TECHNISCHE HOCHSCHULE NÜRNBERG
GEORG SIMON OHM

Fakultät Informatik

Automatisierte Provisionierungsmechanismen für Laufzeitumgebungen von Legacy z/OS Anwendungen mit „IBM Cloud Provisioning and Management for z/OS“ am Beispiel der „Rechnungsschreibung“ bei DATEV eG

Bachelorarbeit im Studiengang Informatik

vorgelegt von

David Krug

Matrikelnummer 3036355

Erstgutachter:	Prof. Dr. Korbinian Riedhammer
Zweitgutachter:	Prof. Dr. Friedhelm Stappert

Hinweis: Diese Erklärung ist in alle Exemplare der Abschlussarbeit fest einzubinden. (Keine Spiralbindung)

TECHNISCHE HOCHSCHULE NÜRNBERG
GEORG SIMON OHM

Prüfungsrechtliche Erklärung der/des Studierenden

Angaben des bzw. der Studierenden:

Name: Vorname: Matrikel-Nr.:

Fakultät: Studiengang:

Semester:

Titel der Abschlussarbeit:

Ich versichere, dass ich die Arbeit selbständig verfasst, nicht anderweitig für Prüfungszwecke vorgelegt, alle benutzten Quellen und Hilfsmittel angegeben sowie wörtliche und sinngemäße Zitate als solche gekennzeichnet habe.

Ort, Datum, Unterschrift Studierende/Studierender

Erklärung zur Veröffentlichung der vorstehend bezeichneten Abschlussarbeit

Die Entscheidung über die vollständige oder auszugsweise Veröffentlichung der Abschlussarbeit liegt grundsätzlich erst einmal allein in der Zuständigkeit der/des studentischen Verfasserin/Verfassers. Nach dem Urheberrechtsgesetz (UrhG) erwirbt die Verfasserin/der Verfasser einer Abschlussarbeit mit Anfertigung ihrer/seiner Arbeit das alleinige Urheberrecht und grundsätzlich auch die hieraus resultierenden Nutzungsrechte wie z.B. Erstveröffentlichung (§ 12 UrhG), Verbreitung (§ 17 UrhG), Vervielfältigung (§ 16 UrhG), Online-Nutzung usw., also alle Rechte, die die nicht-kommerzielle oder kommerzielle Verwertung betreffen.

Die Hochschule und deren Beschäftigte werden Abschlussarbeiten oder Teile davon nicht ohne Zustimmung der/des studentischen Verfasserin/Verfassers veröffentlichen, insbesondere nicht öffentlich zugänglich in die Bibliothek der Hochschule einstellen.

Hiermit ☐ genehmige ich, wenn und soweit keine entgegenstehenden Vereinbarungen mit Dritten getroffen worden sind,

☐ genehmige ich nicht,

dass die oben genannte Abschlussarbeit durch die Technische Hochschule Nürnberg Georg Simon Ohm, ggf. nach Ablauf einer mittels eines auf der Abschlussarbeit aufgebrachten Sperrvermerks kenntlich gemachten Sperrfrist

von Jahren (0 - 5 Jahren ab Datum der Abgabe der Arbeit),

der Öffentlichkeit zugänglich gemacht wird. Im Falle der Genehmigung erfolgt diese unwiderruflich; hierzu wird der Abschlussarbeit ein Exemplar im digitalisierten PDF-Format auf einem Datenträger beigefügt. Bestimmungen der jeweils geltenden Studien- und Prüfungsordnung über Art und Umfang der im Rahmen der Arbeit abzugebenden Exemplare und Materialien werden hierdurch nicht berührt.

Ort, Datum, Unterschrift Studierende/Studierender

Kurzdarstellung

Deutsche Kurzzusammenfassung Zitattest 1 [Kuhn 19] Zitattest 2 [Roge 11]

Abstract

english translation of ‘kurzzusammenfassung‘

Inhaltsverzeichnis

Kapitel 1

Einleitung

Vor mehr als fünfzig Jahren wurde der allererste Großrechner, auch Mainframe genannt, vorgestellt. Seit dieser Zeit setzen sich die monolitisch aufgebauten Systeme in Bezug auf Leistungsfähigkeit und Zuverlässigkeit gegenüber andere Systeme ab. Obwohl die Systeme immer weniger Platz brauchten, anfangs waren es ganze Gebäudestockwerke, heute sind es ungefähr die Ausmaße eines großen Kleiderschrankes. Und weiteren Verbesserung bei der Handhabbarkeit, von reinen Druckausgaben über text-basierenden Terminals bis hin zu benutzerfreundlichen GUI´s. Auch hat sich die Weise, wie Programme entwickelt werden verändert. Zu Beginn mussten diese noch auf Lochkarten (ABBILDUNG !!) gestanzt und umständlich über ein Lesegerät eingelesen werden. Heute stehen dem Entwickler moderne IDE´s zur Verfügung. Trotz dieser Veränderungen verlor der Mainframe durch die Dezentralisierung der IT hinzu Client-Server-Umgebungen in den 1990-er Jahren an Bedeutung. Dieser Prozess führte soweit, dass in den frühen 1990-er Jahren bereits Vorhersagen über die Abschaltung des letzten Mainframes getroffen wurden.[1] [Ceru 03]

Trotz dieser Vorhersagen verarbeiten heutzutage Großrechner weltweit circa 1,2 Millionen CICS[2] Transaktionen pro Sekunde.[3] Im Vergleich hierzu werden 63.000 Google Suchanfragen pro Sekunde abgesetzt. [4] Wie hat es diese schon seit den frühen 1990-er Jahren totgesagte Technologie geschafft auch heute noch diese Relevanz zu haben? Hier kommen die klar definierten Vorteile und Use-Cases des Mainframes zum Tragen. Zunächst ist 'RAS'[5] zu nennen. Dies beschreibt grundsätzlich, die Stabilität eines Hard- und Softwaresystems. Hierzu zählt vor allem das Verhalten bei einem Hardware-/Softwaredefekt und möglichst automatische Erkennung und möglichst effektive Behebung von diesen. Zusätzlich sollte dies keinen oder nur selten einen kompletten Systemausfall zur Folge haben. Abbildung 1.1 zeigt die ungeplante Server Ausfallzeit in Minuten pro Server im Jahr 2019. Wie zu sehen ist, schneidet der IBM z Systems w/Linux oder z/OS, das Mainframesystem der IBM, am besten ab.

[1] [Also 93]
[2] Begrifferklärung zu CICS in 2.1
[3] [IBM 19a]
[4] [Sull 16]
[5] reliability, availability and serviceability

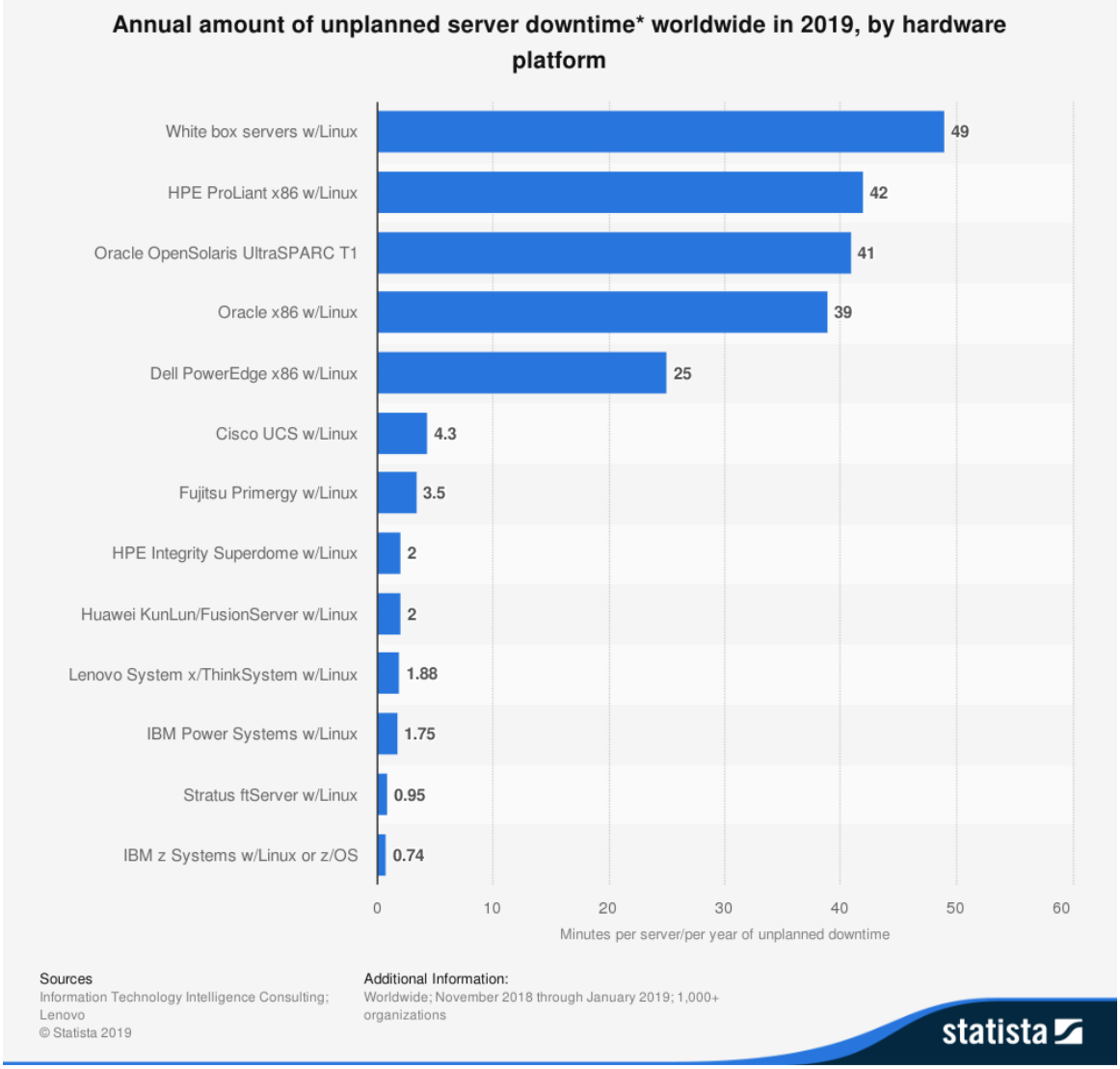

Abbildung 1.1: Annual amount of unplanned server downtime worldwide in 2019, by hardware platform

Hinzu kommen spezielle Sicherheitsmechanismen und Skalierbarkeit. All dies verbunden mit der durch (HIER SPECS EINFÜGEN) gewehrleisteten Performance, ermöglicht spezielle Use-Cases. Unteranderem Massendatenverarbeitung, die dazugehörige Ressourcenverwaltung und Breitband Kommunikation. Das macht den Mainframe vor allem für Banken, das Gesundheitswesen, Versicherungen, Fluggesellschaften usw. attraktiv. Zu diesen Unternehmen zählt auch die DATEV eG. [IBM 14]

Die DATEV eG wurde am 14.02.1966 von 65 Steuerbevollmächtigten gegründet. Sie verfolgten mit der Gründung das Ziel Buchführungsaufgaben mit Hilfe der EDV zu bewältigen. Aufgrund hohen Mitgliederwachstums wurde hierfür 1969 in einen firmeneigenen IBM-Großrechner investiert.[DATE 17] Heute umfasst das Leistungsspektrum der DATEV eG unter anderen das Rechnungswesen, Personalwirtschaft, Consulting, IT-Sicherheit, Weiter-

bildung. Ein nicht unbeträchtlicher Teil (PROZENTSATZ ?) der betriebswirtschaftlichen Anwendungen laufen bis heute auf einem IBM Großrechner. So werden pro Tag circa 150.000 Batch Jobs und circa 90 Millionen CICS-Transaktionen verarbeitet. Hierfür stehen dem System 114.000 MIPS an CPU-Kapazität zur Verfügung. Diese Last wird von circa 14.000 aktiven Modulen erzeugt. Wie in der Abbildung **??** zu sehen ist, ist COBOL mit XXX Prozent die am häufigsten verwendete Programmiersprache am Großrechner bei der DATEV eG. Durch diese Module werden unteranderem im Monat circa 13 Millionen Lohnabrechnungen erstellt und circa 1 Millionen Umsatzsteuer-Voranmeldungen durchgeführt.

Die Risiken, die sich für die DATEV eG durch die Nutzung eines IBM Großrechners ergeben, werden im Folgenden dargestellt.
Zunächst ist zu nennen, dass die Verfügbarkeit von Skills im Mainframebereich immer schlechter wird. Die aktuellen Wissensträger fallen durch ihr Alter langsam aus. Durch die geringe Beliebtheit und wenig Präsenz an Universitäten und Hochschulen sind junge Nachfolger nur schwer zu finden. So ist zum Beispiel die Programmiersprache COBOL auf den TIOBE Index Platz 28.[6] Zum anderen gibt es in Deutschland nur XXX Universitäten und Hochschulen, die einen Studiengang mit Schwerpunkt Mainframe anbieten.[**?**]

Als nächstes ist die Herstellerabhängigkeit von IBM zu erwähnen. Die DATEV eG ist nicht nur in der Wahl des Betriebssystems eingeschränkt, sondern auch einem Datenbanksystem oder einer Messaging Lösung. Außerdem hat die IBM eine Quasi-Monopolstellung[**?**] im Mainframebereich, so ist die DATEV eG auch in ihrer Preisverhandlungspolitik eingeschränkt. Hinzu kommt die Abhängigkeit von der IBM Mainframe Strategie, also ob die IBM selbst noch weitere Ressourcen in ihren Großrechnerbereich investiert. Dies wird durch eine sinkende Kundenzahl am Markt verstärkt.[7]

Für die DATEV eG ist trotz der Risiken eine Ablösung der Mainframe Bestandsanwendungen durch cloud-native Anwendungen aktuell nicht absehbar. Ein weiterer Punkt ist, dass die Sicherung des Bestandsgeschäfts über einen Zeitraum von Jahren oder Jahrzenten notwendig sein wird. Aufgrund der Stärken des Großrechners will man bei Modernisierungsprojekten eine Alternative für die cloud-native Entwicklung anbieten. Um dieses Ziel zu erreichen, wurde bei der DATEV eG in den letzten Jahren der Entwicklungsprozess für Mainframe-Projekte überarbeitet und diesen an übliche Entwicklungsstandards anzupassen. So wird auf eine von der IBM entwickelten auf eclipse-basierende Entwicklungsumgebung gesetzt. Somit ist die DATEV eG auch in diesem Bereich von der IBM abhängig. Außerdem läuft seit einiger Zeit die Umstellung auf git für die Verwaltung von COBOL und IBM-Assembler Sourcen.

Zum Entwicklungsprozess zählt jedoch nicht nur das Erzeugen von Code, sondern auch die Bereitstellung der dazugehörigen Laufzeitumgebung. So gewinnen bei der DATEV eG,

[6] [TIOB 19]
[7] weltweite installierte MIPS-Zahl sei laut IBM steigend

außerhalb des Mainframeumfelds, PaaS (Plattform as a Service) Ansätze immer mehr an Bedeutung. Ein Vorteil dabei ist die unkomplizierte, automatisierte Provisionierung von Laufzeitumgebungen. Dadurch wird die Entwicklungsgeschwindigkeit erhöht und die Bereitstellung von isolierten Testumgebungen vereinfacht. Außerdem können während eines laufenden Entwicklungsprozesses Komponenten, wie zum Beispiel ein Datenbanksystem, hinzugefügt oder ausgetauscht werden. Um sich auch in diesem Bereich der großrechnerfremden Entwicklung anzupassen, sucht die DATEV eG hier nach einer Lösung. Für die automatisierte Provisionierung von Laufzeitumgebungen im Mainframeumfeld stellt die IBM seit dem Jahr 2019 das Tool 'IBM Cloud Provisioning and Management for z/OS'[8] zur Verfügung. Anhand des Beispiels der Rechnungsschreibung [9] wird in dieser Arbeit untersucht, ob und wie es möglich ist, ähnlich wie bei einem PaaS Ansatz, solche Provisionierungsmechanismen für z/OS Anwendungen zu automatisieren.

[8] Beschreibung in Absatz 2.2
[9] Beschreibung im Kapitel 4.1.1

Kapitel 2

Grundlagen

In diesem Kapitel werden für diese Arbeit wichtige Begriffe und Systeme erläutert.

2.1 Customer Information Control System

Das Customer Information Control System, kurz CICS, ist ein Applikationsserver für einen IBM-Großrechner. Ein Applikationsserver stellt eine Umgebung zur Verfügung, in der Anwendungen gehostet werden können. Dabei kümmert sich dieser unter anderem um Transaktionalität, Webkommunikation und Sicherheit. Hierfür stellen Applikationsserver eine API zur Verfügung. CICS hat einen weiteren Vorteil, es unterstützt verschiedene Programmiersprachen. So können Programme innerhalb einer Anwendung in der für ihren Use-Case am besten geeigneten Sprache implementiert werden. Zu den unterstützten Sprachen zählen neben COBOL und IBM Assembler auch Java und Java EE. [Rayn 11]

2.1.1 CICS Transaktion

Ein Businessablauf wird im CICS in einer Transaktion gekapselt. So kann eine Transaktion mehrere Programme unterschiedlicher Programmiersprachen umfassen. Eine Transaktion besitzt ein eindeutiges Kürzel, die TransaktionsID. Über die TransaktionsID kann der Ablauf gestartet werden. Dies kann sowohl per Webanfrage oder per Messaging Queue als auch aus einem anderem Programm heraus oder per Hand geschehen. In der Transaktion werden alle Änderungen die Programme an Resourcen, wie zum Beispiel einer Datenbank oder Dateien, tätigen protokolliert. So wird im Fehlerfall sichergestellt, dass diese rückgängig gemacht werden können. [Rayn 11]

2.1.2 Voraussetzungen

Im Umfeld der DATEV eG sind die Hard- und Softwarevorausetzungen um ein CICS beziehungsweise eine CICS Instanz zu erstellen und zu starten vorhanden. Der Fokus dieser Arbeit liegt auf letzterem somit werden nur die dafür notwendigen Voraussetzungen dargelegt. Außerdem liegt der Fokus nur auf Systemen, die vorerst nicht für die produktiven Systeme der DATEV eG vorgesehen sind. Aus diesem Grund werden nur Schritte, die für ein solches Testsystem benötigt werden, dargestellt. Eine weitere Eingrenzung besteht darin, dass nur die Arbeitsschritte, die mit z/OSMF[1] automatisiert werden, erläutert werden.

2.1.3 Einrichtung CICS Instanz

Die in diesem Absatz benötigten Informationen stammen aus Gesprächen mit Mitarbeiter 2 aus der Abteilung, die für die CICS Administration zuständig ist. Um eine lauffähige CICS Instanz den Vorausetzungen aus dem Absatz 2.1.2 entsprechend einzurichten, sind mehrere Schritte notwendig. Diese werden im Folgendem beschrieben.

2.1.3.1 CICS spezifische Dateien

Zunächst müssen CICS spezifische Dateien im z/OS angelegt werden. Im Falle dieser Arbeit handelt es sich um 17 verschiedene. Diese Dateien benötigt die CICS Instanz um zum Beispiel Systemfehler zu protokollieren. Eine weitere Datei ist dafür zuständig, dass ein Debugger innerhalb der Instanz verwendet werden kann.

2.1.3.2 CSD

In der CICS system defintion, kurz CSD, Datei muss jede Ressource, die dem System zur Verfügung stehen soll, definiert werden. Eine CSD Datei kann für mehrere CICS Instanzen verwendet werden. Eine solche allgemeine CSD Datei hat ca. 22.600 Einträge. Ein Eintrag besteht aus einer Gruppe und einer Liste. Die Gruppe ist hierbei die Definition einer Systemressource und muss händisch angelegt werden. Bei der Liste handelt es sich um das System, welches diese Ressource benötigt. Dort ist unter anderem für jede CICS Instanz hinterlegt, zu welchem Db2 Datenbanksystem und welchem MQ Messagingsystem sich diese Instanz verbinden soll.

[1] Beschreibung in Absatz 2.2

2.1.3.3 STC Job

Bei einem Started Task Controll-Job, kurz STC Job, handelt es sich um einen Batch Job, der mit Hilfe des 'START'-Konsolenkommandos innerhalb von z/OS gestartet werden kann. Dieser Batch Job wird deshalb auch als Started Task bezeichnet.[Cass 07] Bei der DATEV eG existiert für jedes CICS ein solcher Job. In diesem werden zunächst einige zur Laufzeit benötigten Bibliotheken und Dateien eingebunden, unter anderem die CICS spezifischen Dateien[2]. Außerdem werden hier die SIT [3] Parameter definiert. Zunächst wird festgelegt welche Standard SIT verwendet werden soll. Anschließend können diese Standardwerte überschrieben werden. Zu diesen Parameter zählen unter anderem, der eindeutige Name der CICS Instanz, der Speicherort der dazugehören CSD und ob eine Verbindung zu einem Db2 Datenbanksystem hergestellt werden soll.

2.1.4 Entfernung CICS Instanz

Um eine CICS Instanz zu entfernen muss diese zunächst gestoppt werden. Dies ist über das 'STOP'-Konsolenkommando von z/OS möglich. Anschließend müssen alle im Absatz 2.1.3 beschriebene Schritte rückgängig gemacht werden. Also müssen die für diese Instanz spezifischen Dateien, die Einträge für die CICS Instanz aus der CSD Datei und schließlich auch der STC Job gelöscht werden.

2.2 IBM Cloud Provisioning and Management for z/OS

In diesem Absatz wird zunächst auf die für dieses Kapitel grundlegenden Begriffe eingegangen. Anschließend wird IBM Cloud Provisioning and Management for z/OS, kurz z/OSMF, erläutert. Im Anschluss darauf wird das auf Kommandozeilenbefehle basierende z/OS Provisioning Toolkit, kurz z/OS PT, und dessen Möglichkeiten dargestellt.

2.2.1 Begrifferklärung

Im folgenden werden einige allgemeine Begriffe, die im Umfeld von IBM Cloud Provisioning and Management for z/OS vorkommen, erläutert.

[2] Beschreibung in Absatz 2.1.3.1
[3] CICS system initialization table

2.2.1.1 Workflow

Ein Workflow ist eine beliebig komplexe eindeutige Aneinanderreihung von sogenannten Steps. Nach der Ausführung dieser wird ein bestimmtes Ziel erreicht, zum Beispiel die erfolgreiche Bereitstellung eines CICS Systems. Die Definition eines Workflows, den dazugehörigen Steps und ihrer Variablen wird in XML umgesetzt. Ein Step beschreibt einen Teilablauf eines Workflows. Innerhalb eines Steps können sowohl interne und externe Scripte als auch JCLs und somit Programme ausgeführt werden. Des weiteren besteht die Möglichkeit REST-Calls zu tätigen. Außerdem können Bedingungen für die Durchführung eines Steps definiert werden. So ist es zum Beispiel möglich einen Step nur durchzuführen, wenn eine bestimmte Variable einen bestimmten Wert besitzt. Ein weiteres Beispiel ist, es können erforderliche Steps definiert werden, so dass bevor ein Step auf eine Datei zugreift, mittels eines vorherigen Steps geprüft wird ob diese vorhanden ist und wenn nicht diese erzeugt. Es wird ein XML Schema verwendet um sicherzustellen, dass zur Laufzeit keine syntaktischen Fehler vorhanden sind. [Rott 18]

Ein Nachteil von Workflows ist, dass diese statisch sind, das heißt, dass die Variablenzuweißungen immer zum Zeitpunkt der Erstellung stattfindet. Dadurch ergibt sich, dass für jede kleine Änderung ein eigener Workflow erzeugt werden muss. Somit ist ein Workflow eher ein Einmal- bzw. Wegwerfprodukt.

2.2.1.2 Template

Bei dem Nachteil von Workflows als Wegwerfprodukt setzen die sogenannten Templates an. Ein Template besteht aus drei Dateien.

Einer Datei für Eingabevariablen. In dieser Datei können Workflowvariablen Werte zugewiesen werden. Diese Variablen müssen bei ihrer Definition im Workflow entsprechend gekennzeichnet sein.

Die nächste Datei ist die sogenannte Aktion-Definitions-Datei. Hier werden die Aktionen, die ein Anwender mit diesem Template durchführen kann, festgelegt. Einer Aktion wird eine Workflow Definitions Datei und somit ein Workflow zugewiesen. Dabei ist zu beachten, dass die Datei für die Eingabevariablen und welche Variablen davon verwendet werden, anzugeben ist.

Als letzte Datei ist die Manifest-Datei zu nennen. In dieser wird dem Template mitgeteilt an welchem Speicherort sich die oben genannten Dateien befinden. Da ein Template immer provisioniert werden kann, wird hier auch der Speicherort des Bereitstellungsworkflows angeben. Zusätzlich kann noch eine Beschreibung des Templates hinzugefügt werden.

Somit bildet ein Template einen Rahmen um mehrere Workflows und ermöglich so schnellere De-/provisionierung. Zudem können die Variablen nur an einer Stelle geändert werden. Außerdem besteht die Möglichkeit, den Variablen zum Zeitpunkt der Provisionierung als Anwendereingabe einen Wert zuzuweisen. Somit ist ein Template flexibler als ein Workflow. [IBM 19a]

2.2.1.3 Instance

Hierbei handelt es ich um das Ergebnis nach der Provisionierung eines Templates. Zum Beispiel eines funktionsfähigen CICS.

2.2.2 IBM Cloud Provisioning and Management for z/OS

Das IBM Cloud Provisioning and Management for z/OS bietet die Möglichkeit mehrere Systeme innerhalb eines z/OS Betriebssystems zu provisionieren, unter anderem Laufzeitumgebungen wie CICS. Jedoch nicht die Bereitstellung eines kompletten z/OS Betriebssystems. Für diese Aufgaben stehen zwei Schnittstellen zur Verfügung. Zum einem z/OS Provisioning Toolkit, im Weiteren z/OSPT genannt, und zum anderen z/OS Management Facility, im Weiteren z/OSMF genannt. [Keit 16]

2.2.2.1 z/OS Provisioning Toolkit

z/OSPT bietet ein Kommandozeileninterface für die Bereitstellung und das Verwalten von Laufzeitumgebungen. In Abbildung 2.1 werden die möglichen Kommandozeilenbefehle mittels des Befehls ‘zospt -h‘ in einem Kommandofenster angezeigt. Mit z/OSPT werden noch zwei weitere Begriffe eingeführt.
Zum einen sogenannte ‘images‘. Dabei handelt es sich grundsätzlich um ein Template, jedoch kann dieses Template über eine weitere Inputfile verändert werden. Dadurch kann ein Template mit kleineren Änderungen provisioniert werden, ohne dass ein neues Template erzeugt werden muss. Dies erhöht die Flexibilität weiter.
Des anderen die sogenannten ‘container‘. Dabei handelt es sich eins zu eins um eine Instance. [IBM 19b]

2.2.2.2 z/OS Management Facility

Der Hauptaugenmerk dieser Arbeit liegt jedoch bei z/OSMF. Da dieses die Verwaltung von Workflows und Templates über eine browserbasierende Schnittstelle ermöglicht. Durch diese

```
$ zospt -h
IBM z/OS Provisioning Toolkit V1.1.5

Usage: zospt [OPTIONS] COMMAND [arg...]

Options:
 --version   : Displays the command line version.
 -h (--help) : Displays the command line help.

Commands:
    build       PATH [-h (--help)] -t (--tag) <imageName>               Build an image
    images      [-h (--help)]                                           List all images
    inspect     <imageName> | <containerName> | <containerId>           Inspect an image or a container
                [-h (--help)]
    rm          <containerName> | <containerId> ... [-f (--force)]      Remove one or more containers
                [-h (--help)]
    rmi         <imageName> ... [-h (--help)]                           Remove one or more images
    run         <imageName> [--draft]                                   Run an image in a new container
                [--link <containerName> | <containerId>:<alias>]
                [--name <containerName>] [-h (--help)] [-q (--quiet)]
    start       <containerName> | <containerId> ... [-h (--help)]       Start one or more containers
    stop        <containerName> | <containerId> ... [-h (--help)]       Stop one or more containers
    ps          [-a (--all)] [-f (--filter) <filter>] [-h (--help)]     List containers

Run 'zospt COMMAND --help' for more information on a command.
```

Abbildung 2.1: z/OSPT mögliche Kommandozeilenbefehle

Oberfläche, in Abbildung **??** dargestellt, ist es einfacher zu bedienen und somit wird der Einstieg in die Provisionierung erleichtert.

//hier zosmf welcomepage screenshot

Wie auf der rechten Seite der Abbildung **??** zu sehen ist, bietet z/OSMF viele Funktionen an. Für diese Arbeit besitzt nur der Menüpunkt ‘Cloud Provisioning‘ Relevanz. Unter diesem Punkt sind die Funktionalitäten für die automatisierte Bereitstellung von Templates zu finden. [Rott 18]

Dabei handelt es sich zunächst um das ‘Resource Management‘. Darunter werden sogenannte ‘Domains‘ und die dazugehörigen ‘Tenants‘ verwaltet. Unter einer ‘Domain‘ ist ein System, das Systemressourcen in Ressourcenpools gliedert, zu verstehen. ‘Tenants‘ sind die dazugehörigen Rechtegruppen, die dem Anwender den Zugriff und die Nutzung von zugeordneten Templates ermöglicht. Einem Template muss sowohl eine ‘Domain‘ als auch ein ‘Tenant‘ zugewiesen werden. [Rott 18]

Zur Verwaltung der Templates und Instances kommen die ‘Software Services‘ zum Einsatz. Dort können neue Templates über Manifest Datei hinzugefügt werden. Dann muss, wie oben beschrieben, eine ‘Domain‘ und ein ‘Tenant‘ zugwiesen werden. Anschließend kann das Template, falls es keine Fehler beinhaltet, veröffentlicht werden. Es ist zu empfehlen vorher einen ‘Test Run‘ durchzuführen. Dabei wird eine Instance testweise provisioniert. Diese Instance verhält sich genauso wie eine Instance, die aus einem veröffentlichten Template erzeugt wurde. Somit kann damit das Template und die in der Aktion-Definitions-Datei definierten Aktionen getestet werden. [Rott 18]

Kapitel 3

Vorgehensweise

Zu Beginn dieser Arbeit war eine Einarbeitung nicht nur in das verwendete Toolkit notwendig. Sondern es musste sich viel mit den verschiedenen Systemen, also CICS, Db2 und MQ, im Hinblick auf administrative Aufgaben auseinandergesetzt werden. Nachdem eine Beispielanwendung gefunden wurde, folgte die Analyse des Ist-Zustandes inklusive der Beschreibung dieser Anwendung. Während der Analyse wurde der momentane Bereitstellungsprozess untersucht.

(evtl. Workshop erwähnen)

Die Installation des Toolkits geschah bereits vor dem Beginn dieser Arbeit. Somit ist es möglich auf dem Testplex, einer Systemumgebung für Test von neuen Betriebssystemversionen oder ähnlichem, zu beginnen. Dieser wird hauptsächlich von Administratoren genutzt. Zusätzlich ist dieser komplett von anderen Systemumgebungen abgekapselt, um unvorhersehbare Fehler zu vermeiden. In dieser Umgebung wird zunächst untersucht, wie es möglich ist ein CICS zu provisionieren. Hierfür wird vorerst ein von der IBM bei der Installation von z/OSMF mitgeliefertes minimales CICS Template verwendet. Anschließend wird ein umfangreicheres mitgeliefertes Template an die Anforderungen der Anwendung angepasst. Da im Testplex jedoch keine Anwendungsdaten vorhanden sind, wird vorerst nur die benötigten Systeme provisioniert, um so die Grundvoraussetzungen zu schaffen.

Nachdem eine CICS Instanz, eine Db2 Datenbank und MQ Queues auf dem Testplex sowohl provisioniert als auch deprovisioniert werden können, folgt der nächste Schritt. Dabei handelt es sich um den Wechsel der Systemumgebung vom Testplex auf die Entwicklungssystemumgebung. Letzteres ist die Testumgebung für alle Mainframe Entwickler. Hier werden vor allem neue Programmversionen entwickelt und damit kleinere Tests durchgeführt. Außerdem sind in der Entwicklungssystemumgebung alle Anwendungsdaten, die für diese Tests notwendig sind, vorhanden. Somit kann die Integrierung der Beispielanwendung in das provisionierte CICS stattfinden.

Zuletzt wird eine Diskussionsrunde stattfinden. Hierbei werden Kollegen aus allen beteiligten Gruppen teilnehmen. Dazu zählen CICS-Administratoren, Db2-Administratoren, MQ-Administratoren, Entwickler der Beispielanwendung und Architekten. Zunächst wird das

Ergebnis dieser Arbeit vorgestellt. An Hand dessen wird diskutiert, ob und wenn ja wie das ‘IBM Cloud and Management for z/OS Toolkit‘ verwendet werden soll.

Kapitel 4

Analyse

Im Folgendem erfolgt eine Beschreibung der Beispielanwendung ‘Rechnungsschreibung‘. Die dafür benötigten Informationen stammen aus Gesprächen mit Mitarbeiter 1 aus der Abteilung, die für die Rechnungsschreibung zuständig ist. Hierbei wird vor allem der technische Aspekt beleuchtet. Anschließend wird der aktuelle Bereitstellungsprozess für Laufzeitumgebungen, den dazugehörigen Datenbanksystem und einer Messaging Lösung dargestellt.

4.1 Rechnungsschreibung

Für diese Arbeit wurde die Rechnungsschreibung als Beispielanwendung herangezogen, weil sie folgenden Anforderungen entspricht. Es handelt sich zum einem um eine in sich abgeschlossene Anwendung, die nur zu Beginn des Prozesses von anderen Anwendungen abhängig ist. Zum anderen benötigt die Rechnungsschreibung ein CICS als Laufzeitumgebung, eine Db2-Datenbank und MQ als Messaginglösung. Somit kann ein umfangreicher Bereitstellungsmechanismus in dieser Arbeit untersucht werden.

4.1.1 Beschreibung

Die Erzeugung der Rechnungen lässt sich in mehrere Schritte unterteilen, gesammelt werden diese Schritte als Rechnungsschreibung bezeichnet.
Bei dem Ablauf handelt es sich um einen Batch[1]-Ablauf, der auf einem Großrechner läuft. Nur die Preisermittlung wird in ein CICS ausgelagert. Zunächst wird nach jeder kostenpflichtigen Leistungserbringung durch die dazugehörige Anwendung ein Berechnungssatz erzeugt. Ein Berechnungssatz beinhaltet die Metainformationen der Berechnung unter anderem die Artikelnummer, Menge und den Ordnungsbegriff. Der Preis und der Rechnungsempfänger wird zu einem späteren Zeitpunkt innerhalb der Rechnungsschreibung ermittelt.

[1] Stapelverarbeitung

4.1.1.1 Einpflegung Berechnungssätze

Für das Einpflegen der Berechnungssätze in den Rechnungsschreibungsablauf stehen den Anwendungen drei Möglichkeiten zur Verfügung.
Bei der Ersten Möglichkeit handelt es sich um die Verwendung des DMVINF[2]-Moduls und der dazugehörigen Schnittstelle. Dieses Modul ist in der Programmiersprache Assembler entwickelt worden. Das Ergebnis dieses Moduls ist eine sequenzielle Datei am Großrechner, dieses Format lässt sich mit einer .txt Datei unter Windows vergleichen. Diese Datei, auch Berechnungsdatei genannt, hat folgenden Aufbau. Der erste Satz enthält Steuerinformationen, wie zum Beispiel Datum/Uhrzeit, Produkt usw. Danach kommen die eigentlichen Berechnungssätze. Schließlich folgt noch die Anzahl der Sätze und die Summe der einzelnen Artikel in einem Satz mit Kontrollinformationen. Diese Kontrollinformationen werden im weiteren Verlauf mit den eingelesenen Werten abgeglichen, dadurch wird Datenverlust und unkontrollierte Eingriffsmöglichkeiten von außen ausgeschlossen. Aus dem Aufbau einer solchen Datei lässt schließen, dass verschiedene Schritte für die Erzeugung innerhalb der Anwendung notwendig sind.

Für die nachfolgenden Schritte stellt das DMVINF-Modul jeweils Schnittstellen zur Verfügung. Zuerst wird beim sogenannten Open die Datei erstellt und der Steuersatz geschrieben. Danach folgt das eigentliche Schreiben der Berechnungssätze, dabei dürfen nur bestimmte Felder (Ordnungsbegriffe, Länderschlüssel und Mengen) verändert werden. Um unzulässige Veränderungen zu verhindert, haben diese einen Abbruch der Verarbeitung zur Folge. Schließlich folgt noch der ‘Close‘ bei dem die Kontrollinformationen geschrieben werden. Hinzuzufügen ist, dass die variablen Informationen einer Formalprüfung unterzogen werden. So entstehen je nach fachlicher Logik und Laufhäufigkeit der Anwendung mehrere Berechnungsdateien.

Eine weitere Möglichkeit die Berechnungsinformationen in den Ablauf einzupflegen ist die Übergabe über einen mit der Programmiersprache Java realisierte WebService. Hier werden die Berechnungsinformationen im XML-Format bereitgestellt. Das Ergebnis der entsprechenden Plausibilitätsprüfungen, die in einem Onlineverfahren durchgeführt werden, wird direkt an die aufrufende Anwendung zurückgegeben. Sind die Daten korrekt werden diese vorerst in einer Datenbank gespeichert. Vor dem nächsten Schritt wird diese Datenbank ausgelesen und mit der ersten Möglichkeit in den Kernablauf eingespeist.

Bei der letzten Möglichkeit handelt es sich um die Übergabe mittels einer CSV-Datei. Die Datei wird auf den Großrechner übertragen und dort mit dem DMVINF-Modul verarbeitet. Dieses Verfahren wird kaum von produktiven Anwendungen sondern hauptsächlich für Test- oder Qualitätssicherungszwecke genutzt.

[2] DatevMakroVerarbeitungsinformation

Mittels dieser drei Möglichkeiten werden insgesamt monatlich circa 30 Millionen Datensätze bereitgestellt und weiterverarbeitet. Diese Datensätze stehen innerhalb der durch das DMVINF-Modul erzeugten Berechnungsdateien dem weiteren Verlauf als Input zur Verfügung. Um sicher zu stellen, dass all diese Dateien auch verarbeitet werden, wird bei Erstellung einer solchen ein Eintrag in eine Kontrolldatei vorgenommen. In dieser Kontrolldatei wird jedes Lesen und somit auch das Lesen im weiteren Verlauf gekennzeichnet. Eine monatliche Überprüfung führt die zuständige Abteilung durch.

4.1.1.2 Tägliche Bewertung

Der nächste Schritt des Rechnungsschreibungsprozesses ist die sogenannte Tägliche Bewertung. Dieser Ablauf läuft einmal täglich von Montag bis Freitag und ist für die Preis- und Rechnungsempfängerermittlung zuständig. Zur Realisierung wurden die Programmiersprachen Assembler und COBOL genutzt. Am Ende dieses Ablaufes steht die ARUBA[3]-Db2-Datenbank. Dort werden die Berechnungsdaten der letzten 36 Monate aufbewahrt. Dabei handelt es ich um insgesamt circa 3,8 Milliarden Datensätze von einer Gesamtgröße von circa 400 GB mit Indizes. Diese Datensätze beinhalten alle Informationen für die endgültige Erzeugung der Rechnungen.

Der erste Schritt der Täglichen Bewertung ist das Zusammenführen der Berechnungsdateien aus dem vorherigen Schritt und aus den bereits vorhandenen Daten des laufenden Monats aus der ARUBA-Db2-Datenbank. Zusätzlich werden während dieser Zusammenführung den Berechnungssätzen auf Basis der abgebenden Anwendung die entsprechenden Rechnungsstellungsrythmen (täglich oder monatlich) zugewiesen. Anschließend wird mit Hilfe der Beraternummer die zugehörigen Betriebsstätte-, Rechnungsempfänger-, Hauptberater- und Mitglieds- bzw. Geschäftspartnernummer ermittelt. Die Beraternummer ist als oberster Ordnungsbegriff in den Berechnungssätzen enthalten. Außerdem wird die Debitorenkontonummer entweder durch die Mitglieds- oder durch die Geschäftspartnernummer zugeordnet. Für die Preisermittlung werden die Datensätze nach Geschäftspartner gruppiert. Im DATEV eG Umfeld ist ein Geschäftspartner entweder eine Kanzlei oder ein einzelner Mandant, dieser ist jedoch meist einer Kanzlei zugeordnet.

Dann werden die gruppierten Daten auf Grund der Performance an ein CICS übertragen. Die Architektur wird in 4.1.2 beschrieben. Dort findet die Preisermittlung mit den dazugehörigen kundenindividuellen Abhängigkeiten, wie zum Beispiel Rabatte, statt. Anschließend werden die Daten wieder zurück an den Batch-Ablauf übertragen. Hier werden die Rechnungsnettobeträge geprüft, ob diese über einem bestimmten Rechnungslimitbetrag liegen. Falls dies nicht der Fall ist, werden die Berechnungssätze als BUL[4] gekennzeichnet und in die folgende Rechnungsperiode vorgetragen. Schließlich wird noch die Umsatzsteuer ermittelt.

[3] Abrechnungs- und Umsatz-Basis
[4] Berater unter Limit

Abschließend werden die neu erzeugten Berechnungsdaten in die ARUBA-Db2-Datenbank eingepflegt und entsprechend gekennzeichnet.

4.1.1.3 Rechnungsaufbereitung

Als letzter Schritt folgt die Rechnungsaufbereitung. Diese erfolgt am ersten Werktag im Monat. Mit Hilfe der ARUBA-Db2-Datenbank wird ermittelt, welchen Kunden eine Rechnung zugestellt werden muss. Außerdem wird dabei der Zustellungsweg, per Post oder E-Mail, bestimmt. Darauf folgt die Aufbereitung der Druckrohdaten und letztlich das Versenden der Rechnungen an die Berater. Zusätzlich werden die Rechnungen noch im PDF-Format archiviert.

4.1.2 Architektur

In dieser Arbeit steht das automatisierte Provisionieren von Laufzeitumgebungen im Fokus. In diesem Fall handelt es sich um die Laufzeitumgebung CICS. Deshalb wird im Folgenden nur darauf eingegangen.

Das System muss an Lasttagen bis zu 180.000 Geschäftspartner verarbeiten können. Um all diese an das CICS zu übertragen stehen dem System mehrere IBM MQ Queues zur Verfügung. Darunter ist eine allgemeine Queue in der alle Aufträge, die für die Weiterverarbeitung zur Verfügung stehen, geschrieben werden. Pro Geschäftspartner wird ein Auftrag angelegt. In diesem Auftrag befinden sich die Namen vier weiterer Queues. Eine dieser Queues beinhaltet alle Informationen, die für die Preisermittlung des dazugehörigen Geschäftspartners notwendig sind. In den restlichen drei Queues sind die Ergebnisse der Preisermittlung gespeichert. Die Queuenamen werden nicht dynamisch generiert, da dies zu Performanceproblemen führt. Deshalb existieren für jede der vier Queues jeweils 100 vorgefertigte Namen. Somit können auch maximal nur 100 Aufträge gleichzeitig auf Weiterverarbeitung warten. Falls dieses Limit erreicht ist, wartet der Batch-Ablauf solange bis einer der Aufträge fertig gestellt wird. Sobald ein Auftrag in die allgemeinen Auftragsqueue geschrieben wird, wird eine CICS-Transaktionen gestartet. Diese führt die Preisermittlung durch und schreibt das Ergebnis auf die dazugehörigen Queues. Ist dies geschehen stehen die Queues wieder für einen neuen Auftrag zur Verfügung. Es können maximal 30 Transaktionen zeitgleich arbeiten.

Für die Preisermittlung wird auch eine Db2-Datenbank, in der die Einzelpreise der Artikel gespeichert sind, verwendet. Wenn alle Transaktionen direkt auf diese Datenbank zugreifen würden, hätte dies über 60 Millionen Datenbankzugriffe zur Folge. Dies führt zu massiven Einbußen bei der Performance. Deshalb werden bevor die eigentliche Preisermittlung stattfindet, alle benötigten Einzelpreise und Preisabhängigkeiten ermittelt. Diese Informationen werden dann in einen sogenannten ‘SHARED GETMAIN‘-Bereich gespeichert. Dabei

handelt es sich im Prinzip um einen Hauptspeicherbereich des Großrechners. Die Adresse von diesem Bereich wird dem Ablauf zur Verfügung gestellt. Somit greifen die einzelnen Transaktionen nicht mehr direkt auf die Datenbank zu, sondern auf den schnelleren Hauptspeicher.

4.2 Aktueller Bereitstellungsprozess

Mit vielen anderen Abteilungen sprechen
Viel auf ‘Zuruf‘ und Besprechungen
Genauere Infos noch von den CICSAdmins nachfragen

Kapitel 5

Realisierung

In diesem Kapitel wird beschrieben, wie die Aufgabe dieser Arbeit gelöst wurde. Dazu wird nach der im Kapitel 3 beschriebenen Reihenfolge, der Arbeitsschritte vorgegangen. Es ist nochmal zu erwähnen, dass zunächst die Provisionierung einer CICS Instanz untersucht wird. Danach wird in weiteren Schritten zuerst eine Db2 Datenbank und schließlich MQ Queues dem Bereitstellungsprozess hinzugefügt.

5.1 Testplex

Vor Beginn der eigentlichen Untersuchung mussten zunächst alle benötigten Rechte beantragt werden. Hierzu zählen unter anderem die Rechte für die Nutzung des Testplexes, die Nutzung von z/OSMF und z/OSPT und die Rechte für die Templateverwaltung innerhalb von z/OSMF. Außerdem war es auf dem Testplex möglich, die Rechte für das Erstellen der CICS Dateien, das Recht, um ein CICS starten zu dürfen und die Rechte für die Administration von Db2 und MQ einer persönlichen UserID zu geben. Dies stellt kein Problem dar, weil es sich bei dem Testplex um eine reine Systemtestumgebung handelt. Außerdem benötigt das IBM Cloud and Management for z/OS lesenden Zugriff auf den Speicherpfad der Template Dateien. Schließlich konnte mit dem ersten Versuch ein bei der Installation von z/OSMF mittgeliefertes minimales CICS Template zu provisionieren begonnen werden.

5.1.1 IBM Standard CICS Template

Trotz der Vorteile durch die Weboberfläche von z/OSMF wurde zunächst auf z/OSPT gesetzt. Diese Entscheidung fiel auf Grund der höheren Flexibilität, durch Images. Da es sich um ein mitgeliefertes Template handelt, sind alle benötigten Workflow Definitionsdateien und Template Dateien vorhanden. Somit konnte der Konsolenbefehl ‘zospt build‘ auf dieses Template durchgeführt werden. Dadurch sollte ein Image erzeugt werden. Jedoch zeigte sich ein weiterer Nachteil des Kommandozeileninterfaces, es ist nicht möglich Templates eine Domain und einen Tenant zuzuweisen. Dies hatte zur Folge das der Befehl ‘zospt build‘

Variablenname	Kurzbeschreibung
DFH_REGION_SEC	Legt fest, ob für das CICS Sicherheit im Allgemeinen aktiviert ist.
DFH_REGION_SECPRFX	Wenn DFH_REGION_SEC gesetzt ist, legt den Namen Perfix bei Authentificationanfragen für Ressourcen fest.
DFH_REGION_APPLID	Applikations ID der zu provisionierenden CICS Instance.
DFH_LE_HLQ	High-level qualifier[1] für die Sprachumgebung[2]
DFH_REGION_HLQ	High-level qualifier für die CICS Dateien.
DFH_REGION_LOGSTREAM	Legt fest, wie die Log Dateien für das provisionierte CICS erstellt werden sollen.
DFH_STC_ID	User ID mit dem die CICS Instanz startet.
DFH_REGION_DFLTUSER	Default User ID für das CICS.
DFH_REGION_VTAMNODE	Name des VTAM Knotens, wenn das CICS hochfährt.
DFH_REGION_MEMLIMIT	Dem CICS maximal zur Verfügung stehender Speicherplatz.
DFH_ZOS_PROCLIB	Datei auf dem Großrechner, die den Job enthält, der für das Erzeugen der CICS Instanz zuständig ist.
DFH_ZOS_VSAM_VOLUME	Speichersystem auf welchem die Dateien gespeichert werden sollen. Entscheidung kann auch an das System abgeben werden.
DFH_CICS_USSHOME	Homeverzeichnes des Unix System Services
DFH_CICS_HLQ	High-level qualifier von dem CICS Installationsort.

Tabelle 5.1: Zu verändernde Variablen im minimalen CICS Template

fehlschlug. Zusätzlich führte es dazu, dass für alle folgenden Aufgaben z/OSMF genutzt wird.

Das z/OSMF auf die Template- und Workflow-Dateien zugreifen kann, sind diese in einem Unix Dateisystem auf dem Großrechner gespeichert. Das Template konnte dann ohne weitere Probleme in die Software Services aufgenommen werden. Dabei wurden ihm eine Domain und ein Tenant zugewiesen. Bevor das Template provisioniert werden kann, müssen Änderungen in der Eingabevariablen Datei vorgenommen werden. Dazu mussten die Werte, der in Tabelle 5.1 genannten Variablen angepasst werden. Die Kurzbeschreibungen und die Beschreibungen aller Variablen, die im Standard Template vorhanden sind, ist in [IBM 19a] zu finden. Das diese Änderungen greifen, muss das Template aktualisiert werden. Dies ist in der Oberfläche per Knopfdruck durchgeführt worden.

Als nächster Schritt wurde ein Testlauf und somit ein erster Versuch das Template zu provisionieren durchgeführt. Hierbei trat beim ersten Step, der einen Job starten wollte, ein

Fehler auf. Nämlich um einen Rechte Verstoß bezüglich des Jobnames. Bei der DATEV eG benötigt eine User ID die Rechte, um Jobs mit bestimmten Namen starten zu dürfen. Da im Template von der IBM vorgeschlagene Jobnamen verwendet werden, kommt es zum Verstoß. Um dieses Problem zu lösen, wurden die Jobnamen innerhalb des gesamten Templates an DATEV eG Standards angepasst. Nachdem das Template aktualisiert wurde, wurde erneut versucht zu provisionieren. Dabei stellte sich heraus, dass der Befehl, um ein CICS zu starten innerhalb der DATEV eG einen weiteren Parameter benötigt. Dieser wurde hinzugefügt und danach funktionierte das Provisionieren und alle definierten Aktionen des minimalen IBM Standard CICS Templates.

5.1.1.1 DATEV eG spezifischen CICS Template

Nachdem das minimale IBM Standard CICS Template funktionsfähig war und erste Erfahrungen mit z/OSMF gesammelt wurden, wurde ein allgemeines mitgeliefertes Template untersucht. Wie in der Tabelle **??** dargestellt ist, ist dieses Template mit insgesamt 76 verwendeten Dateien sehr umfangreich. Zu diesen Dateien zählen alle, die direkt mit dem Template in Verbindung stehen. Das Template beinhaltet nicht nur die Möglichkeit verschiedene CICS Typen zu provisionieren, sondern auch, ob dies mit Skripten oder mit der REST-Api geschieht. Dadurch verliert das Template an Übersichtlichkeit. Zusätzlich kommt am Ende bei der Provisionierung keine CICS Instanz heraus, die einer DATEV eG spezifischen Instanz entspricht. Auf Grund dessen wurden alle für ein DATEV eG spezifischen CICS nicht notwendigen Dateien entfernt. Wie in der Tabelle **??** zu sehen ist, hatte das unter anderem die Löschung von knapp der Hälfte der Dateien zur Folge. Des Weiteren wurden auch viele nicht benötigte Variablen und Steps entfernt. Dadurch schrumpft die provision.xml um circa ein Drittel. Zusammenfassend lässt sich sagen, dass das Template an Übersichtlichkeit gewonnen hat.

Als nächstes wurde mit der Modifizierung der restlichen Dateien begonnen. Als Ziel davon steht eine funktionsfähige DATEV eG spezifische CICS Instanz. Zunächst wurden die Namen der CICS Dateien[3] an die DATEV eG internen Namenskonventionen angepasst.

In Zusammenarbeit mit den CICS Administratorenteam wurde festgelegt, dass eine jede provisionierte CICS Instanz ihre eigene CSD Datei zur Verfügung gestellt bekommen soll. Hierfür soll die bestehende von den Kollegen gepflegte Datei kopiert und mit bestimmten Namenkonventionen gespeichert werden. Somit ist sichergestellt, dass durch die neu provisionierten Instanzen die Alten nicht beeinflusst werden. Außerdem kann jeder Anwender so ohne Seiteneffekte seine CSD bearbeiten. Ein weiterer Vorteil ist, dass bei der Deprovisionierung diese Kopie der Standard Datei ohne Nebenwirkungen gelöscht werden kann. Dadurch dass die Datei, die von den Kollegen gepflegt wird als Grundlage verwendet wird,

[3] Beschreibung in Absatz 2.1.3.1

sind alle provisionierten Instanzen immer auf dem aktuellsten Stand. Um dies Umzusetzen musste ein JCL Job geschrieben werden, der den Kopiervorgang abbildet. Anschließend wurde dieser mittels eines neuen Steps in den Workflow eingebunden. Außerdem mussten bestimmte Gruppen zu der CSD Liste der CICS Instanz hinzugefügt werden. Die JCL ist in Abbildung **??** abgebildet. Die Reihenfolge ist relevant, da es der Initialisierungsreihenfolge entspricht.

die JCL genau erkären..... bzw job im grundlagen teil erkären

Ein spezielles Augenmerk lag auf der Editierung der ‘createCICS.jcl‘-Datei. In dieser befindet sich die Definition des STC Jobs für das provisionierte CICS. Im Standard IBM Template beinhaltet diese zunächst ein Makro für die Validierung von den SIT Parametern. Noch bevor die Jobdefinition beginnt, werden alle aus der Datei für die Eingabevariablen benötigten Variablenwerte in temporäre Zwischenvariablen eingefügt. Dadurch ist eine Änderung nur an einer Stelle notwendig, falls sich etwas an der Variablen ändert. Danach folgt die Definition des Jobs, diese setzt sich aus folgenden Hauptbestandteilen zusammen:

- Einbindung der benötigten Bibliotheken
- Einbindung der zuvor angelegten CICS spezifischen Dateien
- Definition der SIT Parameter

In Abbildung **??** ist zu sehen, dass es vor allem bei der Definition der SIT Parameter zu tief verschachtelten if-Bedingungen kommen kann. Es handelt sich um den Code, der für das Einlesen der Variable ‘DFH_REGION_SITPARAMS‘ aus der Eingabedatei zuständig ist. In dieser Variablen werden die SIT Parameter als Komma separierter String angegeben. Für die Erzeugung eines DATEV eG spezifischen CICS, wurde das Makro für die Validierung von SIT Parametern beibehalten. Alles danach wurde zunächst durch eine zur Verfügung gestellten DATEV eG Standard JCL, für die Erzeugung eines CICS, ersetzt. Nach und nach ist die Logik, wie die aus Abbildung **??**, hinzugefügt worden. Zusätzlich wurden die vorher statische DATEV eG Standard JCL durch Verwendung der Templatevariablen dynamisiert.

Zu der Definition der SIT Parameter ist zu sagen, dass hier nur die wirklich benötigten mit aufgenommen wurden. Die anzunehmenden Werte wurden einzeln mit den CICS Administratorenteam besprochen und festgelegt. Es ist zu beachten, dass es im IBM Standard Template zwei Möglichkeiten gibt, die Parameter zu setzen. Für bestimmte SIT Parameter besteht eine Variable innerhalb des Templates. Für alle anderen ist die Variable ‘DFH_REGION_SITPARAMS‘ vorgesehen. In dieser Arbeit wurde hauptsächlich auf letztere Möglichkeit gesetzt. Dadurch sind die SIT Parameter nur an einer Stelle im Template zu verwalten beziehungsweise die Verwaltung wird nicht auf zwei Arbeitsweisen verteilt.

Schließlich hat die Provisionierung eines DATEV eG spezifischen CICS Instanz funktioniert. Dies wurde mit einem Anmeldevorgang an diese CICS sichergestellt. Außerdem sind alle Standard DATEV eG Transaktionen funktionsfähig. Es wurden alle Dateien wieder pflichtgerecht gelöscht.

5.1.1.2 Bereitstellung Db2

In diesem Absatz wird die Provisionierung deiner Db2 Datenbank beschrieben. Da die Systemumgebung noch der Testplex ist, nur die Datenbank ohne Tabelle, ohne Daten.

Für die Erstellung einer Db2 Datenbank existiert innerhalb der DATEV eG eine REST-API. Wie im Absatz 2.2.1.1 beschrieben, ist es möglich innerhalb eines Workflow Steps einen REST-Request abzusenden. Der Code ist in Abbildung **??** zu sehen. So muss im Body des Requests unter anderem der Datenbankname und eine UserID übergeben werden. Der Code für das Löschen der Datenbank sieht ähnlich aus, nur handelt es sich um einen DELETE-Request. So wurden zwei neue Steps erzeugt und in den Workflow eingebunden.

Die API ist nur dazu fähig Datenbanken auf einem bestimmten Datenbanksystem zu erzeugen. Um die Datenbank aus der CICS Instanz heraus nutzen zu können, muss dem CICS dieses Datenbanksystem mitgeteilt werden. Hierfür ist, wie in Abbildung **??** bereits zu sehen ist, das Hinzufügen einer weiteren CSD Gruppe notwendig. Des Weiteren müssen weitere Bibliotheken in der ‘createCICS.jcl‘ aufgenommen werden. Um den Aufruf möglichst dynamisch zu gestalten wurden, zusätzlich neue Variablen im Template definiert. Diese werden in der Eingabedatei des Templates gesetzt.

In Abilldung zeile SOUNSSO

5.1.1.3 Bereitstellung MQ

In diesem Absatz wird die Provisionierung einer MQ Queue beschrieben. Es ist auch möglich einen MQ Queuemanager zu provisionieren, der Fokus dieser Arbeit liegt aber auf der Bereitstellung von Queues. Bei einem Queuemanager handelt es sich um ein Subsystem, deshalb wird vorerst von einer automatischen Bereitstellung abgesehen. Auf dem Testplex wird des Weiteren die benötigte Funktion, dass eine CICS Transaktionen über eine Queue gestartet wird, nicht untersucht.

Da, wie im Absatz 4.1.2 beschrieben, sehr viele gleichartige Queues benötigt werden, wurde für die Erstellung dieser von den MQ Administratorenteam ein Rexx Skript angefertigt. Dieses Skript steht dieser Arbeit zur Verfügung. Es wurde mit Hilfe von neu erstellten Templatevariablen dynamisiert. Anschließend wurde es in den Provisionierungsworkflow mit Hilfe eines neuen Steps aufgenommen. Für die Deprovisionierung besteht noch kein Script.

Hierfür wurde auf Grundlage des Provisionierungsskriptes ein Eigenständiges erzeugt und ebenfalls in das Template aufgenommen.

Ähnlich wie in Absatz 5.1.1.2 für die Datenbank beschrieben, muss der CSD Datei eine weitere Gruppe für den Queuemanager angegeben werden. Dadurch hat die CICS Instanz auf alle Queues, die sich innerhalb dieses Managers befinden, Zugriff. Des Weiteren ist die Aufnahme weiterer Bibliotheken in der ‘createCICS.jcl‘ notwendig.

5.2 Entwicklungssystemumgebung

Innerhalb der Entwicklungssystemumgebung sind die Sicherheits- und Rechtevorschriften schärfer als auf dem Testplex. So wäre es zwar möglich alle für die administrativen Aufgaben notwendigen Rechte einer persönlichen UserID zu geben. Dies würde bedeuten, dass alle Anwender dieses Templates diese Rechte auch benötigen. Dies würde zu einem Chaos auf dem System führen, da sie auch außerhalb des Templates diese Rechte besitzen würden. Somit wurde in Absprache mit den Administratorenteams für CICS und MQ festgelegt hierfür jeweils einen technischen User[4] zu beantragen. Diesem werden nur die benötigten Rechte übergeben und ist somit sehr anwendungsspezifisch. Um als Anwender das Template nutzen zu können, werden nur die Rechte benötigt Jobs mit diesen technischen Usern ausführen zu dürfen. Für Db2 ist ein solcher User nicht notwendig, da das Datenbanksystem hinter der REST-API für alle zugänglich ist und jeder darauf Datenbanken erstellen darf.

Bei der Übertragung des Templates von der Testsystemumgebung auf die Entwicklungssystemumgebung waren in allen drei Bereichen des Templates notwendig.

5.2.1 CICS Anpassung

Zunächst wurden alle Steps modifiziert, so dass sie den CICS spezifischen technischen User verwenden. Als nächstes musste ein Parameter bei der Erstellung der CICS spezifischen Dateien hinzugefügt werden. Es handelt sich um den Massageclass Parameter mit dem Wert ‘NONE’. Dadurch sind die Daten von täglichen Datensicherung der Entwicklungssystemumgebung ausgenommen. Da die Dateien bei einem Deprovisioning gelöscht werden, ist keine Sicherung notwendig. Außerdem ändert sich die CSD Datei, die als Vorlage gilt, auf die Standard Entwicklungssystemumgebung CSD Datei. Die Db2 und MQ Bibliotheken, die das CICS anzieht, besitzen einen anderen Namen. So musste dies in der ‘createCICS.jcl‘ angepasst werden. Zusätzlich musste ein SIT Parameter angepasst werden, so dass die Log Dateien funktionisfähig sind.

[4] User ID mit zunächst keinen Berechtigungen

5.2.2 Db2 Anpassung

Für die Datenbank Provisionierung waren keine Änderungen am Template notwendig. TABELLEN UND DATEN ABER SCHON

5.2.3 MQ Anpassung

Bei der grundsätzlichen Provisionierung der Queues gab es keine Veränderung. Jedoch wird ein anderer Queuemanager benötigt, dies zieht eine Änderung der Gruppe in der CSD nach sich. Da eine der Queues, jedoch die Transaktion und somit die Anwendung starten soll, sind sogenannte MQ Prozesse notwendig. Diese wurden in das Script aufgenommen.

Das diese funktionieren, muss CICS seitig eine Queue vorhanden sein, die sich um diese speziellen Fälle kümmert. Jede CICS Instanz benötigt somit eine spezifische Queue, dies muss in der MQ CSD Gruppe hinterlegt werden. In diesem Schritt wurde beschlossen, die Verwaltung der MQ CSD Gruppe komplett dem Template zu übergeben. Diese Entscheidung hatte eine Änderung des in Abbildung **??** gezeigten Codes zu Folge. So wird, wie in Abbildung **??** abgebildet, zunächst eine Gruppe angelegt und erst anschließend dem CSD hinzugefügt.

5.3 Nutzwertanalyse

Kapitel 6

Ausblick

Bezug auf den Anfang Integrieren in eine Buildpipeline Möglichkeit (NUR MÖGLICHKEIT)
Bereitstellen der CICSe auch in Produktion

Kapitel 7

Zusammenfassung

Abbildungsverzeichnis

Tabellenverzeichnis

Quellcodeverzeichnis

Literaturverzeichnis

[Also 93] S. Alsop. “IBM still has the brains to be a player in client/server platforms”. *InfoWorld*, Vol. 15, No. 10, p. 4, 1993.

[Cass 07] P. Cassier. *System programmer’s guide to Workload manager. IBM redbooks*, IBM International Technical Support Organization, United States?, 4th ed. Ed., 2007.

[Ceru 03] P. E. Ceruzzi. *A history of modern computing. History of computing*, MIT Press, Cambridge, Mass., 2. ed. Ed., 2003.

[DATE 17] DATEV eG. “Geschichte der Datev”. 2017.

[IBM 14] IBM. “Who uses mainframes and why do they do it?”. 2014.

[IBM 19a] IBM. “DATEV eG”. 2019.

[IBM 19b] IBM. “Using IBM z/OS Provisioning Toolkit”. 2019.

[Keit 16] Keith Winnard, Gary Puchkoff, Hiren Shah. *IBM Cloud Provisioning and Management for z/OS: An Introduction. Redbooks*, IBM International Technical Support Organization, Poughkeepsie, N.Y., 2016.

[Kuhn 19] J. B. Kühnapfel. *Nutzwertanalysen in Marketing und Vertrieb. essentials*, Springer Fachmedien Wiesbaden GmbH, Wiesbaden, 2. auflage 2019 Ed., 2019.

[Rayn 11] C. Rayns. *CICS transaction server from start to finish. Redbooks*, IBM International Technical Support Organization, Poughkeepsie, N.Y., 2011.

[Roge 11] P. Rogers. *ABCs of z/OS system programming: Volume 4. IBM redbooks*, IBM International Technical Support Organization, Poughkeepsie, N.Y.?, 2011.

[Rott 18] R. J. T. Rotthove. *IBM z/OS Management Facility V2R3. Redbooks*, IBM Redbooks, [Place of publication not identified], 2018.

[Sull 16] D. Sullivan. “Google now handles at least 2 trillion searches per year - Search Engine Land”. 2016.

[TIOB 19] TIOBE Software BV. “TIOBE Index | TIOBE - The Software Quality Company”. 25.11.2019.