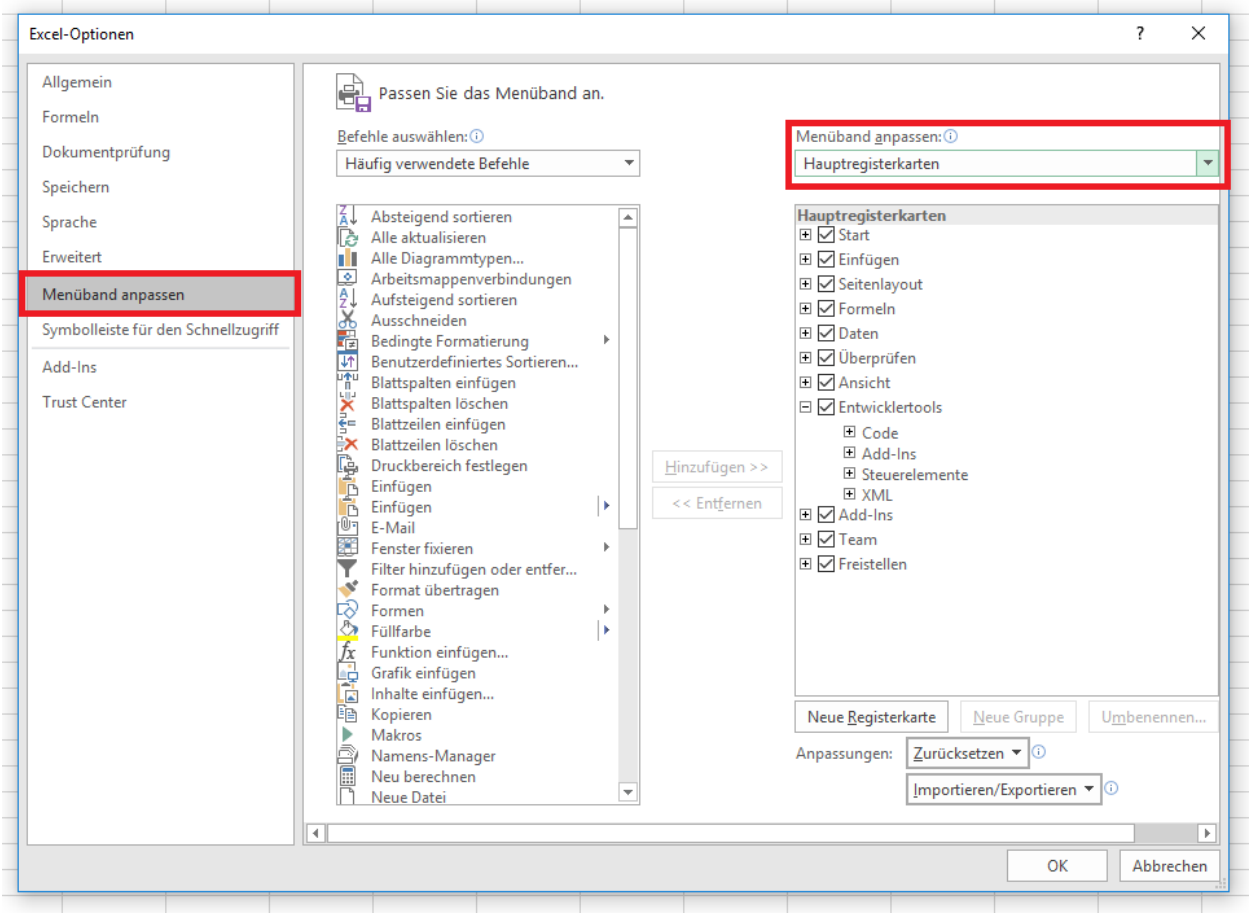

Abbildung 9: Menüband anpassen in Microsoft Excel

4.2 Implementierung in Microsoft Visio

4.2.1 Grundlegende Bedingungen

Bevor Sie mit der Implementierung beginnen sind drei Bedingungen zu beachten, die für die Konsistenzprüfung notwendig sind. Die Visio Datei muss als Dateityp*: „Visio-Zeichnung mit Makros“* gespeichert werden.

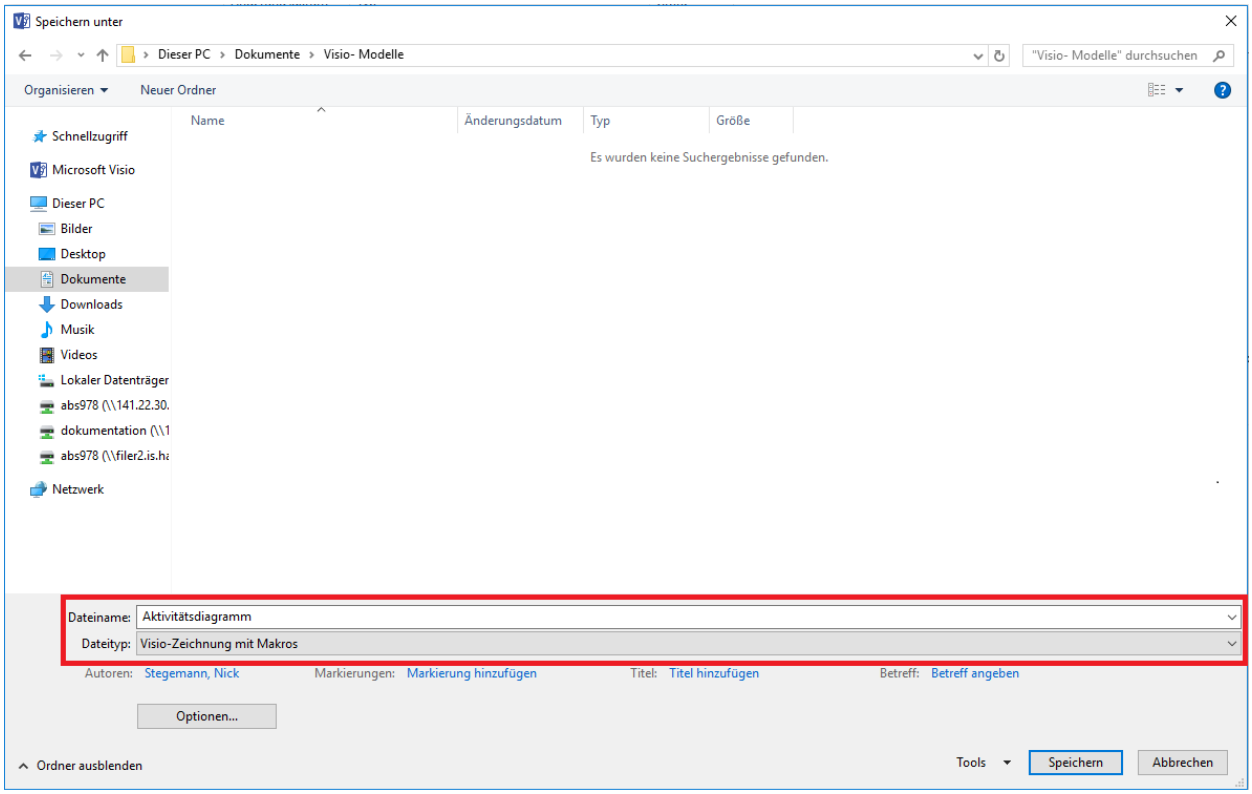

Abbildung 10: Microsoft Visio Dateien im erforderlichen Format speichern

Des Weiteren muss von Visio aus ein Verweis zu Access und Excel eingerichtet werden, damit Makro Befehle und Funktionen in VBA auf Access und Excel Dateien zugreifen können. Dafür gehen Sie auf: **Entwicklertools** → **Visual Basic**. Es öffnet sich der VBA Editor. Jetzt gehen Sie auf: **Extras** → **Verweise** und aktivieren die folgenden sechs Verweise:

„Visual Basic For Applications“
„Microsoft Visio 16.0 Type Library“
„OLE Automation“
„Microsoft Office 16.0 Object Library“
„Microsoft Access 16.0 Object Library“
„Microsoft Excel 16.0 Object Library“

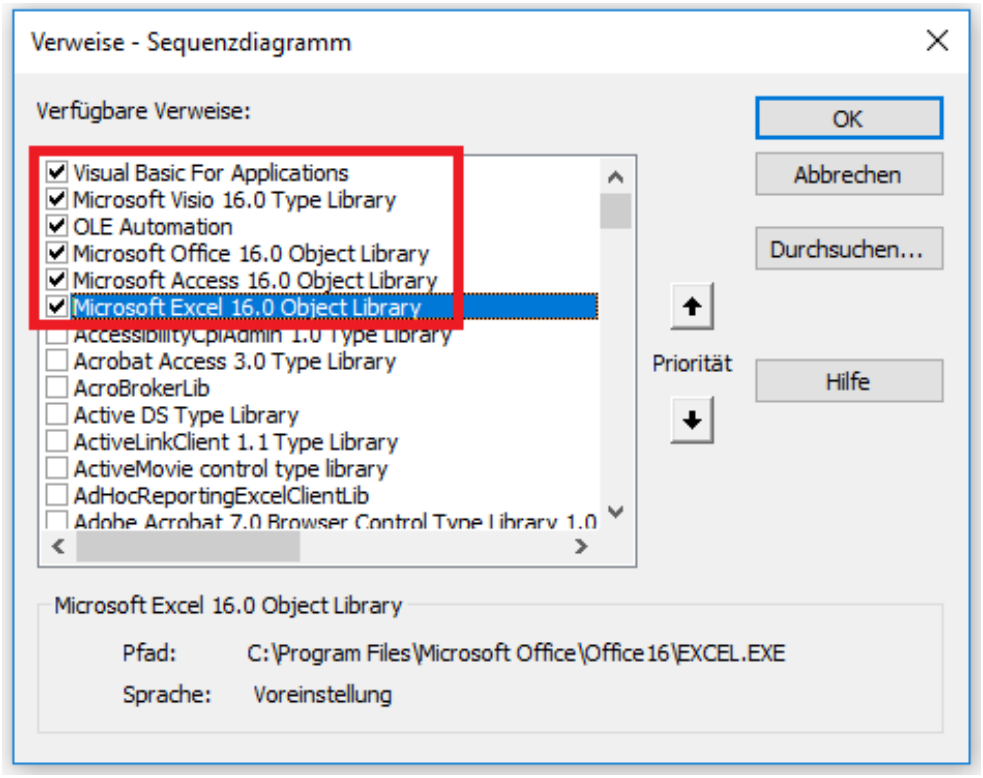

Abbildung 11: Visio Verweise einrichten

Falls Sie zwei Zeichenblätter von verschiedenen Visio Dateien auf Konsistenz hin prüfen, müssen Sie die Verweise in beiden Visio Dateien einrichten. Die nachfolgenden Schritte führen Sie dann ebenfalls in beiden Visio Dateien aus. In einigen Schritten gibt es Änderungen zwischen der Implementierung des ersten und des zweiten Zeichenblattes. Diese sind gekennzeichnet.

4.2.2 Makroimplementierung in Visio

In Microsoft Visio gehen sie im VBA Editor in der **Projekt Seitenleiste** Ihrer Zeichnung auf **Visio Objekte → ThisDocument (*„Ihr Dateiname“*)**.

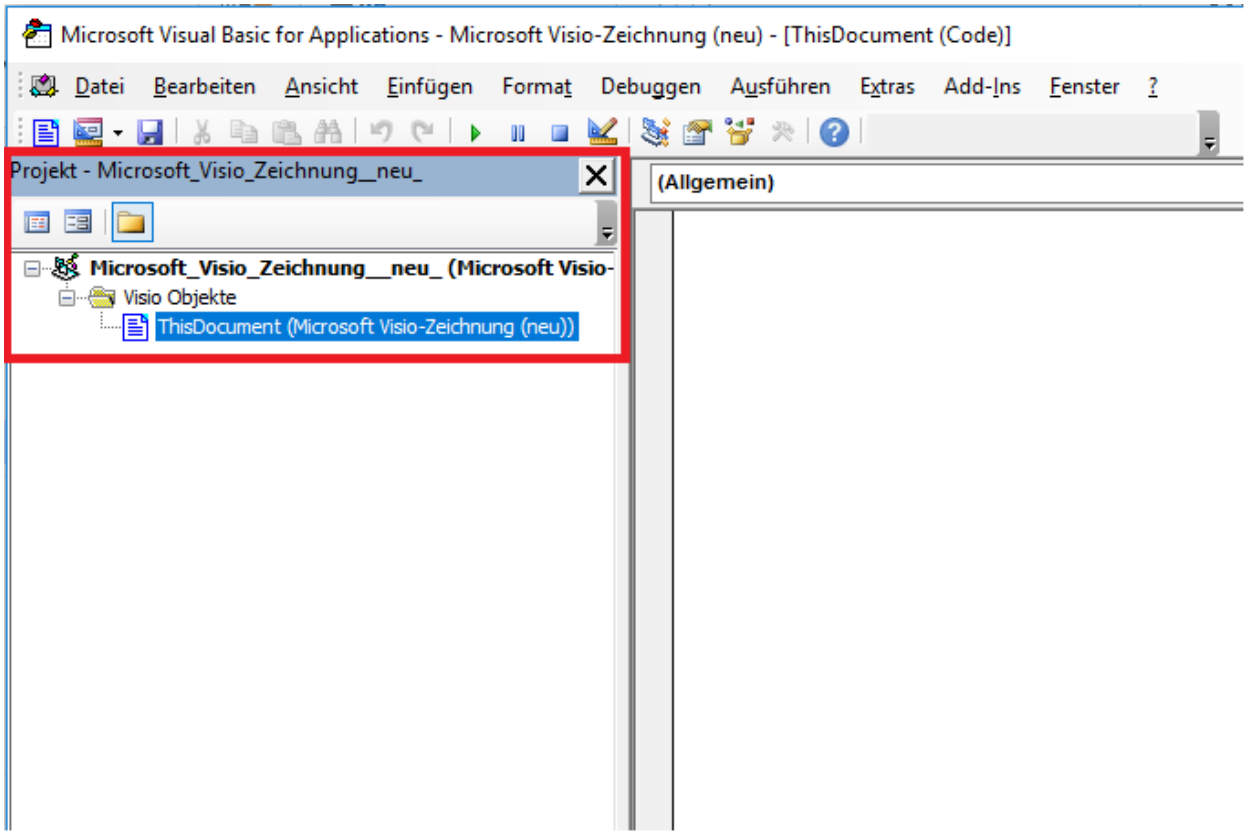

Abbildung 12: Visio Makroprogrammierung

Geben Sie in der Programmierfläche folgenden Code ein:

```
Sub Hola()

Dim App As New Access.Application

App.OpenCurrentDatabase "(„Ihr Dateipfad zum Ordner Konsistenzprüfung")\Konsistenzprüfung\Access\AccessDatenbank.accdb", True

End Sub
```

Die folgende Abbildung zeigt ein Beispiel dieses Codes.

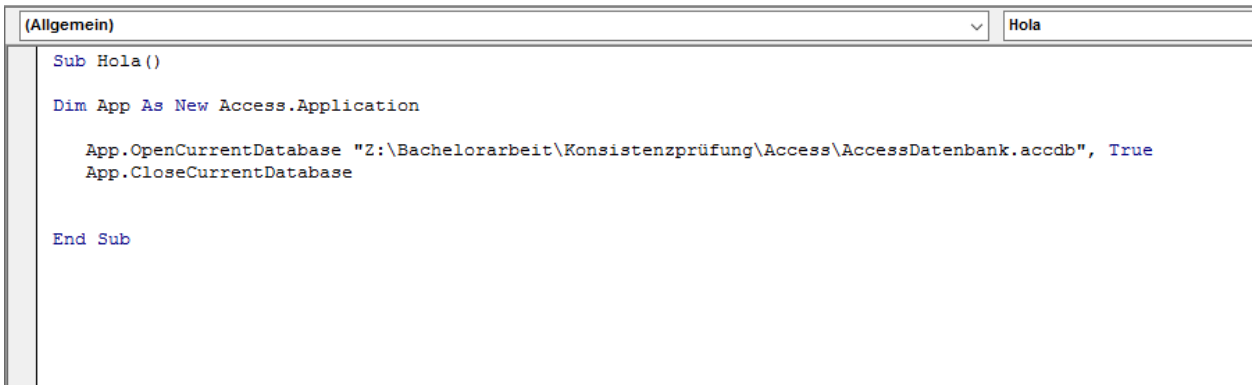

Abbildung 13: Visio Makro Programmierbeispiel

Gehen Sie anschließend auf **Entwicklertools** → **Makros**. Wählen Sie das Makro *ThisDocument.Hola* aus und klicken Sie auf **Optionen**. Im Feld **Tastenkombinationen** tragen sie den Buchstaben „k“ ein. Klicken Sie auf **Ok** und anschließend auf **Abbrechen**.

4.2.3 Datenbankexport- Assistent

Sie müssen den Datenbankexport- Assistenten für jedes Zeichenblatt ausführen. Dabei ist es egal, ob die Zeichenblätter in einer oder in zwei verschiedenen Visio Dateien sind!

In der normalen Microsoft Visio Umgebung, nicht im VBA Editor, gehen Sie auf **Ansicht** → **Add- Ons** → **Visio- Extras** → **Datenbankexport- Assistent**. Klicken Sie im ersten erscheinenden Fenster auf **Weiter**. Im zweitem Fenster wählen sie im oberem Auswahlmenü ihre Zeichnung und im unterem ihr Zeichenblatt aus. Wenn sie zwei Zeichenblätter aus einer Visio Datei auf Konsistenz hin prüfen, wählen Sie hier bei der zweiten Ausführung des Datenbankexport- Assistenten das andere Zeichenblatt aus.

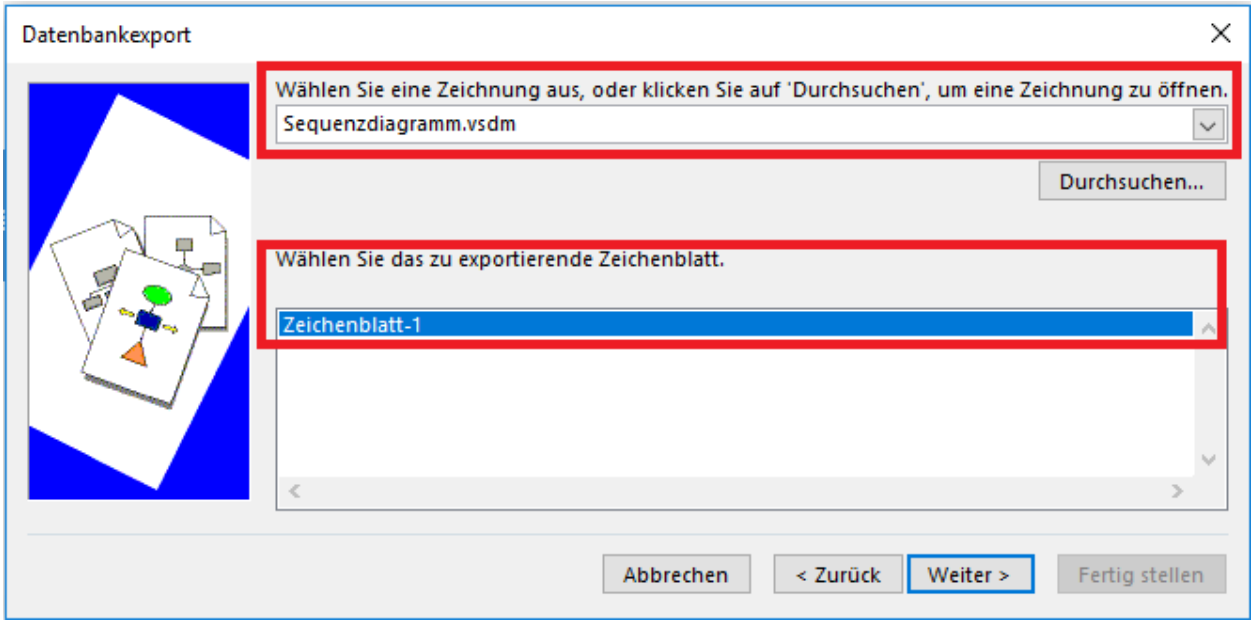

Abbildung 14: Datenbankexport- Assistent, zweites Fenster

Klicken Sie auf **Weiter**. Sie werden danach gefragt welche Shapes Sie übertragen wollen. Wenn sie den Menüpunkt: **Ausgewählte Shapes auf dem Zeichenblatt** aktivieren können Sie, wenn sie anschließend auf den Button **Shapes auswählen** klicken, die einzelnen zu vergleichenden Shapes auswählen. Im Regelfall werden alle Shapes verglichen, daher wird für diese Anleitung der Menüpunkt**: Alle Shapes auf dem Zeichenblatt** ausgewählt. Nachdem Sie Ihre Auswahl getroffen haben, klicken sie auf **Weiter**.

Im nächstem Fenster werden Sie danach gefragt welche Visio- Daten sie exportieren möchten. Diese Konsistenzprüfung arbeitet nur mit dem **Shape.Text**. Ein Auszug über die Möglichkeiten zum Export von weiteren Daten befindet sich in Kapitel 8. Fügen Sie daher nur den **Shape.Text** hinzu. Die folgende Abbildung zeigt, wie Ihr Fenster aussehen sollte.

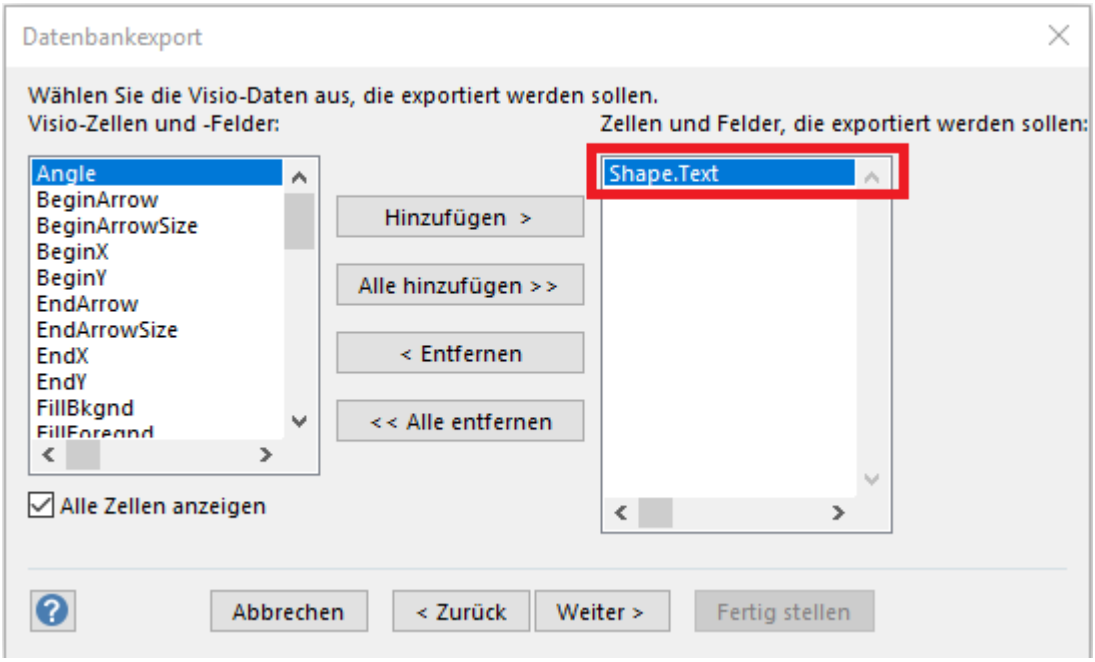

Abbildung 15: Datenbankexport- Assistent, **Shape.Text** hinzufügen

Wenn sie anschließend auf **Weiter** klicken, können Sie im nächsten Fenster die ODBC-Datenquelle auswählen, in die Sie exportieren möchten. Wählen Sie die **MS Access Database** aus und klicken Sie auf **Weiter**.

Es öffnet sich das „Datenbank auswählen“ Fenster. Hier wählen Sie die *AccessDatenbank* aus dem Ordner ***Konsistenzprüfung*** aus und drücken anschließend auf den Button **OK**.

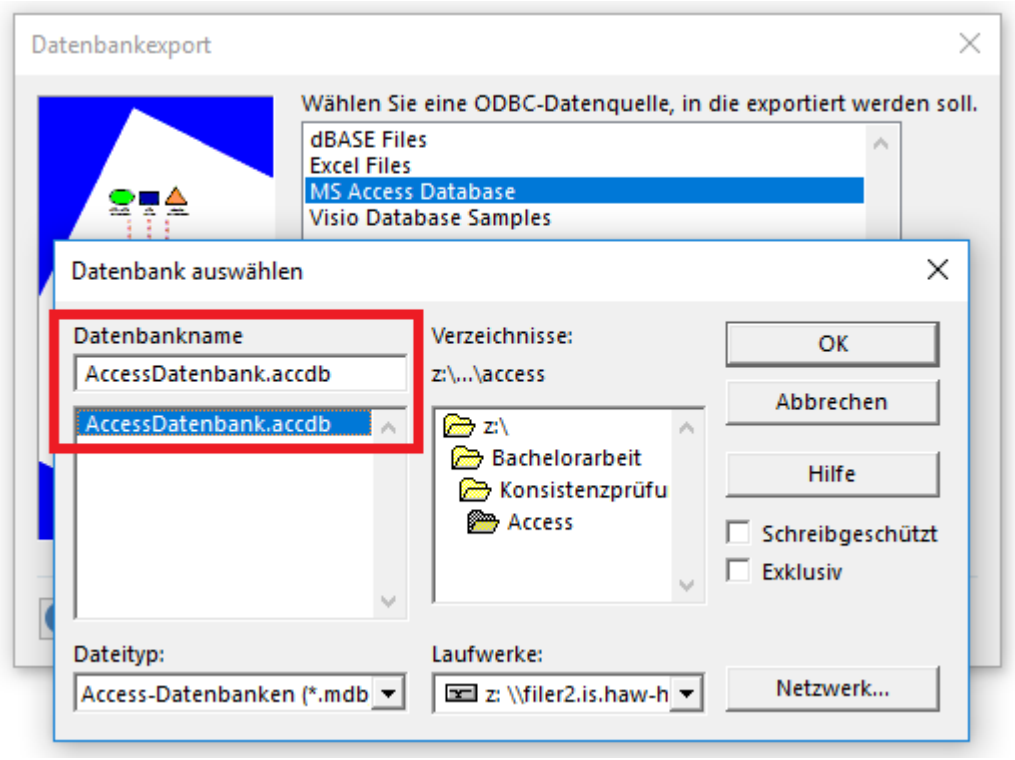

Abbildung 16: Datenbankexport- Assistent, *AccessDatenbank* auswählen

Im nächsten Fenster sollte im Auswahlmenü **Datenbank** der Pfad der *AccessDatenbank* angegeben sein. Ist dies nicht der Fall, geben Sie den Pfad manuell ein. Im Auswahlmenü **Tabellenname** geben Sie für die Implementierung des ersten Zeichenblattes *Modell1* und für die zweite *Modell2 ein*. Alternativ können Sie die Reihenfolge auch vertauschen. Wichtig ist nur, dass die beiden verschiedenen Zeichenblätter nicht mit demselben **Tabellennamen** implementiert werden. Im Auswahlmenü **Schlüsselfeld** sollte automatisch *ShapeKey* stehen und im Auswahlmenü Schlüsseltyp *ShapeID*. Ist dies nicht der Fall, geben Sie den die Daten manuell ein. Den Auswahlpunkt **Schlüsselfeld als Primärschlüssel verwenden für Tabelle verwenden** lassen sie deaktiviert.

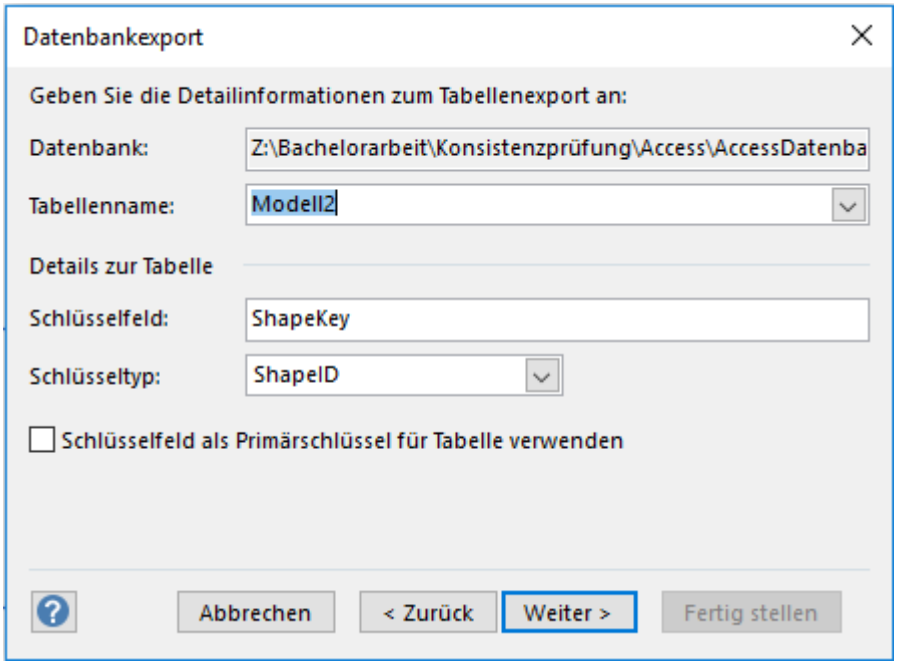

Abbildung 17: Datenbankexport- Assistent, Detailinformationen

Klicken Sie auf **Weiter**. Es erscheint die Fehlermeldung: „Fehler beim Export der Datenbank“, die abfragt ob die bestehende Tabelle ersetzt werden soll. Klicken Sie bei Erscheinen dieser Fehlermeldung immer auf **Ja**.

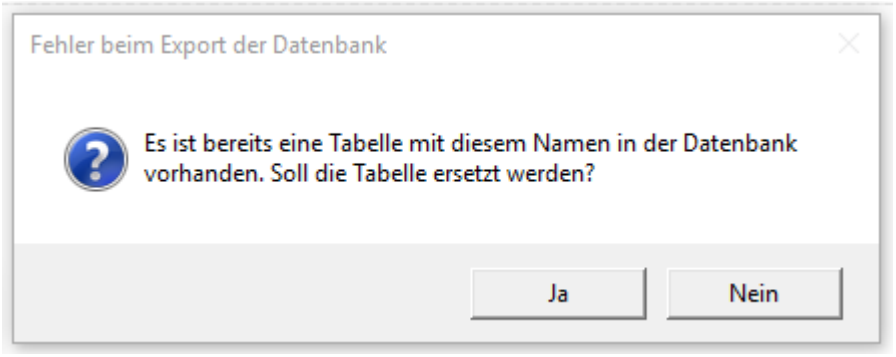

Abbildung 18: Fehler beim Export der Datenbank

Im nächstem Fenster sollen folgende Auswahlmenüs folgende Daten zugewiesen werden:

Daten auswerten als: *Wert* **Feldname:** *Halo* **Feldtyp:** *LONGCHAR*

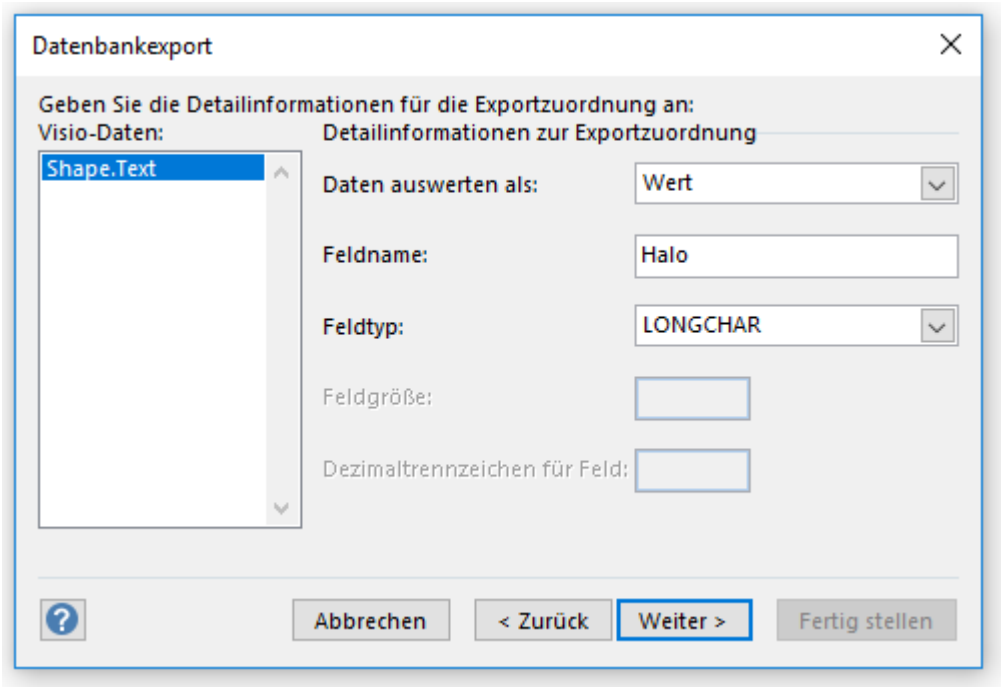

Abbildung 19: Datenbankexport- Assistent, Datenzuweisung

Nachdem Sie auf **Weiter** geklickt haben, werden Sie im nächsten Fenster gefragt, ob Sie die **Kontextmenüaktion ‚Export' in das Zeichenblatt** aufnehmen wollen. Diese Funktion aktivieren Sie. Mithilfe dieser lässt sich der Datenbankexport- Assistent in das Rechtsklickmenü in der Zeichnungsansicht einbinden und ermöglicht so eine benutzerfreundliche Bedienung. Klicken Sie anschließend auf **Weiter**.

Im letzten Fenster erscheint eine Übersicht über die ausgewählten Auswahlkriterien. Klicken Sie auf **Fertig stellen**. Wiederholen Sie anschließend die Implementierung mit der zweiten Visio Datei. Generell müssen Sie den Datenbankexport- Assistenten lediglich einmal pro neu erstelltem Visio- Zeichenblatt implementieren.

4.3 Implementierung in Access

4.3.1 Grundlegende Bedingung

Sie müssen von Access einen Verweis zu Excel einrichten, damit Makro Befehle und Funktionen in VBA auf Excel Dateien zugreifen können. Dafür gehen Sie auf: **Datenbanktools** → **Visual Basic**. Es öffnet sich der VBA Editor. Jetzt gehen Sie auf: **Extras** → **Verweise** und aktivieren die folgenden sechs Verweise:

„Visual Basic For Applications“
„Microsoft Access 16.0 Object Library“
„OLE Automation“
„Microsoft Office 16.0 Access database engine Object Library“
„Microsoft Visual Basic for Applications Extensibility 5.3“
„Microsoft Excel 16.0 Object Library“

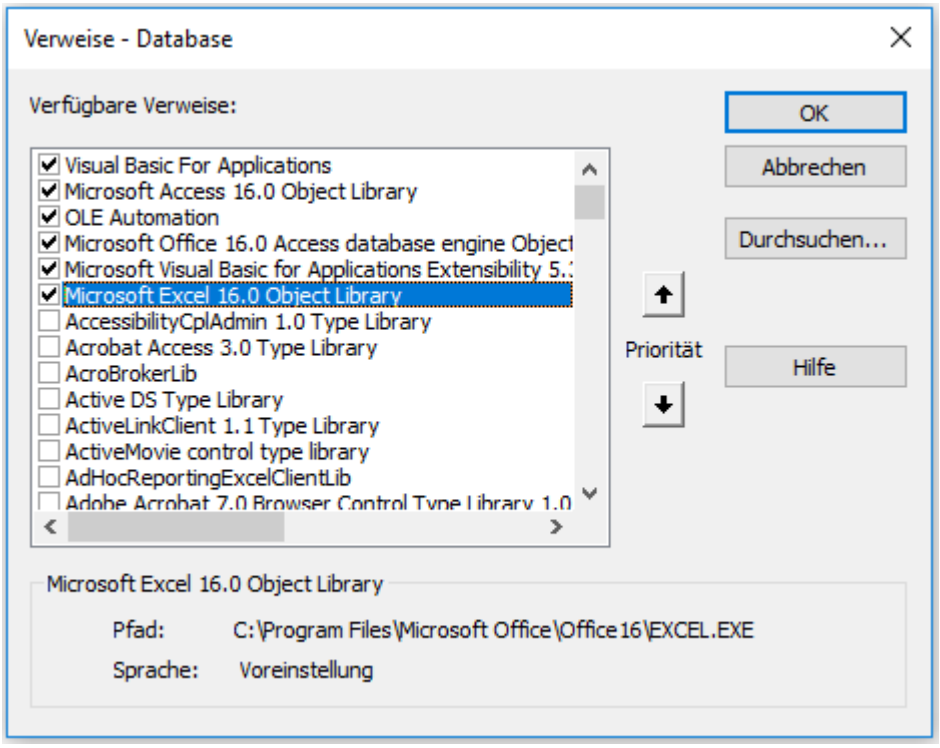

Abbildung 20: Access Verweise

4.3.2 Makroimplementierung in Access

Sie müssen in dem Makro „Modul2“ der Datei *AccessDatenbank* den Pfad zu dem Ordner ***Konsistenzprüfung*** angeben. Dies tun sie in den in der folgenden Abbildung markierten Zeilen.

```
Option Compare Database

Public Function ExcelDatenAusführen() As String

Dim dbTable As String
Dim dbTable2 As String


Dim xlWorksheetPath As String
Dim xlWorksheetPath2 As String

'Modell1 in Excel erstellen

xlWorksheetPath = "Z:\Bachelorarbeit\Konsistenzprüfung\"
xlWorksheetPath = xlWorksheetPath & "Modell1.xls"
dbTable = "Modell1"
DoCmd.TransferSpreadsheet transfertype:=acExport, spreadsheettype:=acSpreadsheetTypeExcel9, tablename:=dbTable, FileName:=xlWorksheetPath, hasfieldnames:=True

'Modell2 in Excel erstellen

xlWorksheetPath2 = "Z:\Bachelorarbeit\Konsistenzprüfung\"
xlWorksheetPath2 = xlWorksheetPath2 & "Modell2.xls"
dbTable2 = "Modell2"
DoCmd.TransferSpreadsheet transfertype:=acExport, spreadsheettype:=acSpreadsheetTypeExcel9, tablename:=dbTable2, FileName:=xlWorksheetPath2, hasfieldnames:=True


'ExcelMakroausführen

Dim xlApp As Object
Dim sFile As String

sFile = "Z:\Bachelorarbeit\Konsistenzprüfung\Excel\ExcelDatenbank.xlsm"
Set xlApp = CreateObject("Excel.Application")
xlApp.Workbooks.Open sFile
xlApp.Run "Mehrere_Dateien_einlesen"
xlApp.Workbooks.Close

Set xlApp = Nothing

'Datei öffnen

Dim oAppXL As Excel.Application
   Set oAppXL = CreateObject("Excel.Application")
   oAppXL.Workbooks.Open "Z:\Bachelorarbeit\Konsistenzprüfung\Excel\ExcelDatenbank.xlsm"
   oAppXL.Visible = True
   Set oAppXL = Nothing


End Function
```

Abbildung 21: Access Makro „Modul2", Pfadänderungen

4.4 Implementierung in Excel

4.4.1 Grundlegende Bedingung

Gehen Sie auf: **Entwicklertools** → **Visual Basic**. Es öffnet sich der VBA Editor. Jetzt gehen Sie auf: **Extras** → **Verweise** und aktivieren die folgenden vier Verweise:

„Visual Basic For Applications“
„Microsoft Excel 16.0 Object Library“
„OLE Automation“
„Microsoft Office 16.0 Object Library“

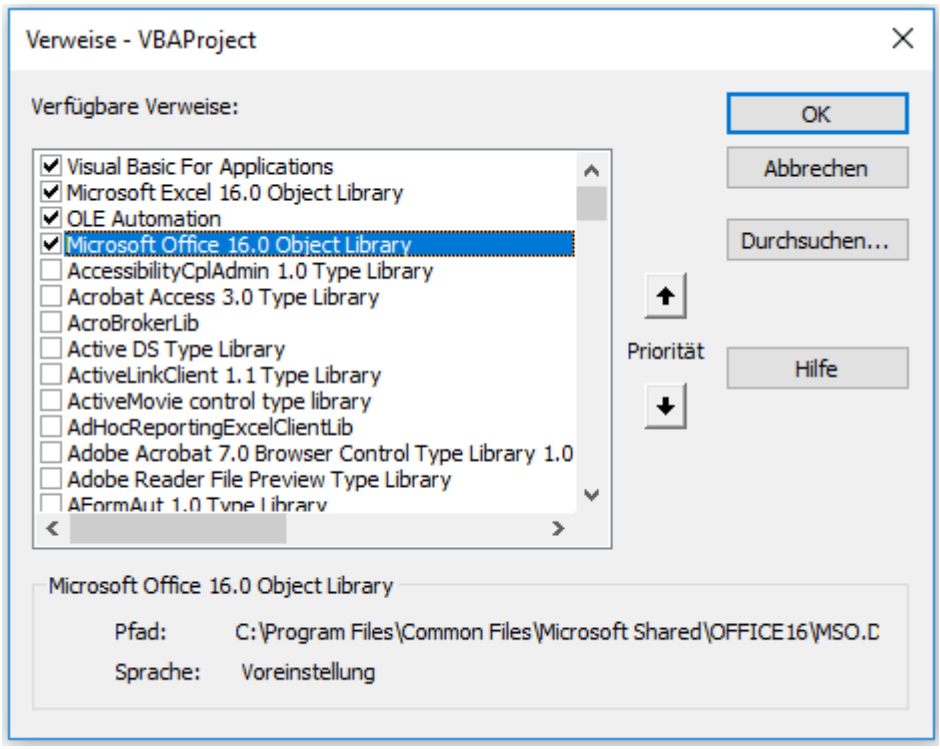

Abbildung 22: Excel Verweise

4.4.2 Makroimplementierung in Excel

Sie müssen in dem Makro „Modul1“ der Datei *ExcelDatenbank* den Pfad zu dem Ordner ***Konsistenzprüfung*** angeben. Dies tun sie in der in der folgenden Abbildung markierten Zeile.

```
Sub Mehrere_Dateien_einlesen()


    'Hier werden in der ExcelDatenbank die Tabellen Modell1 und Modell2 erstellt

    Dim strFile As String

    strPath = "Z:\Bachelorarbeit\Konsistenzprüfung\"
    strExt = "*.*"

    If strPath = "" Then
        Exit Sub
    Else
```

Abbildung 23: Excel Makro „Modul1", Pfadänderungen

5 Ausführung der Konsistenzprüfung

Machen Sie einen Rechtsklick in der Zeichnungsansicht in Microsoft Visio. Klicken Sie anschließend auf **Datenbankexport- Tabelle**.

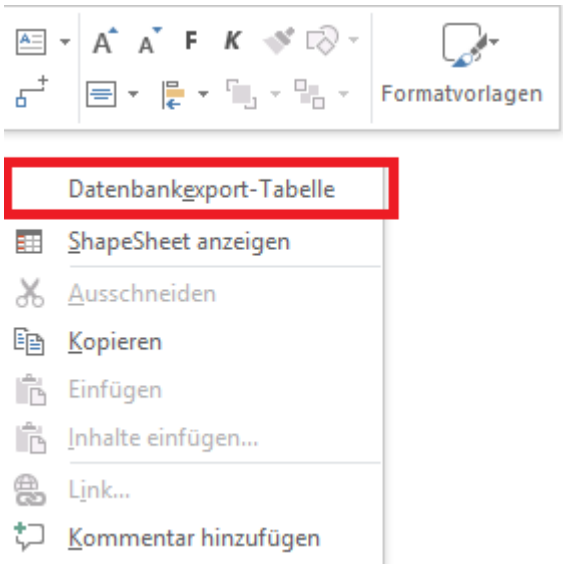

Abbildung 24: Rechtsklick, Datenbankexport- Tabelle

Es erscheint die Fehlermeldung: „Fehler beim Export der Datenbank“ (VGL: Abbildung 12: Fehler beim Export der Datenbank). Klicken Sie auf **Ja**. Haben Sie den Datenbankexport- Assistenten für Ihre Visio Datei eingerichtet und Visio anschließend beendet, erscheint nach einem Neustart von Visio nach der Ausführung von **Datenbankexport- Tabelle** folgende Datenbank- Assistent- Warnung:

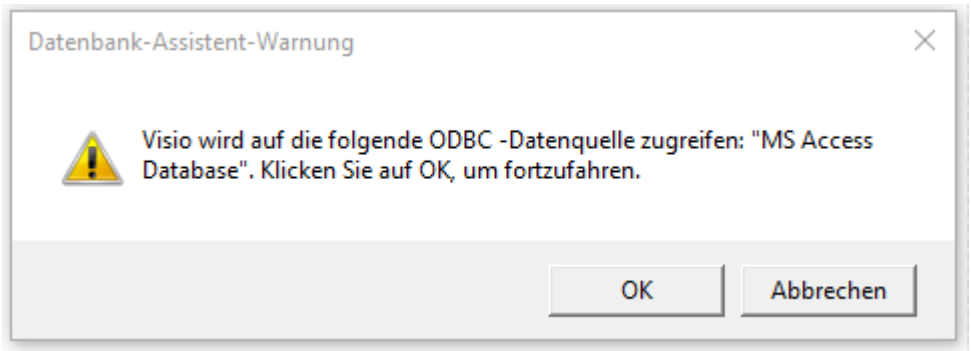

Abbildung 25: Datenbank- Assistent- Warnung

Klicken Sie auf **OK** und geben Sie anschließend den Pfad zur *AccessDatenbank* ein (VGL: Abbildung 10: Datenbankexport- Assistent, *AccessDatenbank* auswählen). Jetzt erscheint die Fehlermeldung: „Fehler beim Export der Datenbank“. Klicken Sie wieder auf **Ja**.

Drücken Sie anschließend in Visio in der Zeichnungsansicht gemeinsam die Tasten „Strg“ und „k“. Es öffnet sich die Excel Datei *ExcelDatenbank* mit der Excel Tabelle *Konsistenzprüfung*.

Sie können die Excel Tabelle *Konsistenzprüfung* jetzt speichern, modifizieren oder anderweitig verwenden. Um die Konsistenzprüfung noch einmal auszuführen müssen Sie die Tabelle *Konsistenzprüfung* löschen, sodass nur noch die leere *Tabelle1* in *ExcelDatenbank* enthalten ist. Speichern sie anschließend die Excel Datei und schließen sie die *ExcelDatenbank*.

6 Verifizierung anhand eines Fallbeispiels

In dem folgenden Fallbeispiel werden eine Fehlerbaumanalyse und ein Sequenzdiagramm auf Konsistenz hin geprüft.

6.1 Fehlerbaumanalyse

Im folgenden Diagramm ist die Fehlerbaumanalyse eines nicht startenden Triebwerkes dargestellt. Die Fehlerbaumanalyse beginnt mit einem einzigen unerwünschten Ereignis (Hauptereignis) und führt zu immer spezifischer werdenden Ereignissen (Zwischenereignisse). Diese können auch in ein oder mehrere Subsysteme unterteilt werden, je nach Komplexität des Systems. Daraus ergibt sich eine Baumstruktur. Am Ende der Fehlerkette steht die nicht funktionierende Komponente des Systems (Grundereignis). [Wik19b] Diese letzte Komponente wird entweder als Kreis (Primärereignis), Raute (nicht weiter entwickeltes Ereignis) oder als Dreieck (Transferereignis) dargestellt. Dabei deuten nach oben hin gerichtete Dreiecke auf eine andere Fehlerbaumanalyse hin, die das Ereignis genauer beschreibt. [Lös19]

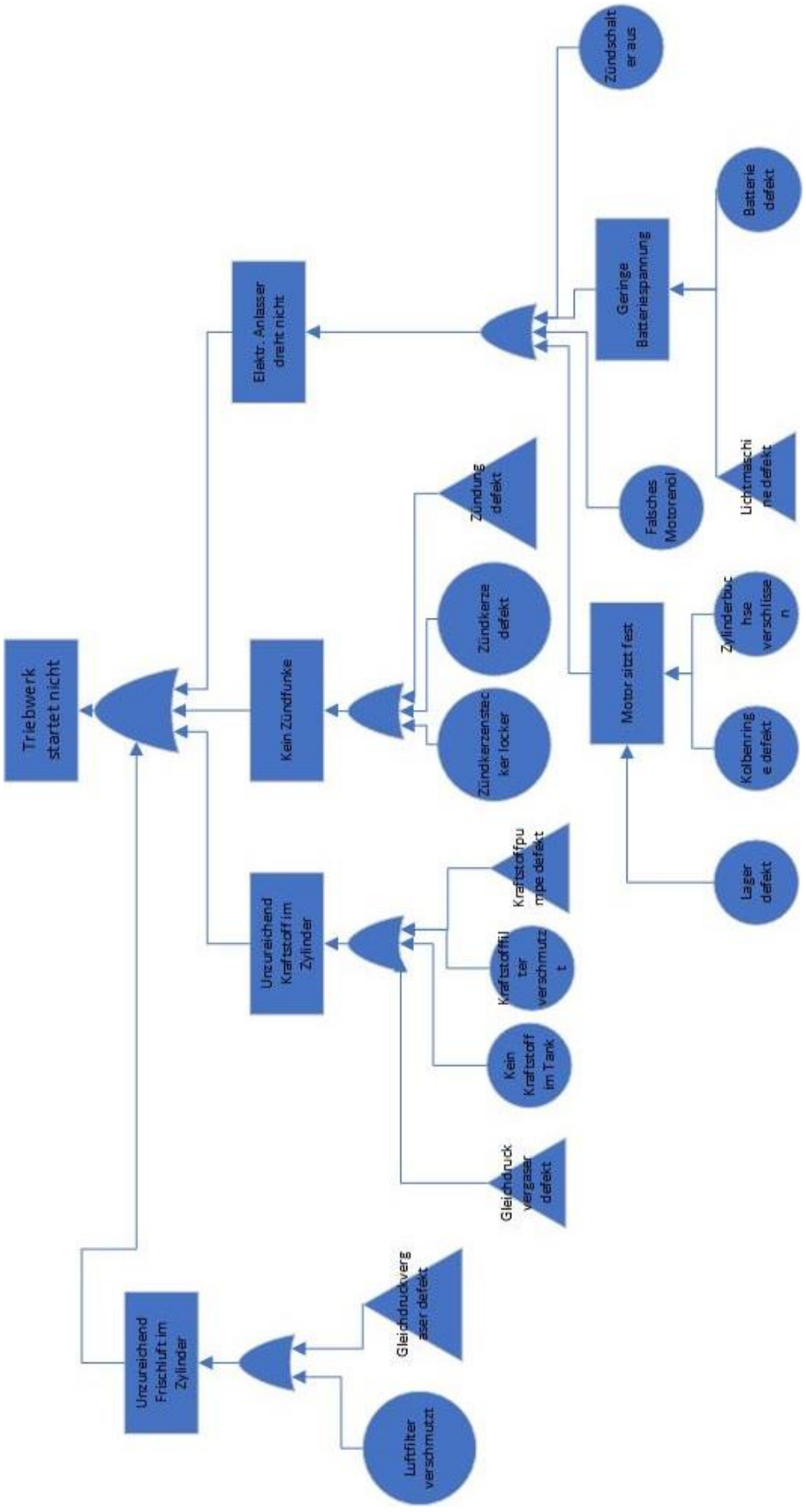

Abbildung 26: Fehlerbaumanalyse – Triebwerk [Bom15]

6.2 Sequenzdiagramm

Das Sequenzdiagramm wird in der SysML Notation den Verhaltensdiagrammen zugeordnet. [Wik19] In diesem Fallbeispiel wird sich auf die Darstellung von einem Entscheidungspfad durch die Fehlerbaumanalyse beschränkt. Um die komplette Fehlerbaumanalyse zu modellieren würde man mehrere Sequenzdiagramme einsetzen.

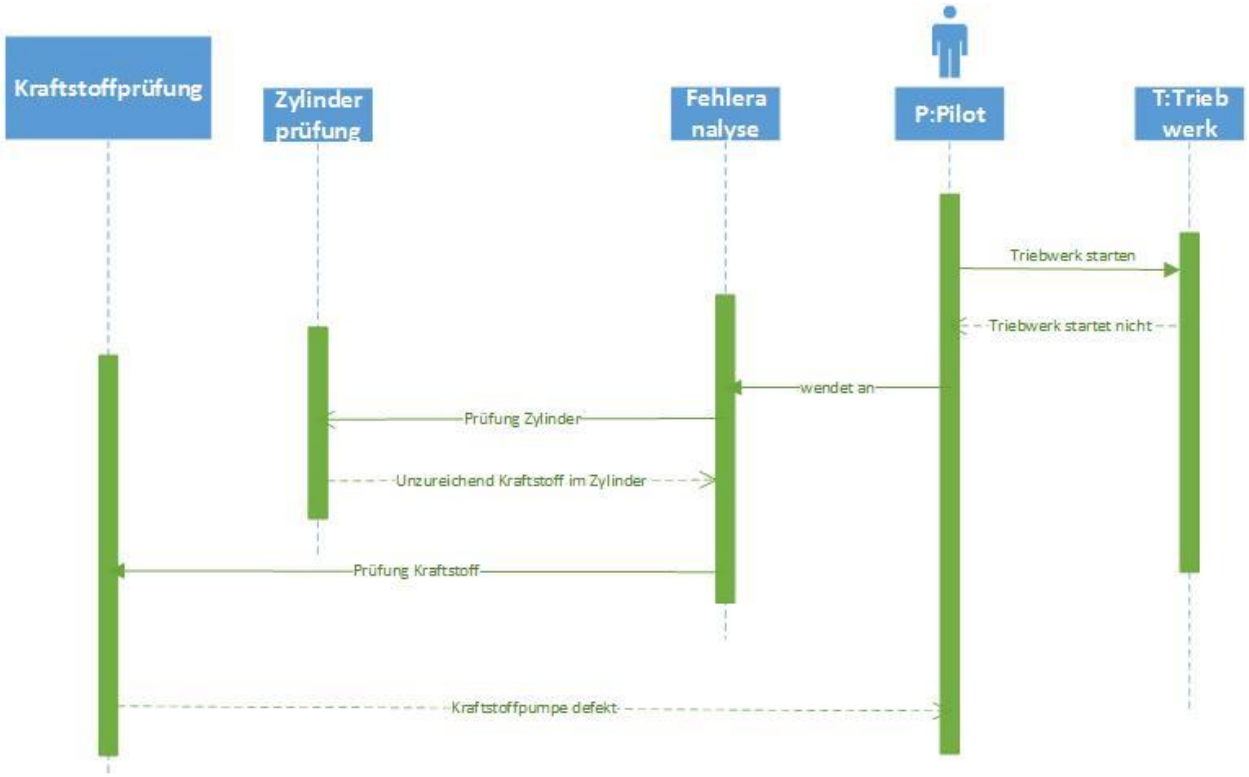

Abbildung 27: Sequenzdiagramm - Entscheidungspfad des Fehlerbaums

Dieses Sequenzdiagramm stellt den Entscheidungspfad bis zum Grundereignis: „Kraftstoffpumpe defekt“ da. Die Fehlerbaumanalyse und das Sequenzdiagramm werden mit der Konsistenzprüfung auf Konsistenz hin geprüft.

6.3 Ergebnis der ersten Konsistenzprüfung

1	Modell1				Modell2
2	Triebwerk startet nicht				T:Triebwerk
3					P:Pilot
4					
5	Unzureichend Frischluft im Zylinder				
6	Unzureichend Kraftstoff im Zylinder				Triebwerk starten
7	Kein Zündfunke				Triebwerk startet nicht
8	Elektr. Anlasser dreht nicht				Fehleranalyse
9					
10					wendet an
11					Zylinderprüfung
12					
13					Prüfung Zylinder
14					Unzureichend Kraftstoff im Zylinder
15	Gleichdruckvergaser defekt				Kraftstoffprüfung
16	Luftfilter verschmutzt				
17					Prüfung Kraftstoff
18					Kraftstoffpumpe defekt
19					
20					
21	Gleichdruckvergaser defekt				
22	Kein Kraftstoff im Tank				
23	Kraftstofffilter verschmutzt				
24	Kraftstoffpumpe defekt				
25					
26					
27					
28					
29					
30					
31	Zündkerzenstecker locker				
32					
33	Zündkerze defekt				
34					
35	Zündung defekt				
36					
37					
38					
39	Motor sitzt fest				
40	Falsches Motorenöl				
41	Geringe Batteriespannung				
42	Zündschalter aus				
43					
44					
45					
46					
47	Lager defekt				
48					
49	Kolbenringe defekt				
50					
51	Zylinderbuchse verschlissen				
52					
53	Lichtmaschine defekt				
54					
55	Batterie defekt				

Abbildung 28: Konsistenzprüfung, Ergebnis 1

Die in beiden Systemmodellen vorkommenden Systemkomponentennamen sind jeweils grün markiert und signalisieren so, dass sie konsistent sind. Die nicht markierten sind dementsprechend inkonsistent. Es liegt in der Natur der Systemmodellierung, dass zwangsläufig einige Systemkomponentennamen des einen Modells, nicht mit denen des anderen Systemmodells übereinstimmen, da verschiedenen Systemmodelle verschiedenen Aspekte des Systems modellieren und dementsprechend andere Systemkomponentennamen haben.

Wie in Zeile 24 und 18 in Abbildung 28 zu sehen ist, ist der Systemkomponentenname „Kraftstoffpumpe defekt“ konsistent. Zur Verifizierung wird in dem Sequenzdiagramm der Systemkomponentenname „Kraftstoffpumpe defekt“ in „Kraftstoffpumpe defekt01“ umbenannt und die Konsistenzprüfung erneut ausgeführt.

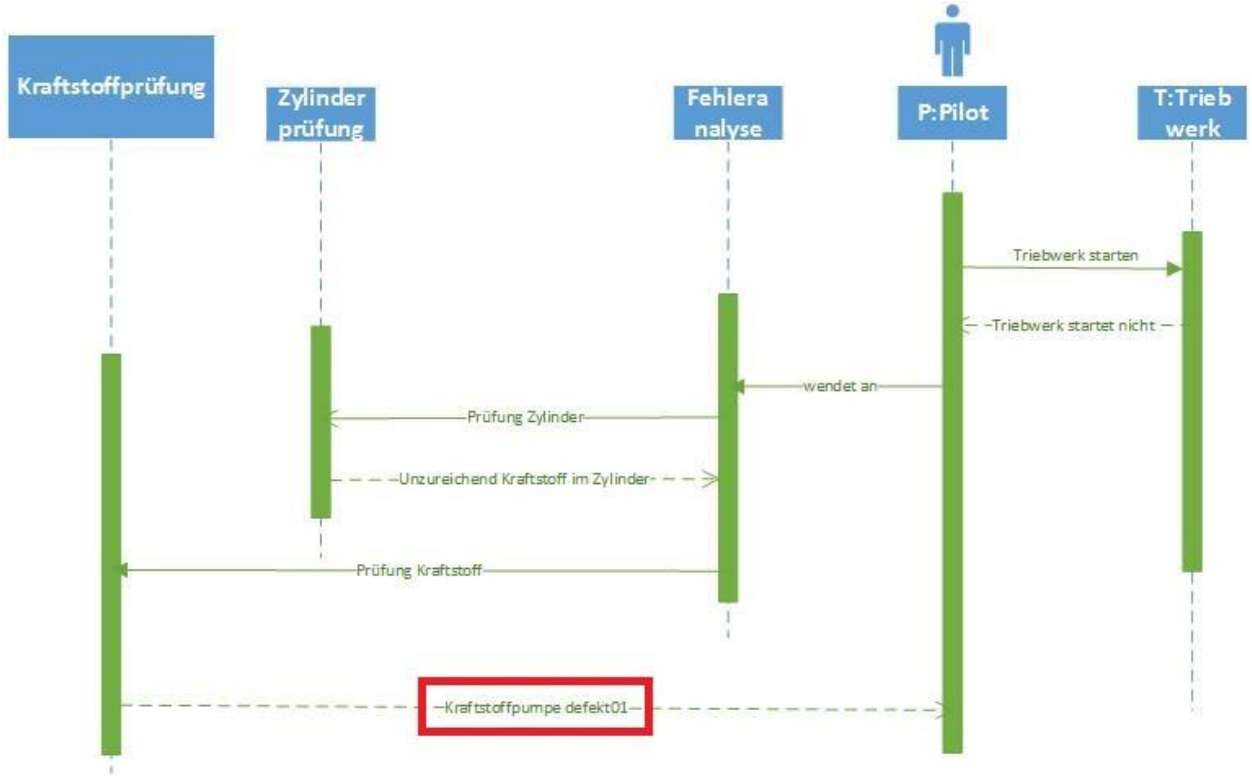

Abbildung 29: Sequenzdiagramm, "Kraftstoffpumpe defekt01"

6.4 Ergebnis der zweiten Konsistenzprüfung

1	Modell1				Modell2
2	Triebwerk startet nicht				T:Triebwerk
3					P:Pilot
4					
5	Unzureichend Frischluft im Zylinder				
6	Unzureichend Kraftstoff im Zylinder				Triebwerk starten
7	Kein Zündfunke				Triebwerk startet nicht
8	Elektr. Anlasser dreht nicht				Fehleranalyse
9					
10					wendet an
11					Zylinderprüfung
12					
13					Prüfung Zylinder
14					Unzureichend Kraftstoff im Zylinder
15	Gleichdruckvergaser defekt				Kraftstoffprüfung
16	Luftfilter verschmutzt				
17					Prüfung Kraftstoff
18					Kraftstoffpumpe defekt01
19					
20					
21	Gleichdruckvergaser defekt				
22	Kein Kraftstoff im Tank				
23	Kraftstofffilter verschmutzt				
24	Kraftstoffpumpe defekt				
25					
26					
27					
28					
29					
30					
31	Zündkerzenstecker locker				
32					
33	Zündkerze defekt				
34					
35	Zündung defekt				
36					
37					
38					
39	Motor sitzt fest				
40	Falsches Motorenöl				
41	Geringe Batteriespannung				
42	Zündschalter aus				
43					
44					
45					
46					
47	Lager defekt				
48					
49	Kolbenringe defekt				
50					
51	Zylinderbuchse verschlissen				
52					
53	Lichtmaschine defekt				
54					
55	Batterie defekt				

Abbildung 30: Konsistenzprüfung, Ergebnis 2

In Zeile 18 in Abbildung 30 steht nach der zweiten Ausführung der Konsistenzprüfung der Systemkomponentenname „Kraftstoffpumpe defekt01“, welcher nicht mehr grün markiert ist. In Zeile 24 bleibt der Systemkomponentenname „Kraftstoffpumpe defekt“ unverändert, da dieser in dem Fehlerbaum auch nicht verändert worden ist. Er ist ebenfalls nicht mehr grün markiert, weil es nach der zweiten Konsistenzprüfung keinen anderen konsistenten Systemkomponentenamen mehr gibt. In den folgenden Abbildungen sind die Ergebnisse der ersten und der zweiten Konsistenzprüfung explizit dargestellt.

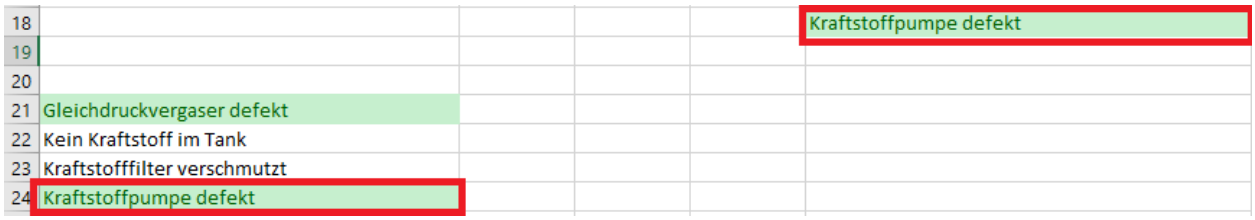

Abbildung 31: Konsistenzprüfung, Ergebnis 1, Ausschnitt

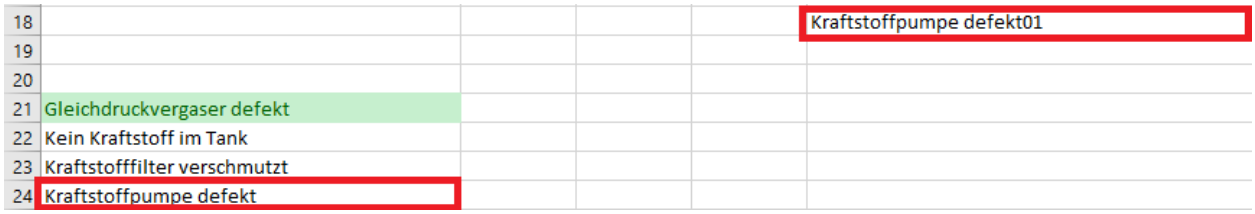

Abbildung 32: Konsistenzprüfung, Ergebnis 2, Ausschnitt

Die Konsistenzprüfung hat den neuen Systemkomponentennamen erkannt und als nicht mehr konsistent eingestuft. Dies ist das gewünschte Ergebnis. Die Konsistenzprüfung gilt dementsprechend als verifiziert.

7 SysML Shape Schablonen von Pavel Hruby

Der Doktorrand Pavel Hruby stellt auf seiner Internetseite [Hru19] Microsoft Visio Schablonen in SysML zur Verfügung. Die in diesem Kapitel abgebildeten Systemmodelle sind mit diesen erstellt und nach der Notationsübersicht von Weilkiens [Wei19] modelliert worden. Sie müssen um die Konsistenzprüfung mit den Schablonen von Hruby in Visio auszuführen, den Ordner mit den Schablonen als „vertrauenswürdig“ einstufen lassen. Dazu gehen Sie in Visio auf **Datei → Optionen → Trust Center → Einstellungen für das Trust Center → Vertrauenswürdige Speicherorte → neuen Speicherort hinzufügen…** und geben den Pfad zu dem Ordner an. Aktivieren Sie anschließend die Funktion **Vertrauenswürdige Speicherorte im Netzwerk zulassen (nicht empfohlen)**.

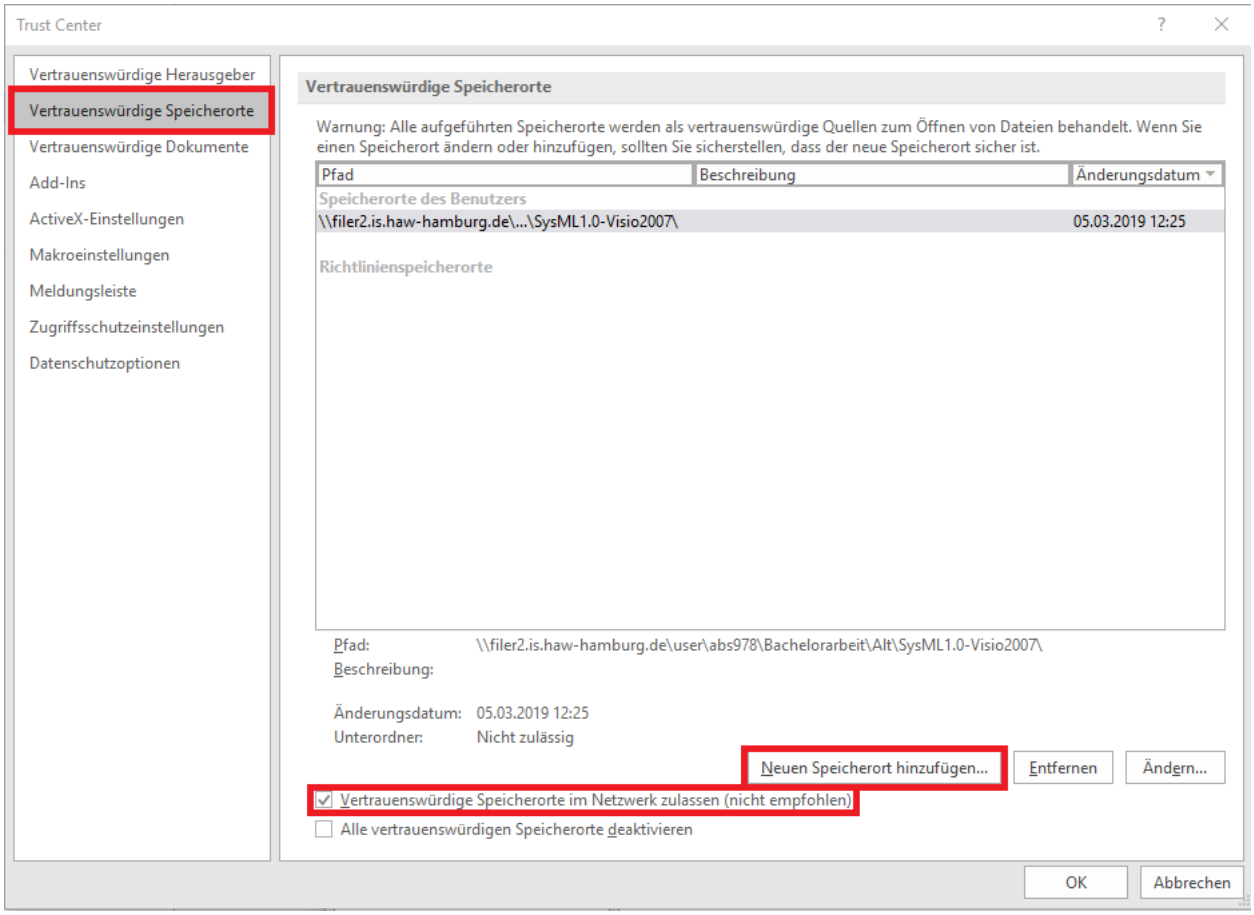

Abbildung 33: Visio, vertrauenswürdigen Speicherort einrichten

7.1 Blockdefinitionsdiagramm

In der folgenden Abbildung sind in einem Blockdefinitionsdiagramm die einzelnen Programmkomponenten dargestellt.

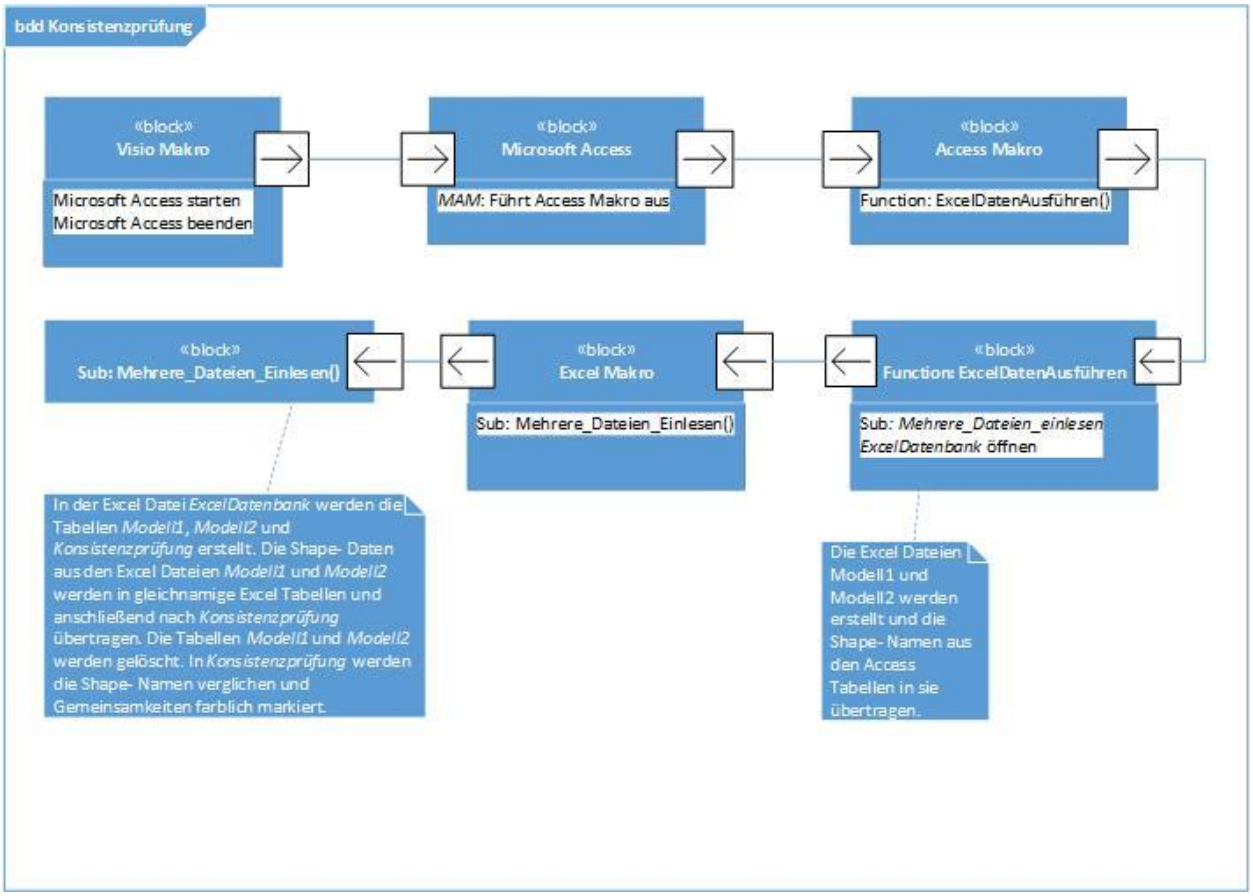

Abbildung 34: Programmkomponenten im Blockdefinitionsdiagramm

7.2 Aktivitätsdiagramm

In der folgenden Abbildung ist in einem Aktivitätsdiagramm der Programmablauf der Konsistenzprüfung modelliert.

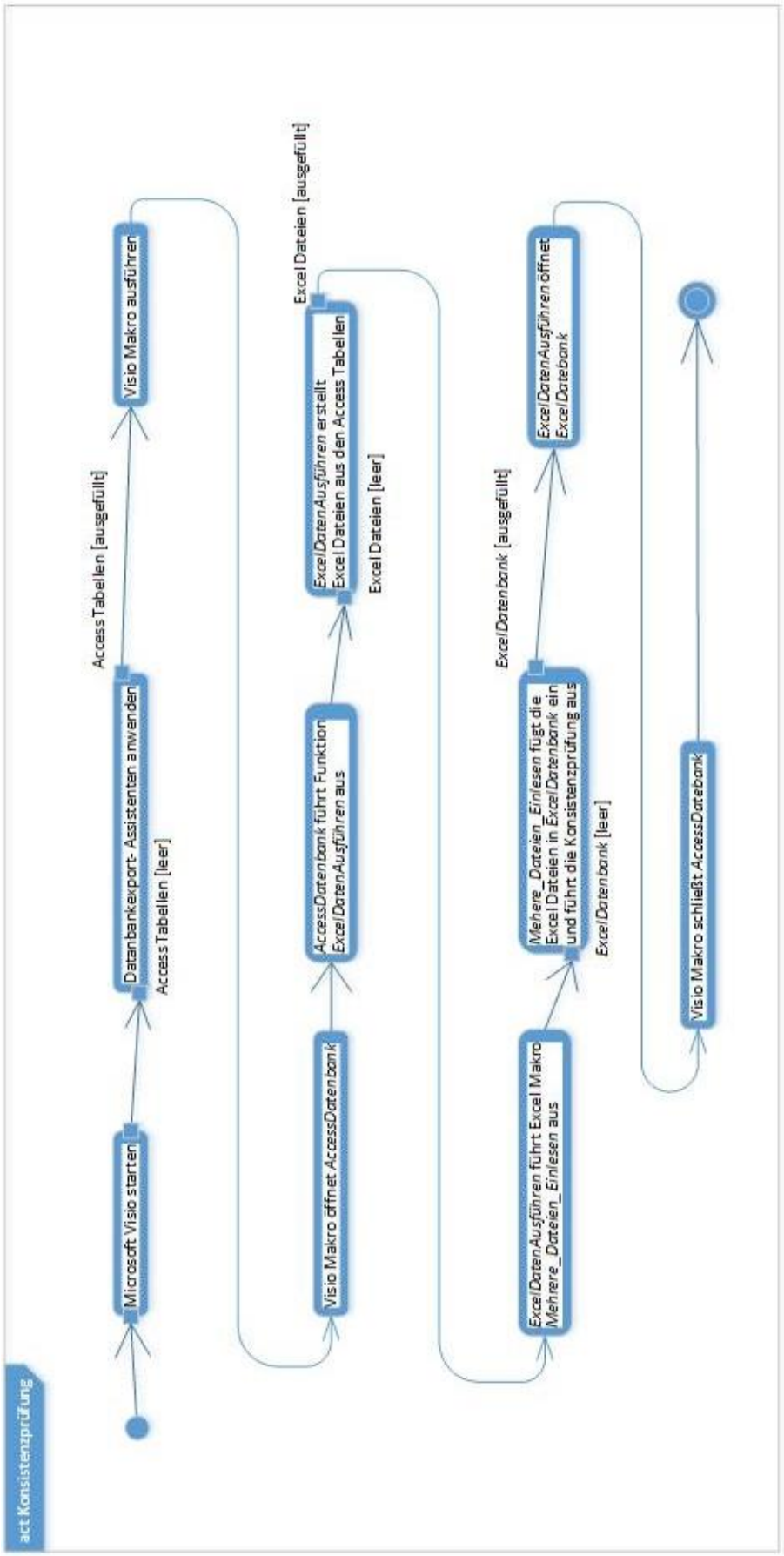

Abbildung 35: Programmablauf als Aktivitätsdiagramm

7.3 Use- Case- Diagramm

In der folgenden Abbildung sind einem Use- Case- Diagramm die einzelnen Funktionen des Programmes dargestellt.

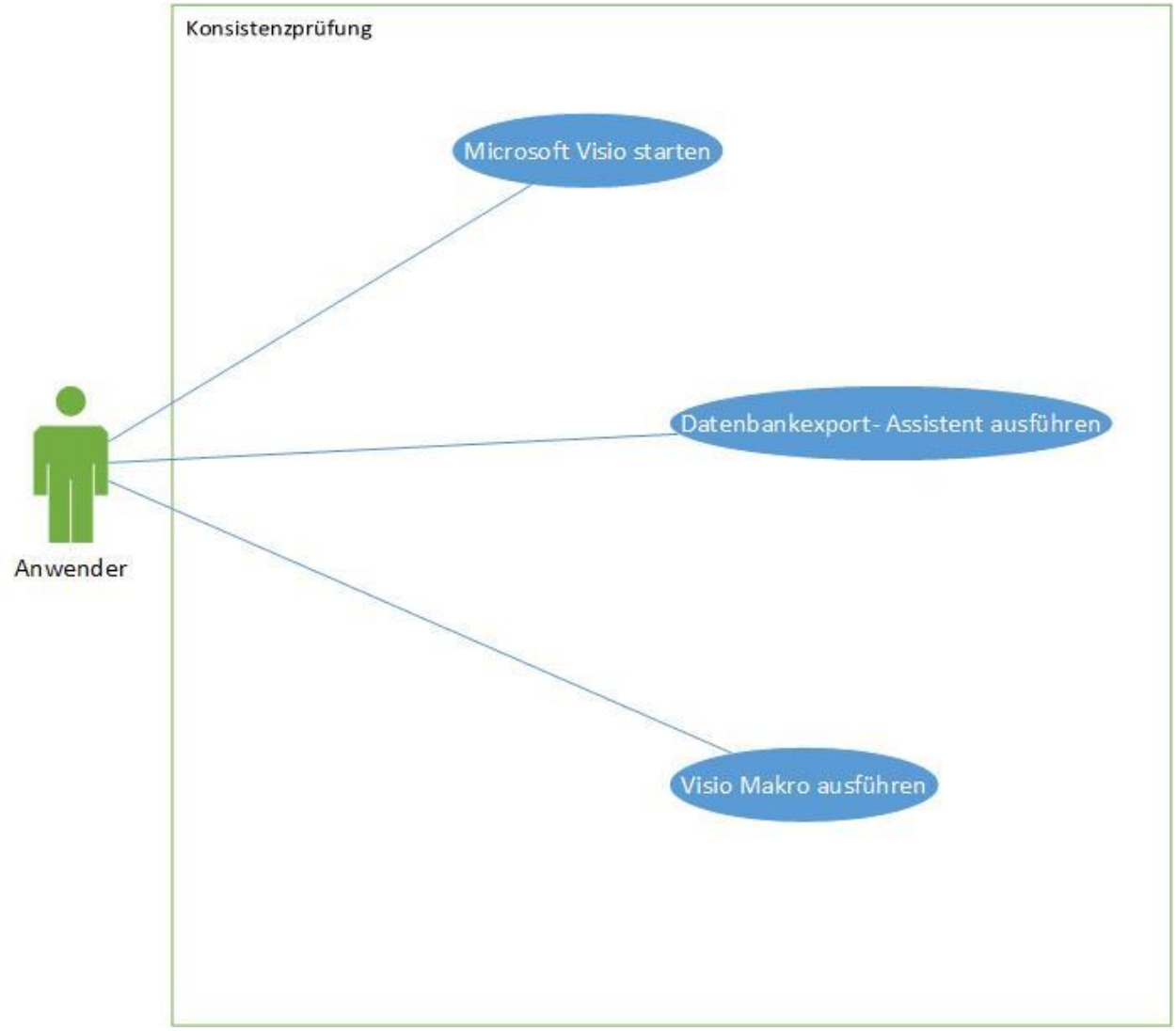

Abbildung 36: Programmfunktionen als Use- Case- Diagramm

7.4 Verifizierung der Konsistenzprüfung mit den Schablonen von Hruby

Die Konsistenzprüfung wird mit dem Aktivitäts- und dem Use- Case- Diagramm durchgeführt. Das Ergebnis ist in folgender Abbildung dargestellt.

1	Modell1				Modell2
2					
3	Anwender				Microsoft Visio starten
4	Datenbankexport- Assistent ausführen				
5					
6	Visio Makro ausführen				Datanbankexport- Assistenten anwenden
7					
8	Microsoft Visio starten				
9					Access Tabellen [ausgefüllt]
10					Visio Makro ausführen
11					
12					
13					Visio Makro öffnet AccessDatenbank
14					Access Tabellen [leer]
15					
16					AccessDatenbank führt Funktion ExcelDatenAusführen aus
17					ExcelDatenAusführen erstellt Excel Dateien aus den Access Tabellen
18					Excel Dateien [leer]
19					Excel Dateien [ausgefüllt]
20					ExcelDatenAusführen führt Excel Makro Mehrere_Dateien_Einlesen aus
21					Mehere_Dateien_Einlesen fügt die Excel Dateien in ExcelDatenbank ein und führt die Konsistenzprüfung aus
22					ExcelDatenbank [leer]
23					ExcelDatenbank [ausgefüllt]
24					ExcelDatenAusführen öffnet ExcelDatebank
25					Visio Makro schließt AccessDatebank

Abbildung 37: Konsistenzprüfung mit Schablonen von Hruby

Alle Systemkomponentennamen, welche in beiden Systemmodellen konsistent sind, sind farblich markiert. Auch mit den Schablonen von Hruby lässt sich die Konsistenzprüfung ausführen.

8 Ansätze für folgende Arbeiten

8.1 Mehrere Systemmodelle auf Konsistenz hin prüfen

Die mit dieser Bachelorarbeit zur Verfügung gestellte Konsistenzprüfung kann zwei Systemmodelle auf Konsistenz hin prüfen. Es ist möglich, weitere Systemmodelle einzubinden. Die Programmierung des Excel Makros erlaubt ein Einlesen von allen in dem Ordner ***Konsistenzprüfung*** enthaltenen Excel Dateien. In dieser Konsistenzprüfung sind dort nur zwei, *Modell1* und *Modell2*. Wenn mehrere Excel Dateien in dem Ordner wären, würden diese auch in *ExcelDatenbank* erstellt werden. Diese könnten mit dem Excel Makro- Rekorder in der Tabelle *Konsistenzprüfung* entsprechend bearbeitet werden.

8.2 Weitere Shape- Daten vergleichen

Für diese Bachelorarbeit ist nur mit dem **Shape.Text** gearbeitet worden. Dieser stellt den Namen der Systemkomponente dar und reicht somit für eine Konsistenzprüfung aus. Es ist möglich weitere Shape- Daten mit dem Datenbankexport- Assistenten zu exportieren, siehe Abbildung 9. Diese könnten ebenfalls auf Konsistenz hin geprüft oder anderweitig verwertet werden.

9 Zusammenfassung

In dieser Bachelorarbeit ist ein Programm vorgestellt worden, welches eine automatische Konsistenzprüfung zwischen zwei Systemmodellen in Microsoft Visio ermöglicht. Nach der Erläuterung der Grundlagen von Visio, der SysML sowie von VBA wurden die einzelnen Entwicklungsschritte erklärt. Dabei hat sich als Datenexportierungsmöglichkeit aus Visio der Datenbankexport- Assistent als beste Lösung herausgestellt. Mit ihm lassen sich die Shape- Namen der Systemmodelle in Visio fehlerlos exportieren. Eine benutzerfreundliche Ausführung ist gewährleistet, da die Funktion in das rechtsklick Menü der Visio Zeichnungsansicht eingebunden ist. Wie in Kapitel 8 beschrieben kann dieser nicht nur die Namen der Shapes, sondern auch noch weitere Shape- Daten exportieren. Des Weiteren bietet er den Vorteil, dass er lediglich einmal pro neuem Visio- Zeichenblatt eingerichtet werden muss. Er sollte daher in anderen oder einer weiterführenden Arbeit, falls ein Datenexport aus Visio gewünscht ist, berücksichtigt werden.

Die Installation des Programmes wurde in der Implementierungsanleitung Schritt für Schritt erklärt. Die wichtigsten Punkte hierbei sind die Aktivierung der entsprechenden Verweise und die Angabe des korrekten Pfades zum Ordner ***Konsistenzprüfung*** in den einzelnen Makros. Dann wurde die ordnungsgemäße Ausführung der Konsistenzprüfung beschrieben. Es ist zu berücksichtigen, dass die, bei erstmaliger Nutzung der Konsistenzprüfung, erscheinende Fehlermeldung normal ist. Danach lässt sie sich, in derselben Computersitzung, ohne Fehlermeldung ausführen.

Mit einem Fallbeispiel ist das Ergebnis einer vollständig durchgeführten Konsistenzprüfung dargelegt worden. Falls ein Systemkomponentenname eines Systemmodells geändert und nicht mehr konsistent ist mit dem Systemkomponentennamen eines anderen Systemmodells, ist dieser Systemkomponentenname anschließend nicht mehr farblich markiert, wenn diese beiden Systemmodelle auf Konsistenz hin geprüft werden. So wird der inkonsistente Systemkomponentenname aufgedeckt.

Mit den von Dr. Pavel Hruby zur Verfügung gestellten Visio Schablonen für die SysML ist die Konsistenzprüfung in mehreren Systemmodellen in Microsoft Visio visualisiert worden. Wichtig ist hierbei, dass der Ordner, in dem sich die Schablonen befinden, vor der Benutzung durch Visio als „vertrauenswürdig“ eingestuft werden muss, da sie ansonsten nicht in Visio verwendet werden können. Die Konsistenzprüfung kann auch mit den Systemmodellen, die mit diesen Visio Schablonen erstellt worden sind, ausgeführt werden. Dies wurde verifiziert, indem zwei von den Systemmodellen auf Konsistenz hin geprüft worden sind.

Zuletzt wurden mögliche Ansätze für nachfolgende Arbeiten vorgestellt, wobei als erstes die Möglichkeit in Betracht gezogen worden ist, mehrere Modellierungen auf Konsistenz hin prüfen zu können, da einige Segmente der Konsistenzprüfung schon in der Lage sind mehrere Visio- Zeichenblätter zu verarbeiten.

10 Abkürzungsverzeichnis

Access	Microsoft Access
Excel	Microsoft Excel
MAM	Microsoft Access Makro
OMG	Object Management Group
SysML	System Modified Language
UML 2.0	Unified Modeling Language 2.0
VBA	Visual Basic for Applications
Visio	Microsoft Visio

11 Abbildungsverzeichnis

12 Literaturverzeichnis

[Wei19] Weilkiens, T.: *„OMG Systems Modeling Language (OMG SysML™) 1.3“*, unter: https://model-based-systems-engineering.com/wp-content/uploads/2012/08/sysmod-sysml-1.1-notations%C3%BCbersicht-oose.pdf (abgerufen am 02.02.2019)

[Wik19] Wikipedia: *„Systems Modeling Language“*, unter https://de.wikipedia.org/wiki/Systems_Modeling_Language (abgerufen am 30.02.2019)

[Luc19] Lucidchart: *„Alles über Microsoft Visio ® für Diagramme“*, unter https://www.lucidchart.com/pages/de/was-ist-microsoft-visio (abgerufen am 30.02.2019)

[Rum19] Rum, S.: *„Einige Erklärungen zur VBA- Programmierung“*, unter http://userpage.fu-berlin.de/~ram/pub/pub_jf47ht81Ht/charakterisierung_vba (abgerufen am 03.03.2019)

[Wik19a] Wikipedia: *„Visual Basic for Applications“*, unter https://de.wikipedia.org/wiki/Visual_Basic_for_Applications (abgerufen am 03.03.2019)

[Hru19] Hruby, P.: *„Software Stencils“*, unter http://softwarestencils.com/ (abgerufen am 17.04.2019)

[Mic19] Microsoft Support*: „Exportieren eines Datenbankobjekts in eine andere Access- Datenbank“*, unter https://support.office.com/de-de/article/exportieren-eines-datenbankobjekts-in-eine-andere-access-datenbank-4e2cd6dd-e482-441e-88b5-aa5319a428a6 (abgerufen am 10.03.2019)

[Wik19b] Wikipedia: *„Fehlerbaumanalyse“*, unter https://de.wikipedia.org/wiki/Fehlerbaumanalyse (abgerufen am 15.03.2019)

[Lös19] Lösungsfabrik: „Risikoanalyse- Tools: Fehlerbaumanalyse (FTA)“, unter http://www.mpl.loesungsfabrik.de/blog/iso-13485/fehlerbaumanalyse (abgerufen am 09.04.2019)

[Bom15] Bombardi, J.: Systems Engineering Hausarbeit. HAW Hamburg: Dept. F+F, 2015

13 Verzeichnis für die als Informationsquelle genutzten Videos

[Pro19] Programming: *„45. VBA- Application Part 1 (Programming In Access 2013)“*, unter https://www.youtube.com/watch?v=BSwgy8LYqbA&list=PLYMOUCVo86jEeMMdaaq03jQ_t9nFV737s&index=45 (abgerufen am 10.03.2019)

[Exc19] Excelpedia: *„Werte aus Tabellenblatt per Button in ein anderes übertragen – Excel VBA | Excelpedia“*, unter https://www.youtube.com/watch?v=GQYkURMTmoU (abgerufen am 12.03.2019)

[The19] Andreas Thehos: *„Excel # 334 – Makro automatisch bei Änderungen ausführen – VBA Worksheet_Change“*, unter https://www.youtube.com/watch?v=L2mYkaOD9N4 (abgerufen am 12.03.2019)

[The19a] Andreas Thehos: *„Excel # 337 – Makros beim Starten und Beenden einer Arbeitsmappe ausführen“*, unter https://www.youtube.com/watch?v=xEg4PnvKp68 (abgerufen am 13.03.2019)

Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Erklärung zur selbstständigen Bearbeitung einer Abschlussarbeit

Gemäß der Allgemeinen Prüfungs- und Studienordnung ist zusammen mit der Abschlussarbeit eine schriftliche Erklärung abzugeben, in der der Studierende bestätigt, dass die Abschlussarbeit „– bei einer Gruppenarbeit die entsprechend gekennzeichneten Teile der Arbeit [(§ 18 Abs. 1 APSO-TI-BM bzw. § 21 Abs. 1 APSO-INGI)] – ohne fremde Hilfe selbständig verfasst und nur die angegebenen Quellen und Hilfsmittel benutzt wurden. Wörtlich oder dem Sinn nach aus anderen Werken entnommene Stellen sind unter Angabe der Quellen kenntlich zu machen."

Quelle: § 16 Abs. 5 APSO-TI-BM bzw. § 15 Abs. 6 APSO-INGI

Dieses Blatt, mit der folgenden Erklärung, ist nach Fertigstellung der Abschlussarbeit durch den Studierenden auszufüllen und jeweils mit Originalunterschrift als letztes Blatt in das Prüfungsexemplar der Abschlussarbeit einzubinden.
Eine unrichtig abgegebene Erklärung kann -auch nachträglich- zur Ungültigkeit des Studienabschlusses führen.

Erklärung zur selbstständigen Bearbeitung der Arbeit

Hiermit versichere ich,

Name: _____

Vorname: _____

dass ich die vorliegende – bzw. bei einer Gruppenarbeit die entsprechend gekennzeichneten Teile der Arbeit – mit dem Thema:

ohne fremde Hilfe selbständig verfasst und nur die angegebenen Quellen und Hilfsmittel benutzt habe. Wörtlich oder dem Sinn nach aus anderen Werken entnommene Stellen sind unter Angabe der Quellen kenntlich gemacht.

- die folgende Aussage ist bei Gruppenarbeiten auszufüllen und entfällt bei Einzelarbeiten -

Die Kennzeichnung der von mir erstellten und verantworteten Teile der ist erfolgt durch:

Ort	Datum	Unterschrift im Original