unibz
Freie Universität Bozen
Libera Università di Bolzano
Università Liedia de Bulsan

Fakultät für Informatik
Facoltà di Scienze e Tecnologie informatiche
Faculty of Computer Sciences

BACHELOR THESIS

Erweiterung eines Testdokumentationstools zur besseren/einfacheren Strukturierung und Wartung dokumentierter Testfälle

Jan Schmid Niederkofler

Supervisor: Andrea Janes

Juli, 2017

Contents

List of Tables

List of Figures

Listings

Abstract deutsch

An abstract should be structured as follows[1]:

1.

2. **Problem statement**: Den Verlauf der Testautomatisierung kann man in 2 Phasen unterteilen. Erstere die Dokumentation der Testfälle und letztere das Automatisieren. Während die bereits standartisierten Tests aus der Testfactory gewartet werden riskiert man durch den ständigen Wandel der zu Testenden Programme die Dokumentation zu vernachlässigen da diese nicht nur Zeitintensiv ist sondern auch nicht so oft benötigt wird wie ersteres, aber nicht weniger wichtig.

3. **Motivation**: Die Motivation hinter diesem Projekt ist die Wartbarkeit der Dokumentation zu erleichtern bzw. zu ermöglichen. Dessen Funktion vielleicht nicht so viel Einfluss auf den Alltag hat, jedoch große Hilfe bieten kann für den Fall das Fehler entdeckt werden und/oder der zuständige für den Bereich nicht Anwesend ist.

4. **Approach**: Die ersten Schritte bei der Problemlösung war das Erfassen was bereits vorhanden ist womit man Arbeiten kann und was selbst noch zu machen ist. Sobald man ein Gefühl dafür hat macht man sich mit der Umgebung vertraut, wie kann man was Steuern, welche Tools können helfen, welche Sprache ist hier am besten geeignet. Die Wahl fiel schnell auf VBS das es von der Applikation am besten unterstütz wird. Obwohl diese in viele Hinsicht C-Sprachen unterlegen wäre. Hier ist auch nützlich zu Wissen wie weit man das System auslasten kann. Da wir hier von tausenden Elementen sprechen müssen wir schaugen ab wann wird der Ablauf durch die neue Feature langsamer oder es Limitationen vom System aus gibt.

   Hier hatten wir noch dazu die Möglichkeit das Tool intern oder extern zu erweitern, und entschieden uns auf so viel Möglich auf extern, da sich intern bereits über die Zeit sich viel Code angesammelt und die externen VBS Scripte einfach einzuspielen sind.

[1] https://www.ece.cmu.edu/ koopman/essays/abstract.html

Bei der Implementation ist aufzupassen welche Fehler passieren könnten und wie man mit ihnen Umgehen sollte. Da dieses Projekt großes Erweiterungpotential hat war es auch wichtig versuchen im Rahmen der Zeit zu bleiben.

5. **Results**: Das Erebnis ist eine leichtere und weniger Fehleranfällige Eingabe bei der Dokumentation. Auch hat man später eine leichtere Wartung da der Pfad standartiesiert eingeben ist und man so Testfälle ausmachen kann die von der Änderung bestimmter Masken betroffen sind. Ältere Systeme müssen überarbeitet werden da die Daten in verschiedenen Formaten abgespeichert worden sind.

6. **Conclusions**: What are the implications of your answer? Is it going to change the world (unlikely), be a significant "win", be a nice hack, or simply serve as a road sign indicating that this path is a waste of time (all of the previous results are useful). Are your results general, potentially generalizable, or specific to a particular case?

   Das erreichte Ergebnis bietet die Grundlage einer besserung Wartung. Die erleichtert die Eingabe bietet aber sogleich Möglichkeit ausgebaut zu werden um mehr Arbeiten weniger Fehleranfällig sein zu lassen oder gar zu automatisieren.

Chapter 1

Einleitung

1.1 Einleitung

Testfälle sind die einzelnen Tests, die durchgeführt werden um sicherzustellen, dass Anwendungen funktionieren wie erwartet. Die Teställe werden erstellt von den jeweiligs für die Applikation Verantwortlichen erstellt. Diese werden dann von einer eigenen Abteilung standartisiert und besitzen dann einen fixen Ablauf. Mit HP Application Lifecycle Manager werden diese dokumentiert. Diese Prozedur funktionier wird aber durch die Erweiterung verbessert. Die Wartbarkeit der dokumentierten Testfälle wird vereinfacht. Auch verbessert wird die Eingabe über Dropdown Menüs mit Vorschlägen anstatt es selbst eintippen zu müssen. Immer noch erschwerend wirkt die ständige Veränderungen bei den MAX Anwendungen.

Supply Chain Lufthansa Lufthansa ist mit fast 126.000 Mitarbeitern eine der größten Fluggesellschaften weltweit. Diese wird aufgeteilt in Passagierbeförderung, Fracht, Technik, Catering, IT- Dienstleistungen und Service –und Fianzgesellschaft. Dieses Projekt wurde in der Lufthansa Technik realisiert. Genauer im Bereich der Supply Chain. Der Einsatzbereich der Supply Chain ist die Versorgung von Materialen für Kunden. Diese werden unterteilt in Material das repariert und wieder im Umlauf gebracht wird, wie ein Flugzeuggetriebe, als auch Material das einfach aufgestockt wird wie Schrauben etc. Um den Kunden ein Interface bieten zu können über denen sie gewünschte Bestellungen abgeben können, bietet Lufthansa verschiedene Software-Lösungen an. Diese unterstehen regelmäßigen Updates. Um sicherzugehen das nach einem Update alles noch funktioniert wie es sollte wurden bis vor kurzen Integrationstests abgehalten, die daraus bestanden die Applikationen manuell durchzuklicken. Diese Arbeitszeit soll nun mit automatischen Tests reduziert werden. ((insert source about Lufthansa))

1.2 Testdokumentation / Testautomatisierung

Die Vorteile der Testautomatisierung sind in erster Linie Zeitersparnis durch die Abnahme der Testfälle per Computer. Automatisierte Tests laufen, ohne menschliche Interaktion, sehr viel schneller). Manueller Testfälle hingegen können Tage/Wochen andauern und bedürfen Menschen die sie durchlaufen. Durch die Reduzierung der menschlichen Interaktion auf ein Minimum entstehen auch weniger Probleme falls ein Experte für einen bestimmten Bereich nicht anwesend sein kann, sondern wird nur noch benötigt falls die Tests nicht erfolgreich abgeschlossen werden können. Ein Automatisierungszyklus beginnt mit der Auffindung aller Testfälle. Diese werden dann dokumentiert. Der erste Schritt der Dokumentation beginnt mit einem Tool namens „SAP Workforce Performance Builder“. Dieses ist ein Werkzeug um automatisch bei jeden User Input Screenshots zu erstellen. So kann ein Testfall in erster Instanz abgebildet sein. Diese Dokumentation ist fundamental zur Automatisierung. Hauptsächlich liegt der Vorteil bei Dokumentation dabei, dass auch wenn ein Experte oder Wissensträger verloren geht, nicht das Wissen selbst verloren ist. Die Dokumentation beginnt mit Übertragung des Testfalls in das HP Tool „Application Lifecycle Manager“ ((insert screenshot)). Hier werden die Schritte einzeln in eine Tabelle eingegeben.

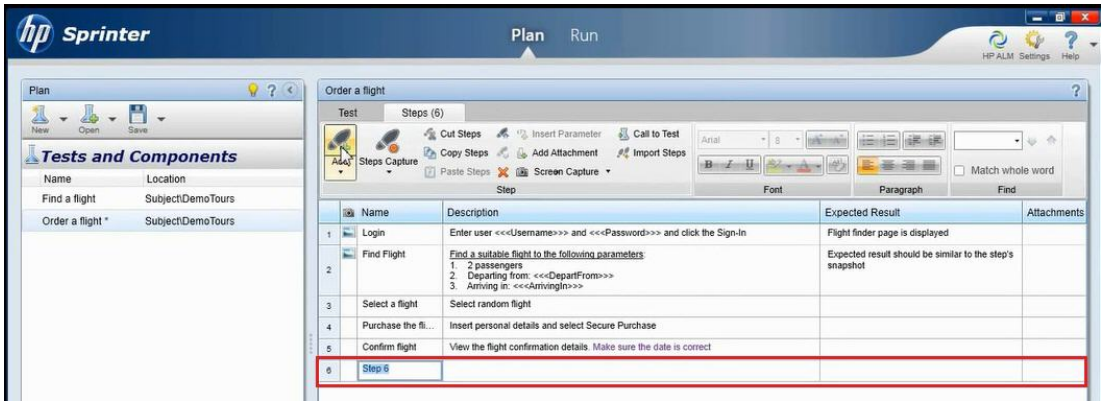

Figure 1.1: In diesen Bild können wir ein Beispiel sind die einzelnen Schritte zu erkennen. Die Felder die wir im späteren Verlauf verändern werden sind Namen und Pfad

Ein einzelner Testschritt besteht aus: wo genau die Aktion stattfindet, was für ein Input kommt, welches Objekt an der Maske angesprochen wird. Ein Testschritt kann auch eine SQL abfrage sein oder der Start eines Scripts. Diese Dokumentation wird später verwendet um Testfälle manuell durchzuspielen. Manuelle Tests wird es vorrausichtlich immer geben, da nicht alles zu automatisieren geht, sprich wo es menschliche Interaktion braucht. Parallel dazu werden sie verwendet um bei

fehlerhaften Durchläufen den Fehler zu suchen.

Ist ein Testfall komplett übertragen und kontrolliert wird er nach Budapest zur Testfactory freigegeben, wo mit Hilfe von Visual Basic jeweils ein Script erstellt wird, der den Testfall nachspielt. Dieses Script ist unabhängig von der Dokumentation im HPQC, auch wenn dieses darauf basiert. Änderungen an der Maske werden nur in den angefertigten Scripten eingespielt, nicht aber in der Dokumentation.

Object Action genauer beschreiben + Beschreibungssprache und Formale Sprache (aus dem Wiki) + outsourcing probleme

1.3 Ausgangssituation

Die Übertragung der Testfälle in das HP Quality Center funktioniert, aber ist durch die Eingabe über Textfelder Fehleranfällig. Da sich auch bei kleinen Änderungen an den Anwendungen viele Testfälle verändern ist der Aufwand zu hoch, sodass diese bei der großen Anzahl an Testfällen die Dokumentation Anpassung wegbleibt. Dies betrifft nicht automatisierte Tests, da hier die Visual Basic Scripts angepasst werden.

1.4 Schwer bis unmöglich manuelle Testfälle zu warten

Ein entscheidender Punkt in der Dokumentation ist die Angabe wo ein bestimmtes Element sich befindet. Dies wird mit Object Path und Object Name angegeben. Bei der Übertragung in das HP tool wird dies manuell eingepflegt. Ändert sich die Maske, durch die regelmäßigen Updates, wird diese Angabe falsch. Dies richtig zu Pflegen beinhaltet alle Testfälle durchzugehen und wo auf die geänderte Oberfläche zugegriffen wird dementsprechend richtig zu stellen. Folglich sind diese Tests nicht skalierbar und bereits bei wenigen Testfällen kann bei Veränderungen großer Aufwand verbunden.

Chapter 2

Grundlagen und Stand der Technik

2.1 Hp Tool

Das HP Tool Application Life Cycle Manager, oder kurz HP ALM, ist eine Sammlung von Software Applikationen die den Lebenszyklus einer Software unterstützt. In diesen Fall die Erstellung von automatisierten Testfällen. Spezifisch wird hier HP Quality Center verwendet das eine einzige Plattform für Test- und Software Zyklus Management bereitstellt. Aufgabe der Software Factory ist nicht die genutze Software nach zu bauen sondern eigentständige Programme schreiben die auf einer existierend Testumgebung die gewünschten Abläufe rekonstruieren.

2.2 Daten

Eine Sammlung von allen Objekten die bei der Automatisierung angesprochen worden sind gibt es in Form von XML-Dateien. Diese liegen extern und nicht im HP-QC selbst. Da sie von der Testfactory angefertigt worden sind, eignen sie sich optimal da selbige beim Automatisieren verwendet und somit Konflikte mit der Namensgebung der Elemente vermieden werden.

Da reine XML nicht in dieser Form eingespeichert und verwendet werden können müssen diese erst geparst werden. Am besten eigenen sich hier “Project List” die von HP bereitgestellt werden. Diese können auch mit externen Scirpten erstellt,gelöscht oder bearbeitet werden. Intern sind diese einfach aufrufbar und somit ideal geeignet um das Eingabefenster je nach Aufruf dynamisch zu verändern.

Der Workflow reguliert den Ablauf der Übertragung der Testfälle ins HPQC. Mit diesen kann man Felder an Bedingungen knüpfen, lehr lassen, ausgrauen, autovervollständigen etc.

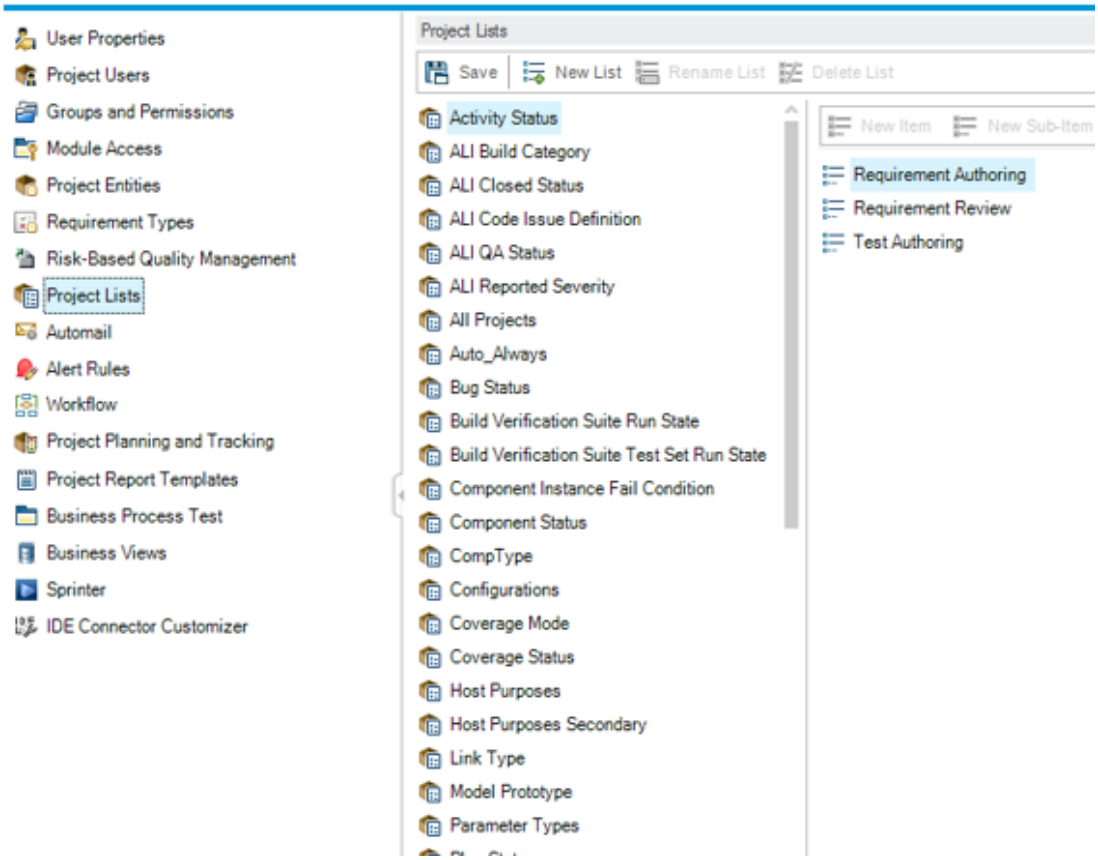

Figure 2.1: In diesem Bild sehen wir ein Beispiel von HP eigenen Listen. Diese können vom Workflow aus zugegriffen werden und von extern bearbeitet oder erstellt werden.

2.3 HP ALM XML

```
1 <qtpRep:ObjectRepository xmlns:qtpRep="http://www.\
  mercury.com/qtp/ObjectRepository">
2 <qtpRep:Objects>
3 <qtpRep:Object Class="Link" Name="\
  matRequest_info_second_IBO_Materialstamm_Tab">
4 <qtpRep:ChildObjects>
5 <qtpRep:Object Class="WinButton" Name="\
  matRequest_info_NotificationBar_FileSave_OpenButton">
```

Im oberen Beispiel können wir die Grundstruktur der XML Dateien erkennen. In der ersten Zeile wird die Repository URL angegeben. Für unsere Listen benötigen wir den Unterpunkt “qtpRep:Objects”. In diesem werden alle Objekte, von Buttons

über Explorer eingeordnet. Die sind unter “qtpRep:Object” zu erkennen.

Die wichtigsten Elemente die wir aus der XML auslesen können sind die Attribute Class und Name. Das Attribut Class gibt mit welche Art von Objekt wir es hier zu tun haben. Im Code-Beispiel ist es einmal ein Link und darunter ein Button. Das nächste ist für uns das wichtigste Element, der Namen mit der ein Objekte eindeutig gekennzeichnet wird. Am Ende von diesen steht auch eine Art Klasse in der genau spezifiziert wird was dies darstellt.

Anklickbaren Elemente wie Textboxen etc. werden alle erst untergeordnet in Explorer, Applikation etc. unter Childobjects. Ein größerer Panel besteht immer aus mehreren kleinenren Interaktiven Elementen. Diese können selbst Unterelemente haben. Mit der Klasse, Namen und dem Pfad werden sie eindeutig verifiziert und zugeordnet. werden.

2.4 Methodisches Vorgehen

Für dieses Projekt benutzen wir die für diese Art von Aufgabe untypische Programmiersprache Visual Basic Script. Aus dem einfachen Grund, dass man mit dieser einfach auf den Lifecycle Manager zugreifen kann. Auch C(sharp) bietet diese Möglichkeit ist aber im Funktionsumfang und Ausgereiftheit deutlich VBS unterlegen. Auch sind VBS Scripte im HP Alm einfacher einzubinden.

Noch dazu bietet HP ALM eine einfache Schnittstelle für VBS um mit dem Programm auch von extern kommunizieren zu können, die Open Test Arcitecture. Der Vorteil von extern auf die Software zuzugreifen liegt auf dementsprechend externe Dokumente leichter Zugriff zu haben. In unseren Fall XML-Files. Diese Prozedur wird unter Implementation genauer erklärt.

Ein weitere Vorteil die Scripte nicht direkt in den Hauseigennen “Worflow” einzubinden ist das dieser schon mehrere Tausend Zeilen schwer ist. Dies würde die Übersicht des bereits bestehenden Code verringen.

Bevor wir mit der Implementation beschäftigen, müssen wir und vorerst die Einschränkungen unseres Programmes kennen. Unter diesen versteht man hier die maximale Azahl an Elemente die das Programm erfassen kann, ebenso die in einer einzigen Iteration uploaden kann oder ob die Ladezeiten dabei nicht zu groß werden. Auch dürfen diese später nicht die Performance für User beinträchtigen.

Die Listen die eingetragen werden müssen werden auf unter >10.000 geschätzt. Die Tests mit HP ALM zeigen keine negativen Auswirkungen auf HP ALM auch über 100.000 Elementen. Auch die Einspielzeit, sofern es in wenigen Iteration geschieht, bleiben immer unter 15 Minuten. Um bei Listen Elemente hinzuzufügen, wie ein Button etc. muss diese aber bereits im HP QC commited sein, nicht nur hinzugefügt. Da aber Listen einzeln commiten mehrere Stunden dauern würde muss man die einzelnen Elemente später hinzufügen.

Chapter 3

Herausforderungen

[In der Zusammenfassun so lange nicht aktive Probleme. Dargestellt wie in Zukunft ausbaufähig und könnte und den Umständen Probleme machen. Alles komplett abschließen, keine Offenen Enden]

3.1 Rechte in einen Großkonzern

Während der Entwicklung des tools wird das Hp Quality Center Programm noch weiterhin verwendet. In diesen sind Daten gespeichert die nicht verändert werden sollen. Auch ein lahmlegen oder verlangsamen der Produktion kann zu monetären Schäden führen. Deshalb ist hier eine Sandbox benützt worden in der Daten regelmäßig aktualiesert worden sind und sie trotzdem bei Problemen resetiert worden konnte.

3.2 Semaphore / Visual Basic

Visual Basic ist konzeptiert als Script Sprache die Aufgaben automatisiert ausführen kann, oder für Zeitersparnis sorgen kann. Sie ist von HP ausgewählt worden um als Schnittstelle für die Life Cycle Applikation zu fungieren bzw. sie auf Wunsch maßzuschneidern. Dafür bietet Hp eine Reihe von Bibliotheken mit Zugang auf allen möglichichen Bereichen, sei es von extern über Scripts, sei es es intern übern den Hauseigenen Workflow.

Mit dies erweist sich VBS als eindeutig geigneste Wahl für dieses Projekt. Nun hat eine Scritpsprache gegenüber anderen Programmiersprachen wie Java einige Nachteile. Einmal ist der einzige Datentyp Variant, dieser kann zu Problemen führen wenn sich HPQC bestimmte Datentypen erwartet aber VBS sie nicht richtig interpretiert. Dies muss dann richtig konvertiert werden oder es kann zu Fehlern kommen.

Ein weiterer möglicher Konfliktpunkt könnte der Mangel an Abstrakten Datentypen sein. Da wir mit größeren Mengen an Daten mit vielen einzelnen und untereinander verknüpften Elementen zu arbeiten haben wären diese eine natürliche Wahl. Hier müssen wir Alternativen suchen die z.B. mit String manipultation funktioniert.

3.3 XML

Eine weitere Herausforderung bringen die XML Dateien. Mit VBS verlieren wir die "Tiefe" der Elemente. Da einmal eingelesen VBS alles Zeile für Zeile behandelt und keinen unterschied macht ob untereinander Zeilen verschachtelt sind. Sie macht die XML Datei eindimensional. Hier wären ADT nützlich beim einlesen. Gelöst wurde es mit der Namen der Listen bzw. der Elemente auf denen die vorhergehende Ebene verweist.

Des wurde das Format wie es jetzt standartisiert ist erst im Laufe der Zeit entwickelt und ältere XML Dateien sind grundsätzlich sehr verschieden aufgebaut. Um dies entgegen zu wirken müssen diese auf dem neusten Stand gebracht werden.

3.4 Zeigen von großen Datenmengen auf wenig Feldern

Schwierigkeiten kann auch eine verständlich Darstellung der eingespielten Daten bringen. Diese gehen in die Tausende Die Auswahlmöglichkeiten sind auf Eingabefelder und Comboboxen beschränkt. So kann es schnell passieren das einem die Übersicht verloren geht und die geplante vereinfachte Eingabe das Gegenteil erreicht. Eine Möglichkeit ist mehrere Comboboxen, an Stelle wie vorher einem Textfeld, die sich nach unten skalieren und die Auswahl mit einent Top-down approach vergleichbar ist. Es wird von der Applikation über Panel zu einzelnen Objekten immer die nächste Ebene über ausgegraut und mit der vorhergehenden Auswahl schränkt sich die Auswahl genug ein um übersichtlich zu bleiben. Die nächsten Auswahlmöglichkeiten werden dynamisch generiert im Workflow.

3.5 Workflow in HPQC

Über den Workflow in HPQC lassen sich Elemente ansteuern und verändern. Dieser ist mit der Zeit gewachsen, da das Programm auf eigene Wünsche maßgeschneidert worden ist. So bestand das Risiko den noch weiter aufzublasen was Wartung und Entwicklung erschweren könnten. Deshalb wurde beschlossen Soviel wie möglich mit externen Scripten zu machen.

Chapter 4

Lösungen

4.1 Object Repository

Im Fokus steht ein Object Repository für Object Path und Object Name. Pro Eintrag in dieser Liste muss jedes Element genau identifiziert sein. Aus diesem soll der User später die Einträge vom Element in das HPQC nehmen. Zugriff hat dieser direkt im HPQC bei der Eingabe-Maske über ein Drop-Down Menü.

4.2 UTF XML

Die Liste wird dynamisch aus bereits automatisierten Tests erstellt. Zugriff darauf hat man mit UFT (Unified Functional Testing). Über diese Software kann man über bereits erstellte Scripts in VBS die angesprochenen Elemente auslesen. Hier kann man alle Test-Scripts ansprechen und hat somit alle Elemente die angesprochen werden abgedeckt. Die Ausgabe erfolgt in einem XML Format.

4.3 Workflow

Der Workflow ist eine Schnittstelle an der man sich anbinden kann um die Übertragung ins HPQC zu verändern. In unseren Fall die Eingabemaske. Einfache Textfelder (in HTML Editfields) werden durch Listen ersetzt. Aus diesen wählt man sich das Ergebnis gewünschte Ergebnis. So ist es möglich ein Element zu verändern und über alle Testfälle übergreifenden ändern mit einem einzelnen Eingriff in das Object Repository. (wo denn?). Parallel dazu wird die Fehlerquote reduziert, die mit manueller Eingabe vorkommen.

4.4 Binary Repräsentation

Der Object Path wird in Zukunft mit mehreren Feldern angezeigt, dass die Eingabe erleichtert. Wird eine Applikation ausgewählt gibt es im ersten Object Path Feld nur die Elemente auszuwählen die auch in der Maske sind. So geht es mit Unterseiten oder Reitern weiter bis wir beim gewünschten Objekt sind. Dieser wird mit der Element- Art verknüpft, sprich Button, Editfield usw. Das soll dazu führen das nur die Objekte dargestellt werden die auch ausgewählt werden die auch mit der vorher eingegeben Objekt Art übereinstimmen. Vor dieser Auswahl bleibt das Feld Object Name ausgegraut. Es entsteht eine Baumstruktur die das Navigieren in diesen einfacher gestaltet als eine große Liste.

Chapter 5

Problemstelliung

5.1 Problemstellung

5.2 Warum HP Quality Center

Durch eine einzige Plattform die dieses Tool für Lebenszyklen und Test Management bereitstellt macht es für die Testautomatisierung geeignet. Die Testdokumentation und dessen Automatisierung folgt gewissen Zyklen [explaining here or somewhere else, or at all?] die sich in HP QC unter „Status“ wiedergeben lässt. Das Projekt ist nicht zentral und wird teilweise outsourcend und bei Setzung dieses Status weiß wann welche Personen jetzt für den nächsten Schritt verantwortlich sind. Diese werden benachrichtigt sobald dieser in den eigenen Aufgabenbereich fällt.

5.3 Requierments

Im Rahmen dieser Arbeit wird über UFT eine XML aus den bereits automatisierten Testfällen alle bekannten Objekten ausgelesen und über in ein Object Repostory eingespielt. Auf diesen wird über den ALM Explorer mit Dropdown Menüs dem Endnutzer Zugriff gegeben. Der gesamte Pfad zu einem Objekt hin muss über dieselbe HP QC GUI erkenntlich sein. Es muss eine Möglichkeit bestehen Objekte die neu hinzugekommen sind trotzdem einzuspielen. Am besten ohne externe Anbindungen und direkt über den Input an der grafischen Oberfläche. Im Besonderen darf das Projekt HP QC nicht deutlich langsamer machen. Wird das Tool bis zur Unbrauchbarkeit langsamer überwiegen die Nachteile den Vorteilen und das Projekt wird überflüssig. [Noch nicht sicher wo das hingehört]

Chapter 6

Implementation

6.1 OTA / tdconnect, Visual Basic Script

Die Open Test Architecture API (OTA) erlaubt es mit tdconnect Zugriff auf HP Application Lifecycle Manager über externe Scripte zu haben. Die Schnittstelle wird mit Visual Basic Script angesprochen. Für diesen Zweck wurde Visual Basic Script C(sharp) bevorzugt da diese Sprache für HPQC am geeignetsten ist. VBS ist gut dokumentiert und verfügt über eine viel größere Bibliothek was die Steuerung des Tools anbelangt. Der Workflow von HPQC ist auch mit VBS beschrieben, was eine Andockung mit der selbigen vereinfacht. C(sharp) scheint zu diesen Zeitpunkt noch nicht für die Software ausgearbeitet genug zu sein.

Creating a connection

```
1 ' Create a Connection object to connect to Quality \
Center
2    Set tdConnection = CreateObject("TDApiOle80.\
    TDConnection")
3 'Initialise the Quality center connection
4    tdConnection.InitConnectionEx qcURL
5 'Authenticating with username and password
6    tdConnection.Login qcID, qcPWD
7 'connecting to the domain and project
8   tdConnection.Connect qcDomain, qcProject
```

Über dieses tdConnection Objekt können wir Objekte für HP ALM erstellen, löschen oder verändern. In unseren Fall sind das selbstgebaute Listen.

6.2 Listen adden + Elemente

```
1 ' custom is the connection with the elements, oList \
  are the lists access
2    Dim custom
3     Set custom = tdConnection.Customization
4     Dim oLists
5     Set oLists = custom.Lists
6
7     'add list for each
8     for each item in listToAdd
9                                     oLists.RemoveList(\
                                      lhtListS & item)
10                                    oLists.AddList(lhtListS\
                                       & item)
11                                    'AddList RemoveList; \
                                      Adding or removing List\
                                       to the queue on the \
                                      list object
12                 next
13    custom.Commit
14 End Sub
```

In diesen Beispiel kann man erkennen das die Listen bevor sie geuploadet werd erst gelöscht werden. Was erst Gegenintuitive erscheinen mag wird so gehandhabt weil Listen die es bereits gibt einfach übersprungen werden. Das hat zur Folge, falls eine Liste nuie Elemente hinzugefügt bekommt, würde diese nicht aktualisiert werden. Praktisch hier kommt das HP-Alm beim Löschen von Listen den Befehl einfach überspringt falls die Liste nicht vorhanden ist. So kommt es zu keinen Problemen.

6.3 Aus XML auslesen

```
1 for each child in xmlDoc.SelectNodes("/qtpRep:\
  ObjectRepository/qtpRep:Objects/qtpRep:Object")
2   objectLevelName = child.getAttribute("Name")
3   objectLevel1.add(filenameXML & "|" & objectLevelName\
    )
4
5   'first iteration, getting the second level
6   For Each child2 In child.SelectNodes("./qtpRep:\
    ChildObjects/qtpRep:Object")
7     objectLevel2Name = child2.getAttribute("Name")
8     objectLevel2.add(objectLevelName & "|" & \
      objectLevel2Name )
```

```
9
10      'third iteration, getting the objects
11        For Each child3 In
12        child2.SelectNodes("./qtpRep:ChildObjects/qtpRep:Object") objectName =
13        child3.getAttribute("Name") objectList.add(objectLevel2Name & "|" & objectName )
14
15        'fourth iteration, getting the objects in case objects have objects
16          For Each child4 In
17            child3.SelectNodes("./qtpRep:ChildObjects/qtpRep:Object") objectName2
18           = child4.getAttribute("Name") objectList2.add(objectName & "|" & objectName2 )
19         next
20       next
21     next
22 next
```

Das Herzstück dieses Projektes, das Durchlaufen der XML. Der erste Schritt ist das auslesen aller Knotenpunkte mit dem Befehl Select Nodes. Dieser beinhaltet zuert allte Objekte in einer einzigen Dimension. Wir können jetzt den ersten Punkt in die Liste als root Listenname setzen. Dieses Rootelement besteht aus dem Dateinamen und dem Attributname. Hier gilt zu beachten das maximal 4 Tiefen existieren wobei nur 3 üblich sind und die letze ein Ausnahmefall ist. Dieser kann bei einer Combobox mit vorgefertigten Elementen vorkommen.

In diesen Fall haben wir eine rekursive linked List erstellt ohne abstrakte Datentypen, sondern nur über den Namen, da VBS diese nicht native unterstützt. An der linken Seite der Pipe sehen wir wo wir gerade sind, auf der rechten das gewünschte Objekt. Der Listenname gibt an an welchen Panel/Browser wir uns gerade befinden. Diese Manipulation ist nötig da beim einlesen die Level der XML verloren gehen und alles Eindimensional dargestellt wird und wir nur Knotenpunkt für Knotenpunkt linear verarbeiten können.

Die Variante wurde einem rekursiven Aufruf bevorzugt da die Schreibweise intuitiver erscheint und damit leichter zu warten ist.

6.4 Listnamen Manipulation

```
1 for each list in listToAdd
2 listName = getRightSideOfPipe(list)
3 for each item in itemToAdd
```

```
4   objectList = getLeftSideOfPipe(item)
5   if(listName = objectList) Then
6     if oLists.IsListExist(lhtListS & list) Then
7     Set myList  = oLists.List(CStr(lhtListS & list))'
      nameList
8     Set listRoot = myList.RootNode
9     objectName = getRightSideOfPipe(item)
10    On Error Resume Next
11      listRoot.AddChild(CStr(objectName))'subname
12    On Error GoTo 0
13
14    else
15      MsgBox("The List: " + lhtListS & list + " does
        not exist or is not yet commited") End If
16    End If
17    next
18  next
19 custom.Commit
```

Der Namen ist mit einer Pipe “|” getrennt. Auf der rechten Seite zeigt dieser an welches Objekt gerade im Focus steht. Im Gegensatz auf der linken Seite steht das Objektlevel, also im welchen Panel, Browser etc das Object platziert ist. Dieser funktioniert auch als Listenname. Sobald der Listenname nicht mehr mit dem Objectlevel übereinstimmt, kreieren wir eine neue Liste.

Der Commit ist der Teil des Programms der am längstem braucht, viele Commits würden viel Zeit beanspruchen, deshalb wird dieser erst ausgeführt sobald wir alle Ebenen und Objekte im Zwischenbuffer oList abgelegt haben.

*instert graphic with name description

6.5 Ergebnis

Chapter 7

Nächste Schritte

7.1 Zukünftiger Einsatz und Ausbau

7.2 Allen Testfällen

Im Moment können nicht alle Testfälle standartisiert werden da ältere XML Dateien andere Formate bzw. der Standard sich erst mit der Zeit so wie er ist entwickelt hat. Diese müssen erst angepasst werden um ältere bereits eingespielte Tests später leichter zu warten sind. [Anstatt dokumentierte Testfälle -> Dokumentation „Synonyme“ -> Standardisieren, einheitliche Beschreibung der Testfälle, Formalisieren-> sollen alle gleich Aussehen]

7.3 Manuelle Eingabe aus der End-User Maske

Zukünftige Veränderung an der Graphischen Oberfläche wir nicht automatisch bei der Testautomatisierung mitverändert. Bei dem Übertragen eines Testfalles wird anstelle des Pfades ein Feld Sonstiges ausgefüllt, das später von der Testfactory eingespielt wird. Eine möglich Feature besteht darin End-user direkt über die Eingabemaske die Dateien im Hintergrund zu verändern. Trotzdem wird eine Rückbestätigung der Testautomatiser sicherer sein.

7.4 Direkt Automatisiert

Im HP ALM sind jetzt Listen aller Möglichen Objekte und den Customisierbaren Listen gepseichert, vorher waren diese nur extern über XML zugänglich. Die bietet einen möglich Ausbau für die Testautomation um diese teil oder volls vom HPQC übernehmen zu lassen.

Fazit

Bibliography