Inhaltsverzeichnis

1 Automatisierung

1.1 Einführung

- Arbeit handelt um Automatisierung
- Warum wurde dieses Thema für die Arbeit gewählt?
- Automatisierung von Programminstallationen
- Betreuung und Analyse eines gesamten zu Automatisierenden Prozesses
- Daraufsicht auf den Prozess der Automatisierung managen und arbeiten nach ISO bzw. ITIL Konformität Praxis 4.

Häufig gibt es gewisse Aufgaben oder Prozesse, welche keine großen Unterschiede in der Abwicklung aufweisen. Für jene kann sich mit der Anwendung von Automatisierung beschäftigt werden. Die generelle Anforderung an Automatisierung ist es Arbeits- oder Produktionsprozesse für den Menschen so durchzuführen, dass dieser nicht unmittelbar tätig werden muss, um Aufgaben zu verrichten. Spezifischer auf das Themenfeld der Projektarbeit bezogen, stellt sich die Aufgabe dar, mit Hilfe von Skripten Installationsprozesse auf ein Minimum von Zeitaufwand zu beschränken. https://m.bpb.de/nachschlagen/lexika/lexikon-der-wirtschaft/18743/automatisierung Das Skript soll also insoweit die Prozesse automatisieren, dass vom Nutzer lediglich ein paar Parameter angepasst werden müssen, die sich bei jeder Installation unterscheiden

2 Task Automation mit Powershell

Der Schwerpunkt dieser Praxisarbeit wurde auf die Implementierung von zu automatisierenden Prozessen in PowerShell gelegt. Es wird beginnend bei der Ideenfindung für einen solchen Prozess, bis zur abschließenden Einbindung, ein umfassender Überblick gegeben, wie Projekt und Aufgabe abgelaufen sind.

2.1 Aufgabenstellung

Als allgemeine Frage, welche es zunächst zu beantworten galt, wurde die Folgende formuliert:

Wie können Prozesse mit Hilfe von Powershell weitestgehend automatisiert werden?

Aus jener Frage lassen sich mehrere Aufgaben ableiten. Einerseits kristallisiert sich daraus das Recherchieren und Verstehen von Prozessen, sowohl im übergeordneten Zusammenhang mit Anderen, als auch der Eigentliche an sich. Geht man beispielsweise von einer Programm Installation aus ist dies der Kernprozess, er hängt aber übergeordnet mit den Berechtigungen die auf dem Betriebssystem gelten zusammen. Andererseits spielt die Aufgabe des strukturierten Dokumentieren und Vorgehen bei der Umsetzung eine große Rolle.

Nicht zuletzt muss sich mit der Verwendung von Powershell, vielmehr der Implementierung in der Powershell ISE, befasst werden. Auf die Aufgabe des Implementierens wurde in diesem Teil der Arbeit das Hauptaugenmerk gelegt.

Ziel der Aufgabe ist es also den gewählten Prozess weitestgehend zu automatisieren. Durch die implementierten Skripte soll dabei eine schnellere und qualitativ gleichbleibende Installation, sowie Konfiguration der Software ermöglicht werden.

2.2 Einordnung der Aufgabenstellung in übergeordnete Prozesse

Ausgewählt wurde diese Aufgabe in der Abteilung Virtualisierung, da bei Digitalen Arbeitsplätzen die Automatisierung solcher Prozesse eine generell recht übliche Prozedur ist. Zudem ist das Aufsetzen und die Verwaltung, insbesondere von virtuellen Desktops, meist nur minimal Abweichend, wenn nicht sogar nahezu identisch in der Durchführung.

Demnach kann das Unternehmen durch die Automatisierung kosten sparen und bietet zudem die Verlagerung von Ressourcen. Ganzheitlich gesehen können durch die gewonnene zeitliche Ersparnis, die Arbeitskräfte in die Umsetzung komplexerer Aufgaben gesetzt werden.

Arbeiten im AD soll vereinfacht werden, dass der Nutzer nicht mehr auf einzelne Schritte zurückgreifen muss.

2.3 Praktische Umsetzung

Zunächst soll im Analyseteil der Aufgabe ein grundlegendes Verständnis zu den ablaufenden Prozessen aufgebaut werden. Hierfür wurde eine kurze Liste mit den in der Abteilung verwendeten Produkten ausgehändigt:

- Citrix Cloud Connector (CC)
- Powershell
- PowerShell ISE

Festgelegt wurde sich auf die Installations Automatisierung des Citrix Cloud Connectors. In den folgenden Unterkapiteln soll ein genereller Überblick über die in dem Projekt verwendete Software geschaffen werden. Dabei soll dem Leser ein klarer Nutzen und Mehrwert des Produkts, sowie eine kurze Heranführung über die verwendeten Funktionen, in dem Projekt, erläutert werden.

2.3.1 Citrix Cloud Connector

Der Citrix Cloud Connector ist ein Produkt, das die Verbindung zur Citrix Cloud ermöglicht.

Allgemein ist das Netzwerk so aufgebaut, dass es Virtuelle Apps und Desktop-Services gibt, die einem Nutzer zur Verfügung gestellt werden sollen. Diese Apps und Services werden von einem Ressourcenblock, der je nach Bedarf extern von Drittanbietern oder auch vom Anbieter selbst direkt bezogen werden kann, bereitgestellt. Der Cloud Connector bietet dabei die Schnittstelle zwischen Nutzer und Ressourcen. Er entnimmt die benötigten Ressourcen und stellt diese dem Nutzer auf seinen Endgeräten zur Verfügung. Der Vorteil eines solchen Systems ist, dass durch diese Netzwerkstruktur der Zugriff, von nahezu jedem Gerät, welches einen Netzwerkzugriff durch den Cloud Connector hat, ermöglicht.

2.3.2 PowerShell

Zur Umsetzung des Skripts wurde das plattformübergreifende Framework PowerShell verwendet. Es ist eine moderne Befehlsshell, die aus verschiedenen Vorteilen anderer Shells entwickelt wurde. Ein Vorteil von PowerShell ist das Akzeptieren und Zurückgeben von .NET-Objekten. [1]

In Bezug auf Skriptsprachen wird sie gerne als Standard zur Konfiguration und automatisierten Verwaltung von jeglichen Systemen genutzt. Dadurch ist sie ideal für die Aufgabe der automatisierten Installationskonfigurationen geeignet.

Für die spätere praktische Anwendung sei zu beachten, dass der Nutzer des Skripts auf dem Zielsystem Administrationsrechte benötigt, um im Vorhinein jeglichen Berechtigungsproblemen präventiv aus dem Weg zu gehen. Zudem sollte das System unter einem Betriebssystem der Firma Microsoft laufen. [2] Sind diese Kernaspekte eingehalten kann das Ausführen von Skripten gewährleistet werden.

2.3.3 PowerShell ISE

Die ISE (Integrated Scripting Environment) ist eine Hostanwendung für PowerShell. Ausgeführt wird diese Anwendung durch den cmdlet ise. Sie bietet eine grafische Oberfläche und dient der Skript Bearbeitung. Durch beispielsweise Syntaxhighlighting sorgt die ISE für eine übersichtlichere Einsicht in das Skript. Funktional bietet sie einen Debugger, Skriptvervollständigung und eine selektive Skriptausführung. [3]

Diese funktionalen Vorteile können jedoch durch Erweiterungen noch weiter verbessert werden. Eine dieser Erweiterungen wird im folgenden Kapitel erläutert.

2.3.3.1 ISE Steroids Um das Erstellen des Skripts weitestgehend zu vereinfachen, wurde auf das Plugin ISESteroids zurückgegriffen. Diese Erweiterung kann durch einen cmdlet, in PowerShell, heruntergeladen und installiert werden. Sobald die ISE gestartet wurde, kann über den Befehl Start-Steroids das Plugin ausgeführt werden. Dies erweitert die ISE um viele verschiedene Funktionen und Einstellungsmöglichkeiten. Beispielsweise können Funktionen einfacher erzeugt, Skriptteile können durch au-

tomatische Vervollständigung gebildet und vorgefertigte Funktionen können direkt erzeugt werden. Die jedoch wichtigste Funktion, die für eine gute und schnelle Implementierung geboten wird, ist der vollwertige Debugger. Mit ihm kann das Skript an gewissen Stellen angehalten und Variablenwerte überprüft werden. Einerseits wird dadurch ein klareres Verständnis über das Skript ermöglicht und andererseits lassen sich logische Zusammenhänge schneller Nachvollziehen. [4]

2.3.4 Vorbereitungen

Zur besseren Vorbereitung, der späteren Implementierung, wurde sich mit einem Entwickler aus der Abteilung vernetzt. Dieser zeigte mir einige praktisch bereits umgesetzte Projekte und händigte Skript Beispiele aus, anhand derer sich orientiert, sowie weitergebildet werden konnte. Eines der vorgegebenen Skriptteile konnte spezifisch für die eigene Aufgabe mit einbezogen werden. Die Inhalte des vorgegebenen Skripts waren allgemeine Installation und Download cmdlets des Cloud Connectors.

Nach dem Austausch setze ich mich mit meinem Betreuer in Verbindung, um die Implementierung und weiteres Vorgehen zu planen. Wie bereits gesagt lag das eigentliche Installationsskript bereits vor, daran mussten nur kleine spezifische Optimierung für die Unternehmensumgebung vorgenommen werden. Diese Aufgabe wurde jedoch zurückgestellt, da sie zunächst nicht elementar war.

Es wurde gemeinsam eine Excel Tabelle mit den zu erledigenden Aufgaben erstellt. Zu den einzelnen Aufgaben sollten klare Termine festgelegt werden. Um diese zu eindeutigen Zeitpunkten erledigt zu haben. Dabei ging es zunächst nicht um die endgültige Fertigstellung und Abgabe der Aufgabe, sondern viel mehr um das gewähren einer besseren Übersicht und klareren Strukturierung des weiteren Vorgehen. Bezogen auf die Ebene des Projektmanagements kann man dabei von festgelegten Sprints reden. Ebenfalls wurde sich zu Daily Huddles verabredet, um sich stetig mit dem Betreuer auszutauschen.

Die Tabelle sieht wie folgt aus:

Aufgabe	Wer	Status	Zieldatum
Erstellung Generalisierung Script zur Übergabe von Parametern.	Louis Löhr	[onGoing]	<24.01>
Erstellung / Überarbeitung Script Basisinstallation Cloud Connectoren	Louis Löhr	[offen]	<04.02>
Erstellung Script Cloud Connector Preferences Check	Louis Löhr	[offen]	<18.02>
Erstellung Script Storefront Preferences Check	Louis Löhr	[offen]	<18.02>

Aufgabe	Wer	Status	Zieldatum
Erstellung Script Storefront Basis Installation	Louis Löhr	[offen]	<25.02>
Erstellung Script Storefront Konfiguration	Louis Löhr	[offen]	<25.02>
Erarbeitung Prozessablauf Diagramm	Louis Löhr	[offen]	
Erarbeitung Dokumentation	Louis Löhr	[offen]	

2.4 Logischer Aufbau

Wie der Tabelle zu entnehmen ist wurde sich zuerst mit der Übergabe der Installationsparameter beschäftigt. Diese Aufgabe beanspruchte in der Praxisphase am meisten Zeit. Die einzige Vorgabe hierfür war es eine Datei mit den Installationsparametern zu erstellen. Diese Datei sollte entweder das Format XML oder JSON haben. Die Parameterdatei darf später für den Endnutzer die einzige zu editierende Datei sein. Ausgelesen werden diese Parameter aus der Datei in dem Hauptskript, welches die Installation dann durchführt. Je nach Art der Installation gibt es zudem einen Source Ordner mit den Installationsdateien. Im Idealfall soll zudem noch eine log-Datei erstellt werden um den gesamten Prozess zu Dokumentieren. Des Überblicks halber wurde ein Technisches Ablauf Diagramm erstellt.

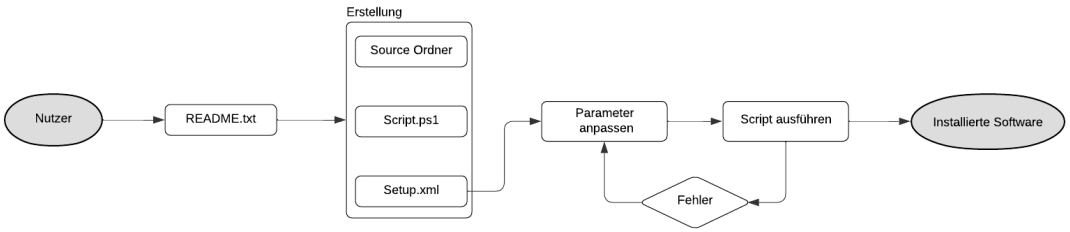

Figure 1: Technisches Ablauf Diagramm

Dieses Ablaufdiagramm soll final eine maximale Übersicht des implementierten Projekts ermöglichen und dem Nutzer dazu dienen die später gegebene Ordnerstruktur besser zu verstehen. Es ist im Verlauf des Projekts stetig ergänzt und angepasst worden.

Zum besseren Verständnis wurden ebenfalls eine ReadMe Datei hinterlegt. Ergänzend ist noch zu Sagen, dass dem Technischen Diagramm die Idee des Parameter Eintragens visualisiert wurde. Fehler die der Nutzer beim Eintragen der Parameter in die Datei macht sollen abgefangen werden und zum korrigieren der Parameter forcieren.

2.5 Automatisierung Citrix Cloud Connector

Dieser Abschnitt thematisiert die gesamte Implementierung des Skripts. Durchweg ist der Aufbau der einzelnen folgenden Abschnitte mit der angeführten Idee und Erklärung, warum sich genau für diese Art der Umsetzung entschieden wurde, strukturiert.

Nachdem ein grundlegendes Verständnis mit Powershell aufgebaut wurde, ging dies über in die Auswahl des Dateityps der Parameterdatei. Festgelegt wurde sich auf das Format XML. Durch die hierarchische Strukturierung der Daten in einer XML kann eine klare Übersicht geschaffen werden. Zudem lassen sich Daten zugänglich und individuell, durch die simple Markup Language, bearbeiten. [5]

2.5.1 Setup.xml

Die Bezeichnung Setup.xml wurde gewählt, um dem Nutzer zu vermitteln, dass dies der Ort zur Anpassung der wesentlichen Parameter für die Installation ist.
Der Aufbau der XML wurde wie folgt gewählt. Setup bildet den Wurzelknoten, auf den die einzelnen Installationsverzeichnisse folgen. Über diesen Knoten kann also später größer skaliert werden. Geht man nun in das Element CloudConnector lassen sich acht Attribute finden. In diesen Attributen werden die Werte/Parameter, welche individuell für jede Installation sind, angepasst.
Bei der größeren Skalierung muss lediglich ein neuer Installationsknoten mit Attributen festgelegt werden.

2.5.2 Main Skript

Im Main Skript vorhanden sind Funktionen zur XML-Zerlegung, dem Downloaden der Installationsdatei und Anweisungen zur Installation im Grundsätzlichen. Der genaue Aufbau wird demnach in den folgenden drei Unterkapiteln erläutert.

2.5.2.1 XML-Parser Zuvor wurden in der Setup.xml alle Attribute mit Werten festgelegt. Die Aufgabe des XML-Parsers ist es nun, sowohl die Attributnamen, als auch deren Werte auszulesen. Der Fokus lag also zuerst auf dem Auslesen der XML-Daten im Allgemeinen. Gelöst wurde dies durch eine foreach-Schleife, die über XML-Pfadbeschreibungen durch die gesamte XML iteriert und die gesammelten Knoten in einem Array speichert. Dieses Array wird dann über eine for-Schleife ausgelesen, um die einzelnen Werte der Knoten in einer neuen Liste zu speichern. Diese Liste wird dann später bei der Installation weiterverwendet.

Nach der Kernimplementierung wurde sich damit beschäftigt das Skript zu optimieren. So wurde sich um die Fehlerbehandlung und Datenprüfung gekümmert. Einen genaueren Einblick hierfür kann sich im Kapitel Optimierung und Fehlerbehandlung eingeholt werden. Dasselbe gilt für das Schreiben und

Erzeugen der log-Datei. Fortgefahren wurde dann damit, dass Skriptteile, die eine spezifische Teilaufgabe verfolgen, in einzelnen Funktionen gebündelt wurden. Dies soll die Erstellung von späteren (anderen) Installationen erleichtern, da man gewisse Funktionen universell anwenden kann.

Begonnen wird mit der Funktion LesenderEingabe{}. Zuerst wird die einzulesende XML-Datei ausgewählt. Abgespeichert muss diese dafür in dem aktuellen Dateiverzeichnis sein. Also jenes in dem der Nutzer das Skript abspeichert und ausführt. Sofern eingehalten wird der gesamte XML-Inhalt in der Variable $xmlFile gespeichert. Über diese Variable wird dann mit Hilfe einer Pfadbeschreibung auf den Installationsknoten verwiesen. Der Knoten beinhaltet den spezifischen Input mit den Attributwerten, die für die Installation wichtig sind. Die ausgelesenen Attribute werden mit ihren Werten als Liste dann in $xmlElements abgespeichert und von der Funktion ausgegeben.

Als nächstes werden von der Funktion Get-Nodes{} die in $xmlElements gelisteten Attribute Ausgelesen und in dem Array $returnElemente abgespeichert. Dies dient dazu um in der darauf folgenden Funktion AusgebenvonNullwerten{} die einzelnen Attribute dahingehend zu überprüfen ob diese mit Werten festgelegt sind. Ist dies nicht der Fall so bricht das Skript ab und fordert den Nutzer dazu auf die Attribute mit fehlenden Werten zu belegen.

Sind alle Attribute mit Werten belegt geht es in die letzte Funktion des Parsers Get-NodeData{}. Get-NodeData{} ist von der Struktur ähnlich zu Get-Nodes{}. Es werden hier jedoch nun die einzelnen Werte der Attribute in $returnValue in Form von einer Liste #text abgespeichert. Dabei ist zu beachten, dass die reinen Werte nur durch den Befehl .'#text' aus dem Array gefiltert werden. So können durch den einfachen Aufruf von Get-Nodes{} überall im Skript die Werte der Attribute genutzt werden.

2.5.3 Optimierung / Fehlerbehandlung

2.6 Formale Komponenten

2.6.1 README.txt

2.6.2 Prozessdiagramm

2.6.3 XML Parsing

2.7 Automatisierung Active Directory

2.7.1 Algorithmus zur Sortierung

- sortieren einzelner AD Richtlinien [**CKursTaschenrechnerMusterlosung?**]

3 Implementierung von automatischen Standard Service Requests

3.1 Aufgabenstellung

Der Kunde besitzt keine Standard Service Requests. Die Aufträge kommen als Incidents und können demnach nicht automatisiert abgearbeitet werden. Das Reporting ist demnach falsch und die ITIL Konformität nicht gegeben. In Audit ist dies als zu verbessernd identifiziert.

Das Ziel der Arbeit ist es sowohl die vollständige Automatisierung als auch Trennung von Incidents und Standard Service Requests durchzuführen.

3.2 Einordnung der Aufgabenstellung in übergeordnete Prozesse

Es soll die Analyse stattfinden, um dies an die jeweiligen Betriebseinheiten zur Umsetzung weiterzugeben.

Die Analyse setzt sich zusammen aus der Absprache mit den einzelnen Betriebseinheiten, zu aktuellem Vorgehen. Im weiteren Verlauf soll dann das erlangte Wissen an Spezialisten weitergetragen und entwickelt werden. Im Nachgang soll dann mit dem Kunden ein Ende zu Ende Test gemacht werden, um den Erfolg und die vollständige Abdeckung der Arbeit zu prüfen. Der Kunde aktiviert das Formular für den erhaltenen Prozess.

3.3 Praktische Lösung

Die erste Aufgabe war das ganzheitliche Nachvollziehen von Prozessen und Strukturen. Gemacht wurde dies um die Analyse korrekt und zielgeführt durchzuführen. Dafür wurden einige verwendete Frameworks, die bei einem SSR (Standard Service Request) kurz erläutert. Im Folgenden werde ich diese kompakt erläutern und aufzeigen inwiefern jene im Zusammenhang mit dem angebotenen Produkt stehen.

Top Desk Service Now Beatbox –> Microsoft Orchestrator

Im Verlauf der Praxiphase kristallisierte sich immer weiter heraus wie das Vorgehen bei einer Prozessautomatisierung gehandhabt wird. Einleitend

Usecaseanalyse

Danach wurde sich mit der Entwicklerabteilung ausgetauscht in wie weit der Prozess aktuell gehandhabt wird

3.4 Themen die beleuchtet werden könnten

- Wie werden zu automatiseriende Prozesse ausgewählt? (Atos arbeitet mit Konzept alles automatiseren was geht) Kunde legt in SLA von Atos fest was automatisert werden soll

3.5 Verknüpfung zu Vorlesungsinhalten

Softwareengeneering Userstories

3.6 Kritische, inhaltliche Reflexion von Theorie und Praxis

Links für 4.Praxis

interner Sharepoint https://atos365.sharepoint.com/sites/190118162/Shared%20Documents/Forms/AllItems.aspx?Fold

4 General Structure

Problemstellung aufführen

Grundlagenaufbau -Literatur mit Ansätzen (Was gibt es schon) -Warum habe ich was wie gemacht (Wenn von Firma vorgegeben trotzdem eventuelle eigene Aspekte alternativ Wege integrieren) -gelerntes im Studium aufbringen

Codehandeling: -in Anhang verlinken -nur interessante/spezifische Aspekte aufzeigen (Segmente des Codes)

Lösung Struktur nach eigenem ermessen Grundlagen wenn notwendig erklären

5 Ausblick von Automatisierung

6 Quellen

[1] sdwheeler. Was ist PowerShell? - PowerShell. Im Internet: https://docs.microsoft.com/de-de/powershell/scripting/overview; Stand: 21.07.2022

[2] Iwaya A. Does PowerShell Work on Other Operating Systems Besides Windows? Im Internet: https://www.howtogeek.com/306261/does-powershell-work-on-other-operating-systems-besides-windows/; Stand: 21.07.2022

[3] sdwheeler. Einführung in die Windows PowerShell ISE - PowerShell. Im Internet: https://docs.microsoft.com/de-de/powershell/scripting/windows-powershell/ise/introducing-the-windows-powershell-ise; Stand: 21.07.2022

[4] . Im Internet: http://www.powertheshell.com/; Stand: 21.07.2022

[5] Extensible Markup Language (XML) 1.0. 1998. Im Internet: https://web.archive.org/web/20060615212726/http://www.w3.org/TR/1998/REC-xml-19980210; Stand: 22.07.2022