



**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

Dieses Werk einschließlich seiner Teile ist **urheberrechtlich geschützt**. Jede Verwertung außerhalb der engen Grenzen des Urheberrechtsgesetzes ist ohne Zustimmung des Autors unzulässig und strafbar. Das gilt insbesondere für Vervielfältigungen, Übersetzungen, Mikroverfilmungen sowie die Einspeicherung und Verarbeitung in elektronischen Systemen.

## 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 beigelegt. 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 [?] Zitattest 2 [?]

## **Abstract**

english translation of ‘kurzzusammenfassung‘



# Inhaltsverzeichnis

<b>1. Einleitung</b>	<b>1</b>
<b>2. Firmenkontext</b>	<b>5</b>
<b>3. Grundlagen</b>	<b>7</b>
3.1. Mainframe / Großrechner	7
3.1.1. Batch	7
3.1.2. Batch-Job / Job	8
3.1.3. Resource Access Control Facility	8
3.1.4. REXX	8
3.2. IBM Mainframe Architektur bei der DATEV eG	9
3.2.1. Stages, Sysplexe und Logical Partitions	9
3.2.2. Middleware / Subsysteme	10
3.2.3. Architekturüberblick	14
3.3. IBM Cloud Provisioning and Management for z/OS	14
3.3.1. Begriffserklärung	15
3.3.2. z/OS Provisioning Toolkit	16
3.3.3. z/OS Management Facility	17
<b>4. Vorgehensweise</b>	<b>19</b>
<b>5. Analyse</b>	<b>21</b>
5.1. DATEV-Rechnungsschreibung	21
5.1.1. Beschreibung	21
5.1.2. Architektur der Preisermittlung	24
5.2. Aktueller Bereitstellungsprozess	25
<b>6. Realisierung</b>	<b>29</b>
6.1. Test-Plex	29
6.1.1. IBM Standard CICS Template	29
6.2. Entwicklungsumgebung	35
6.2.1. CICS Anpassung	35
6.2.2. Db2 Anpassung	36
6.2.3. MQ Anpassung	36
6.2.4. Testablauf	37

6.3. Bereitstellungsprozess aktuelles Template . . . . .	38
6.3.1. 1. Fall . . . . .	39
6.3.2. 2. Fall . . . . .	39
6.3.3. 3. Fall . . . . .	39
6.4. Fazit Realisierung . . . . .	40
6.5. Interviews . . . . .	42
6.5.1. Durchführung . . . . .	42
6.5.2. Meinungsbild . . . . .	46
<b>7. Ausblick . . . . .</b>	<b>47</b>
<b>8. Zusammenfassung . . . . .</b>	<b>49</b>
<b>A. Anhang . . . . .</b>	<b>51</b>
A.1. Produktstammdaten data definition language . . . . .	51
A.2. Interview Fragebögen . . . . .	66
<b>Abbildungsverzeichnis . . . . .</b>	<b>67</b>
<b>Tabellenverzeichnis . . . . .</b>	<b>69</b>
<b>Quellcodeverzeichnis . . . . .</b>	<b>71</b>



# Kapitel 1.

## Einleitung

„I recently predicted the last mainframe will be unplugged on March 15, 1996“<sup>1</sup>, ein in der Großrechner-Welt bekannt gewordenes Zitat. Es handelt sich um eine 1993 getroffene Vorhersage, nämlich dass der letzte Mainframe, auch Großrechner genannt, am 15 März 1996 abgeschaltet werden wird. Wieso wird sich im Jahre 2020 dennoch mit dieser Technologie beschäftigt? Und was genau ist ein Großrechner?

In einem Satz ist ein Großrechner<sup>2</sup> ein leistungsstarkes, zentralisiertes Serversystem. In dieser Arbeit wird nur auf Mainframes aus dem Hause von IBM eingegangen. Damit ist auch der Technologiestack festgelegt. Das verwendete Betriebssystem ist z/OS, darauf werden Middleware Produkte wie CICS<sup>3</sup>, das Datenbanksystem Db2<sup>4</sup> sowie die Messaging Lösung „IBM MQ“<sup>5</sup> betrieben. Als Programmiersprachen werden COBOL, IBM Assembler, C, C++ verwendet, seit ca. 23 Jahren auch Java<sup>6</sup>.

Der IBM Mainframe hat eine lange Geschichte. Vor mehr als fünfzig Jahren wurde der allererste Großrechner, der sog.(Kommentar: Bezeichnung such ich noch raus) vorgestellt. Bis in die 90iger Jahre spielte der IBM Mainframe eine Hauptrolle auf dem Computermarkt, dann gewannen zunehmend verteilte Client-Server-Systeme an Bedeutung.<sup>7</sup> Seitdem gilt der Mainframe bereits als „legacy“- veraltet -. Im Jahre 2020 ist der größte Konkurrent für den Mainframe die Cloud.

Wieso also wird sich mit dieser Technologie noch beschäftigt? - Eine Antwort: Der Mainframe wird weltweit genutzt. So verarbeiten Großrechner auch heutzutage weltweit circa 1,2 Millionen CICS Transaktionen pro Sekunde.<sup>8</sup> Im Vergleich hierzu werden 63.000 Google Suchanfragen pro Sekunde abgesetzt.<sup>9</sup>

---

<sup>1</sup>[?]

<sup>2</sup>Beschreibung im Absatz 3.1 zu finden

<sup>3</sup>Anwendungsserver, CICS Beschreibung Absatz 3.2.2.1

<sup>4</sup>Beschreibung Absatz 3.2.2.2

<sup>5</sup>Beschreibung im Absatz 3.2.2.3 zu finden

<sup>6</sup>[?]

<sup>7</sup>[?]

<sup>8</sup>[?]

<sup>9</sup>[?]

Aus der Kombination von hohem Workload, dem proprietären, IBM-abhängigen Technologiestack und dem Ruf eines veralteten Systems entstehen jedoch zunehmend Risiken. Es wird immer schwieriger Nachwuchs in diesem Bereich zu finden. Zum einem, da es kaum noch an Universitäten gelehrt wird. Die Seite des Hochschulkomasses<sup>10</sup> liefert sogar weder für „Mainframe“ noch für „Großrechner“ einen Treffer. Zum anderen ist der demographische Faktor, der Wissensträger nicht zu vernachlässigen. Ein weiteres Problem ist, dass eine Firma, die einen IBM Großrechner mit z/OS betreibt, von dem oben genannten proprietären Technologiestack abhängig ist, dass heißt es entsteht eine starke Herstellerabhängigkeit.

Offentsichtlich betreiben dennoch einige Firmen einen IBM Großrechner. Darunter zählen hauptsächlich Banken, das Gesundheitswesen, Versicherungen, Fluggesellschaften usw. Der gemeinsame Nenner dieser Unternehmen ist, dass sich über die Jahre und Jahrzehnte enorme Investitionen auf dem Mainframe angesammelt haben. Die entstandenen Kernsysteme haben hohe Anforderung an Massendatenverarbeitung, Sicherheitsstandards und Hochverfügbarkeit. All diese Punkte sprechen nach wie vor für die Nutzung eines Großrechners, auch bei der DATEV eG. [?]

Die DATEV eG wurde am 14.02.1966 von 65 Steuerbevollmächtigten gegründet. Sie verfolgten mit der Gründung das Ziel, Buchführungsaufgaben mit Hilfe der neu aufkommenden EDV zu bewältigen. Aufgrund hohen Mitgliederwachstums wurde hierfür bereits 1969 in einen firmeneigenen IBM-Großrechner investiert.[?] Heute umfasst das Leistungsspektrum der DATEV eG unter anderen das Rechnungswesen, Personalwirtschaft, Consulting, IT-Sicherheit, Weiterbildung. Ein nicht unbeträchtlicher Teil dieser betriebswirtschaftlichen Anwendungen läuft bis heute auf einem IBM Großrechner im DATEV Rechenzentrum. So werden pro Tag circa 150.000 Batch Jobs<sup>11</sup> und circa 90 Millionen CICS-Transaktionen verarbeitet. Diese Last wird von circa 14.000 aktiven Modulen erzeugt. Wie in der Abbildung 1.1 zu sehen ist, ist COBOL mit circa 46% Prozent die am häufigsten verwendete Programmiersprache am Großrechner bei der DATEV eG. Durch diese Module werden unter anderem im Monat circa 11 Millionen Lohnabrechnungen erstellt und circa 1 Millionen Umsatzsteuer-Voranmeldungen durchgeführt.

Ich persönlich habe eine Ausbildung zum Fachinformatiker der Anwendungsentwicklung im Großrechnerbereich bei der DATEV eG abgeschlossen und parallel dazu ein Studium der Informatik begonnen. Während dieser Ausbildung lernte ich unter anderem die Entwicklungsprozesse für den Mainframe kennen. Kurz bevor ich die Ausbildung begonnen hatte, wurde eine auf Eclipse basierende Entwicklungsumgebung für COBOL und IBM Assembler in der DATEV eG flächendeckend eingeführt. Zuvor wurde mit Hilfe der in Abbildung ?? gezeigten Oberfläche, dem sog. ISPF gearbeitet. Diese stellte z.B. nur ein Syntaxhighlighting zur Verfügung. Kurz vor Abschluss meiner Ausbildung wurde git, das Standard-Sourceverwaltung für verteilte Systeme und Cloud Entwicklung, auch für COBOL

---

<sup>10</sup>[?]

<sup>11</sup>Beschreibung in Absatz 3.1.2



Abbildung 1.1.: Anteil der verwendeten Programmiersprachen auf dem Mainframe bei DATEV eG in Prozent

und IBM Assembler Sourcen eingeführt. Dadurch wurde ein bis dato verwendetes eigenentwickeltes Tool für die Sourceverwaltung der z/OS Sourcen abgelöst. Jedoch wurde damit zwar das Tooling modernisiert, jedoch nicht der Entwicklungsprozess selbst. Entwickelt wird in einer sogenannten „Entwicklungsstage“<sup>12</sup>. So teilen sich zumindest die Entwickler einer Anwendung die gleiche Test-CICS/Db2/MQ Ressourcen. Das heißt auch, dass Parallelentwicklung an unterschiedlichen Features nur mit viel Abstimmungs-Aufwand und Absprachen innerhalb eines Entwicklungsteams, teilweise auch abteilungsübergreifend, möglich ist. Die Verantwortung der Systemressourcen, unter anderem auch der Test Ressourcen, liegt bei extra dafür entstandenen Administratorenteams. Werden Änderungen an bestehenden Ressourcen durchgeführt oder werden neue Systemumgebungen benötigt, entsteht weiterer Abstimmungs-Aufwand und weitere Absprachen. Dadurch wird der Prozess fehleranfällig und langsam. In der Cloud Entwicklung sind dagegen moderne Prozesse, bei denen eine automatisierte Bereitstellung von Systemumgebungen, zum Beispiel einer Datenbank, über einen „Marktplatz“, Standard.

Moderne Entwicklungsprozesse durfte ich nach dem Abschluss meiner Ausbildung in einem „Cloud Native“-Projekt selbst erleben. Hier hatte jeder Entwickler seine eigene Testumgebung und konnte isoliert und ungestört von seinen Kollegen, ein neues Feature testen. Die Datenbank konnte per Knopfdruck bereitgestellt werden, ganze Laufzeitumgebungen konnten so erzeugt werden. Solche Prozesse unterstützen die bei DATEV e.G. eingesetzte agile Softwareentwicklung, und von den zuständigen Abteilungen, die sich mit der Modernisierung von z/OS Anwendungen und -prozessen beschäftigen, kam die Frage auf, ob es möglich

<sup>12</sup>Beschreibung der Stages in Absatz 3.2.1

wäre, solche Mechanismen auch für legacy z/OS Anwendungen aufbauen zu können. In Zusammenarbeit mit dem Bereich IT-Services und Technologiestrategie ergab sich damit das Thema für diese Arbeit, in der folgende Fragen beantwortet und bewertet werden sollen.

Ist es generell technisch möglich mit dem „IBM Cloud Provisioning and Management for z/OS“-Toolkit Laufzeitumgebungen automatisiert bereitzustellen? Wird dadurch der aktuelle Bereitstellungsprozess schneller und auch sicherer? Erzeugt die Nutzung einen Mehrwert bei den Stakeholdern, also den Entwicklerteams und den Administratorenteams.

Um diese Fragen zu beantworten wird die Provisionierung einer z/OS Laufzeitumgebung für eine speziellen Anwendung untersucht. Die Anwendung sollte ein CICS als Anwendungsserver, eine Db2 Datenbank und IBM MQ als Messaginglösung benötigen, um für diese 3 Haupt-Technologien (Middleware-Komponenten) eine Aussage treffen zu können. Die genaue Vorgehensweise wird im Kapitel 4 beschrieben.

## Kapitel 2.

### Firmenkontext

Die Informationen stammen aus Gesprächen mit einem Mitarbeiter aus dem Technologiestrategieteam und der im Anhang ?? befindlichen Präsentation. Die in der Einleitung genannten Fragen passen zur momentanen Mainframestrategiediskussion innerhalb der DATEV eG. Der DATEV eG sind die zunehmenden Risiken, durch schwierigere Bereitstellung von Skills und der Herstellerabhängigkeit, bewusst. Die Frage ist, wie wird mit den vielen Mainframebestandsanwendungen in Zukunft umgegangen. Die komplette Ablösung dieser Anwendungen durch cloud-native Lösungen ist aktuell nicht absehbar. Die Funktionsfähigkeit dieses Bestandsgeschäfts, welches eine große Einnahmequelle der DATEV eG ist, muss durch effiziente Weiterentwicklung und Wartung gewährleistet werden. Auch im Falle einer Ablöse muss je nach Strategie das Alt-System weiterhin über Jahre oder Jahrzehnte gepflegt werden. Daraus folgt, dass aus Sicht der DATEV eG keine Investition in die IBM Mainframe Plattform keine Alternative darstellt.

In diesem Umfeld wurde die in der Einleitung bereits genannte Umstellung auf eine moderne auf eclipse basierende Entwicklungsumgebung und auf git als Sourceverwaltung umgesetzt. Des Weiteren läuft eine Untersuchung bezüglich automatisierter Builds mit Hilfe von Jenkins basierten Pipelines. Diese Maßnahmen erleichtern Mainframe fremden Entwicklern den Einstieg in diese spezielle Welt. Dennoch ist der Prozess für die Bereitstellung von Middleware noch unverändert und noch nicht modernisiert. Hier setzt diese Arbeit an und klärt die in der Einleitung genannten Fragen.



## Kapitel 3.

### Grundlagen

In diesem Kapitel werden für diese Arbeit wichtige Begriffe erläutert.

#### 3.1. Mainframe / Großrechner

Mainframe und Großrechner werden in dieser Arbeit gleichbedeutend verwendet. Im modernen Sprachgebrauch kann ein Großrechner als größte zur Verfügung stehende Serverart betrachtet werden. Er wird von Unternehmen verwendet, um kommerzielle Datenbanken, Transaktionsserver und Anwendungen, die einen hohen Grad an Sicherheit und Verfügbarkeit benötigten, zu hosten. Im Gegensatz zu verteilten Serversystemen, bei denen die Funktionalitäten auf einzelne Server, wie zum Beispiel einen E-Mail-Server, einen Datenbank-Server, einen Web-Server usw. aufgeteilt sind, handelt es sich bei einem Mainframe um ein zentralisiertes System. Unter anderem sind Datenbanksysteme und Anwendungsserver sogenannte „Subsysteme“. [?]

##### 3.1.1. Batch

Batch beziehungsweise Batch-Verarbeitung bezeichnet in der IT die sog. „Stapelverarbeitung“. Das heißt, dass Programme mit minimalem menschlichen Eingreifen nacheinander abgearbeitet werden. Dies geschieht meist zu einer vorher festgelegten Zeit, gesteuert wird dies über sog. „Scheduling“-Systeme. Zum Beispiel wird einmal am Tag zu einer ganz bestimmten Uhrzeit die tägliche Bewertung<sup>1</sup> der DATEV Rechnungsschreibung durchgeführt. Die auszuführenden Programme laufen in sogenannten „Batch-Jobs“. [?]

---

<sup>1</sup>Beschreibung in Absatz 5.1.1.2

### 3.1.2. Batch-Job / Job

In einem Batch-Job, in dieser Arbeit wird „Job“ gleichbedeutend verwendet, wird dem System mitgeteilt welches Programm mit welchen Ein- und Ausgabedateien und Parametern gestartet werden soll. Die Skriptsprache, die diese Jobs definiert, ist im IBM-Mainframe-Umfeld die sog. „Job Control Language“, kurz JCL. Die drei Grundbausteine der JCL werden im Folgendem beschrieben.

Zunächst ist „JOB“ zu nennen, auch Jobkarte genannt. Hier wird der Name des Jobs, Berechnungsinformationen, maximal zur Verfügung stehende CPU-Zeit und weitere Jobweite Parameter gesetzt.

Innerhalb eines Jobs wird mit Hilfe des „EXEC“ Befehls dem System mitgeteilt, welches Programm gestartet werden soll. Es können mehrere „EXEC“ Befehle in einem Job vorkommen, dabei wird jeder einzelne als sogenannter „Job step“ bezeichnet. Dabei können dem Programm neben den Ein-/Ausgabedateien auch weitere Parameter übergeben werden.

Als letztes ist der „DD“ Baustein zu nennen. „DD“ steht für Data Definition. Ein DD-Statement verknüpft den sog. DD-Namen mit einer Datei oder einem I/O Gerät und ist somit ein Alias für diese. Ein „DD“ Baustein ist immer an ein „EXEC“ Befehl gebunden. Einem „EXEC“ können mehrere „DD“ Bausteine zugeordnet sein. [?]

### 3.1.3. Resource Access Control Facility

Die Resource Access Control Facility, kurz RACF, ist ein externer Sicherheitsmanager für z/OS. Dieser bietet eine Rechteverwaltung für das z/OS Betriebssystem an. Damit werden unter anderem Zugriffsrechte auf Dateien und Subsysteme gesteuert. [?]

### 3.1.4. REXX

Die Restructures Extend Executer Programmiersprache, kurz REXX, ist eine prozedurale Programmiersprache. Die Sprache wird interpretiert. Mittlerweile existieren jedoch auch REXX-Compiler. [?]

REXX versteht sich als Skriptsprache und kommt nicht nur am Mainframe zum Einsatz. So kann es als Bindeglied für Betriebssystembefehle, grafischen Interfaces, Objekten, Funktionen und Servicerroutinen gesehen werden. So wird es, wie auch in dieser Arbeit, unter anderem für die Automation von sich wiederholenden Systemadministrations Aufgaben eingesetzt. [?]



## 3.2. IBM Mainframe Architektur bei der DATEV eG

Folgende Begriffe werden in diesem Absatz im Umfeld der DATEV eG erläutert:

- Stages, Sysplexe und Logical Partitions
- Middleware / Subsysteme
- Architekturüberblick

Die DATEV eG bezogenen Informationen stammen aus Gesprächen mit Mitarbeitern der einzelnen Administratorenteams.

### 3.2.1. Stages, Sysplexe und Logical Partitions

Innerhalb der DATEV eG existieren vier sogenannte „Stages“ auf dem Mainframe. Dabei handelt es sich um abgekapselte Systemumgebungen. Eine Stage besteht aus einer oder mehreren „Logical Partitions“, kurz LPARs, mit eigenen Subsystemen und eigener Ressourcenverwaltung. Zu den Subsystemen zählen unter anderem CICS<sup>2</sup>, Db2<sup>3</sup> und IBM MQ<sup>4</sup>. Dadurch wird eine stricte Trennung zwischen den Stages realisiert. Die vier Stages werden im Folgenden beschrieben.

#### „Entwicklung“

Stage auf der alle z/OS Entwickler an neuen Features arbeiten und erste kleinere Tests durchführen.

#### „Qualitätssicherung“

Hier werden vor allem Integrationstests mit extra dafür erstellen Testdaten durchgeführt.

#### „Vorproduktion“

Die Vorproduktion steht für weitere Integrationstests zur Verfügung. Jedoch produktionsnäher und auch mit Echtdaten, also Daten aus der Produktion.

---

<sup>2</sup>Beschreibung in Absatz [3.2.2.1](#)

<sup>3</sup>Beschreibung in Absatz [3.2.2.2](#)

<sup>4</sup>Beschreibung in Absatz [3.2.2.3](#)

**„Produktion“**

Hier befindet sich die Software, die von den Kunden verwendet wird. Ein Programm muss die Entwicklungs-, Qualitätssicherungs- und Vorproduktionsstages durchlaufen bevor es dem Kunden in der Produktion zur Verfügung gestellt wird.

Die LPARs der vier Stages sind auf zwei sogenannte „Sysplexe“ aufgeteilt. Ein Sysplex kümmert sich um die Kommunikation zwischen den LPARs der einzelnen Stages und verwaltet Ressourcen LPAR übergreifend. [?]

Neben den oben erwähnten zwei Sysplexen existiert noch ein weiterer Sysplex. Diese werden alle im Folgenden beschrieben.

**„LAB/Test-Plex“**

Der Test-Plex ist als Labor für Änderungen am System zu betrachten. So werden zum Beispiel neue Betriebssystemsversionen auf diesem geprüft.

**„QS-Plex“**

Enthält unter anderem die LPARs der Entwicklung und der Qualitätssicherung.

**„Prod-Plex“**

Im Prod-Plex sind die LPARs der Vorproduktion und der Produktion enthalten.

**3.2.2. Middleware / Subsysteme**

In diesem Absatz wird die verwendete Middleware beziehungsweise die verwendeten Subsysteme beschrieben. Diese setzen auf dem IBM Mainframe Betriebssystem z/OS auf.

**3.2.2.1. Customer Information Control System**

Das Customer Information Control System, kurz CICS, ist ein Applikationsserver für einen IBM-Großrechner mit Betriebssystem z/OS und damit eine IBM Middleware. 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 Vorteil gegenüber anderen Anwendungsservern, es unterstützt verschiedene Programmiersprachen. CICS ist ein Multi-Language Application Server und unterstützt z.B. COBOL, Assembler, Java und PLI. So können Programme innerhalb einer Anwendung in der für ihren Use-Case am besten geeigneten Sprache implementiert werden. [?]

Das CICS Subsystem einer Stage umfasst mehrere CICS Instanzen.

#### **3.2.2.1.1. CICS Instanz**

Unter einer CICS Instanz ist ein einzelner Bereich, der auf dem z/OS Kernel aufsetzt, zu verstehen. Dieser Bereich ist mittels einer eindeutigen CICS ApplicationID gekennzeichnet und kann darüber explizit angesprochen werden. Eine CICS Instanz verwaltet mehrere CICS Transaktionen.

#### **3.2.2.1.2. CICS Transaktion**

Ein Businessablauf wird im CICS in einer Transaktion gekapselt. Eine Transaktion kann mehrere Programme unterschiedlicher Programmiersprachen umfassen und wird über eine eindeutige „TransaktionsID“ identifiziert..

Über die TransaktionsID wird der Ablauf gestartet. Dies kann sowohl per Webanfrage oder per Messaging Queue als auch aus einem anderen Programm heraus oder manuell geschehen. In der Transaktion werden alle Änderungen, die Programme an Ressourcen, wie zum Beispiel einer Datenbank oder Dateien tätigen, protokolliert. So wird im Fehlerfall die Möglichkeit eines Rollbacks sichergestellt. [?]

#### **3.2.2.1.3. Voraussetzungen**

Bei der DATEV eG kümmert sich ein Team, die „CICS Administration“ um das Erstellen einer CICS-Instanz, starten dieser Instanz und ist generell für alles rund um die Administration des CICS Transactions Servers zuständig. Der Fokus dieser Arbeit liegt auf dem Erstellen („Provisionieren“) einer CICS-Instanz. Es werden nur die dafür notwendigen Voraussetzungen dargelegt. Außerdem liegt der Fokus auf Entwicklungs-CICS-Systemen, auf der sog. Entwicklungs-Stage, nicht auf den produktiven CICS-Systemen der DATEV eG. Aus diesem Grund werden nur Schritte, die für ein solches Testsystem benötigt werden, dargestellt. Eine weitere Rahmenbedingung besteht darin, dass nur die Arbeitsschritte, die mit z/OSMF<sup>5</sup> automatisiert werden, erläutert werden.

#### **3.2.2.1.4. 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 Voraussetzungen aus dem Absatz 3.2.2.1.3 entsprechend einzurichten, sind mehrere Schritte notwendig. Diese werden im Folgendem beschrieben.

#### **CICS spezifische Dateien**

Zunächst müssen CICS spezifische Dateien im z/OS angelegt werden. Im Fall des dieser Arbeit zugrunde liegende Beispiels handelt es sich um 17 verschiedene VSAM<sup>6</sup> Dateien.

---

<sup>5</sup>Beschreibung in Absatz 3.3

<sup>6</sup>Virtual Storage Access Method, spezielle Dateart, die schnelle I/O-Zugriffe ermöglicht.[?]

Diese Dateien benötigt die CICS Instanz um zum Beispiel Systemfehler zu protokollieren oder den Debugger aktivieren zu können.

## CSD

In der Datei „CICS System Definition“, kurz CSD, muss jede Ressource, die dem System zur Verfügung stehen soll, definiert werden. Eine CSD Datei kann für mehrere CICS Instanzen verwendet werden. Eine CSD Datei besteht aus mehreren Einträgen. Ein Eintrag besteht aus einer Gruppe und einer Liste. Die Gruppe ist hierbei die Definition einer Systemressource und muss manuell 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 IBM MQ Messagingsystem sich diese Instanz verbinden soll.

## STC Job

Bei einem Started Task Control-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.[?] Bei der DATEV eG existiert für jede Instanz eines Subsystems ein solcher Job, so also auch für CICS. In diesem werden zunächst einige zur Laufzeit benötigten Bibliotheken und Dateien eingebunden, unter anderem die CICS spezifischen Dateien<sup>7</sup>. Außerdem werden hier die SIT<sup>8</sup> 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 dazugehörigen CSD und die Information, ob eine Verbindung zu einem Db2 Datenbanksystem hergestellt werden soll.

### 3.2.2.1.5. 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 [3.2.2.1.4](#) 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.

---

<sup>7</sup>Beschreibung in Absatz [3.2.2.1.4](#)

<sup>8</sup>CICS system initialization table

### 3.2.2.2. Db2

Db2 ist ein relationales Datenbanksystem, welches unter anderem als Subsystem eines z/OS Betriebssystems läuft. Einer Stage können mehrere Datenbanksysteme, auch Instanz genannt, zugeordnet werden. In diesem befinden sich die Datenbanken und Tabellen.

### 3.2.2.3. IBM MQ

IBM MQ ist eine Messaging-Lösung der IBM. Diese ermöglicht den asynchronen Datenaustausch zwischen Anwendungen mittels sogenannter Queues. Alle IBM MQ Begrifflichkeiten, die in dieser Arbeit verwendet werden, werden im Folgenden erläutert. [?]

Das IBM MQ Subsystem einer Stage setzt sich aus einem oder mehreren Queue Managern zusammen. Ein Queue Manager kann so als IBM MQ Instanz gesehen werden.

#### 3.2.2.3.1. Queue Manager

Bei einem Queue Manager handelt es sich um die zentrale Ressource eines IBM MQ Systems. Er verwaltet alle anderen IBM MQ Ressourcen. Dazu gehören unter anderem die Speichersteuerung der Daten und die Wiederherstellung dieser im Falle eines Fehlers. Desweiteren koordiniert er den Zugriff aller Anwendungen auf die Nachrichten in den von ihm verwalteten Queues. Um hierbei die Konsistenz sicherzustellen, sorgt er für Locking und die notwendige Isolation der Queues. [?]

#### 3.2.2.3.2. Queues

In Queues werden die Nachrichten, die von Programmen gesendet und gelesen werden gespeichert. Es gibt verschiedene Arten von Queues, die im Kontext dieser Arbeit relevanten Queues sind folgende:

##### Die Local Queue.

Dabei handelt es sich um die einzige Queue Art, bei der die Nachrichten physikalisch gespeichert werden. Die anderen Queue Arten nutzen als Basis immer eine Local Queue.

##### Initiation Queue

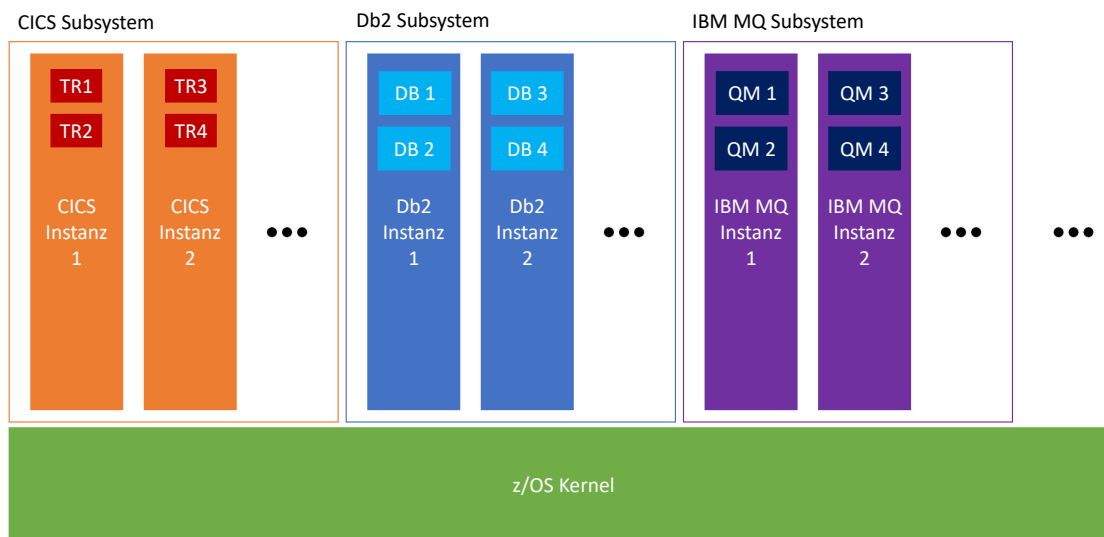
Die sogenannte „Initiation Queue“ ist eine spezielle Art der Local Queue. Diese dient dem Queue Manager dazu, um unter bestimmten Bedingungen eine Trigger-Nachricht darauf zu schreiben. So kann eine andere Local Queue so definiert sein, dass sobald eine Nachricht auf sie geschrieben wird eine solche Trigger-Nachricht erzeugt wird. Dies ermöglicht, dass Anwendungen nur starten, wenn wirklich Daten zum Verarbeiten vorhanden sind. [?]

### 3.2.2.3.3. Process

Für das Auslösen von Anwendungen wird nicht nur die Initiation Queue benötigt, sondern auch sogenannte „Processes“. So muss der Local Queue, die den Start einer Anwendung auslösen soll, bei der Definition nicht nur die Initiation Queue bekannt gemacht werden, sondern auch ein Process. Ein Process legt den „Type“ und den Namen der zu startenden Anwendung fest. Als „Type“ können beispielhaft CICS oder auch WINDOWSNT für Windows unterstützte Plattformen genannt werden. Ist der „Type“ CICS, muss der Name der Transaktion angegeben werden, für Windows Plattformen der Dateipfad der auszuführenden exe. [?]

### 3.2.3. Architekturüberblick

In Abbildung 3.1 sind die einzelnen Subsysteme mit ihren Instanzen dargestellt. Als Grundbaustein dient allen der z/OS Kernel.



TR = Transaktion; DB = Datenbank; QM = Queue Manager

Abbildung 3.1.: Architekturübersicht über die Subsysteme bei DATEV eG

## 3.3. IBM Cloud Provisioning and Management for z/OS

In diesem Absatz wird zunächst auf die für dieses Kapitel grundlegenden Begriffe eingegangen. Das IBM Cloud Provisioning and Management for z/OS bietet die Möglichkeit, mehrere Systeme (sog. Middleware) innerhalb eines z/OS Betriebssystems zu provisionieren, unter anderem Laufzeitumgebungen wie CICS. Für diese Aufgaben stehen zwei Schnittstellen zur

Verfügung. Zum einen das z/OS Provisioning Toolkit, (z/OSPT) und zum anderen z/OS Management Facility (z/OSMF). [?]

### 3.3.1. Begriffserklärung

Im Folgenden werden einige allgemeine Begriffe, die im Umfeld des Tools „IBM Cloud Provisioning and Management for z/OS“ vorkommen, erläutert.

#### 3.3.1.1. Provisioning

Ins Deutsche übersetzt bedeutet es Bereitstellung, in dieser Arbeit wird auch Provisionierung verwendet. In dieser Arbeit umfasst dieser Begriff die Bereitstellung einer Laufzeitumgebung, beziehungsweise den Prozess, der hierfür benötigt wird.

#### 3.3.1.2. Workflow

Ein Workflow ist eine beliebig komplexe, eindeutige Aneinanderreihung von sogenannten Steps. Nach der Ausführung dieser Steps wird ein bestimmtes Ziel erreicht, zum Beispiel die erfolgreiche Bereitstellung eines CICS Systems. Für die Definition eines Workflows, der dazugehörigen Steps und ihrer Variablen wird XML genutzt. Ein Step beschreibt einen Teilablauf eines Workflows. Innerhalb eines Steps können sowohl interne und externe Scripte als auch JCLs und somit Programme ausgeführt werden. Darüber hinaus besteht die Möglichkeit REST-Calls auszuführen. Es 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. Es wird ein XML Schema verwendet um sicherzustellen, dass in dem XML-Skript zur Laufzeit keine syntaktischen Fehler vorhanden sind. [?]

Ein Nachteil von Workflows ist, dass diese statisch sind, das heißt, dass die Variablenzuweisungen 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.

#### 3.3.1.3. Template

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

1. Datei für Eingabevariablen.

In dieser Datei kann Workflowvariablen Werte zugewiesen werden. Diese Variablen müssen bei ihrer Definition im Workflow entsprechend gekennzeichnet sein.

2. 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. Darin ist die Datei für die Eingabevariablen und die Festlegung, welche Variablen davon verwendet werden, anzugeben.

3. Die Manifest-Datei.

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 angegeben. Zusätzlich kann noch eine Beschreibung des Templates hinzugefügt werden.

Somit bildet ein Template einen Rahmen um mehrere Workflows und ermöglicht eine schnellere De-/provisionierung. Das hat den Vorteil, dass Variablen nur an einer Stelle geändert werden. Darüber hinaus besteht die Möglichkeit, per Awendereingabe den Variablen zum Zeitpunkt der Provisionierung einen Wert zuzuweisen. Somit ist ein Template deutlich flexibler als ein Workflow. [?]

#### 3.3.1.4. Instance

Hierbei handelt es sich um das Ergebnis nach der Provisionierung eines Templates, zum Beispiel um ein funktionsfähiges CICS. Als Abgrenzung zu einer Instanz eines Subsystems, kann eine Template Instance auch Instanzen mehrerer Subsystemen enthalten.

#### 3.3.2. z/OS Provisioning Toolkit

z/OSPT bietet ein Kommandozeileninterface für die Bereitstellung und das Verwalten von Laufzeitumgebungen. In Abbildung 3.2 werden die möglichen Kommandozeilenbefehle mittels des Befehls „zospt -h“ in einem Kommandofenster angezeigt. Mit z/OSPT werden noch zwei weitere Begriffe eingeführt.

„Images“. Dabei handelt es sich grundsätzlich um ein Template, jedoch kann dieses Template über eine weitere Inputdatei verändert werden. Dadurch kann ein Template mit spezifischen Änderungen provisioniert werden, ohne dass ein neues Template erzeugt werden muss. Dies erhöht die Flexibilität der Templates weiter.

„Container“. Dabei handelt es sich um eine „Instance“<sup>9</sup>. [?]

---

<sup>9</sup>Beschreibung in Absatz 3.3.1.4



```

$ 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 3.2.: z/OSPT mögliche Kommandozeilenbefehle

### 3.3.3. z/OS Management Facility

Der Hauptaugenmerk dieser Arbeit liegt bei z/OSMF, da damit die Verwaltung von Workflows und Templates über eine browserbasierende Schnittstelle möglich ist. Durch diese Oberfläche, in Abbildung ?? dargestellt, ist es einfacher zu bedienen als das Kommandozeileninterface von z/OSPT und somit wird der Einstieg in die Provisionierung erleichtert.

//hier zosmf welcomepage screenshot

Die linke Seite der Abbildung ?? zeigt den Umfang der z/OSMF Funktionen. Für diese Arbeit besitzt nur der Menüpunkt „Cloud Provisioning“ Relevanz. Unter diesem Punkt sind die Funktionalitäten für die automatisierte Bereitstellung von Templates zu finden. [?]

Zuerst ist das „Resource Management“ zu nennen.. Darunter werden sogenannte „Domains“ und die dazugehörigen „Tenants“ verwaltet. Unter einer „Domain“ ist ein System zu verstehen, das Systemressourcen in Ressourcenpools gliedert. „Tenants“ sind die dazugehörigen Rechtegruppen, die dem Anwender den Zugriff auf und die Nutzung von zugeordneten Templates ermöglicht. Einem Template muss sowohl eine „Domain“ als auch ein „Tenant“ zugewiesen werden. [?]

Zur Verwaltung der Templates und Instances kommen die „Software Services“ zum Einsatz. Dort können neue Templates über die „Manifest Datei“ hinzugefügt werden. Dann muss, wie oben beschrieben, eine „Domain“ und ein „Tenant“ zugewiesen 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 können damit das Template und die in der Aktion-Definitions-Datei definierten Aktionen getestet werden. [?]

## Kapitel 4.

### Vorgehensweise

Zu Beginn dieser Arbeit wurde zunächst das „IBM Cloud Provisioning and Management for z/OS“-Toolkit mit seinen Varianten zOSMF und zOSPT analysiert. Es folgte die Analyse und Darstellung des Ist-Zustandes für die Provisionierung einer Beispielanwendung bei DATEV e.G., inklusive der Beschreibung des momentan etablierten Bereitstellungsprozesses für CICS, Db2 und MQ. Anschließend wurde an Hand einer Beispielimplementierung geprüft, ob es technisch möglich ist eine Laufzeitumgebung mit benötigten Middleware-Komponenten (CICS, MQ, Db2) automatisiert mit dem oben genannten Toolkit für diese Beispielanwendung bereitzustellen. Zuletzt werden Interviews mit Vertretern aus dem Administratorenteam, Entwicklerteam und dem Team für die Technologiestrategie, durchgeführt. Dadurch erfolgt eine Bewertung bezüglich des möglichen Nutzwertes, sowie der Akzeptanz der Technologie durch die Stakeholdern bei DATEV e.G.

Im Folgenden wird speziell auf die Vorgehensweise für die Beispielimplementierung eingegangen. Begonnen wird auf dem Test-Plex, einer Systemumgebung für Tests von neuen Betriebssystemversionen oder ähnlichem.

Dieser ist komplett von anderen Systemumgebungen abgekapselt, um die Auswirkungen von Fehlern in den Tests auf die Entwicklung und Produktion zu vermeiden. In dieser Umgebung wird zunächst untersucht, wie es möglich ist eine CICS-Instanz 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. Für die Provisionierung von Db2 Datenbanken existiert innerhalb der DATEV e.G. bereits eine REST-API. Es wird versucht diese innerhalb des Templates zu nutzen. IBM MQ Queues werden mittels Standard IBM Jobs provisioniert. Da es sich beim Test-Plex, wie im Absatz 3.2.1 beschrieben, nur um eine Testumgebung für Systemtests handelt, werden vorerst nur die benötigten Middleware-Systeme provisioniert.

Nachdem eine CICS Instanz, eine Db2 Datenbank und IBM MQ Queues auf dem Testplex sowohl provisioniert als auch deprovisioniert werden können, folgt der nächste Schritt. Dabei handelt es sich um den Wechsel des Sysplexes vom Test-Plex in den Qs-Plex und somit in

die Entwicklungsstage. In der Entwicklungsstage sind alle Anwendungsdaten, die für diese Tests notwendig sind, vorhanden. Somit kann hier die Integration der Beispielanwendung in die provisionierte CICS-Instanz stattfinden.

## Kapitel 5.

### Analyse

Im Folgendem erfolgt eine Beschreibung der Beispielanwendung „DATEV-Rechnungsschreibung“. Die dafür benötigten Informationen stammen aus Gesprächen mit Mitarbeiter 1 aus der Abteilung, die für die DATEV-Rechnungsschreibung zuständig ist. Es wird vor allem der technische Aspekt beleuchtet. Anschließend wird der aktuelle Bereitstellungsprozess für die Laufzeitumgebung, dem dazugehörigen Datenbanksystem und einer Messaging Lösung dargestellt.

#### 5.1. DATEV-Rechnungsschreibung

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

##### 5.1.1. Beschreibung

Bei dem Gesamtablauf handelt es sich um einen Batch-Ablauf auf dem Großrechner der DATEV e.G. Ein wesentlicher Teil der Preisermittlung ist aus Grund einer besseren Performance als CICS-Anwendung implementiert. 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 Kundenbegriff. Preis und die Kundenzuordnung werden zu einem späteren Zeitpunkt innerhalb der DATEV-Rechnungsschreibung ermittelt.

#### 5.1.1.1. Einpflegung Berechnungssätze

Für das Einpflegen der Berechnungssätze in den Rechnungsschreibungsablauf stehen den Anwendungen drei Möglichkeiten zur Verfügung.

- DMVINI-Schnittstelle
- Webservice-Schnittstelle
- CSV-Datei

Bei der Ersten Möglichkeit handelt es sich um die Verwendung der DMVINI<sup>1</sup>-Schnittstelle. Die Schnittstelle ist in der Programmiersprache Assembler entwickelt. Somit ist diese nur auf dem Mainframe verfügbar. Das Ergebnis der Schnittstelle ist eine sequentielle Datei am Großrechner, vergleichbar mit einer .txt Datei unter Windows. 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 werden Fehler und mutwillige Änderung der Daten von außen ausgeschlossen. Aus dem Aufbau einer solchen Datei lässt sich schließen, dass verschiedene Schritte für die Erzeugung innerhalb der Anwendung notwendig sind.

Für die nachfolgenden Schritte stellt die DMVINI-Schnittstelle jeweils Funktionen 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 (Kundenbegriffe und Mengen) verändert werden. Unzulässige Änderungen haben einen Abbruch des Ablaufs 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 durch einen in der Programmiersprache Java realisierten SOAP-WebService. Hier werden die Berechnungsinformationen in XML-Format bereitgestellt. Dadurch ist eine plattformübergreifende Weitergabe der Daten möglich. 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.

---

<sup>1</sup>DatevMakroVerarbeitungsinformation

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

Mittels dieser drei Möglichkeiten werden insgesamt monatlich circa 30 Millionen Datensätze bereitgestellt und weiterverarbeitet. Diese Datensätze stehen innerhalb der durch die DMVINF-Schnittstelle, den Webservice oder die CSV-Datei erzeugten Berechnungsdateien dem weiteren Verlauf als Input zur Verfügung. Um sicherzustellen, dass all diese Dateien auch verarbeitet werden, wird bei Erstellung einer solchen Berechnungsdatei 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 manuell durch.

#### 5.1.1.2. 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 Preisermittlung und Kundenzuordnung zuständig. Zur Realisierung wurden die Programmiersprachen Assembler, COBOL und Java genutzt. Am Ende dieses Ablaufes steht die ARUBA<sup>2</sup>-Db2-Datenbank. Dort werden die Berechnungsdaten der letzten 36 Monate aufbewahrt. Dabei handelt es sich 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 Kundenbegriff 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 e.G. Umfeld ist ein Geschäftspartner entweder eine Kanzlei oder ein einzelner Mandant.

Dann werden die gruppierten Daten in eine CICS-Anwendung übertragen. Die Entscheidung, die Anwendung in CICS zu hosten hatte die zuständige Abteilung aus Performancegründen getroffen. Die Architektur der CICS-Anwendung wird in 5.1.2 beschrieben. Dort

---

<sup>2</sup>Abrechnungs- und Umsatz-Basis

findet die Preisermittlung mit den dazugehörigen kundenindividuellen Abhängigkeiten, wie zum Beispiel Rabatten, statt. Anschließend werden die Daten wieder zurück an den Batch-Ablauf übertragen. Hier werden die Rechnungsnettoeträge geprüft, z.B. ob diese über einem bestimmten Rechnungslimitbetrag liegen. Wird dieser Rechnungslimitbetrag unterschritten werden die Berechnungssätze als BUL<sup>3</sup> gekennzeichnet und in die folgende Rechnungsperiode vorgetragen. Schließlich wird noch die Umsatzsteuer ermittelt. Abschließend werden die neu erzeugten Berechnungsdaten in die ARUBA-Db2-Datenbank eingepflegt und entsprechend gekennzeichnet.

#### 5.1.1.3. 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, zum Beispiel 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.

#### 5.1.2. Architektur der Preisermittlung

In dieser Arbeit steht das automatisierte Provisionieren von Laufzeitumgebungen im Fokus. In diesem Fall handelt es sich um die Laufzeitumgebung CICS mit den dazugehörigen Elementen. Im Folgenden wird nur auf den Aufbau dieser Umgebung und dessen Gründe eingegangen.

Das System muss an Lasttagen bis zu 180.000 Geschäftspartner verarbeiten können. Die Verarbeitung eines Geschäftspartners wird von einer Transaktion durchgeführt. So sind an Lasttagen 180.000 Transaktionen vom System zu verarbeiten. Um all diese an das CICS zu übertragen, stehen dem System mehrere IBM MQ Queues zur Verfügung. Bevor die eigentliche Ermittlung der Preise stattfindet, werden zunächst die Preisinformationen und die kundenindividuellen Preisabhängigkeiten, wie zum Beispiel Rabatte, ermittelt. Für die Verarbeitung werden zwei weitere Queues verwendet. Eine startet eine Transaktion im CICS, die andere wartet auf deren Antwort. Innerhalb der Transaktion werden alle benötigten Preisinformationen und -abhängigkeiten mit Hilfe einer Db2 Datenbank ermittelt. Diese Informationen werden dann in einen sogenannten „SHARED GETMAIN“-Bereich gespeichert. Dabei handelt es sich im Prinzip um einen Hauptspeicherbereich, der dem CICS zur Verfügung steht. Die Adresse von diesem Bereich wird den Transaktionen zur Verfügung gestellt. Somit greifen die einzelnen Transaktionen nicht mehr direkt auf die Datenbank zu,

---

<sup>3</sup>Berater unter Limit



sondern direkt auf den schnelleren Hauptspeicher. Diese Vorarbeitung ist notwendig, da es aufgrund von bis zu 60 Millionen Datenbankzugriffen zu massiven Einbußen bezüglich der Performance führen würde.

Die Queues, die für die Bestimmung der Preise benötigt werden, werden im Folgenden beschrieben. 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. Hierzu zählt unter anderen die Adresse des vorher beschriebenden Hauptspeicherbereichs. In den restlichen drei Queues sind die Ergebnisse der Preisermittlung gespeichert. Die Ergebnisse stehen somit den Batch-Ablauf zur Weiterverarbeitung zur Verfügung. Für jede der vier Queues existieren jeweils 100 vorgefertigte Namen. Somit können auch maximal nur 100 Aufträge gleichzeitig auf Weiterverarbeitung warten. Falls dieses Limit erreicht ist, wartet der Batch-Ablauf so lange, 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.

## 5.2. Aktueller Bereitstellungsprozess

Wie in den drei Diagrammen zu erkennen ist, ist der aktuelle Bereitstellungsprozess noch mit vielen manuellen Schritten verbunden. Außerdem ist der Hauptaufwand in den Administratorenteams angesiedelt. Das Entwicklerteam ist der Initiator des Ablaufs. So kümmert es sich um Formulare und die erste Kontaktaufnahme zum Administratorenteam. Bei dem Bereitstellungsprozess einer Db2 Datenbank muss es außerdem Projekt Informationen, unter anderem Daten der Voruntersuchung, bereitstellen. Hinzu kommt das Erstellen eines Datenbankmodells. Hierfür wird Datenbankwissen benötigt, hierzu wird gegenfalls die Unterstützung eines Administrator benötigt.

Zusätzlich zu den vielen manuellen Schritten sind die vielen Absprachen zwischen mehreren Abteilungen zu nennen. Steht ein beteiligtes Team nicht zu Verfügung, kommt es zu Verzögerungen, das Team muss warten, der komplette Zeitplan kann sich dadurch nach hinten verschieben. Der Prozess für die Bereitstellung einer CICS-Instanz, mit einer Db2 Datenbank und IBM MQ Queues dauert in der Summe circa sechs Arbeitstage. Es setzt sich aus der Dauer der Einzelprozesse zusammen, für jedes Subsystem wird mit circa zwei Arbeitstage gerechnet. Diese Einschätzung beruht auf der Annahmen, dass zum einen jedes beteiligte Team nur diese Aufgabe zu erledigen hat und zum anderen jeweils ein Tag für die Beratung durch das jeweilige Administratorenteam veranschlagt wird. Natürlich ist

ein parallelisierter Ablauf der einzelnen Teilprozesse möglich, so kann die Gesamtdauer im besten Fall auf circa zwei bis drei Arbeitstage verkürzt werden.

Ein weiterer Punkt ist, dass die Kommunikation beziehungsweise der Initiator für den Start des gesamten Prozesses meist per Zuruf stattfindet. So existiert für die erste Kontaktaufnahme kein Formular, keine Automation oder ähnliches. Zur Kommunikation wird auf E-Mail, Telefon oder mittels Terminen zurückgegriffen.

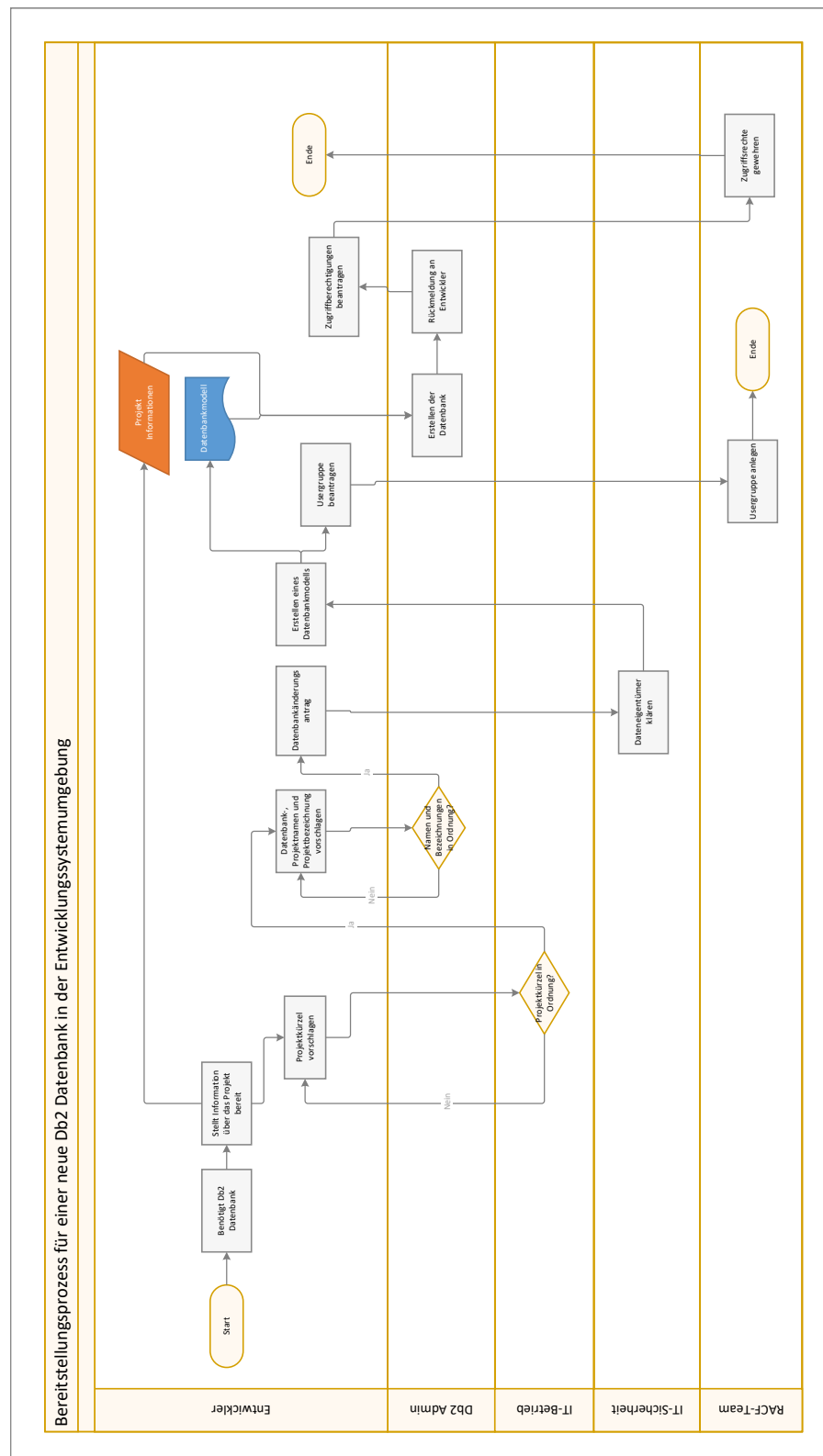


Abbildung 5.1.: Bereitstellungsprozess einer Db2 Datenbank

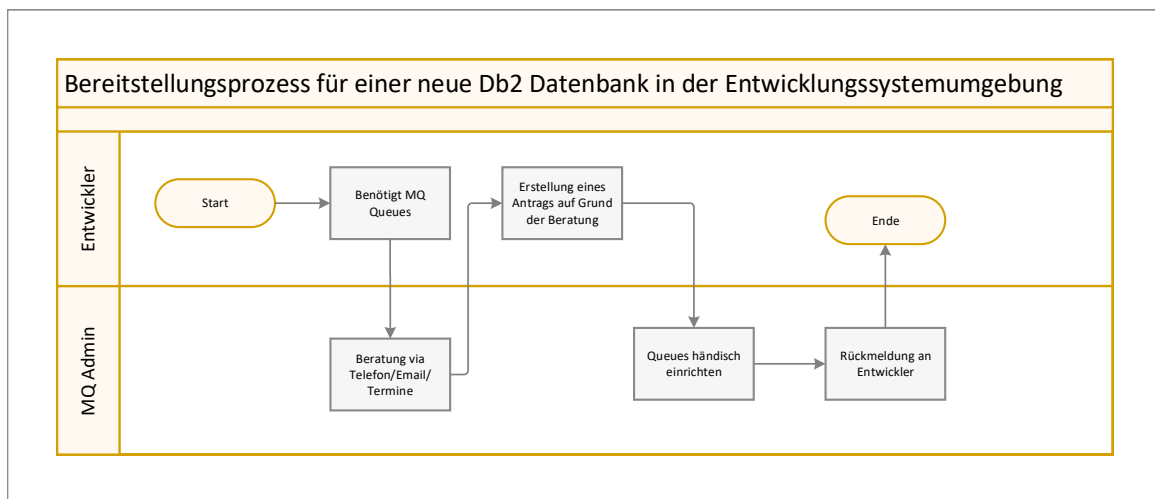


Abbildung 5.2.: Bereistellungsprozess von IBM MQ Queues

## Kapitel 6.

### Realisierung

In diesem Kapitel wird beschrieben, wie die Aufgabe dieser Arbeit gelöst wurde. Dazu wird nach der im Kapitel 4 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. Die Funktionsfähigkeit der so generierten Laufzeitumgebung wird mittels eines Testablaufes am Beispiel der DATEV Rechnungsschreibung sichergestellt. Es folgt eine Bewertung der implementierten Provisionierungslösung durch die Stakeholder bei DATEV e.G. (Entwickler, Administration, Technologiestrategie) Zuletzt wird noch ein Fazit zur Realisierung gezogen und ein Ausblick im Bezug auf die technischen Aspekte gegeben.

#### 6.1. Test-Plex

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 mitgeliefertes minimales CICS Template zu provisionieren begonnen werden.

##### 6.1.1. 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 von z/OSPT durch das

Konzept der Images<sup>1</sup>. 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, dass der Befehl „zospt build“ fehlschlug. Für alle folgenden Aufgaben wurde deshalb z/OSMF genutzt wird.

Damit 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<sup>2</sup> 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 der Tabelle 6.1 genannten Variablen angepasst werden. Die Kurzbeschreibungen und die Beschreibungen aller Variablen, die im Standard Template vorhanden sind, ist in [?] zu finden. Damit diese Änderungen greifen, muss das Template aktualisiert werden. Dies kann in der Oberfläche per Knopfdruck durchgeführt werden.

Als nächster Schritt wurde ein Testlauf und somit ein erster Versuch das Template zu provisionieren durchgeführt. Dabei kam es anfangs zu Rechtproblemen, da die Anforderungen und Rahmenbedingungen der DATEV e.G. in dem standardisierten IBM Template natürlich nicht berücksichtigt waren, beispielsweise ist die Berechtigung für das Starten von Jobs von DATEV Vorgaben abhängig und CICS-Start-Mechanismen haben spezifische Anforderungen an die Eingabeparameter. Nach den notwendigen Anpassungen funktionierte das Provisionieren und die definierten Aktionen des IBM Standard CICS Templates im DATEV Umfeld.

#### 6.1.1.1. DATEV eG spezifischen CICS Template

Nachdem das minimale IBM Standard CICS Template funktionsfähig war und erste Erfahrungen mit z/OSMF gesammelt worden waren, wurde ein allgemeines, mitgeliefertes Template untersucht. Ziel dieses nächsten Schritts war die Provisionierung einer funktionsfähigen DATEV eG spezifische CICS Instanz. Um dieses Template an die DATEV Umgebung an-

---

<sup>1</sup>Beschreibung siehe Absatz 3.3.2

<sup>2</sup>Beschreibung in Absatz 3.3.3

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 Authentificatio- nanfragen für Ressourcen fest.
DFH_REGION_APPLID	Applikations ID der zu provisionierenden CICS Instance.
DFH_LE_HLQ	High-level qualifier <sup>3</sup> für die Sprachumgebung <sup>4</sup>
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	Homeverzeichnis des Unix System Services
DFH_CICS_HLQ	High-level qualifier von dem CICS Installationsort.

Tabelle 6.1.: Zu verändernde Variablen im minimalen CICS Template

zupassen und letztendlich eine „DATEV CICS Instanz “ zu provisionieren, wurden folgende Schritte durchgeführt.

- Analyse des bestehenden Templates und der darauffolgenden Minimalisierung
- Umgang und Anpassung der CSD Datei
- Anpassung der Jobs und Skripte mit Schwerpunkt auf der „createCICS.jcl“-Datei.

Die Analyse ergab, dass das mitgelieferte Template mit insgesamt 76 verwendeten Dateien sehr umfangreich ist. Es zählen alle Dateien, die direkt mit dem Template in Verbindung stehen. Im Zentrum des Templates steht die Workflow Definitionsdatei „provision.xml “ mit circa 583 Zeilen Code. In dieser sind alle Steps, die bei einer Provisionierung durchgeführt werden, definiert. Das Template beinhaltet nicht nur die Möglichkeit, CICS-Instanzen mit unterschiedlichen Konfigurationen zu provisionieren, sondern auch, ob dies mit Skripten

oder mit der REST-API geschieht. Zu diesen unterschiedliche Konfigurationen zählt unter anderem die Unterscheidung, ob die CICS-Instanz einem Sysplex hinzugefügt werden soll oder nicht. Dadurch verliert das allgemeine Template allerdings deutlich an Übersichtlichkeit. So wurden neben allen für ein DATEV eG spezifischen CICS nicht notwendigen Dateien auch nicht benötigte Variablen und Steps entfernt. Wie in der Tabelle ?? zu sehen ist, konnten dadurch circa die Hälfte der Dateien gelöscht werden und bei der provision.xml wurde ein Drittel an Quellcode eingespart werden. Diese Version dient dem weiteren Vorgehen als Grundlage

Als nächster Schritt wurde in Zusammenarbeit mit den CICS Administratorenteam festgelegt, wie im Umfeld der automatisierten Bereitstellung die CSD Dateien verwaltet werden soll. So wurde festgelegt, dass 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 existierenden und bei DATEV im Einsatz befindlichen Instanzen nicht beeinflusst werden. Das ermöglicht auch jedem Nutzer der Provisionierung, die CSD dieser eigenen CICS-Instanz ohne Seiteneffekte zu 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, sind alle provisionierten Instanzen immer auf dem aktuellsten Stand. Um dies umzusetzen, musste ein JCL Job geschrieben werden, der den Kopiervorgang implementiert. Anschließend wurde dieser mittels eines neuen Steps in den z/OSMF 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 erklären..... bzw job im grundlagen teil erklären

Im nächsten Schritt wurden die Jobs und Skripte angepasst. Zunächst wurden die Namen der CICS Dateien<sup>5</sup> an die DATEV eG internen Namenskonventionen angepasst. Ein spezielles Augenmerk lag auf der Anpassung 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 der SIT Parameter. Noch bevor die Jobdefinition beginnt, werden alle aus der Datei für die Eingabevariablen benötigten Variablenwerte

---

<sup>5</sup>Beschreibung in Absatz 3.2.2.1.4



in temporäre Zwischenvariablen eingefügt. 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 Variable 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 wurde damit die notwendige Logik, wie die aus Abbildung ??, hinzugefügt. Damit wurde die vorher statische DATEV eG Standard JCL durch Verwendung von Template internen Variablen flexibilisiert.

Es wurden nur die wirklich benötigten SIT Parameter aufgenommen. 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, diese 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 mit letzterer Variante gearbeitet. Dadurch sind die SIT Parameter nur an einer Stelle im Template zu verwalten, beziehungsweise wird die Verwaltung nicht auf zwei Arbeitsweisen verteilt.

Durch die beschriebene Vorgehensweise wurde erfolgreich die Provisionierung eines DATEV eG spezifischen CICS Instanz umgesetzt. Sichergestellt wurde dies wurde mit einem Anmeldevorgang an dieses CICS, wie in Abbildung ?? zu sehen ist. Außerdem sind alle Standard DATEV eG Transaktionen in dieser Instanz funktionsfähig. Die Deprovisionierung verlief nach Plan.

#### 6.1.1.2. Bereitstellung Db2

In diesem Absatz wird die Provisionierung einer Db2 Datenbank beschrieben. In der Systemumgebung Testplex bedeutet dies die Provisionierung der Datenbank ohne Tabelle und Daten.

Für die Erstellung einer Db2 Datenbank existiert innerhalb der DATEV eG eine REST-API. Wie im Absatz 3.3.1.2 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 in diesem Fall um einen DELETE-Request. Die zwei notwendigen Steps wurden 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 sowie 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 Abildung zeile SOUNSSO

#### 6.1.1.3. Bereitstellung MQ

In diesem Absatz wird die Provisionierung einer MQ Queue im TestPlex beschrieben. Es ist auch möglich einen MQ Queuemanager zu provisionieren, der Fokus dieser Arbeit liegt aber auf der Bereitstellung von Queues. Für die Bereitstellung eines MQ Queuemangers bei DATEV e.G. ist laut MQ-Administration vorerst keine automatische Bereitstellung vorgesehen ist, gegenfalls kann dies in einem zukünftigen Szenario umgesetzt werden.. Ebenfalls in Abstimmung mit MQ- und CICS-Administration wurde entschieden, die Funktion eines Starts einer CICS Transaktionen über eine Queue nicht umzusetzen. Der Fokus lag auf der Prüfung, wie es möglich ist, eine einzelne Queue zu provisionieren, nicht die voll umfängliche Umsetzung der Anforderung der Anwendung DATEV Rechnungsschreibung.

Die IBM stellt Programme für die Verwaltung und das Nutzen von Queues zur Verfügung. Diese können mittels eines Jobs und bestimmten Parametern gestartet werden. In Abbildung ?? ist die JCL des Jobs für das Erstellen einer Queue zu sehen. Das auszuführende Programm ist „CSQUTIL“ und als Parameter wird der Queuemanager übergeben. Unter dem DD Namen „MQSCIN“ ist der MQ Befehl für das Erzeugen einer Queue zu sehen. Um zu Prüfen, ob die Queue auch funktionsfähig ist, wurde nach dem Erstellen auch mit Hilfe eines Jobs, eine Messages auf die Queue geschrieben und wieder abgeholt. Der Job für das Löschen der Queues ist analog aufgebaut.

Ähnlich wie in Absatz 6.1.1.2 für die Datenbank-Provisionierung beschrieben, muss der CSD Datei eine weitere Gruppe für den Queuemanager angegeben werden. Dadurch hat die CICS Instanz Zugriff auf alle Queues, die sich innerhalb dieses Managers befinden. Des Weiteren ist die Aufnahme weiterer Bibliotheken in der „createCICS.jcl“ notwendig.

## 6.2. Entwicklungssystemumgebung

Innerhalb der Entwicklungssystemumgebung sind die Sicherheits- und Rechtsvorschriften schärfer als auf dem Test-Plex. 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. Damit bestünde eine potentielle Gefahr für das System, da sie damit 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<sup>6</sup> zu beantragen. Diesem werden nur die für das Template benötigten Rechte übergeben und er ist somit use-case-spezifisch. 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 Anpassungen in allen drei Bereichen des Templates notwendig.

### 6.2.1. CICS Anpassung

Zunächst wurden alle Steps modifiziert, so dass sie den CICS spezifischen technischen User verwenden. Hierfür musste der „JOB“ Baustein jeder JCL angepasst werden. Dafür bietet zOSMF beim Zuweisen des „Tenants“ eine Standard Jobkarte, die vor jeden Job des Templates eingefügt wird, hinterlegt werden. Als nächstes musste ein Parameter bei der Erstellung der CICS spezifischen Dateien hinzugefügt werden. Es handelt sich um den Messageclass Parameter mit dem Wert „NONE“. Dadurch sind die Daten von der täglichen Datensicherung der Entwicklungssystemumgebung ausgenommen. Da die Dateien bei der Deprovisionierung gelöscht werden, ist keine Sicherung notwendig. Außerdem wird die CSD Datei, die als Vorlage gilt, durch CSD Datei der Standard Entwicklungssystemumgebung geändert. Für die Db2 und MQ Bibliotheken, die das CICS anzieht, wurden die spezifischen Namen auf der Entwicklungs-Stage angegeben. So musste dies in der „createCICS.jcl“ angepasst werden. Zusätzlich musste ein SIT Parameter angepasst werden, so dass die Log Dateien funktionsfähig sind. Außerdem kam noch ein Ordner neu hinzu. Dieser dient später als Ablageort der kompilierten Programme.

---

<sup>6</sup>User ID mit zunächst keinen Berechtigungen

### 6.2.2. Db2 Anpassung

Eine genauere technische Analyse der Rechnungsschreibungsdatenbank kam zu dem Ergebnis, dass es zwar möglich wäre diese Datenbank zu provisionieren, dies aber den zeitlichen Rahmen dieser Arbeit übersteigen würde. Der Grund hierfür ist die Komplexität der benötigten Tabellen. So wird auf drei Tabellen für die Ermittlung der Produktstammdaten lesend zugegriffen, auf neun weitere bei der Bestimmung der Preisabhängigkeiten. Auf die Tabellen wird nicht direkt zugegriffen, sondern über Views<sup>7</sup>. Bei allen anderen werden innerhalb der View noch weitere Tabellen, teilweise aus anderen Datenbanken, gejoint. Insgesamt besteht das System aus 14 Tabellen, die auf vier Datenbanken aufgeteilt sind, und 12 Views für den Zugriff auf diese Tabellen.

Die Db2 Administration muss dafür Vorarbeit leisten. Mit dieser wurde begonnen, jedoch stellte sich heraus, dass die Komplexität (600 Zeilen Code<sup>8</sup> für einen kleinen Teil an Tabellen) der Anwendung DATEV Rechnungsschreibung im Rahmen dieser Arbeit als zu umfangreich angesehen wurde. Sollte sich die Provisionierung generell als zielführend erweisen wird dieser Einmalaufwand erbracht werden

Für die weitere Arbeit werden, die in einem anderen Datenbanksystem bereits vorhanden, Datenbanken genutzt. Hierfür mussten die dafür vorgesehenen Variablen in der Eingabedatei des Templates angepasst werden. Außerdem wurden sowohl in der Provisionierungs- als auch in der Deprovisionierungsdatei die Datenbanksteps auskommentiert und somit kommen diese nicht mehr zum Einsatz.

### 6.2.3. MQ Anpassung

Da für die DATEV Rechnungsschreibung, wie im Absatz 5.1.2 beschrieben, sehr viele gleichartige Queues benötigt werden, wurde für die Erstellung dieser von den MQ Administratoren ein Rexx Skript angefertigt. Dies geschah unabhängig dieser Arbeit zum Zeitpunkt der Einführung des Rechnungsschreibungsprozesses. Dieses Skript steht dieser Arbeit zur Verfügung. Für die Provisionierung MQ Queues waren folgende Arbeitsschritte notwendig.

- Anpassung des zur Verfügung stehenden Skriptes
- Implementierung von Jobs für restliche Queues
- Anpassung der CICS CSD Datei

Hierfür wurden zunächst die Eingabeparameter durch vorher angelegte Templatevariablen ersetzt. Diese steuern, wie viele Queues jeweils angelegt werden, auf welchen Queuemanager

---

<sup>7</sup>Alias eines Datenbankabfrage, auf die wie auf eine normale Tabelle zugegriffen werden kann

<sup>8</sup>Data Definition Language im Anhang ?? zu finden

die Queues angelegt werden und den ersten Qualifier des Queuenamens. Für den restlichen Queuenamen existiert auch eine Variable, in dieser werden die Namen als Komma separierte Liste angegeben und ausgelesen. Anhand dieser Namen wird dann die maximale Queuetiefe und die maximale Länge einer einzelnen Nachricht festgelegt. Im alten Skript wurden die Queues mit Hilfe einer Queue, die als Vorlage dient, angelegt. Im Fall einer Provisionierung kann nicht davon ausgegangen werden, dass diese Vorlagen zur Verfügung stehen. Deshalb wurden die benötigten Parameter explizit manuell angegeben. Um die damit erstellten Queues zu testen, wurde eine Routine entwickelt, die eine Nachricht auf die Queue schreibt und diese wieder abholt. Anschließend wurde das Skript in den Provisionierungsworkflow mit Hilfe eines neuen Steps aufgenommen.

Für die Deprovisionierung der Queues besteht noch kein Skript. Als Grundlage kann das vorher angepasste Provisionierungsskript dienen. Hierfür musste der „Define“-Befehl für die Erstellung von Queues durch den „Delete“-Befehl ausgetauscht werden. Die Logik für die Ermittlung der maximalen Queuetiefe und der maximalen Nachrichtenlänge wird dafür nicht mehr benötigt und konnte entfernt werden.

Die durch die beiden Skripte erstellten Queues sind nur für den Datenaustausch zwischen der CICS Transaktion für die Preisermittlung und dem Batch Ablauf zuständig. Wie in Absatz ?? beschrieben, benötigt der Ablauf noch weitere Queues. Da es sich hierbei um spezielle Queues handelt, wurde auf die im Absatz 6.1.1.3 gezeigte Technik zurückgegriffen. Bei der Antwort-Queue für die Ermittlung der Listenpreise handelt es sich um eine Queue ohne besondere Parameter. Es werden noch zwei Trigger-Queues benötigt, die über Prozesse eine Transaktion im CICS starten. Als letzter Baustein für das Triggering der Transaktion wird noch eine Initiation Queue benötigt. Diese muss im CICS hinterlegt sein.

Jeder CICS Instanz kann nur eine Initiation Queue zugewiesen sein. Dadurch benötigt jedes CICS eine eigene Initiation Queue. Die Zuweisung geschieht in der MQ CSD Gruppe. Somit müsste für jede provisionierte CICS Instanz im Voraus eine solche CSD Gruppe angelegt werden. In Absprache mit der MQ-Administrations wurde entschieden, die Verwaltung der MQ CSD Gruppe komplett dem Template zu übergeben. Diese Entscheidung hatte eine Änderung des in Abbildung ?? gezeigten Codes zur Folge. So wird, wie in Abbildung ?? abgebildet, zunächst eine Gruppe angelegt und erst anschließend dem CSD hinzugefügt.

Für jeden MQ bezogenen Job wurde zuguterletzt die Jobkarte angepasst und der technische User der CICS-Administration durch den technischen User der MQ-Administration, der für administrative Aufgaben berechtigt ist, ausgetauscht.

#### 6.2.4. Testablauf

Für die Prüfung der Funktionsfähigkeit der so generierten Laufzeitumgebung steht dieser Arbeit ein Testablauf zur Verfügung. Dieser wurde von den Mitarbeitern der Rechnungs-

schreibung beigesteuert. Dabei handelt es sich um einen Teilablauf des gesamten Rechnungsschreibungsprozesses. In diesem Ablauf wird nur die Preisermittlung, die die Laufzeitumgebung CICS benötigt, getestet. Als Eingabe dienen vordefinierte Dateien und die Ergebnisse werden ebenfalls in Dateien geschrieben. Der Ablauf liegt in Form von zwei Jobs vor. Beide sind in der gleichen JCL Datei definiert, somit starten beide zeitgleich. Dies ist notwendig, da der erste Job die Verarbeitung im CICS über die Queues startet und der zweite auf die Ergebnisqueues lauscht.

Um den Ablauf auch auf der vorher provisionierten Laufzeitumgebung zu starten, musste lediglich der verwendete QueueManager angepasst werden. Über die Queues und das verwendete Triggering wird die Transaktion im richtigen CICS gestartet. Um die Ausgabe zu prüfen wurde der gleiche Testablauf mit den gleichen Eingabedateien auf der für Testzwecke üblichen Laufzeitumgebung durchgeführt. Anschließend wurden die Ausgabedateien beider Läufe verglichen.

### 6.3. Bereitstellungsprozess aktuelles Template

Bei dem Bereitstellungsprozess der durch das aktuelle Template möglich gemacht wird, sind drei Fälle zu unterscheiden:

1. Fall:

Dem Entwicklerteam steht das Template in z/OSMF zur Verfügung und es wurde noch keine Instance dieses Templates provisioniert. Es wird eine neue Instanz benötigt.

2. Fall:

Dem Entwicklerteam steht das Template in z/OSMF zur Verfügung und es steht bereits eine Instanz dieses Templates zur Verfügung. Es wird eine weitere Instanz benötigt.

3. Fall:

Das Administratorenteam führt Änderungen an einer Workflow Definitiondatei durch. Hier ist zwischen zwei weiteren Fällen zu unterscheiden:

a) Das Template wurde noch nicht veröffentlicht.

b) Das Template wurde veröffentlicht.

### 6.3.1. 1. Fall

Der Mitarbeiter meldet sich an der zOSMF Oberfläche an und klickt auf den Menüleistepunkt „Cloud Provisioning“. Anschließend öffnet er die „Software Services“ und wählt dort das oben genannte Template aus. Er kann es ohne Änderungen provisionieren und damit seine Programmabläufe testen.

### 6.3.2. 2. Fall

Mit dem aktuellen Stand muss der Mitarbeiter wissen, an welchem Speicherort das Template abgelegt ist, da er die Template- nicht die Workflowdateien kopieren muss. Anschließend sind Änderungen an den Eingabevariablen des Templates notwendig. Unter anderem ist eine andere CICS Application Id zu wählen. Um die Queues und MQ Prozesse aus Fall eins nicht zu überschreiben, muss ein anderer Queue Manager gesetzt werden. Dieser Queue Manager muss von den zuständigen Administratorenteam manuell bereitgestellt werden. Anschließend muss mit den veränderten Dateien ein neues Template in zOSMF aufgenommen werden. Schließlich kann der Mitarbeiter eine Instanz erzeugen. Diese ist unabhängig von der Instanz aus Fall eins.

### 6.3.3. 3. Fall

Ein Template ist dann veröffentlicht, wenn es den berechtigten Teams zur Verfügung steht. Zunächst muss der Speicherort der zu bearbeiteten Dateien bekannt sein. Anschließend kann die Änderung mit einem Editor nach Wahl durchgeführt werden.

#### 6.3.3.1. 3. Fall a)

Hier kann das Template in der zOSMF Oberfläche per Mausklick aktualisiert werden.

#### 6.3.3.2. 3. Fall b)

Um die Funktionsfähigkeit der veralteten Instanzen weiterhin sicherzustellen, muss eine neue Version des Templates erzeugt werden. Dies ist auch per Mausklick zu lösen.


## 6.4. Fazit Realisierung

Am Ende der Realisierung steht ein funktionsfähiges Template. Dieses Template provisioniert ein CICS und die benötigten MQ Queues. Wie in Absatz 6.2.2 beschrieben, wurde eine Db2 Datenbank wegen hoher Komplexität außen vorgelassen. Auf dem Test-Plex wurde bewiesen, dass die Provisionierung einer Datenbank möglich ist. Des Weiteren wäre die Provisionierung von Tabellen mit hohem einmaligen Arbeitsaufwand ebenfalls möglich. Ein Testablauf der Beispielanwendung DATEV Rechnungsschreibung in einer provisionierten, isolierten CICS-Laufzeitumgebung konnte korrekt durchgeführt werden.

Folgende Probleme im Rahmen der Implementierung wurden erkannt:

- Nicht sprechende Fehlermeldungen von z/OSMF
- Nicht identifizierbare Programmiersprache
- Nicht optimales Zugriffsrechtekonzept

Als erstes Problem sind nicht sprechenden Fehlermeldung von z/OSMF, Abbildung 6.1, zu nennen. z.B. wird bei dem Hinzufügen und Aktualisieren eines Templates in zOSMF



✖ During template evaluation, one or more errors were detected in the workflow definition file.

Abbildung 6.1.: Beispiel einer Fehlermeldung von zOSMF

das Template und damit alle davon benötigten Dateien auf Syntaxfehler geprüft. Die in Abbildung 6.1 gezeigte Meldung tritt dann ein, wenn ein solcher Syntaxfehler vorhanden ist. Es ist aber nicht zu erkennen, welcher Fehler genau vorliegt, noch nicht einmal in welcher Datei dieser auftritt. Zudem auch keine genaue Anzahl an auftretenden Fehlern. Dieser Umstand, kombiniert mit 36 bestehenden Dateien, erschwert die Fehlersuche. Im Gegensatz dazu wird im Fehlerfall beim Ausführen eines Steps immer der Fehlercode und der genaue Ort des Fehlers ausgegeben. Zum Beispiel wird bei einem Step, in dem ein REST Aufruf gemacht wird, und ein Fehler auftritt, der Requestcode und die hinterlegte Fehlermeldung an der zOSMF Oberfläche angezeigt.

Ein weiteres Problem ist eine nicht genau identifizierbare Programmiersprache, die für die dynamische Generierung von Skripten genutzt wird. So ermöglicht diese die dynamische Wertzuweisung von zum Beispiel Rexx-Variablen durch Variablen des Templates. Außerdem besteht eine Art von String Verarbeitung. Zu beachten ist, dass wenn am Zeilenanfang ein '#' steht, kann diese Programmiersprache verwendet werden. In Abbildung ?? ist ein Beispiel zu sehen. Dort werden die Queuenamen, die als kommaseparierte Liste in der Templatevariable „DFH\_MQ\_QUEUE NAMES“ angegeben sind, ausgelesen und in eigenen



REXX Variablen gespeichert. Zu sehen ist zunächst eine „set“ Anweisung, mit der Variablen zugewiesen werden können, If-Bedingungen und eine foreach-Schleife

In Abbildung ?? wird das Ergebnis, welches zur Laufzeit ausgeführt wird, dargestellt. Es ist zu erkennen, dass nur noch die für das REXX Skript notwendigen Codeabschnitte vorhanden sind. Dadurch können sehr dynamische Templates erstellt werden. Jedoch wurde weder eine Dokumentation zu dieser Sprache noch um welche Sprache es sich genau handelt gefunden. Somit liegt dem Wissen über diese Sprache nur der Code aus Beispielen der IBM zu Grunde.

Ein weiterer Problempunkt ist das mit zOSMF und dem Template einhergehende Zugriffsrechtekonzept. Die zOSMF Berechtigungsgruppen sind nicht an die DATEV eG internen Richtlinien angepasst. Die Aufnahme in eine solche Gruppe, um zum Beispiel die zOSMF Oberfläche nutzen zu dürfen, geschieht auf Zuruf und manuelles Hinzufügen einer User Id durch einen Mitarbeiter. Außerdem ist der Einsatz einer für das ganze Template gültigen Standard Jobkarte, um technische User verwenden zu können, nicht optimal. zOSMF bietet hier eigentlich eine Möglichkeit in der Stepdefinition einen „runAsUser“ anzugeben. Unter diesem User würde der Step dann ausgeführt werden, also die Stelle an der zum Beispiel für CICS Steps der technische User für administrative CICS Aufgaben angegeben werden müsste. So würde das Gewähren der expliziten Rechte zum Starten eines Jobs mit der technischen User Id entfallen und damit die manuelle Arbeit des „Gewährens“, was mittels eines Formulars beantragt wird. Jedoch um einen „runAsUser“ in der Stepdefinition angeben zu können, muss in der dem Template zugewiesen „Domain“ ein sogenannter „Cloud Security Admin“ hinterlegt sein. Dieser würde sicherstellen, dass nur die für ein Template zugelassenen User diesen Template auch provisionieren dürfen. In dieser Arbeit wird die mitgelieferte „Default Domain“ genutzt, in dieser ist kein „Cloud Security Admin“ angegeben. Da es sich um die Standard „Domain“ handelt, darf diese nicht geändert werden. Somit müsste eine eigene „Domain“ angelegt werden um einen „Cloud Security Admin“ hinterlegen zu können. Dadurch, dass sich zOSMF bei der DATEV eG noch in einem Teststadium befindet, wird von der Erstellung einer eigenen „Domain“ abgesehen. Dies ist der Grund für den nicht optimalen Einsatz der oben genannten Jobkarten. An diesen beiden Fällen ist zu erkennen, dass das Rechtekonzept noch nicht für einen firmenweiten Einsatz ausgelegt ist und noch überarbeitet und angepasst werden muss. Dies ist jedoch explizit nicht Bestandteil dieser Arbeit.

## 6.5. Interviews

In diesem Absatz wird zunächst erläutert, auf welcher Grundlage die Interviews geführt worden sind. Anschließend werden die Ergebnisse der einzelnen Interviews nach Gruppen aufgeteilt, ausgewertet und interpretiert. Schließlich wird daraus ein allgemeines Stimmungsbild abgeleitet.

### 6.5.1. Durchführung

Interviewt wurden jeweils zwei Mitarbeiter der Gruppen, CICS Administration, Db2 Administration, MQ Administration. Zusätzlich wurde eine Fachberaterin im Bereich Technologiestrategie und ein Mitarbeiter des Entwicklerteams der DATEV Rechnungsschreibung befragt. Für die beiden letztgenannten Interviews waren nur die Fragen 1., 2. und 6. bis 10. des Fragebogens von Relevanz. Sowohl der Fragenkatalog als auch die ausgefüllten und digitalisierten Fragebögen sind im Anhang A.2 zu finden. Bevor die Interviews durchgeführt worden sind, wurde den Teams in getrennten Terminen die Ergebnisse dieser Arbeit vorgestellt. Der Schwerpunkt wurde an das jeweilige Team angepasst. So wurde bei den Administratorenteams vor allem auf den Teil des Templates, der für ihr Arbeitsgebiet zuständig ist, eingegangen. Außerdem wurden neben den im Absatz 6.3 dargestellten Lösung, auch die Lösung aus Kapitel 7 vorgestellt. An Hand des durch diese Arbeit bereit gestellte Template wurde die zOSMF Oberfläche erläutert.

#### 6.5.1.1. CICS Administratoren

Die Meinung bezüglich des aktuell möglichen Ablauf mittels zOSMF von CICS Administrator 1 ist mittelmäßig. So bietet es zwar eine flexible Versionierung und Veröffentlichung. Jedoch ist es durch die verschiedenen Sprachen und Dokumentarten mit Startschwierigkeiten versehen. Im Gegensatz dazu sieht CICS Administrator 2 das momentane Template zumindest für CICS als ablauffähig und mehrfach einsetzbar. Den Vorteil der Konfigurierbarkeit von außerhalb des Templates durch zOSPT nennen beide Administratoren. Als Nachteil sehen sie jedoch die Notwendigkeit eines sehr dynamischen Templates und der damit verbundenen Komplexität. Die Benutzerfreundlichkeit der Oberfläche wird von CICS Administrator 1 in beiden Fällen als sehr gut betrachtet. Auf Grund dessen, dass noch nicht damit gearbeitet wurde, enthält sich CICS Administrator 2 der Bewertung. Bezüglich der Arbeitsweise bei Änderungen an den Workflow Definitionsdateien und dahinterliegenden Skripten usw., liegt die Meinung bei schlecht bis mittelmäßig. Hier fehlt beidene eine geeignete Toolunterstützung und das damit einhergehende Syntaxhighlighting. Der erste Eindruck wird von einem hohen Ersteinrichtungsaufwand und einer zeitaufwändigen Einarbeitungsphase geprägt. Zusammen mit den verschiedenen Sprachen hat dies auf CICS Administrator 1

eine abschreckende Wirkung. Dennoch können sich beide Befragten vorstellen, nachdem der Einarbeitungsaufwand erbracht wurde, täglich mit dem Toolkit zu arbeiten. Da bei dem aktuell etablierte Prozess ein hoher manueller Aufwand zu erbringen ist und eine Abstimmung zwischen den Administratoren und dem Entwicklerteam notwendig ist.

Zusammenfassend lässt sich sagen, dass die CICS Administratoren eine Chance auf Verbesserung des Prozesses durch das Toolkit sehen. Allerdings schreckt der hohe Einarbeitungsaufwand und die Mischung aus verschiedenen Sprachen und fehlendem Syntaxhighlighting ab.

#### 6.5.1.2. Db2 Administratoren

Sowohl Db2 Administrator 1 als auch Db2 Administrator 2 erkennen die Möglichkeit einer Verbesserung durch zOSMF. Jedoch sind sie der Meinung, dass noch einiges an Forschung und Weiterentwicklung notwendig ist um es sinnvoll nutzen zu können. Sie stimmen auch bei ihrer Ansicht bezüglich zOSPT überein. So sehen sie den Vorteil des Kommandozeileninterfaces vor allem bei einer Endausbaustufe mit automatisiertem Deployment innerhalb einer CI/DC-Pipeline und dem Einsatz von Jenkins. Db2 Administrator 2 stört sich an den Begriffen ‘Container‘ und ‘Image‘, da diese teilweise vertauscht und synonym verwendet werden. Bezüglich der Benutzerfreundlichkeit der Oberfläche fällt die Bewertung bei beiden schlecht bis mittelmäßig aus. Db2 Administrator 1 empfindet die gezeigte Arbeitsweise für Änderungen an den Workflow Definitionsdateien als sehr schlecht, da es momentan ohne automatisches Deployment realisiert ist. Die Bewertung von Db2 Administrator 2 ist mittelmäßig, da eine Entwicklungsumgebung sinnvoll wäre, vor allem im Hinblick auf eine Syntaxprüfung. Der erste Eindruck des Toolkits ist sehr positiv. Es wird als mächtiges Tool und als Zukunft des modernen Deployments auf dem Mainframe betitelt. Jedoch wird es als sehr komplex betrachtet. Im Vergleich dazu wird der aktuell etablierte Bereitstellungsprozess ebenfalls als komplex beschrieben. Dieser funktioniert zwar sehr gut, aber es sind viele Abhängigkeiten zu anderen Personen vorhanden, dadurch entstehen Wartezeiten. Außerdem sei ein sehr umfangreiches Wissen über alle beteiligten Subsysteme notwendig. Hinzu kommt ein hoher Konfigurationsaufwand und viel Vorarbeit, zum Beispiel Funktionsuser und ein Rechtekonzept. Beide Db2 Administratoren könnten sich um diese Probleme anzugehen, vorstellen mit dem Toolkit täglich zu arbeiten. Eine Verbindung mit Jenkins wird hierfür von Db2 Administrator 1 vorausgesetzt.

Die Interviews mit den Db2 Administratoren ergaben folgendes Bild. Sie sehen in dem Toolkit eine gute Möglichkeit um den Bereitstellungsprozess zu vereinfachen und weniger zeitaufwändig zu gestalten. Allerdings ist noch viel Forschungsarbeit in dieses Thema zu investieren. Als Hauptpunkt ist die Nutzung mit Jenkins und so die Einbindung und Etablierung einer automatisierten Lösung zu nennen.

### 6.5.1.3. MQ Administratoren

MQ Administrator 1 sieht bereits im aktuell funktionsfähigen Template einen Mehrwert. Zum einen weil mehr Verantwortung im Entwicklerteam liegt und zum anderen sind weniger händische Eingriffe notwendig. Jedoch ist die Lösung, die im Ausblick gezeigt wurde, flexibler und damit etwas besser geeignet. Zudem seien die momentan bereits vorhandenen Features durchaus gut, jedoch kam die Frage auf, ob die IBM das ‘IBM Cloud Provisioning and Management for z/OS’-Toolkit noch weiterentwickelt. Die Benutzerfreundlichkeit der zOSMF Oberfläche wurde als mittelmäßig bis gut eingestuft. Bezüglich des Arbeitens, Verwaltens und Ändern von Workflow Definitionsdateien und den dazugehörigen Skripten konnte keine Bewertung abgegeben werden, da noch nicht selbst damit gearbeitet wurde. Dies hat auch Einfluss auf den ersten Eindruck. So wird zu bedenken gegeben, dass der Zeitaufwand und die zu leistenden Vorarbeiten mit einzubeziehen sind. Vor allem, wenn die Provisionierung von einem MQ Queue Manager hinzu kommt. Jedoch kann sich MQ Administrator 1 vorstellen mit dem Toolkit täglich zu arbeiten, da letztendlich die Werkzeugwahl keine Rolle spielt. Diese Entscheidung wird dadurch begünstigt, dass der aktuell etablierte Prozess schlecht beurteilt wird. Aufgrund des hohen manuellen Aufwands und der dadurch erzeugten Rückfragen. Zuletzt wird noch darauf hingewiesen, dass das Toolkit generell noch Neuland sei. So müssten erst die Grundlagen gelernt und damit Erfahrung gesammelt werden bevor eine qualitativere Bewertung möglich sei. Dies beinhaltet wahrscheinlich eine starke Lernkurve.

Im Vergleich zu MQ Administrator 1 fehlt MQ Administrator 2 noch weitere Automatismen. So sind trotz des Einsatzes von zOSPT noch Absprachen mit Dritten, wie dem RACF-Team und dem Speicher-Team, notwendig. zOSPT sei zudem nur Docker ähnlich, ist jedoch keine vollumfängliche Containerlösung. So könnte sich MQ Administrator 2 zwar vorstellen mit dem Toolkit täglich zu arbeiten, aber es müsste ohne manuelle Eingriffe funktionieren. Die Erstellung der Skripte muss mit einem einmaligen Aufwand verbunden sein, so dass sie keine ständigen Anpassungen benötigen. Davon wird auch der erste Eindruck beeinflusst. So sind zwar viele gute Ansätze vorhanden, aber es fehlen Analogien und eine Ähnlichkeit zu Jenkins und anderen PaaS Lösungen. Dies geht soweit, dass XML nicht mehr als zeitgemäß betrachtet wird, sondern auf Umsetzungen in groovy, yaml oder mit ansible playbooks zu setzen sei.

Zusammenfassend lässt sich sagen, dass sich beide MQ Administratoren einig sind, dass der momentan etablierte Bereitstellungsprozess schlecht ist und ein neuer Prozess durchaus notwendig wäre. Der durch diese Arbeit gezeigte Prozess als Ablöse wird prinzipiell als möglich erachtet. Jedoch nur der Einsatz mit z/OSPT. Außerdem wird vor einer starken Lernkurve und noch fehlender Automation und der damit einhergehenden Ähnlichkeit zu Jenkins oder anderen PaaS Lösungen gewarnt.

#### 6.5.1.4. Entwicklerteam der DATEV Rechnungsschreibung

Aus Sicht des Entwicklers wird für den gezeigten Bereitstellungsprozess viel Wissen über die z/OSMF Oberfläche und das Template selbst benötigt. Dieses Wissen müsse auch bei geringer Nutzung über einen längeren Zeitraum erhalten werden. Der Prozess sei zwar schon ganz gut, jedoch ist weiterhin viel Absprache mit den Administratoren notwendig. Hier wird auch der Nachteil des momentan etablierter Bereitstellungsprozesses gesehen. Jedoch sobald der Erstaufwand geleistet wurde, muss sich nur im Ausnahmefall noch darum gekümmert werden. Diese Verantwortung würde im Fall der Umsetzung mit z/OSPT bei dem Entwicklerteam selbst liegen. Durch die eingesparten Absprachen wird sich eine höhere Effizienz erhofft. Es wurde noch die Nutzung in der Qualitätssicherungs- und Produktionsstage genannt. Hier werden Vorteile einer einfachen Skalierbarkeit gesehen. Vor allem auf Grund der Eigenverantwortung über die Subsysteme könnte sich die tägliche Arbeit mit dem Toolkit vorgestellt werden. Allerdings nur im Hinblick auf eine Integration in eine Jenkins Build Pipeline und sammeln von Erfahrungen bezüglich des Prozesses und des Toolings.

Das Hauptaugenmerk des Entwicklers liegt bei der höheren Eigenverantwortung beziehungsweise der Eigenverwaltung von benötigten Subsystemen. Eine einfache und intuitive Bedienung des Toolkits ist außerdem wichtig.

#### 6.5.1.5. Fachberaterin im Bereich Technologiestrategie

Nach der Fachberaterin im Bereich Technologiestrategie ist der gezeigte Ablauf beziehungsweise die zOSMF Oberfläche für eine solche Aufgabe geeignet. Jedoch sei es besser wenn zOSMF in den bereits existierenden ‘Marketplace’ für DATEV Cloud Lösungen integriert wäre. Der Prozess, der mit Hilfe von zOSPT ermöglicht wird, wird als gut angesehen. Da durch ihn die Entwicklung von z/OS Anwendungen an die Vorgehensweise der Cloud Native Entwicklung angenähert wird. Hier kommt die Rolle des Build Engineers auch für solche Anwendungen ins Spiel. Dieser kümmert sich um die Erstellung und Pflege der Build-Pipeline. Da sich die Fragen drei bis fünf auf die Nutzung und Verwaltung von Templates und somit auf die administrative Seite abzielen sind diese aus Entwicklersicht nicht relevant. Große Nachteile im momentan etablierten Bereitstellungsprozess ist vor allem das eine Anzahl an Entwickler, die an einem Produkt arbeiten, die gleiche Entwicklungsumgebung teilen. So arbeiten alle mit der gleichen CICS Instanz, der gleichen Test-Datenbank und mit den gleichen MQ Queues. Dadurch beeinflussen Änderungen die Tests der anderen Kollegen und müssen koordiniert werden. Falls Änderungen an der Umgebung notwendig sind, kann während dieser Zeit kein Entwickler weiterarbeiten. Hier sei der Vorteil des Toolkits. Es ermöglicht aus Entwicklersicht eine sehr einfache, schnelle Möglichkeit eine isolierte Umgebung bereitzustellen, unabhängig von dem Administratorenteams. Zusätzlich dienen die

Konfigurationsdateien auch als Dokumentation, welche Ressourcen für ein erneutes Erstellen der Umgebung notwendig sind.

Abschließend lässt sich sagen, dass aus Sicht einer Fachberaterin im Bereich Technologiestrategie dieses Toolkit die Entwicklung beziehungsweise den Bereitstellungsprozess deutlich verbessern kann. So ist für den Entwickler ein an die Cloud Native Welt angenäherter Entwicklungsprozess möglich. Dadurch wird der Wechsel zwischen beiden Umgebungen immer fließender.

### 6.5.2. Meinungsbild

Über alle Gruppen hinweg lassen sich folgende Punkte zusammenfassen:

- neuer Prozess notwendig
- z/OSPT Lösung bevorzugt
- erste Erfahrungen sammeln

Es stimmen alle Gruppen überein, dass der momentan etablierte Bereitstellungsprozess für Mainframesubsysteme durch viele Absprachen und Abstimmungs-Aufwand zeitäufwändig ist. Sie würden einen neuen schnelleren Prozess begrüßen.

Jedoch muss dieser Prozess aus Entwicklersicht mit minimalen Konfigurationsaufwand verbunden sein. Dies ermöglicht eine Umsetzung mittels z/OSPT und einer möglichen Integration in eine Jenkins Build Pipeline oder durch die Einbindung in den „DATEV Marktplatz“ mittels eines entwickelten „Service Brokers“. Aus Sicht der Administratoren sind mit dieser Umsetzung nur wenige allgemeine Templates zu verwalten. Da die Entwickler mit z/OSPT Images und keine weiteren Templates erzeugen. Um diese Punkte zu ermöglichen, muss das Template umgestaltet werden. Der dadurch in den Administratorenteams entstehende Aufwand und die damit verbundene steile Lernkurve hat eine abschreckende Wirkung.

Trotz dieser abschreckenden Wirkung sind auch die Administratorenteams bereit, falls die Kapazitäten vorhanden sind, den Bereitstellungsprozess mit Hilfe des „IBM Cloud Provisioning and Management for z/OS“ zu verbessern. Aus Sicht der Technologiestrategie ist das ein wichtiger und notwendiger Schritt in zu einem Cloud Native ähnlichen Prozess.

## Kapitel 7.

### Ausblick

Je weiter das Projekt dem Abschluss näher kam desto mehr kristallisierte sich ein Hauptproblem heraus. Dieses Template ist sehr auf die Rechnungsschreibung spezialisiert, dass heißt es ist funktionsfähig, aber kann nicht ohne zeitaufwendige Eingriffe in das Template, in die Workflowdefinitionsdateien und die eigentlichen REXX Skripte und Jobs, an eine andere Anwendung angepasst werden. Zusätzlich ist der durch den momentanen Stand des Templates ermöglichte Bereitstellungsprozess nicht optimal. Bei der Betrachtung des Falles, wenn zwei Anwender jeweils eine eigene Instanz des gleichen Templates benötigen, muss dieses kopiert werden und neu in zOSMF aufgenommen werden. Dies verlangt Wissen über die zOSMF Oberfläche und den Speicherort des Templates, um es schließlich auch zu ändern. Hinzu kommt, dass die Bereitstellung eines Queue Managers nicht im Template enthalten ist. So müsste dafür ebenfalls ein neuer Queue Manager angelegt werden. Eine Herangehensweise an dieses Problem wird im Folgenden beschrieben. Diese ist als Ausblick zu verstehen, die Umsetzung ist kein Bestandteil dieser Arbeit.

Angenommen das Template beinhaltet die Provisionierung eines Queue Managers. So könnte jeder Mitarbeitern seine eigene Instanz des Templates besitzen und beispielsweise für eigene Tests nutzen. Da die Application Id einer CICS Instanz eindeutig sein muss, müsste dennoch jeder Mitarbeiter ein eigenes Template dahingehend anpassen, dass dies gewährleistet ist. Eine Möglichkeit wäre eine Definition eines Pools mit verfügbaren Application Ids und dann mittels eines Programmes eine ungenutzte zu bestimmen. Dieses Programm kann dann in das Template mittels eines Steps aufgenommen werden. Das würde das Problem mit der eindeutigen Application Id lösen. Jedoch müsste immer noch für jede kleine Änderung an der Konfiguration des Templates ein neues erzeugt werden. Hier schafft zOSPT Abhilfe. Damit kann, wie in Absatz 3.3.2 beschrieben, mit Hilfe einer Konfigurationsdatei das Template von außerhalb gesteuert werden. Dadurch fällt das Kopieren des Template für den Mitarbeiter weg, dieser muss mittels des Kommandozeileninterfaces ein Image bauen und daraus einen Container erzeugen. Das Kommandozeileninterface hat einen weiteren Vorteil. Mit Hilfe von diesem können diese Arbeitsschritte in eine Jenkins-Ablauf aufgenommen werden. Somit läuft der Prozess automatisiert ab und nähert sich modernen Entwicklungsabläufen an.

Die Arbeitsweise und der Bereitstellungsprozess für die Laufzeitumgebung, die der Rechnungsschreibungs-Ablauf benötigt, ist dadurch vereinfacht und es werden weniger Absprachen benötigt. Allerdings ist das Template weit von einer Nutzung außerhalb der Rechnungsschreibung entfernt. Während der Realisierung dieser Arbeit wurde klar, dass anwendungsspezifische Templates nicht optimal sind und nicht den vollständigen Funktionsumfang von ‘IBM Cloud Provisioning and Management for z/OS’ ausreizen. So ist zu empfehlen, dass für jedes Subsystem, also CICS, Db2 und MQ, ein eigenständiges Template zu realisieren ist. Für die Realisierung davon müssten die Template dynamischer implementiert sein. Als Beispiel wird die Provisionierung von MQ Queues herangezogen. Momentan werden die Prozesse und die Trigger Queues sehr statisch angelegt. Das heißt, dass sowohl Namen als auch die damit verknüpften Queueparameter fest hinterlegt sind. Um nur ein Beispiel zu nennen. Dies müsste dahingehend angepasst werden, dass alle Parameter in einer Konfigurationsdatei angegeben werden können. Es ist auch in Betracht zu ziehen, ob für den User nur bestimmte vorgefertigte Profile, wie ‘klein’, ‘mittel’ und ‘groß’, auswählbar sind. Dies müsste alles in dem Template implementiert werden. Hierfür ist noch viel Zeitaufwand von Seiten der Administration einzuplanen.

Angenommen es existieren jeweils ein CICS, ein Db2 und ein MQ Template und diese sind so realisiert, dass sie firmenweit eingesetzt werden können. Dann ist der nächste Schritt, die Aufnahme in den ‘DATEV marketplace’, möglich. Der ‘DATEV marketplace’ ist eine Weboberfläche mit der sich Entwicklerteams ihre benötigte PaaS-Umgebung konfigurieren können. Heute stehen ihnen dort Dienste wie MongoDB, PostgreSQL, Kafka und viele weitere zur Verfügung. In weiter Zukunft könnten hier auch Dienste wie CICS, Db2 und MQ zur Auswahl stehen. Im Hintergrund würden diese über Templates und Images Instanzen bereitstellen. Um dies zu verwirklichen könnte die von zOSMF zur Verfügung gestellte REST-API verwendet werden. Diese ermöglicht den Zugriff auf fast alle zOSMF Funktionalitäten mittels Requests. Für die ‘Tenant’ Zuweisung zu einem Template existiert noch kein Request. Daran ist zu erkennen, dass von Seiten von zOSMF beziehungsweise von IBM ebenfalls noch Verbesserungsmöglichkeiten bestehen.

Diese technische Umsetzung ermöglicht in Zukunft den in Diagramm ?? dargestellten Bereitstellungsprozess. Es ist zu erkennen, dass Verantwortung von den Administratorenteams an die Entwicklerteams übertragen wird. Dadurch wird Kommunikationsaufwand eingespart und einem Entwickler steht binnen weniger Minuten eine funktionsfähige Laufzeitumgebung für seine legacy z/OS Anwendung zur Verfügung. Bei Problemen oder Beratungswunsch unterstützen die Administratorenteams weiterhin.



## Kapitel 8.

### Zusammenfassung

Zusammenfassend lässt sich sagen, dass es generell möglich ist mit dem ‘IBM Cloud Provisioning and Management for z/OS’-Toolkit Laufzeitumgebungen für legacy z/OS Anwendungen automatisiert bereitzustellen. Das funktionsfähige Template verkürzt den Bereitstellungsprozess deutlich. Durch den Abbau von Kommunikation zwischen den Abteilungen und nur einmaligen erstellen der Skripte ist es zudem weniger fehleranfällig. Die Stakeholder sehen in diesem Template auch einen Mehrwert. Jedoch ist es noch nicht perfekt. Der Bereitstellungsprozess ist noch immer mit einigen manuellen Schritten verbunden. So muss das Template manuell kopiert und Änderungen an der Konfiguration müssen innerhalb des Templates stattfinden.

Hierfür wurde in der Arbeit eine Lösung mit Hilfe von zOSPT beleuchtet. Diese sieht in einer Endausbaustufe eine einfache Einbindung des Templates in einen automatisierten Build-Prozess, zum Beispiel mit Jenkins, vor. Außerdem würde der Einsatz von zOSPT das Einbinden in den DATEV eG internen ‘Marketplace’ für Cloud Lösungen ermöglichen. Um diese Ziele zu erreichen muss noch viel Aufwand in die Gestaltung des Templates gesteckt werden. Zusätzlich müsste ein sogenannter ‘Service Broker’ für die Einbindung der einzelnen Subsysteme in den ‘Marketplace’ implementiert werden. Diese beiden Lösungsansätze stoßen sowohl bei den Administratorenteams als auch beim involvierten Entwicklerteam auf fruchtbaren Boden. Dadurch wird eine Ähnlichkeit zum ‘Cloud Native’-Bereitstellungsprozesses hergestellt. Dies ist ein weiterer Schritt um dem Image eines veralteten Systems mit veraltetem langsamen Prozesses zu entgegenkommen.



## Anhang A.

### Anhang

#### A.1. Produktstammdaten data definition language

```
1  CREATE TABLE PSSSCHEMA.TPAUSPSSBASELBART
2      (PARTITIONID INTEGER
3
4      NOT NULL
5  WITH DEFAULT 1
6      ,PID INTEGER
7
8      NOT NULL
9      ,AUSPRAEGUNG SMALLINT
10
11     NOT NULL
12     ,GUELTIGAB DATE
13
14     NOT NULL
15     ,LFDNR INTEGER
16
17     NOT NULL
18     WITH DEFAULT
19     ,GUELTIGBIS DATE
20
21     NOT NULL
22     WITH DEFAULT "9999-12-31 "
23     ,PSSID INTEGER
24     WITH DEFAULT NULL
25     ,ZUSATZID CHARACTER(4) FOR SBCS DATA
26     WITH DEFAULT NULL
27     ,BEZEICHNUNG VARCHAR(100) FOR SBCS DATA
28     WITH DEFAULT NULL
29     ,MWSTANTEILFREI DECIMAL(5, 2)
30     WITH DEFAULT NULL
31     ,MWSTANTEILREDUZIERT DECIMAL(5, 2)
32     WITH DEFAULT NULL
33     ,MWSTANTEILVOLL DECIMAL(5, 2)
34     WITH DEFAULT NULL
```

```

29      ,CONSTRAINT PID PRIMARY KEY
30      (PARTITIONID
31      ,PID
32      ,AUSPRAEGUNG
33      ,GUELTIGAB
34      ,LFDNR
35      )
36  )
37  IN DATABASE PSSBAPRV
38  APPEND NO
39  NOT VOLATILE CARDINALITY
40  DATA CAPTURE NONE
41  AUDIT NONE
42  CCSID EBCDIC
43  PARTITION BY RANGE
44      (PARTITIONID NULLS LAST ASC
45      )
46      ( PARTITION 1
47          ENDING ( 1
48      ) INCLUSIVE
49      , PARTITION 2
50          ENDING ( 2
51      ) INCLUSIVE
52      );
53
54  CREATE UNIQUE INDEX PSSSCHEMA.PPAUSPSSBASELBART
55      ON PSSSCHEMA.TPAUSPSSBASELBART
56      (PARTITIONID ASC
57      ,PID ASC
58      ,AUSPRAEGUNG ASC
59      ,GUELTIGAB ASC
60      ,LFDNR ASC
61      )
62      INCLUDE NULL KEYS
63      CLUSTER
64      PARTITIONED
65      DEFINE YES
66      COMPRESS NO
67      BUFFERPOOL BP2
68      CLOSE YES

```

```

69      DEFER NO
70      COPY NO
71      PARTITION BY RANGE
72      (PARTITION 1
73          USING STOGROUP STAPSA01
74              PRIQTY -1
75              SECQTY -1
76              ERASE NO
77          FREEPAGE 0
78          PCTFREE 10
79          GBPCACHE CHANGED
80      ,PARTITION 2
81          USING STOGROUP STAPSA01
82              PRIQTY -1
83              SECQTY -1
84              ERASE NO
85          FREEPAGE 0
86          PCTFREE 10
87          GBPCACHE CHANGED);
88
89  CREATE TABLE PSSSCHEMA.TMAXNUM
90      (MAXNUMID INTEGER
91
92          ,MAXNUM INTEGER
93
94          ,MAXNUBEZ CHARACTER(42) FOR SBCS DATA
95
96      WITH DEFAULT "X"
97          ,BEZEICHNUNG VARCHAR(100) FOR SBCS DATA
98
99      WITH DEFAULT "X"
100      ,CONSTRAINT MAXNUMID PRIMARY KEY
101      (MAXNUMID
102      )
103      )
104      IN DATABASE PSSBAPRV
105      APPEND NO
106      NOT VOLATILE CARDINALITY
107      DATA CAPTURE NONE
108      AUDIT NONE

```

```

109 CCSID EBCDIC;
110
111 COMMENT ON TABLE PSSSCHEMA.TMAXNUM
112     IS "maximale_□Nummer";
113
114
115 SET CURRENT SQLID = "DB2SADM";
116
117
118 COMMENT ON COLUMN PSSSCHEMA.TMAXNUM.MAXNUMID
119     IS "ID_□für_□maximalen_□Nummer";
120
121
122 SET CURRENT SQLID = "DB2SADM";
123
124
125 COMMENT ON COLUMN PSSSCHEMA.TMAXNUM.MAXNUM
126     IS "maximale_□Nummer";
127
128
129 SET CURRENT SQLID = "DB2SADM";
130
131
132 COMMENT ON COLUMN PSSSCHEMA.TMAXNUM.MAXNUBEZ
133     IS "Bezeichnung_□für_□maximale_□Nummer";
134
135
136 SET CURRENT SQLID = "DB2SADM";
137
138
139 COMMENT ON COLUMN PSSSCHEMA.TMAXNUM.BEZEICHNUNG
140     IS "Bezeichnung_□für_□maximale_□Nummer";
141
142 CREATE UNIQUE INDEX PSSSCHEMA.PMAXNUM
143     ON PSSSCHEMA.TMAXNUM
144     (MAXNUMID ASC
145     )
146     INCLUDE NULL KEYS
147     CLUSTER
148     DEFINE YES

```

```
149      COMPRESS NO
150      BUFFERPOOL BP2
151      CLOSE YES
152      DEFER NO
153      COPY NO
154      USING STOGROUP STALDL01
155          PRIQTY -1
156          SECQTY -1
157          ERASE NO
158      FREEPAGE 0
159      PCTFREE 99
160      GBPCACHE CHANGED
161      PIECESIZE 2097152K;
162
163 CREATE FUNCTION PSS.WHICH_PARTITIONID
164     (
165         MAXID INTEGER )
166     RETURNS INTEGER
167     VERSION V1
168     DISALLOW DEBUG MODE
169     ASUTIME NO LIMIT
170     INHERIT SPECIAL REGISTERS
171     WLM ENVIRONMENT FOR DEBUG MODE DB0TWLM
172     APPLICATION ENCODING SCHEME EBCDIC
173     QUALIFIER UGPSENT
174     DYNAMICRULES RUN
175     WITH EXPLAIN
176     WITHOUT IMMEDIATE WRITE
177     ISOLATION LEVEL UR
178     OPTHINT " "
179     REOPT NONE
180     VALIDATE RUN
181     ROUNDING DEC_ROUND_HALF_EVEN
182     DATE FORMAT ISO
183     DECIMAL( 31 )
184     FOR UPDATE CLAUSE REQUIRED
185     TIME FORMAT ISO
186     CURRENT DATA NO
187     DEGREE 1
188     PACKAGE OWNER UGPSENT
```

```

189 BUSINESS_TIME SENSITIVE NO
190 SYSTEM_TIME SENSITIVE NO
191 ARCHIVE SENSITIVE NO
192 APPLCOMPAT V10R1
193 LANGUAGE SQL
194 NO EXTERNAL ACTION
195 PARAMETER CCSID EBCDIC
196 DETERMINISTIC
197     NOT SECURED
198     CALLED ON NULL INPUT
199     READS SQL DATA
200     SPECIFIC WHICH_PARTITIONID
201 BEGIN
202     DECLARE MAXNUM INTEGER;
203     SELECT MAXNUM
204         INTO MAXNUM
205         FROM AVADMIN.AMAXNUM
206         WHERE MAXNUMID = MAXID;
207     RETURN MAXNUM;
208 END;
209
210 SET PATH = "PSS" , "SYSIBM" , "SYSFUN" , "SYSPROC" , "SYSIBMADM" , "PSSSCHEMA" ;
211
212 CREATE VIEW PSSSCHEMA.VPAUSPSS_BASELBART
213     ( PARTITIONID
214     , PID
215     , AUSPRAEGUNG
216     , GUELTIGAB
217     , LFDNR
218     , GUELTIGBIS
219     , PSSID
220     , ZUSATZID
221     , BEZEICHNUNG
222     , MWSTANTEILFREI
223     , MWSTANTEILREDUZIERT
224     , MWSTANTEILVOLL
225     ) AS
226 SELECT B.* FROM TPAUSPSSBASELBART B WHERE B.PARTITIONID =
227     PSS.WHICH_PARTITIONID ( 3011 )
228 ;

```



```

229
230 CREATE TABLE PSSSCHEMA.TPAUSPSSPREISE
231     (PARTITIONID INTEGER
232                                     NOT NULL
233 WITH DEFAULT 1
234     ,ARTNR INTEGER
235                                     NOT NULL
236     ,PREISTYPID SMALLINT
237                                     NOT NULL
238     ,GUELTIGAB DATE
239                                     NOT NULL
240     ,STAFFELNR INTEGER
241                                     NOT NULL
242     ,PID INTEGER
243                                     NOT NULL
244 WITH DEFAULT
245     ,PREISREGEL CHARACTER(2) FOR SBCS DATA
246                                     NOT NULL
247 WITH DEFAULT "X"
248     ,GUELTIGBIS DATE
249                                     NOT NULL
250 WITH DEFAULT "9999-12-31"
251     ,PRODUKTPREIS DECIMAL(11, 3)
252 WITH DEFAULT NULL
253     ,EINZELPREIS DECIMAL(8, 3)
254 WITH DEFAULT NULL
255     ,PREISINTERVALL DECIMAL(11, 3)
256 WITH DEFAULT NULL
257     ,PREISEINHEIT DECIMAL(8, 3)
258 WITH DEFAULT NULL
259     ,STAFFELVERTEILUNG CHARACTER(1) FOR SBCS DATA
260 WITH DEFAULT NULL
261     ,INTERVALLVON INTEGER
262 WITH DEFAULT NULL
263     ,INTERVALLBIS INTEGER
264 WITH DEFAULT NULL
265     ,PREISTYPBEZ VARCHAR(100) FOR SBCS DATA
266                                     NOT NULL
267 WITH DEFAULT "X"
268     ,PREISAB DECIMAL(8, 3)

```

```

269 WITH DEFAULT NULL
270     ,PREISABRELEVANZ CHARACTER(1) FOR SBCS DATA
271                                     NOT NULL
272 WITH DEFAULT "K"
273     ,EINHEIT INTEGER
274 WITH DEFAULT NULL
275     ,CONSTRAINT PPAUSPSSPREISE PRIMARY KEY
276     (PARTITIONID
277     ,ARTNR
278     ,PREISTYPID
279     ,GUELTIGAB
280     ,STAFFELNR
281     )
282     )
283     IN DATABASE PSSBAPRV
284     APPEND NO
285     NOT VOLATILE CARDINALITY
286     DATA CAPTURE NONE
287     AUDIT NONE
288     CCSID EBCDIC;
289
290 CREATE INDEX PSSSCHEMA.IPAUSPSSPREISE
291     ON PSSSCHEMA.TPAUSPSSPREISE
292     (PARTITIONID ASC
293     ,PID ASC
294     )
295     INCLUDE NULL KEYS
296     NOT CLUSTER
297     DEFINE YES
298     COMPRESS NO
299     BUFFERPOOL BP2
300     CLOSE YES
301     DEFER NO
302     COPY NO
303     USING STOGROUP STAPSA01
304         PRIQTY -1
305         SECQTY -1
306         ERASE NO
307     FREEPAGE 0
308     PCTFREE 10

```

```
309         GBPCACHE CHANGED
310         PIECESIZE 2097152K;
311
312 CREATE INDEX PSSSCHEMA.IPAUSPSSPREISE2
313         ON PSSSCHEMA.TPAUSPSSPREISE
314         (PARTITIONID ASC
315         ,ARTNR ASC
316         ,PREISTYPID ASC
317         ,GUELTIGAB ASC
318         ,INTERVALLVON ASC
319         )
320         INCLUDE NULL KEYS
321         NOT CLUSTER
322         DEFINE YES
323         COMPRESS NO
324         BUFFERPOOL BP2
325         CLOSE YES
326         DEFER NO
327         COPY NO
328         USING STOGROUP STAPSA01
329             PRIQTY -1
330             SECQTY -1
331             ERASE NO
332         FREEPAGE 0
333         PCTFREE 10
334         GBPCACHE CHANGED
335         PIECESIZE 2097152K;
336
337 CREATE UNIQUE INDEX PSSSCHEMA.PPAUSPSSPREISE
338         ON PSSSCHEMA.TPAUSPSSPREISE
339         (PARTITIONID ASC
340         ,ARTNR ASC
341         ,PREISTYPID ASC
342         ,GUELTIGAB ASC
343         ,STAFFELNR ASC
344         )
345         INCLUDE NULL KEYS
346         CLUSTER
347         DEFINE YES
348         COMPRESS NO
```

```

349     BUFFERPOOL BP2
350     CLOSE YES
351     DEFER NO
352     COPY NO
353     USING STOGROUP STAPSA01
354         PRIQTY -1
355         SECQTY -1
356         ERASE NO
357     FREEPAGE 0
358     PCTFREE 10
359     GBPCACHE CHANGED
360     PIECESIZE 2097152K;
361
362     CREATE TABLE PSSSCHEMA.TPAUSPSSBART
363         (PARTITIONID INTEGER
364
365             NOT NULL
366     WITH DEFAULT 1
367         ,PID INTEGER
368
369             NOT NULL
370         ,ARTNR INTEGER
371
372             NOT NULL
373     WITH DEFAULT "1966-02-14"
374         ,OPDATBEN CHARACTER(1) FOR SBCS DATA
375
376             NOT NULL
377     WITH DEFAULT "J"
378         ,AUSDIENSTREL CHARACTER(1) FOR SBCS DATA
379
380             NOT NULL
381     WITH DEFAULT "N"
382         ,VERTRIEBSREL CHARACTER(1) FOR SBCS DATA
383
384             NOT NULL
385     WITH DEFAULT NULL
386         ,KOMMASTELLEN INTEGER
387     WITH DEFAULT NULL
388         ,ERTRNR INTEGER
389
390             NOT NULL

```

389	WITH <b>DEFAULT</b>	
390	,GFEDNR <b>INTEGER</b>	
391		<b>NOT NULL</b>
392	WITH <b>DEFAULT</b>	
393	,UPLONR <b>INTEGER</b>	
394		<b>NOT NULL</b>
395	WITH <b>DEFAULT</b>	
396	,MWSTEUERSATZ <b>INTEGER</b>	
397	WITH <b>DEFAULT NULL</b>	
398	,POLINR <b>INTEGER</b>	
399		<b>NOT NULL</b>
400	WITH <b>DEFAULT</b>	
401	,EXPGNR <b>INTEGER</b>	
402		<b>NOT NULL</b>
403	WITH <b>DEFAULT</b>	
404	,EXPONR <b>INTEGER</b>	
405		<b>NOT NULL</b>
406	WITH <b>DEFAULT</b>	
407	,ARTIKELTYPID <b>INTEGER</b>	
408		<b>NOT NULL</b>
409	WITH <b>DEFAULT</b>	
410	,ARTIKELTYPALT <b>CHARACTER</b> (4) FOR SBCS DATA	
411		<b>NOT NULL</b>
412	WITH <b>DEFAULT</b> "0000 "	
413	,BERBESTEINHID <b>INTEGER</b>	
414		<b>NOT NULL</b>
415	WITH <b>DEFAULT</b>	
416	,BERBESTEINHALT <b>CHARACTER</b> (4) FOR SBCS DATA	
417		<b>NOT NULL</b>
418	WITH <b>DEFAULT</b> "0000 "	
419	,LEISTGRUPID <b>INTEGER</b>	
420		<b>NOT NULL</b>
421	WITH <b>DEFAULT</b>	
422	,LEISTGRUPALT <b>CHARACTER</b> (4) FOR SBCS DATA	
423		<b>NOT NULL</b>
424	WITH <b>DEFAULT</b> "0000 "	
425	,BERFREQID <b>INTEGER</b>	
426		<b>NOT NULL</b>
427	WITH <b>DEFAULT</b>	
428	,NUTZERID <b>INTEGER</b>	

429	WITH <b>DEFAULT</b>	
430	,LEISTARTID <b>INTEGER</b>	
431		<b>NOT NULL</b>
432	WITH <b>DEFAULT</b>	
433	,BERFREQALT <b>CHARACTER</b> (4) FOR SBCS DATA	
434		<b>NOT NULL</b>
435	WITH <b>DEFAULT</b> " 0000 "	
436	,LEISTARTALT <b>CHARACTER</b> (4) FOR SBCS DATA	
437		<b>NOT NULL</b>
438	WITH <b>DEFAULT</b> " 99 "	
439	,NUTZERALT <b>CHARACTER</b> (1) FOR SBCS DATA	
440		<b>NOT NULL</b>
441	WITH <b>DEFAULT</b> "K"	
442	,BARTBEZ_20 <b>CHARACTER</b> (20) FOR SBCS DATA	
443		<b>NOT NULL</b>
444	WITH <b>DEFAULT</b> "X"	
445	,ARTIKELTYPBEZ <b>CHARACTER</b> (50) FOR SBCS DATA	
446		<b>NOT NULL</b>
447	WITH <b>DEFAULT</b> " Keine□Zuordnung "	
448	,BERBESTEINHBEZ <b>CHARACTER</b> (50) FOR SBCS DATA	
449		<b>NOT NULL</b>
450	WITH <b>DEFAULT</b> " Keine□Zuordnung "	
451	,LEISTGRUPBEZ <b>CHARACTER</b> (50) FOR SBCS DATA	
452		<b>NOT NULL</b>
453	WITH <b>DEFAULT</b> " Keine□Zuordnung "	
454	,BERFREQBEZ <b>CHARACTER</b> (50) FOR SBCS DATA	
455		<b>NOT NULL</b>
456	WITH <b>DEFAULT</b> " Keine□Zuordnung "	
457	,NUTZERBEZ <b>CHARACTER</b> (50) FOR SBCS DATA	
458		<b>NOT NULL</b>
459	WITH <b>DEFAULT</b> " Keine□Zuordnung "	
460	,LEISTARTBEZ <b>CHARACTER</b> (50) FOR SBCS DATA	
461		<b>NOT NULL</b>
462	WITH <b>DEFAULT</b> " Keine□Zuordnung "	
463	,BARTBEZ_100 <b>VARCHAR</b> (100) FOR SBCS DATA	
464		<b>NOT NULL</b>
465	WITH <b>DEFAULT</b> "X"	
466	,ERTRBEZ <b>VARCHAR</b> (100) FOR SBCS DATA	
467		<b>NOT NULL</b>
468	WITH <b>DEFAULT</b> "X"	

469	,GFEDBEZ <b>VARCHAR</b> (100) FOR SBCS DATA	
470		<b>NOT NULL</b>
471	WITH <b>DEFAULT</b> "X"	
472	,UPLOBEZ <b>VARCHAR</b> (100) FOR SBCS DATA	
473		<b>NOT NULL</b>
474	WITH <b>DEFAULT</b> "X"	
475	,POLIBEZ <b>VARCHAR</b> (100) FOR SBCS DATA	
476		<b>NOT NULL</b>
477	WITH <b>DEFAULT</b> "X"	
478	,EXPGBEZ <b>VARCHAR</b> (100) FOR SBCS DATA	
479		<b>NOT NULL</b>
480	WITH <b>DEFAULT</b> "X"	
481	,EXPOBEZ <b>VARCHAR</b> (100) FOR SBCS DATA	
482		<b>NOT NULL</b>
483	WITH <b>DEFAULT</b> "X"	
484	,HAKONR <b>INTEGER</b>	
485	WITH <b>DEFAULT</b> NULL	
486	,HAKOBEZ <b>VARCHAR</b> (100) FOR SBCS DATA	
487	WITH <b>DEFAULT</b> NULL	
488	,INPGNR <b>INTEGER</b>	
489		<b>NOT NULL</b>
490	WITH <b>DEFAULT</b>	
491	,INPGBEZ <b>VARCHAR</b> (100) FOR SBCS DATA	
492		<b>NOT NULL</b>
493	WITH <b>DEFAULT</b> "X"	
494	,BEZ035 <b>CHARACTER</b> (35) FOR SBCS DATA	
495		<b>NOT NULL</b>
496	WITH <b>DEFAULT</b> "X"	
497	, <b>CONSTRAINT</b> PPAUSPSSBART <b>PRIMARY KEY</b>	
498	(PARTITIONID	
499	,PID	
500	)	
501	, <b>CONSTRAINT</b> UPAUSPSSBART <b>UNIQUE</b>	
502	(PARTITIONID	
503	,ARTNR	
504	)	
505	)	
506	IN DATABASE PSSBAPRV	
507	APPEND <b>NO</b>	
508	<b>NOT</b> VOLATILE CARDINALITY	

```

509 DATA CAPTURE NONE
510 AUDIT NONE
511 CCSID EBCDIC
512 PARTITION BY RANGE
513     (PARTITIONID NULLS LAST ASC
514     )
515     ( PARTITION 1
516         ENDING ( 1
517     ) INCLUSIVE
518     , PARTITION 2
519         ENDING ( 2
520     ) INCLUSIVE
521     );
522
523 CREATE UNIQUE INDEX PSSSCHEMA.PPAUSPSSBART
524     ON PSSSCHEMA.TPAUSPSSBART
525     (PARTITIONID ASC
526     ,PID ASC
527     )
528     INCLUDE NULL KEYS
529     CLUSTER
530     PARTITIONED
531     DEFINE YES
532     COMPRESS NO
533     BUFFERPOOL BP2
534     CLOSE YES
535     DEFER NO
536     COPY NO
537     PARTITION BY RANGE
538     (PARTITION 1
539         USING STOGROUP STAPSA01
540             PRIQTY -1
541             SECQTY -1
542             ERASE NO
543             FREEPAGE 0
544             PCTFREE 10
545             GBPCACHE CHANGED
546     ,PARTITION 2
547         USING STOGROUP STAPSA01
548             PRIQTY -1

```



```
549                                     SECQTY -1
550                                     ERASE NO
551                                     FREEPAGE 0
552                                     PCTFREE 10
553                                     GBPCACHE CHANGED);
554
555 CREATE UNIQUE INDEX PSSSCHEMA.UPAUSPSSBART
556     ON PSSSCHEMA.TPAUSPSSBART
557     (PARTITIONID ASC
558     ,ARTNR ASC
559     )
560     INCLUDE NULL KEYS
561     NOT CLUSTER
562     DEFINE YES
563     COMPRESS NO
564     BUFFERPOOL BP2
565     CLOSE YES
566     DEFER NO
567     COPY NO
568     USING STOGROUP STAPSA01
569         PRIQTY -1
570         SECQTY -1
571         ERASE NO
572         FREEPAGE 0
573         PCTFREE 10
574         GBPCACHE CHANGED
575         PIECESIZE 2097152K;
576
577 CREATE UNIQUE INDEX PSSSCHEMA.UPAUSPSSBART
578     ON PSSSCHEMA.TPAUSPSSBART
579     (PARTITIONID ASC
580     ,ARTNR ASC
581     )
582     INCLUDE NULL KEYS
583     NOT CLUSTER
584     DEFINE YES
585     COMPRESS NO
586     BUFFERPOOL BP2
587     CLOSE YES
588     DEFER NO
```

589	COPY NO
590	USING STOGROUP STAPSA01
591	PRIQTY -1
592	SECQTY -1
593	ERASE NO
594	FREEPAGE 0
595	PCTFREE 10
596	GBPCACHE CHANGED
597	PIECESIZE 2097152K;

## A.2. Interview Fragebögen

# Abbildungsverzeichnis

1.1. Anteil der verwendeten Programmiersprachen auf dem Mainframe bei DATEV eG in Prozent . . . . .	3
3.1. Architekturübersicht über die Subsysteme bei DATEV eG . . . . .	14
3.2. z/OSPT mögliche Kommandozeilenbefehle . . . . .	17
5.1. Bereistellungsprozess einer Db2 Datenbank . . . . .	27
5.2. Bereistellungsprozess von IBM MQ Queues . . . . .	28
6.1. Beispiel einer Fehlermeldung von zOSMF . . . . .	40



# Tabellenverzeichnis

6.1. Zu verändernde Variablen im minimalen CICS Template . . . . .	31
--	----



## Quellcodeverzeichnis

listings/ddl.txt . . . . .	51
----------------------------	----

