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 e.G.

Bachelorarbeit im Studiengang Informatik

vorgelegt von

David Krug

Matrikelnummer 3036355

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

TECHNISCHE HOCHSCHULE NÜRNBERG
GEORG SIMON OHM

Prüfungsrechtliche Erklärung der/des Studierenden

Angaben des bzw. der Studierenden:

Name: Krug Vorname: David Matrikel-Nr.: 3036355

Fakultät: Informatik Studiengang: Informatik

Semester: Wintersemester 2019/2020

Titel der Abschlussarbeit:

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

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

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

Ort, Datum, Unterschrift Studierende/Studierender

Sperrvermerk

Die vorliegende Arbeit beinhaltet interne vertrauliche Informationen der DATEV e.G. Sie ist nur für die Beteiligten an der Begutachtung bestimmt. Die Weitergabe des Inhalts der Arbeit im Gesamten oder in Teilen sowie das Anfertigen von Kopien oder Abschriften – auch in digitaler Form – vor dem Ablauf der Sperrfrist von 2 Jahren untersagt. Ausnahmen bedürfen der schriftlichen Genehmigung der DATEV e.G.

Kurzdarstellung

Ziel dieser Arbeit ist es, zu bestimmen, ob die Bereitstellung von Laufzeitumgebungen für legacy z/OS Anwendungen über einen cloud nativen, Platform-as-a-Service Ansatz bei DATEV e.G. möglich ist. Es werden folgende Forschungsfragen gestellt:

- Ist es möglich, den Bereitstellungsprozess für z/OS Anwendung bei DATEV e.G. mit Hilfe des „IBM Cloud Provisioning and Management for z/OS“-Tools an cloud native Prozesse anzunähern?
- Erzeugt die Nutzung von „IBM Cloud Provisioning and Management for z/OS“ einen Mehrwert bei den Stakeholdern, also den Entwicklerteams und den Administratorenteams?

Dafür wurde anhand einer Beispielanwendung von der DATEV e.G. das Tool „IBM Cloud Provisioning and Management for z/OS“ untersucht. Es wurden zwei vorhandene Möglichkeiten aufgezeigt, eine davon implementiert. Für ein Meinungsbild bezüglich Mehrwertes und Akzeptanz des Tools, wurden Interviews mit Stakeholdern durchgeführt. Diese Bild zeigt, dass in dem Tool eine Chance auf Verbesserung der aktuellen Prozesse gesehen wird.

Ergebnis war auch, dass die implementierte Variante nicht optimal für den Praxiseinsatz bei DATEV e.G. ist, aber eine wichtige Basis für die automatisierte Bereitstellung von Laufzeitumgebungen für z/OS Anwendungen darstellt. Weiterführende Forschungen könnte darauf aufbauend Variante zwei untersuchen und Möglichkeiten einer weiteren, praxisgeeigneteren Optimierung des z/OS Bereitstellungsprozesses aufzeigen.

Vorwort

Die vorliegende Bachelorarbeit entstand im Rahmen meines Verbundstudiums bei der DATEV e.G. in Nürnberg.

Für die Betreuung meiner Bachelorarbeit möchte ich mich bei Prof. Dr.-Ing. Korbinian Riedhammer und Prof. Dr. rer. nat. Friedhelm Stappert bedanken.

Für ihre Unterstützung möchte ich auch meiner Betreuerin in der Firma DATEV e.G., Sabine Lauterbach, herzlich danken.

Außerdem danke ich allen Kollegen aus dem CICS-,Db2-,IBM MQ-Administratorenteams und dem Entwicklerteam der DATEV-Rechnungsschreibung dafür, dass ich jeder Zeit mit Fragen und Anliegen auf sie zugehen durfte und mir immer freundlich weitergeholfen wurde.

Nicht zuletzt danke ich meiner Familie und Freunde für die Motivation und den Beistand während meines Bildungsweges, welcher mir dadurch deutlich erleichtert wurde.

Inhaltsverzeichnis

Begriffe, die im Glossar erläutert werden, werden bei ihrem ersten Auftreten rot markiert.

Kapitel 1.

Einleitung

„I recently predicted the last mainframe will be unplugged on March 15, 1996“[1] - 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 würde. Warum war diese Vorhersage falsch? Wieso wird sich im Jahre 2020 immer noch mit dieser Technologie beschäftigt? Und was genau ist ein Großrechner?

Kurz gesagt ist ein Großrechner[2] ein leistungsstarkes, zentralisiertes Serversystem. In dieser Arbeit wird nur auf Mainframes aus dem Hause IBM, die sogenannte z-Plattform, eingegangen. Damit ist auch der Technologiestack festgelegt. Das verwendete Betriebssystem ist z/OS, darauf werden Middleware Produkte wie CICS[3], das Datenbankmanagementsystem Db2[4] sowie die Messaging Lösung „IBM MQ“[5] betrieben. Als Programmiersprachen werden z.B. COBOL, IBM Assembler, C und C++ verwendet. Seit ca. 1997 ist auch Java auf dem Mainframe verfügbar. [6]

Der IBM Mainframe hat eine lange Geschichte. Vor mehr als fünfzig Jahren wurde der erste Großrechner, das sog. „System/360“ vorgestellt. Bis in die 90er Jahre spielte der IBM Mainframe eine Hauptrolle auf dem Computermarkt, dann gewannen zunehmend verteilte Client-Server-Systeme an Bedeutung.[7] Seitdem gilt der Mainframe bereits als „legacy“ und damit als „Altlast“[8].

Wieso also wird sich mit der Mainframe Technologie noch beschäftigt? Eine Antwort: Auf dem Mainframe werden auch im Jahr 2020 geschäftskritische Anwendungen in der ganzen Welt gehostet. So verwenden laut IBM 92 der 100 weltweit führenden Banken für ihre Kernabläufe einen IBM Mainframe. Dies beinhaltet 87 Prozent aller Kreditkartentransaktionen und ca. 350.000 Transaktionen pro Sekunde.[9] Inklusive dieser Transaktionen verarbeiten

[1] [Also 93, S. 4]
[2] Beschreibung im Absatz 2.2 zu finden
[3] Anwendungsserver, CICS Beschreibung Absatz 2.4.1
[4] Beschreibung Absatz 2.4.2
[5] Beschreibung im Absatz 2.4.3 zu finden
[6] [Stee 03, S. 6]
[7] [Ceru 03, Kap. 5]
[8] [http 20k]
[9] [http 20a]

Großrechner heutzutage weltweit circa 1,2 Millionen CICS Transaktionen pro Sekunde.[10] Im Vergleich hierzu werden 63.000 Google Suchanfragen pro Sekunde abgesetzt. [11]

Aus der Kombination von hohem Workload, der Abhängigkeit von einem Hersteller (IBM) und dem als veraltet geltenden Technologiestack entstehen jedoch zunehmend Risiken. Es wird immer schwieriger, Nachwuchs in diesem Bereich zu finden. Zum einem, da Mainframe-Know How kaum noch an Universitäten gelehrt wird. Die Seite des Hochschulkompass[12] liefert z.B. weder für „Mainframe“ noch für „Großrechner“ einen Treffer. Zum anderen ist der demographische Faktor bei den Wissensträgern nicht zu vernachlässigen. Diese sind - wie die Technologien auf dem Mainframe - in die Jahre gekommen und erreichen das Rentenalter.[13]

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 existiert eine starke Hersteller- und Plattformabhängigkeit, z.B. in Bezug auf CICS, Db2, IBM-COBOL-Compiler, IBM Assembler.

Offensichtlich betreiben dennoch etliche Firmen einen IBM Großrechner. Dazu zählen hauptsächlich Banken, Versicherungen, Fluggesellschaften usw.[14] Der gemeinsame Nenner dieser Unternehmen ist, dass sich über die Jahre und Jahrzehnte enorme Investitionen auf dem Mainframe angesammelt haben. Die entstandenen, hochgradig geschäftskritischen Kernsysteme haben hohe Anforderungen an Massendatenverarbeitung, Sicherheitsstandards und Hochverfügbarkeit. All diese Punkte sprechen nach wie vor für die Nutzung eines Großrechners, z.B. auch bei der DATEV e.G., weshalb man nach Einführung der IBM z14 Plattform 2019 als Referenzkunde agierte.[15] „Hosting our SaaS and cloud services on the IBM z14 gives us peace of mind that customer information is kept secure, in compliance with data protection regulations.“[16]

Die DATEV e.G. wurde am 14.02.1966 von 65 Steuerbevollmächtigten gegründet. Sie verfolgten mit der Gründung das Ziel, Buchführungsaufgaben für ihre Mandanten mit Hilfe der neu aufkommenden EDV zu bewältigen. Aufgrund hohen Mitgliederwachstums wurde hierfür bereits 1969 in einen firmeneigenen IBM-Großrechner investiert.[http 19b] Heute umfasst das Leistungsspektrum der DATEV e.G. unter anderem das Rechnungswesen, Personalwirtschaft, Consulting, IT-Sicherheit, Weiterbildung für ihre Kunden, in erster Linie Steuerberater, Wirtschaftsprüfer und Rechtsanwälte, und deren Mandanten. Ein nicht unbeträchtlicher Teil dieser betriebswirtschaftlichen Anwendungen läuft bis heute ganz oder als Backend von Client-Anwendungen auf einem IBM Großrechner im DATEV Rechenzentrum.

[10] [http 19c]
[11] [http 19a]
[12] [http 20n]
[13] [http 20e]
[14] [http 20a]
[15] [http 20o]
[16] Jutta Roessner, Executive Committee Member, DATEV

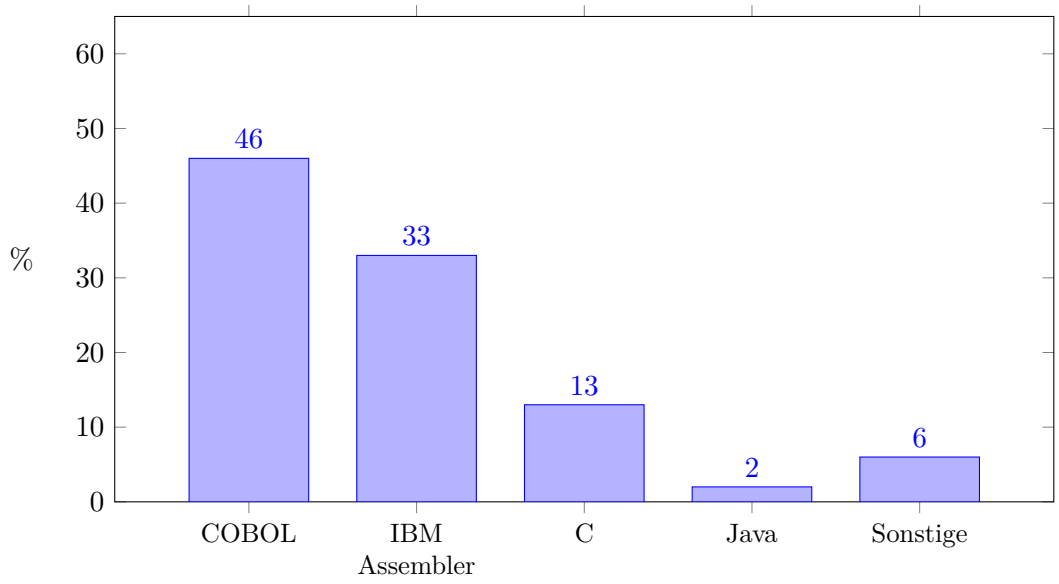


Abbildung 1.1.: Anteil der verwendeten Programmiersprachen auf dem Mainframe bei DATEV e.G. in Prozent

So werden pro Tag circa 150.000 Batch Jobs 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 e.G. Durch diese Module werden unter anderem im Monat circa 11 Millionen Lohnabrechnungen erstellt und circa eine Millionen Umsatzsteuer-Voranmeldungen durchgeführt. 2018 wurde mit den DATEV Produkten erstmals die Umsatz-Milliarde erreicht.[17]

1.1. Problemstellung

Im Jahre 2020 ist der größte Konkurrent für den Mainframe die Cloud. Laut einer Vorhersage aus dem Jahr 2018[18] soll im Jahre 2020 circa 79 Prozent des weltweiten Workloads in einer Cloud verarbeitet werden. Für die Entwicklung von neuen Online-Anwendungen im cloud-native Stil wurde bei der DATEV e.G. eine Platform-as-a-Service (PaaS)-Lösung geschaffen und neue DevOps[19] Prozesse aufgebaut. Dies ist im Cloud-Zeitalter nötig, um mit einer verbesserten Entwicklungseffizienz und neuen Architekturen Anwendungen („Apps“) schneller auf den Markt bringen und auf Kundenanforderungen schneller reagieren zu können. Stichwort: Continuous Integration, Continuous Delivery (CI/CD)[20]. Ein Baustein der effizienteren Prozesse durch PaaS sind die sog. „Self Services“. D.h., Entwicklerteams

[17] [http 20j]
[18] Statistik im Anhang A.1
[19] Siehe Absatz 4.1
[20] Siehe Absatz 2.1.2

können sich über sog. „Cloud Services“, z.B. Datenbanken wie PostgreSQL, Mongo und Messaginglösungen wie Kafka entweder manuell über einen Marktplatz (siehe Abbildung 1.2) oder automatisiert per „Build-Pipeline“ eine Laufzeitumgebung für ihre Anwendung zusammenbauen. Eine genaue Beschreibung der Begrifflichkeiten erfolgt im Absatz 2.1. Den

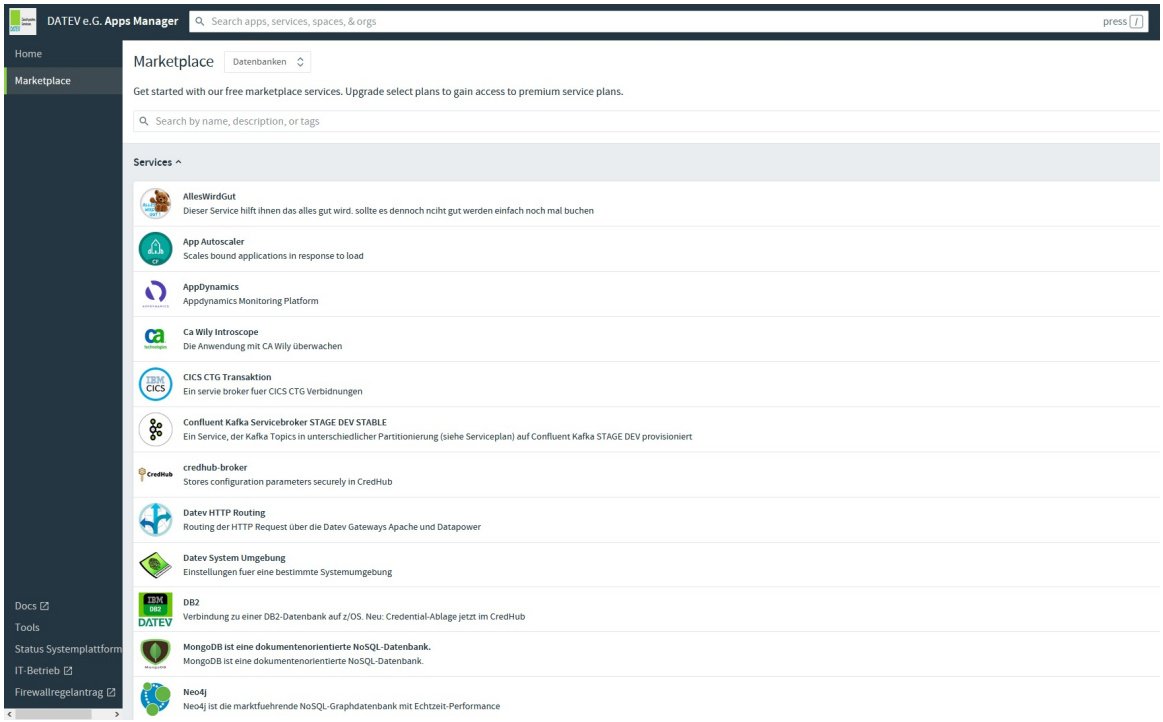
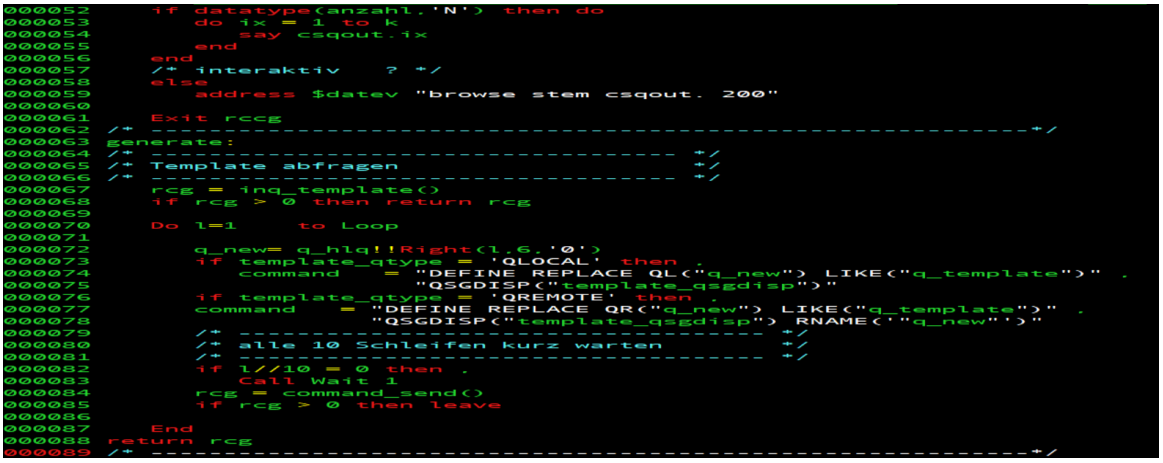


Abbildung 1.2.: Cloud Foundry Marketplace bei DATEV e.G.

Entwicklern steht, neben modernen Entwicklungsumgebungen (IDE[21]) und einer Sourceverwaltung mit Git, auch eine sog. „Toolchain“ zur Verfügung. Diese beinhaltet Tools für Build, Test, Quality Gates und Deployment. Damit wird der Entwicklungsprozess automatisiert und man erhofft sich eine hohe Entwicklereffizienz.

Der Entwicklungsprozess für z/OS Anwendungen bei DATEV e.G. erscheint im Vergleich zu dieser PaaS-Lösung veraltet. So wurde 2010 eine auf Eclipse basierende Entwicklungsumgebung für COBOL und IBM Assembler in der DATEV e.G. flächendeckend bereitgestellt. Zuvor - und teilweise heute noch - wurde mit Hilfe der in Abbildung 1.3 gezeigten Oberfläche, dem sog. ISPF gearbeitet. Diese stellte z.B. nur ein Syntaxhighlighting zur Verfügung. Ein Meilenstein für die modernisierte z/OS Entwicklung bei DATEV war die Einführung von Git im Jahr 2018 auch für z/OS Sourcen. Dieses weit verbreitete Standard-Tool ist im z/OS Umfeld tatsächlich eine entscheidende Neuerung. Dadurch wurde ein bis dato verwendetes eigenentwickeltes Tool für die Sourceverwaltung der z/OS Sourcen abgelöst. Dieses stellte nur ein sehr einfaches Versionierungskonzept ohne aus Git bekannten Features wie Merge, Branch usw. bereit. Parallelentwicklung von verschiedenen Features war

[21] Integrated Development Environment wie IntelliJ oder Eclipse



```
000052      if datatype(anzahl,'N') then do
000053        do ix = 1 to k
000054          say csqout.ix
000055        end
000056      end
000057      /* interaktiv    ? */
000058      else
000059        address $datev "browse stem csqout. 200"
000060
000061      Exit rccg
000062  /* ---------------------------------------------------------------*/
000063  generate:
000064  /* ------------------------------------ */
000065  /* Template abfragen                    */
000066  /* ------------------------------------ */
000067      rcg = inq_template()
000068      if rcg > 0 then return rcg
000069
000070      Do l=1     to Loop
000071
000072        q_new= q_hlq!!Right(l,6,'0')
000073        if template_qtype = 'QLOCAL' then ,
000074          command     = "DEFINE REPLACE QL("q_new") LIKE("q_template")" ,
000075                        "QSGDISP("template_qsgdisp")"
000076        if template_qtype = 'QREMOTE' then ,
000077        command     = "DEFINE REPLACE QR("q_new") LIKE("q_template")" ,
000078                      "QSGDISP("template_qsgdisp") RNAME('"q_new"')"
000079        /* ------------------------------------ */
000080        /* alle 10 Schleifen kurz warten        */
000081        /* ------------------------------------ */
000082        if l//10 = 0 then ,
000083          Call Wait 1
000084        rcg = command_send()
000085        if rcg > 0 then leave
000086
000087      End
000088  return rcg
000089  /* ---------------------------------------------------------------*/
```

Abbildung 1.3.: Auszug aus einem REXX Skript in der ISPF Oberfläche

vorher mit viel Aufwand und Abstimmung möglich. Das Tooling wurde somit modernisiert, jedoch nicht der Entwicklungsprozess selbst. Es teilen sich sehr viele Anwendungen die gleichen Entwicklungs-CICS/Db2/MQ Ressourcen. Das heißt auch, dass eine Parallelentwicklung - trotz jetzt möglicher Git-Branches - an unterschiedlichen Features nur mit viel Abstimmungsaufwand und Absprachen innerhalb eines Entwicklungsteams, teilweise auch abteilungsübergreifend, möglich ist. Werden Änderungen an bestehenden Ressourcen durchgeführt oder werden neue Systemumgebungen benötigt, entsteht weiterer Abstimmungsaufwand und weitere Absprachen. Dadurch wird der aktuelle Prozess im Vergleich zu modernen cloud nativen Prozessen, als fehleranfällig und langsam angesehen.

Es bleibt die Frage, wie wird vor diesem Hintergrund mit den vielen Mainframebestandsanwendungen bei der DATEV e.G. in Zukunft umgegangen? Die komplette Ablösung dieser Anwendungen durch cloud-native Lösungen ist eine Option, deren zeitlicher Rahmen und Machbarkeit aktuell nicht absehbar ist. Für die Funktionsfähigkeit dieses Bestandsgeschäfts, das die Core-Business-Funktionalitäten der DATEV e.G. darstellt, muss also eine effiziente Weiterentwicklung und Wartung gewährleistet werden. Auch im Falle einer geplanten Ablöse von Anwendungen muss je nach Strategie (z.B. „Rebuild“/ „Rearchitect“)[22] das Alt-System parallel dazu über Jahre oder Jahrzehnte gepflegt und funktional aktuell gehalten werden. Daraus folgt, dass aus Sicht der DATEV e.G. weiter in die IBM Mainframe Plattform investiert werden muss. Dies bedeutet Investitionen in die bereitgestellte Infrastruktur (Hardware, Betriebssysteme, Lizenzen), insbesondere aber auch Investitionen, die die oben genannten Anforderungen an Weiterentwicklung, Wartung und Entwicklungseffizienz sowie Effizienz im Betrieb adressieren.

[22] [http 20m]

1.2. Ziel der Arbeit

Es liegt also nahe, sich an den oben beschriebenen Prozessen für cloud native Entwicklung zu orientieren. In diesem Zusammenhang läuft aktuell bei DATEV e.G. ein Proof of Concept bezüglich automatisierter Builds von z/OS Anwendungen auf Basis von Jenkins basierten Pipelines. Dies ist auch die Voraussetzung für automatisierte Tests von z/OS Programmen im Rahmen des „Continuous Integration, Continuous Deployment“ Ansatzes. Was jedoch fehlt, sind „Self Services“ für Laufzeitumgebung und Middleware. Die dafür notwendige automatisierte Provisionierung einer z/OS Anwendungsumgebung, d.h. Laufzeit, Middleware etc., ist aktuell noch weitgehend unerforscht. Hier sind die Prozesse bei DATEV und anderen Kunden oft noch proprietär, hoch spezialisiert, manuell und nicht modernisiert. Gerade bei Mitarbeitern im Betrieb, die als Administratoren für die Middleware-Produkte arbeiten, sind die Bedenken groß, ob man diese Cloud-Vorgehensweise auf hochspezialisierte individuelle Komponenten wie CICS, DB2, IBM MQ anwenden kann, „weil man so etwas bisher nicht vermisst hat“[23]. IBM bietet als eine Lösung „IBM Cloud Provisioning and Management for z/OS“ an. Dies hat sich noch nicht flächendeckend durchgesetzt, aber „Viele Kunden haben [...] im Moment Interesse, jedoch warten viele hier auf die ersten Erfahrungen von anderen“[24]. Hier setzt diese Arbeit an und klärt folgende Fragen:

- Ist es möglich, den Bereitstellungsprozess für z/OS Anwendung bei DATEV e.G. mit Hilfe des „IBM Cloud Provisioning and Management for z/OS“-Tools an cloud native Prozesse anzunähern?
- Erzeugt die Nutzung von „IBM Cloud Provisioning and Management for z/OS“ 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 bestehende Anwendung untersucht. Diese Anwendung sollte CICS als Anwendungsserver, eine Db2 Datenbank und IBM MQ als Messaginglösung nutzen, um für diese drei Haupt-Technologien (Middleware-Komponenten) eine Aussage treffen zu können. Die genaue Vorgehensweise wird im Kapitel 3 beschrieben.

[23] Marcel Amrein, IBM Senior Technical Sales Professional (MQ and CICS) (Quelle: Anhang A.6)
[24] Tobias Leicher, IBM Senior IT Specialist for CICS and zAPI (Quelle: Anhang A.6)

Kapitel 2.

Grundlagen

Um den Unterschied zwischen modernen cloud native Entwicklungsprozessen und dem Mainframe Entwicklungsprozess darstellen zu können, werden zunächst Begriffe aus dem cloud native-Umfeld erläutert. Anschließend werden für die Beantwortung der Forschungsfragen relevante Begriffe des Mainframe-Umfelds eingeführt.

2.1. Cloud native bei DATEV e.G.

Eine cloud native Anwendung ist eine speziell für das Cloud-Computing konzipierte und entwickelte Anwendung. Oft werden damit Online-Anwendungen und mobile „Apps“entwickelt, für die häufig und hoch frequent neue Features bereitgestellt werden sollen. Für solche Anwendungen ist das Architekturpattern der sogenannten „Microservices“, weit verbreitet. Diese einzelnen, entkoppelten Services sind beispielsweise in Containern paketiert. Bekannt geworden ist hier der Ansatz von Docker. Die Docker Container werden über Images beschrieben und beinhalten neben der Anwendung alles das, was die Anwendung an Middleware-Komponenten und Services zur Laufzeit benötigt, wie Bibliotheken, Dateien.[1] Somit können Anwendungen auf verschiedenen „privat“ und „public“ Cloud-Umgebungen, auch von unterschiedlichen Anbietern, ausgeführt werden. Große Anbieter sind hier beispielsweise AWS (Amazon), Google und Microsoft Azure.[2] Dort können bereitstehende Services für Datenhaltung, Security, Messaging usw. genutzt werden. [http 20i]
Eine moderne cloud native Anwendung innerhalb der DATEV e.G. macht folgende Dinge aus:

- „Cloud Foundry“
- CI/CD-Pipeline

[1] [Vohr 16, Kap. 1]
[2] [http 20c]

2.1.1. „Cloud Foundry“

Bei Cloud Foundry handelt es sich um eine quelloffene Platform-as-a-Service, kurz PaaS. Platform-as-a-Service, beschreibt neben Infrastructure-as-a-Service, kurz IaaS und Software-as-a-Service, kurz SaaS, einen Grad an Auslagerung von IT-Systemen in die Cloud. Im Vergleich zu SaaS, bei der ganze Anwendungen in einer Cloud zur Verfügung stehen, und IaaS, bei der die automatisierte Bereitstellung von Infrastrukturkomponenten wie Netzwerk, Speicher usw. im Fokus steht, stellt eine PaaS-Lösung eine Plattform, die sich neben der Infrastruktur auch um das Betriebssystem, die Middleware und die Laufzeitumgebung kümmert, bereit.[3] Für die Verwaltung von Ressourcen bietet Cloud Foundry eine Weboberfläche, den sogenannten „Marketplace“, an. In diesem können mit wenigen Mausklicks Schnittstellen zu Services wie Datenbankmanagementsystemen, Messaging- oder Monitorlösungen zur Anwendung hinzugefügt werden. Diese Schnittstellen, auch „Self Service“ oder „Service Broker“ genannt, können mittels einer von Cloud Foundry zur Verfügung gestellten API selbst entwickelt werden. Daneben kümmert sich Cloud Foundry um das Staging der Anwendungen. D.h., eine Anwendung kann mit den benötigten Komponenten sicher von einer Entwicklungs- in eine QS- bzw. Produktiv-Stage verschoben werden. Die notwendigen stagespezifischen Anpassungen werden konfigurativ beigesteuert. Um eine Anwendung in einer bestimmten Stage bereitzustellen, bietet Cloud Foundry ein Kommandozeileninterface an. Neben der Bereitstellung können mit diesem Interface beispielsweise Anwendungen auch horizontal skaliert werden. Über die Programmiersprache groovy können diese Kommandozeilenbefehle in eine Jenkins basierte CI/CD-Pipeline aufgenommen werden. So kann die Bereitstellung über mehrere Stages hinweg automatisiert werden. Diese Methode wird bei der DATEV e.G. eingesetzt. [http 20h]

2.1.2. „CI/CD-Pipeline“

CI/CD steht für „Continuous Integration und Continuous Delivery“ in manchen Fällen auch für „Continuous Integration, Continuous Delivery und Continuous Deployment“. Die einzelnen Begriffe werden im Folgenden erläutert, als Überblick dient Abbildung 2.1

„Continuous Integration“

Continous Integration beschreibt einen Prozess, bei dem Änderungen von Entwicklern regelmäßig in eine gemeinsame Codebasis integriert und getestet werden. In Abbildung 2.2 ist der Prozess dargestellt. Eine Voraussetzung für den Einsatz von CI ist eine zentrale Sourceverwaltung. Bei der DATEV e.G. handelt es sich dabei um Git. Nachdem ein Entwickler Änderungen am Code vorgenommen hat, lädt er diese in die Sourceverwaltung hoch. Dabei

[3] [http 20l]

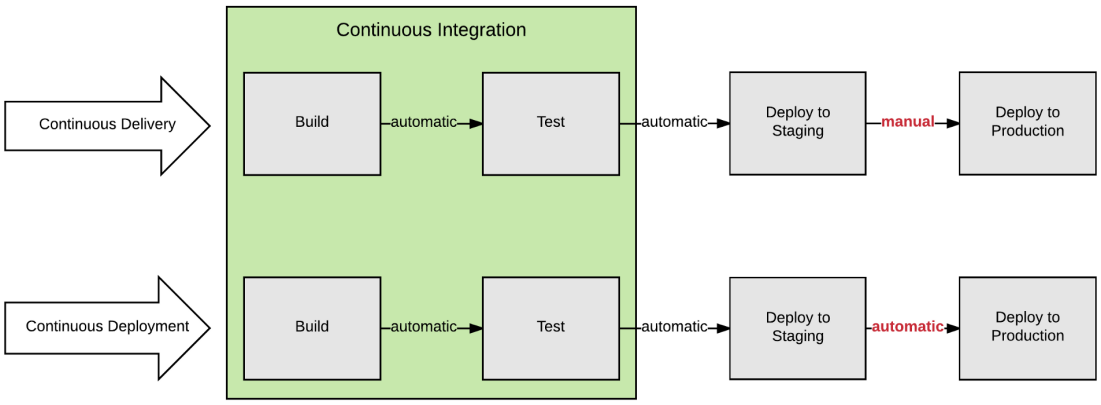


Abbildung 2.1.: Abgrenzung von Continuous Integration, Continuous Delivery und Continuous Deployment (Quelle: [http 20b])

kann eine erste Überprüfung des Codes mittels statischer Codeanalyse durchgeführt werden. Dazu zählt unter anderem die Prüfung vorher definierter Coderichtlinien. Sind diese erfolgreich, wird ein isolierter CI-Server benachrichtigt. Dieser Server holt sich den neuesten Stand des Quellcodes und baut diesen, um anschließend automatisierte Tests durchzuführen. Hier liegt der zentrale Vorteil der Automatisierung. Bei jedem Bauen wird automatisch überprüft, ob das Ergebnis das erwartete ist. Voraussetzung dafür ist die oben beschriebene Paketierung der Anwendung mit der Laufzeitumgebung und den benötigten Komponenten.

Bei den Tests während der Continuous Integration handelt es sich um sogenannte „Unittests“. Dabei wird anhand vordefinierter Eingangsdaten die Ausgabe bestimmter Funktionen geprüft (Soll-Ist-Vergleich). Dazu gehört auch die Überprüfung, ob mit Fehlerbedingungen korrekt umgegangen wird. Dabei werden Abhängigkeiten zu externen Ressourcen, wie beispielsweise einer Datenbank, außen vor gelassen. Wenn solche Ressourcen dennoch benötigt werden, müssen diese mit Hilfe eines sogenannten „Mocking-Framework“ simuliert werden. Schließlich stellt der CI-Server die Ergebnisse dieser Tests (z.B. in einem Jenkins-Dashboard) zur Verfügung. [Last 17, Kap. 2]

„Continuous Delivery“

Bei Continuous Delivery werden die Änderungen, die vorher auf dem isolierten CI-Server getestet und integriert wurden, in eine Stage, z. B. in die Entwicklungsstage, automatisch übertragen. Hier werden weitere Tests, wie Integrationstests und Akzeptanztests, durchgeführt. Am Ende einer Continuous Delivery Pipeline steht ein theoretisch auslieferbarer Stand der Anwendung. Die Übergabe dieses Standes in die Produktion ist bei Continuous Delivery noch manuell umzusetzen. [Last 17, Kap. 3]

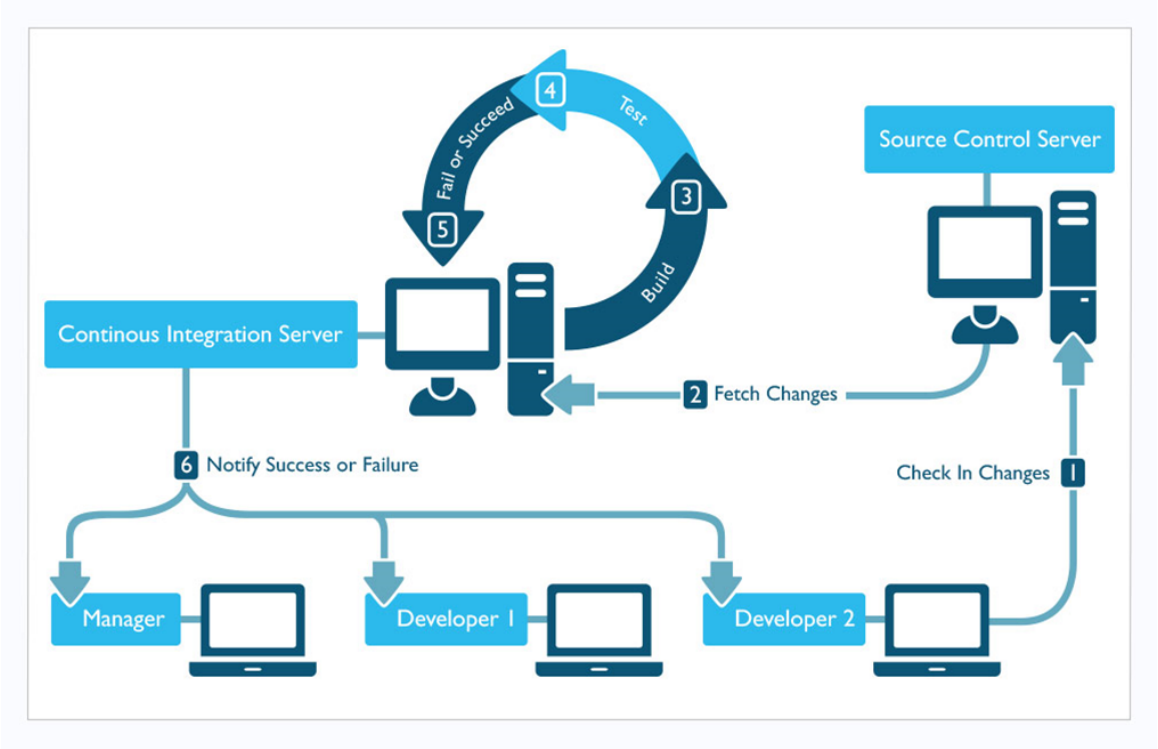


Abbildung 2.2.: Continuous Integration Prozessaufbau (Quelle: [http 20d])

„Continuous Deployment“
Continuous Deployment beschreibt den nächsten Schritt nach Continuous Delivery. Dabei handelt es sich um die automatisierte Bereitstellung in eine Produktions-Stage. [Last 17, Kap. 4]

2.2. Mainframe / Großrechner

Im modernen Sprachgebrauch kann ein Großrechner, oder auch Mainframe, 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 benötigen, 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. Die einzelnen Funktionalitäten werden von sogenannten „Subsystemen“, auch „Middleware“ genannt, zur Verfügung gestellt. Darunter zählen unter anderem Datenbankmanagementsysteme und Anwendungsserver. [Ebbe 11, S. 9-10]

2.3. Mainframe Anwendungen bei DATEV e.G.

Das Betriebssystem des IBM Mainframes ist das für zero downtime stehende z/OS.[4] Darauf aufbauend benötigen klassische z/OS Anwendungen bestimmte Middleware. Bei der DATEV e.G. handelt es sich unter anderem um folgende Middlewarekomponenten:

- Laufzeitumgebung: CICS oder Batch
- Datenhaltung: VSAM oder Db2
- Message Queuing: IBM MQ

Diese Subsysteme stehen in jeder Stage zur Verfügung. Eine Stage beschreibt eine isolierte Systemumgebung mit eigenen Subsystemen und Ressourcenverwaltung. Die DATEV e.G. unterscheidet am Mainframe vier Stages:

- Testplex:
  Labor für Änderungen am System, beispielsweise einer neuen Betriebssystemsversion
- Entwicklung:
  Implementierung neuer Features und Durchführung kleiner Tests
- Qualitätssicherung:
  Durchführung von Integrationstests
- Produktion:
  Software, die für den Kunden bereitsteht

2.4. Subsysteme / Middleware

Für die Beantwortung der Forschungsfragen liegt der Fokus auf dem Erstellen („Provisionieren“) einer anwendungsspezifischen Laufzeitumgebung mit einer Datenhaltung und Message Queuing auf der Entwicklungs-Stage. Als Laufzeitumgebung wird „CICS“, als Datenhaltung „Db2“ und für das Message Queuing „IBM MQ“ verwendet. Wie in Abbildung 2.3 dargestellt, sind mehrere Instanzen pro Subsystem möglich. Bei der DATEV e.G. wird die Anzahl an Instanzen pro Subsystem möglichst gering gehalten, um den Verwaltungsaufwand und den Aufwand bei Änderungen, beispielsweise bei einem Versionswechsel[5], gering zu halten. Daraus folgt, dass sich, wie in der Problemstellung, Absatz 1.1, bereits erwähnt, viele Anwendungen die gleichen Entwicklungs-CICS/Db2/IBM MQ Ressourcen teilen. Diese Instanzen sind langlebig und müssen dahingehend gepflegt und gewartet werden, dass sie die Anforderungen für alle Anwendungen, die sich die Ressourcen teilen, abdecken.

[4] [Ebbe 11, S. 92]
[5] circa alle eineinhalb Jahre erscheint eine neue CICS Version

Diese einzelnen Subsysteme werden im Folgenden erläutert, hierzu dient Abbildung 2.3 als Überblick.

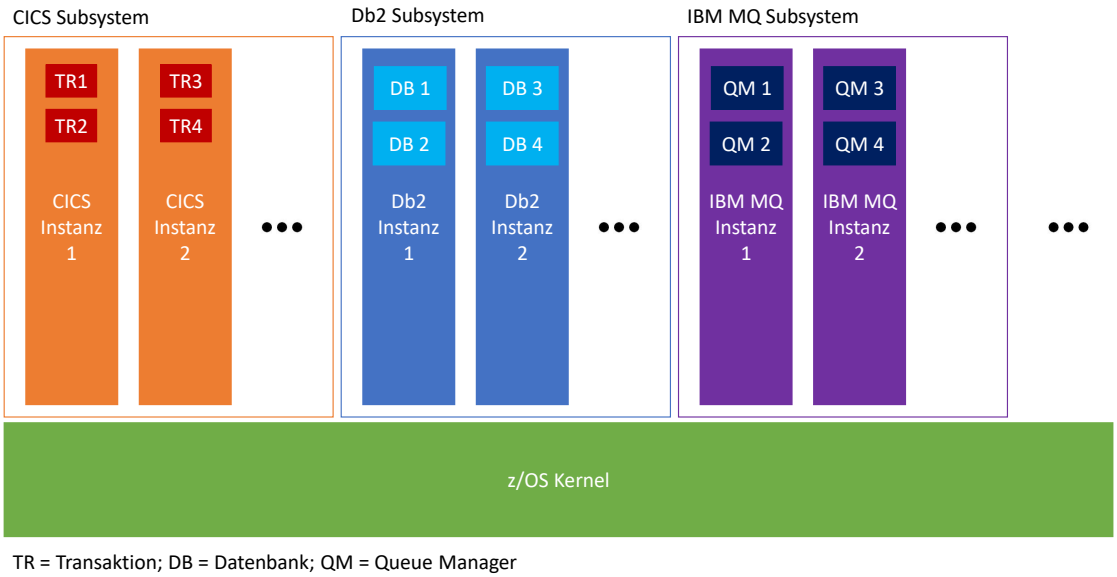


Abbildung 2.3.: Architekturübersicht über die Subsysteme einer Stage bei DATEV e.G.

2.4.1. 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 gegenüber anderen Anwendungsservern, wie zum Beispiel „Apache Tomcat“ oder „IBM WebSphere“, den Vorteil, dass es verschiedene Programmiersprachen unterstützt. Damit ist CICS ein Multi-Language Application Server und kann z.B. von COBOL, Assembler, Java und PLI Programmen genutzt werden. So können Programme innerhalb einer Anwendung in der für ihren Use-Case am besten geeigneten Sprache implementiert werden, für die Kommunikation zwischen den verschiedenen Sprachen stellt CICS mit seinem erwähnten API die Funktionalität zur Verfügung. [Rayn 11, S. 4]

Das CICS Subsystem einer Stage umfasst mehrere CICS Instanzen.

2.4.1.1. 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 verwaltet werden. Eine CICS Instanz verwaltet mehrere CICS Transaktionen.

Wenn in dieser Arbeit im folgenden von dem CICS gesprochen wird, ist damit die CICS-Instanz gemeint.

2.4.1.2. 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 Falle eines Fehlers die Möglichkeit eines Rollbacks, beispielsweise der in der Transaktion genutzten Datenbank, sichergestellt. [Rayn 11, 5-8]

2.4.1.3. CICS Anwendungsdeployment

Zur Beantwortung der Forschungsfragen liegt der Fokus auf der Entwicklungsstage, deshalb wird im Folgenden nur das Deployment von Anwendung in eine Entwicklungs-CICS Instanz dargestellt. Damit eine Anwendung in einer CICS Instanz zur Verfügung steht, muss der kompilierte Sourcecode der Anwendung manuell in einer der CICS Instanz bekannten Bibliothek abgelegt werden. Eine solche Bibliothek muss im „Started Task Control-Job“[6] einer CICS Instanz referenziert werden.

2.4.2. Db2

Db2 ist ein relationales Datenbankmanagementsystem, welches unter anderem als Subsystem eines z/OS Betriebssystems läuft. Einer Stage können mehrere Datenbanksysteme, auch Instanzen genannt, zugeordnet werden. In einer Instanz befinden sich die Datenbanken und Tabellen. In der Entwicklungsstage sind das bei DATEV e.G. unter anderem „DB0C“ und

[6] Siehe Absatz 4.2.1

„DB0T“. DB0C wird als Sandbox betrieben, d.h. jeder Entwickler kann hier eigene Datenbanken erstellen und verwalten. Bei DB0T handelt es sich um ein Datenbankmanagementsystem, dass sich alle Anwendungen der Entwicklungsstage teilen. Die darin enthalten Datenbanken werden von der Db2 Administration verwaltet. Diese Datenbanken entsprechen in der Struktur den Datenbanken aus der Produktions-Stage und werden für Tests bezüglich Strukturänderungen usw. herangezogen.

2.4.3. 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. [Aran 13, Kap. 3.2]

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

2.4.3.1. 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. [Aran 13, S. 36]

2.4.3.2. 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, unter bestimmten Bedingungen eine Trigger-Nachricht auf diese Queue zu schreiben. Eine andere Local Queue kann so definiert sein, dass sobald eine Nachricht auf sie geschrieben wird, eine solche Trigger-Nachricht erzeugt wird. Dies ermöglicht beispielsweise den use-case, dass Anwendungen nur starten, wenn wirklich Daten zum Verarbeiten vorhanden sind. [Aran 13, S. 37-38]

2.4.3.3. 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. [Aran 13, Kap. 4.5]

2.5. „IBM Cloud Provisioning and Management for z/OS“

Die Verwaltung von Subsystemen von z/OS ist eine hochspezialisierte, teilweise manuelle Tätigkeit.[7] Um diese Aufgaben zu bewältigen, sind aktuell in erster Linie proprietäre Tools (siehe Abbildung 2.4) im Einsatz, die schon aus Sicht des User-Interfaces nicht der Erwartungshaltung an moderne Administrationstools entsprechen.

Für die Automation dieser Prozesse bietet die IBM das „IBM Cloud Provisioning and Management for z/OS“ an. Dabei handelt es ich um ein bei der Installation von „z/OS Management Facility“[8], kurz z/OSMF, mitgeliefertes Tool. z/OSMF ist wiederum Teil der Standardauslieferung des Betriebssystems z/OS. Es soll z/OS-Administration über moderne Oberflächen und APIs ermöglichen. z/OSMF (und damit auch „IBM Cloud Provisioning and Management for z/OS“) ist bei DATEV verfügbar, wird aber noch nicht flächendeckend eingesetzt. Die Administratoren kennen und vertrauen auf ihre herkömmlichen Tools und sehen in der Umstellung immer auch Lernaufwand und Risiko. „IBM Cloud Provisioning and Management for z/OS“ ist auch außerhalb der DATEV e.G. kaum verbreitet. Deshalb besteht auch aus Sicht der IBM Interesse an Erfahrungsberichten zur Nutzung von „IBM Cloud Provisioning and Management for z/OS“. So wurde im Rahmen der neunzehnten „AMC“[9]

[7] Analyse des aktuellen Bereitstellungsprozesses, siehe in Absatz 4.2
[8] Siehe Absatz 2.5.1
[9] Academic Mainframe Consortium e.V.

Abbildung 2.4.: Etablierte Konfigurationsoberfläche am Beispiel bei Änderung einer Transaktion

Tagung im IBM Client Center Böblingen am 16.01.2020 bei einem circa 30-minütigen Vortrag ein Arbeitsstand dieser Arbeit vorgestellt. Die Agenda der Veranstaltung ist im Anhang A.2 zu finden. Der AMC e.V. ist ein Förderverein für die akademische Ausbildung auf dem Mainframe.[10] Neben Vertretern der IBM nahmen an der Tagung Vertreter von Hochschulen und verschiedenen Unternehmen teil. Der Vortrag „kam ja sehr gut an und hat auch später noch zu Gesprächen geführt“[11]. In den anschließenden Gesprächen wurde deutlich, dass sich neben der DATEV e.G. und der IBM auch andere Kundenfirmen für die automatisierte Provisionierung von z/OS Middleware mit „IBM Cloud Provisioning and Management for z/OS“ interessieren, aber noch kaum Erfahrung in dem Umfeld besteht. Dies wurde durch eine Nachfrage bei IBM Spezialisten[12] nochmals bestätigt.

Bei „IBM Cloud Provisioning and Management for z/OS“ stehen die sogenannten „Templates“ im Mittelpunkt. Mit Hilfe eines Templates können Instanzen erzeugt werden. Diese Instanz kann eine oder mehrere verschiedene Subsystem-Instanzen enthalten.
Das Template selbst ist eine Art „Verwaltungseinheit“ und besteht aus drei Dateien:

„Manifest-File“
Im Manifest-File ist der Speicherpfad für das sogenannte „Actiondefinitionfile“ und das

[10] [http 20f]
[11] AMC Vorstand Ernst Lugert, 20.1.2020
[12] Siehe Anhang A.6

sogenannte „Variableinputfile“ angegeben. Erläuterungen zu diesen beiden Dateien folgen im Anschluss. Ein Template benötigt zur Provisionierung einen dazugehörigen Provisionierungsworkflow, der Speicherpfad von diesem Workflow wird in der Manifest-File angegeben. [http 20q]

Ein Workflow ist über eine XML Datei, das sogenannte „Workflowdefinitionfile“, definiert. Dieses lässt sich grob in zwei Teile untergliedern:

- Variablendefinition
- Steps

In der Variablendefinition werden, wie der Name schon sagt, alle Variablen, die für diesen Workflow notwendig sind, definiert.

Ein Step beschreibt einen Teilablauf eines Workflows. Ein Workflow kann aus mehreren Steps bestehen. Die Steps werden in Definitionsreihenfolge ausgeführt. Allerdings können Bedingungen für die Durchführung eines Steps definiert werden. So ist es beispielsweise möglich, einen Step nur durchzuführen, wenn eine bestimmte Variable einen bestimmten Wert besitzt. Innerhalb eines Steps können sowohl interne und externe Scripte als auch JCLs[13] und somit z/OS Programme ausgeführt werden. Darüber hinaus besteht die Möglichkeit REST-Calls auszuführen. Durch ein XML Schema wird sichergestellt, dass das Workflowdefinitionfile keine syntaktischen Fehler beinhaltet. Sowohl die Variablendefinition als auch die Steps können in externe XML Dateien ausgelagert werden. Dadurch können Variablen an einer Stelle im Template definiert und in alle Workflowdefinitionfiles aufgenommen werden. [Rott 18, S. 140]

Zusätzlich kann in dem Manifest-File eine Beschreibung des Templates hinzugefügt werden. [http 20q]

„Actiondefinitionfile“

Weitere optionale Aktionen, die ein Anwender mit einer Instanz eines Templates durchführen kann, werden in dem Actiondefinitionfile festgelegt. Standardmäßig handelt es sich dabei um folgende Aktionen. Die Begriffe werden am Beispiel einer Template-Instanz, die eine Datenbank provisioniert, erläutert.

- check_status
  Prüft den Status der Template-Instanz, beispielsweise ob die Datenbank erreichbar ist.

[13] Siehe Glossar Batch Job

- start
  Welche Schritte sollen beim Starten der Template-Instanz durchgeführt werden. Beispielsweise die Erzeugung von Tabellen.

- stop
  Welche Schritte sollen beim Stoppen der Template-Instanz durchgeführt werden. Beispielsweise das Löschen von Tabellen.

- deprovisioning
  Welche Schritte sollen beim Entfernen der Template-Instanz durchgeführt werden. Üblicherweise wird die Template-Instanz zunächst gestoppt, am Beispiel der Datenbank ist dies nicht notwendig, da beim Löschen der Datenbank automatisch die Tabellen auch gelöscht werden.

Neben den Workflowdefinitionfiles muss in einer Aktion auch der Pfad für das sogenannte „Variableinputfile“ angegeben sein. [http 20q]

„Variableinputfile“
In dieser Datei werden den in dem Workflowdefinitionfile definierten Variablen Werte zugewiesen. Somit kann das Template für spezifische Anforderungen, z.B. einer speziellen Anwendung, konfiguriert werden.

Für die Provisionierung eines Templates müssen diesem eine sogenannte „Domain“ und ein „Tenant“ zugewiesen werden. 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. [Keit 16, S. 15]

„IBM Cloud Provisioning and Management for z/OS“ umfasst zwei Lösungsvarianten, „z/OS Management Facility“[14] und „z/OS Provisioning Toolkit“[15].

2.5.1. z/OS Management Facility

Der Funktionsumfang von z/OS Management Facility, kurz z/OSMF, umfasst Systemmanagementfunktionen in einer browserbasierenden Benutzeroberfläche, dargestellt in Abbildung 2.5. Zu diesen Funktionen zählt auch die Verwaltung von Workflows und Templates.

Die linke Seite der Abbildung 2.5 zeigt den Umfang der z/OSMF Funktionen. Für diese Arbeit besitzt nur der Menüpunkt „Cloud Provisioning“ Relevanz. Unter diesem Punkt sind die Funktionalitäten von „IBM Cloud Provisioning and Management for z/OS“ zu

[14] siehe Absatz 2.5.1
[15] siehe Absatz 2.5.2

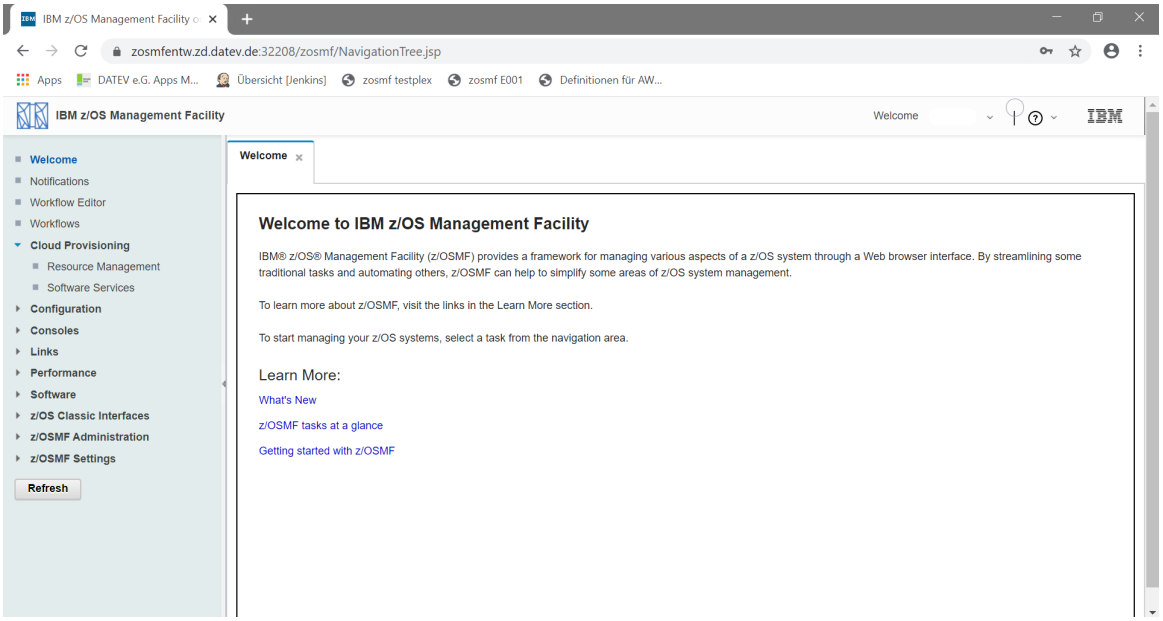


Abbildung 2.5.: z/OSMF Willkommens-Ansicht

finden, und der Begriff „z/OSMF“ wird im folgenden synonym für diese Lösung verwendet. [Rott 18, S. 32]

Unter dem Punkt „Resource Management“ werden die Domains und Tenants verwaltet. Zur Verwaltung der Templates und Template-Instanzen kommen die „Software Services“ zum Einsatz. Dort können neue Templates über die Manifest-Files hinzugefügt werden. Es folgt, wie oben beschrieben, die Zuweisung einer „Domain“ und eines „Tenants“. Anschließend kann das Template, falls es keine Fehler beinhaltet, veröffentlicht werden. Es ist zu empfehlen vorher einen „Test Run“ durchzuführen. Dabei wird eine Instanz testweise provisioniert. Diese Test-Template-Instanz verhält sich genauso wie eine Instanz, die aus einem veröffentlichten Template erzeugt wurde. Somit können das Template und die in dem Actiondefintionfile definierten Aktionen vor der Veröffentlichung getestet werden. [Rott 18, S. 8]

2.5.2. z/OS Provisioning Toolkit

Das z/OS Provisioning Toolkit, kurz z/OSPT bietet für das Provisionieren von Laufzeitumgebungen ein Kommandozeileninterface für die Bereitstellung und das Verwalten von Templates bzw. „Images“, sowie das Starten von Instanzen an. Genau wie bei Cloud Foundry (Siehe Absatz 2.1.1) ermöglicht dieses die Einbindung in eine Jenkins basierende CI/CD-Pipeline. z/OSPT orientiert sich an der Docker Commandline und spricht von Containern (auch wenn es in z/OS diese nicht gibt) und Images. Zur Erläuterung werden die Begriffe hier gegenüber gestellt.

„Docker-Images“
Ein Docker-Image beschreibt die Vorlage für einen Docker-Container und beinhaltet alle Elemente, die für die Ausführung einer Anwendung als Container benötigt werden, so wie den Code, Konfigurationsdateien, Umgebungsvariablen, Bibliotheken und die Laufzeitumgebung.

„Docker-Container“
Mit dem Kommandozeilenbefehl „docker run“ wird aus einem Docker-Image ein Docker-Container erzeugt. Ein Docker-Container beschreibt somit eine lauffähige Instanz eines Docker-Images. [Vohr 16, Kap. 1]

Im Vergleich dazu sind die Definitionen in z/OSPT folgende:

„z/OSPT-Images“
Grundsätzlich ist ein z/OSPT-Image einem Docker-Image nicht unähnlich. Auch ein z/OSPT Image ist für die Erzeugung eines z/OSPT Containers zuständig und beinhaltet alle Elemente, die die Anwendung zur Laufzeit benötigt. Es verknüpft ein Template mit den enthaltenen Dateien (Actiondefinitionfile, Variableinputfile und die Manifest-File) mit einer weiteren nicht im Template enthaltenen Konfigurationsdatei, dem sogenannten „zosptfile“. In diesem muss der Name des zugrundeliegenden Templates angegeben werden. Danach ist es möglich, die Werte aus dem Variableinputfile zu überschreiben und so das Verhalten der Template-Instanz zu verändern. Dadurch kann ein Template mit spezifischen Änderungen provisioniert werden, ohne dass ein neues Template erzeugt werden muss.

„z/OSPT-Container“
Die Beziehung zwischen einem z/OSPT-Container und einem z/OSPT-Image ist die gleiche wie zwischen einem Docker-Container und einem Docker-Image. Ein z/OSPT-Container entspricht einer Template-Instanz, die mit Hilfe eines z/OSPT-Images gestartet wurde.

Um nun einen z/OSPT-Container bereitzustellen, muss ein Template zur Verfügung stehen. Anschließend kann mittels des Konsolenbefehls „zospt build“ und der Angabe des Pfades des zosptfiles ein Image erzeugt werden. Wird nun der „zospt run“-Befehl ausgeführt, wird ein z/OSPT-Container erzeugt (entspricht dem Provisionieren einer Template-Instanz) und gestartet (wenn die start-Aktion in der Actiondefinitionfile definiert wurde). Der Status von vorhandenen Instanzen und Containern kann ebenfalls mittels Kommandozeilenbefehlen abgefragt werden. In Abbildung 2.6 werden die möglichen Kommandozeilenbefehle mittels des Befehls „zospt -h“ in einem Kommandofenster angezeigt.

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

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

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

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

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

Abbildung 2.6.: z/OSPT mögliche Kommandozeilenbefehle

Kapitel 3.

Vorgehensweise

Für die Beantwortung der Forschungsfrage „Ist es möglich, den Bereitstellungsprozess für z/OS Anwendung bei DATEV e.G. mit Hilfe des „IBM Cloud Provisioning and Management for z/OS“-Tools an cloud native Prozesse anzunähern?“ werden zunächst die Vorteile des cloud native Prozesses bei der DATEV e.G. dargestellt. Um letztendlich eine mögliche Verbesserung des momentan in der DATEV e.G. etablierten Breitstellungsprozesses für die Middlewarekomponenten CICS, Db2 und IBM MQ bewerten zu können, wird dieser analysiert.

Wie in Absatz 1.2 beschrieben, wird für die Beantwortung der Forschungsfragen eine Beispielanwendung, die als Laufzeitumgebung CICS, als Datenbank Db2 und als Message Lösung IBM MQ verwendet, benötigt. Hier bietet sich die „DATEV Rechnungsschreibung“[1] an. Dabei handelt es sich um eine legacy z/OS Anwendung. Ein Bereich dieser Anwendung erfüllt die oben genannten Kriterien. Um die genauen Anforderungen an die Middleware zu kennen, wurde dieser Bereich analysiert. Anhand dieser Anforderungen soll mittels z/OSPT oder z/OSMF eine individuelle, auf die Anwendung zugeschnittene Laufzeitumgebung bereitgestellt werden. Konkret verlangt dies die Implementierung eines Templates. Für die Entscheidung, ob z/OSPT oder z/OSMF zum Einsatz kommt, wurden die Vor- und Nachteile beider Lösungen gegenübergestellt.

Im Folgenden wird speziell auf die Vorgehensweise bei der Implementierung des Templates eingegangen.
Um die von allen Entwicklern bei DATEV e.G. verwendeten Subsysteme der Entwicklungsstage bei eventuell auftretenden Fehlern in der neuen, automatisierten Bereitstellung nicht zu beeinflussen, wurden die ersten Schritte auf dem Testplex durchgeführt. Die Anwendung selbst kann auf dem Testplex nicht erprobt werden, da die notwendigen Testdaten auf Db2 dort nicht verfügbar sind. Deshalb wird dort vorerst nur die Provisionierung der benötigten Middleware untersucht. Damit wird das Ziel verfolgt, erste Erfahrungen mit „IBM Cloud Provisioning and Management for z/OS“ zu sammeln und ein Template zu provisionieren, das DATEV spezifische Middleware, also CICS, Db2 und IBM MQ Instanzen beinhaltet.

[1] Beschreibung siehe Absatz 4.3

Da das CICS Subsystem als Laufzeitumgebung im Mittelpunkt der Subsysteme steht, werden zunächst folgende, von der IBM bei der Installation von z/OSMF mitgelierferten, CICS Templates untersucht:

- „cics_getting_started“
- „cics_54“

„cics_getting_started“
Dieses Template wird von der IBM für die ersten Schritte der Provisionierung einer CICS Instanz angeboten. Das Template bietet nur minimale Konfigurationsmöglichkeiten. So entspricht die CICS Instanz einer minimal lauffähigen CICS Instanz nach IBM Standard.

„cics_54“
Hierbei handelt es sich um ein Template, das eine vollumfängliche CICS Instanz nach IBM Standard mit der CICS Version 5.4 provisioniert. Es ermöglicht die Angabe von komplexen Konfigurationen.

Mit den durch die Untersuchung dieser beiden Templates gesammelten Erfahrungen erfolgt die Implementierung eines an das DATEV e.G. Umfeld angepassten CICS Templates. Ist die daraus erzeugte CICS Instanz funktionsfähig, wird die Provisionierung einer Db2 Datenbank und IBM MQ Queues nacheinander in das Template aufgenommen. Für ein Erzeugen von Db2 und IBM MQ existieren bei DATEV e.G. bereits einzelne, voneinander unabhängige Services der jeweiligen Administrations-Teams, die als Bausteine in das Template integriert werden können. Für die Provisionierung von Db2 Datenbanken existiert z.B. bereits eine REST-API. IBM MQ Queues werden mittels Standard IBM Jobs 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 vom Testplex in die Entwicklungsstage. In der Entwicklungsstage sind alle Anwendungsdaten, die für Anwendungsttests notwendig sind, vorhanden. Somit kann hier die Integration der Beispielanwendung in die provisionierte Laufzeitumgebung sowie ein Testlauf stattfinden.

Um ein Meinungsbild bezüglich des Einsatzes von „IBM Cloud Provisioning and Management for z/OS“ bei DATEV e.G. zu bekommen, wurden mit Stakeholdern, also Mitarbeitern der Administratorenteams, Entwicklern und dem Technologiestrategieteam semi-strukturelle Interviews durchgeführt. Aus den Administratorenteams und von den Entwicklern wurden jeweils zwei Vertreter und aus dem Technologiestrategieteam ein Vertreter befragt. Es handelt sich hier um Experten in ihrem Arbeitsbereich, es ist zu beachten, dass hier generell nur eine geringe Anzahl von Kollegen für Befragungen zur Verfügung steht. Es wurden semi-strukturelle Interviews gewählt, um auf einzelne Antworten genauer eingehen

zu können. Auf eine Transkription der Interviews wurde verzichtet, da nicht der genaue Verlauf der Interviews, sondern die inhaltlichen Aussagen in den Antworten auf die Fragen relevant sind.

Die Interviews fanden in Besprechungsräumen innerhalb eines DATEV e.G. Gebäudes statt und dauerten etwa dreißig bis vierzig Minuten. Vor den eigentlichen Interviews wurde sowohl die z/OSMF Lösung (siehe Absatz 5.4) als auch die durch z/OSPT ermöglichte Lösung (siehe Absatz 6) den kompletten Teams erläutert. Der Schwerpunkt der Vorstellung wurde an die jeweilige Zielgruppe (Entwickler, Administrator, Technologie-Strategie) angepasst. So wurde bei den Administratorenteams vor allem auf die Erstellung der Templates, welche für ihr Arbeitsgebiet relevant ist, eingegangen. Sowohl der Fragenkatalog, als auch die ausgefüllten und digitalisierten Fragebögen sind im Anhang A.7 zu finden. Für das Entwicklerteam und die Mitarbeiterin des Technologiestrategieteams waren nur die Fragen 1., 2. und 6. bis 10. des Fragebogens von Relevanz.

Kapitel 4.

Analyse

Im Folgenden wird der Einsatz von cloud native bei der DATEV e.G. analysiert. Im Vergleich dazu wird der aktuelle Bereitstellungsprozess für die Laufzeitumgebung, dem dazugehörigen Datenbanksystem und einer Messaging Lösung erläutert. Anschließend erfolgt eine Beschreibung der Beispielanwendung „DATEV-Rechnungsschreibung“. Die dafür benötigten Informationen stammen aus Gesprächen mit einem Entwickler aus der Abteilung, die für die DATEV-Rechnungsschreibung zuständig ist. Es wird vor allem der technische Aspekt beleuchtet.

4.1. Analyse von cloud native bei DATEV e.G.

Neben den im Absatz 2.1 beschriebenen Begriffe, „cloud native“, „Cloud Foundry“ und „CI/CD“ ist der sogenannte „DevOps“-Gedanke in diesem Zusammenhang bei der DATEV e.G. von Bedeutung. Laut dem Buch „The Phoenix Project“ von Gene Kim[1] lassen sich die Werte und die Philosophie von DevOps in „Three Ways“ erläutern.

„The First Way“ betrachtet den Fluss von der Entwicklung über die Middleware Verwaltung bis hin zum Kunden. Um diesen Fluss zu optimieren, sollten möglichst kleine Änderungen in möglichst kleinen Arbeitsabschnitten entwickelt werden. Dabei helfen unter anderem Continuous Integration[2], Continuous Delivery[3] und das automatische Erstellen von Laufzeitumgebungen nach Bedarf.

„The Second Way“ betrachtet den Feedback-Fluss über alle Stages hinweg. Ziel dabei ist, möglichst schnell Fehler zu erkennen und diese zu beheben oder gar zu vermeiden. Zur Erreichung dieses Ziels gehört unter anderem das Abschaffen der sog. „Schubladendenkweise“. Ein Beispiel aus Entwicklersicht:

[1] [Kim 14, Kap. „The Three Ways Explained“]
[2] Siehe Absatz 2.1.2
[3] Siehe Absatz 2.1.2

„Meine Anwendung läuft nicht... Ach da gibt es Probleme mit dem Webserver, das werden die Kollegen der Administration schon lösen“.

„The Third Way“ betrachtet weniger den Prozess an sich, sondern eher das Schaffen einer Kultur. Eine Kultur, die erlaubt Risiken einzugehen und aus Fehlern zu lernen, aber auch das Verständnis, dass Wiederholung und Übung Voraussetzung für einen guten Entwickler sind. Dabei ist ein hoher Grad an Vertrauen in das Team notwendig.

Bei der DATEV e.G. wird dieser DevOps-Gedanke durch sogenannte „Product-Teams“ umgesetzt. In einem solchen Team kümmert man sich neben der eigentlichen Softwareentwicklung, Testmaßnahmen usw. auch um den Betrieb der Middleware, z.B. die im Cloud Native Umfeld üblichen noSQL Datenbanken wie Mongo oder PostgreSQL. Dem ganzen Team wird viel Vertrauen gegeben und es soll weitestgehend autonom Software erstellen und betreiben können. Dadurch steigt die Effektivität des Teams.

Die Kombination mit einer automatisierten CI/CD-Pipeline birgt weitere Vorteile. So sorgt Continuous Integration dafür, dass individuelle Codeänderungen sofort nach der Integration in die Codebasis auf Korrektheit geprüft werden. Im Fehlerfall werden die Entwickler sofort automatisch benachrichtigt. Verbunden mit regelmäßiger Integration führt das zu einer schnelleren und einfacheren Fehlerbehebung. Ein weiterer Vorteil ist, dass neben der Prüfung auf Korrektheit auch eine statische Codeanalyse möglich ist. Dadurch kann sichergestellt werden, dass der Code den gewünschten Qualitätskriterien entspricht. Durch Continuous Delivery und Continuous Deployment kommen diese Änderungen im Idealfall voll automatisiert beim Kunden an. Auch dies führt zu einer Effizienzsteigerung.

Für die Bereitstellung der benötigten Laufzeitumgebung kommt bei DATEV e.G. die Cloud Foundry Distribution der Firma „pivotal“ als PaaS Plattform zum Einsatz. Durch den darin zur Verfügung gestellten „Marketplace“ können Laufzeitumgebungen effizient „auf Knopfdruck“generiert werden. Mittels des Kommandozeileninterfaces kann die Bereitstellung auch in eine CI/CD-Pipeline integriert werden. So werden z.B. für jeden Entwickler isolierte Testumgebungen geschaffen.

4.2. Aktueller Bereitstellungsprozess

Im Vergleich zu den Product-Teams im cloud native Umfeld, bei denen sich das Team selbst um den Betrieb der Middleware kümmert, verwalten im z/OS Umfeld eigens dafür zuständige Administratorenteams die Middleware auf allen Stages. Die „CICS Administration“ kümmert sich um alles rund um das CICS Subsystem. Die „Db2 Administration“ stellt Datenbanken und Tabellen auf Anfrage der Entwickler bereit. Die „IBM MQ Administration“ verwaltet die IBM MQ Ressourcen. Um die Bereiche, an denen das „IBM Cloud Provisioning

and Management for z/OS“-Tool helfen kann, identifizieren zu können, wird im Folgenden der aktuell etablierte Prozess für z/OS Anwendungen beschrieben. Die in diesem Absatz genannten Informationen zu den Prozessen und Einschätzungen, bzgl. Zeitaufwand stammen aus Gesprächen mit Mitarbeitern aus den jeweiligen Administratorenteams. Wie in Absatz 2.3 beschrieben benötigt eine z/OS Anwendung zunächst eine Laufzeitumgebung, im Fall dieser Arbeit handelt es sich um CICS.

4.2.1. Bereitstellung einer CICS Instanz

Um eine lauffähige CICS-Instanz einzurichten, sind mehrere Schritte notwendig. Der komplette Prozess wird in Abbildung 4.1 dargestellt. Wie zu sehen ist, ist der Initiator des Prozesses das Entwicklerteam. Zunächst wird dort während der Entwicklungsphase festgestellt, dass ein neue CICS-Instanz benötigt wird, z.B. um eine neue Anforderung testen zu können. Hier hilft das CICS Administratorenteam mittels Beratung. Während einer Beratungsphase, die via Telefon, Emails oder Terminen stattfindet, wird sichergestellt, ob wirklich eine neue CICS-Instanz notwendig ist oder ob nicht eine bereits bestehende Instanz genutzt werden kann, die dann aber meist wieder zwischen Anwendungen und Entwicklern geteilt wird. Falls eine neue CICS-Instanz benötigt wird, wird ein RACF Eintrag für diese Instanz beantragt. Dieser Eintrag wird dann vom RACF Team erzeugt. Um sicherzustellen, dass die CICS-Instanz in den täglichen Sicherungen enthalten ist, muss das System Automations-Team benachrichtigt werden.

Nun kann mit dem eigentlichen Anlegen der CICS-Instanz begonnen werden. Dabei müssen folgende Schritte manuell durchgeführt werden. Es werden nur die Schritte, die im Laufe dieser Arbeit durch „IBM Cloud Provisioning and Management for z/OS“ automatisiert werden, dargestellt. Es handelt sich um das Anlegen von:

- CICS spezifische Dateien
- „CICS System Definition“
- Started Task Control-Job

CICS spezifische Dateien

Zunächst müssen CICS spezifische Dateien im z/OS angelegt werden. Im Fall des dieser Arbeit zugrunde liegenden Beispiels handelt es sich um siebzehn verschiedene VSAM Dateien. Diese Dateien benötigt die CICS-Instanz um zum Beispiel Systemfehler zu protokollieren oder den Debugger aktivieren zu können.

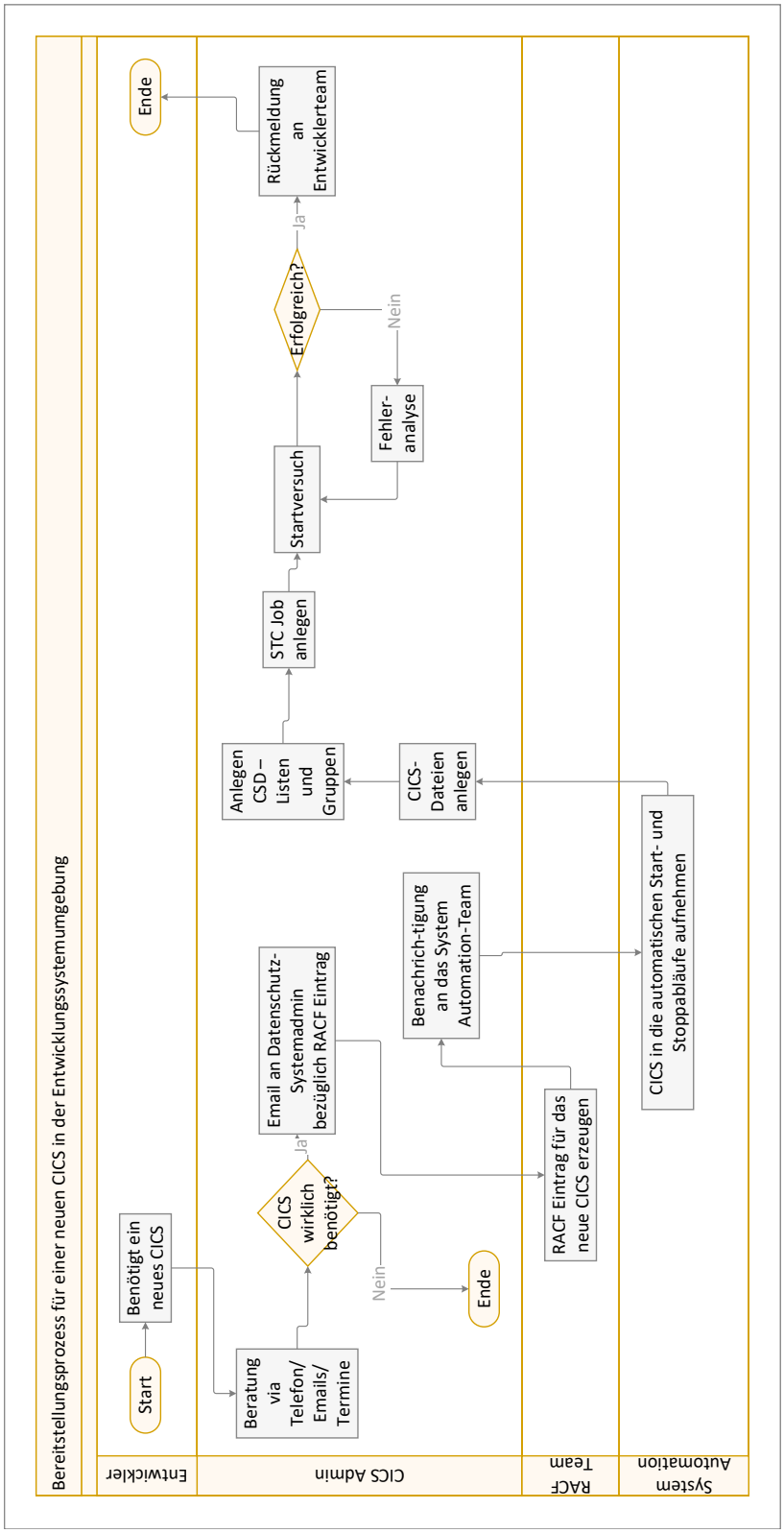


Abbildung 4.1.: Bereitstellungsprozess einer CICS Instanz

„CICS System Definition“

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 und 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, mit welchem Db2 Datenbankmanagementsystem und welchem MQ Queue Manager sich diese Instanz verbinden soll. Somit können Transaktionen auf diese beiden Ressourcen zugreifen.

Started Task Control-Job

Bei einem Started Task Control-Job, kurz STC Job, handelt es sich um einen langlaufenden Batch Job, der mit Hilfe des „START“-Konsolenkommandos innerhalb von z/OS gestartet werden kann. Dieser Batch Job wird deshalb auch als Started Task bezeichnet.[Cass 07, S. 224] Bei der DATEV e.G. existiert für jede Instanz eines Subsystems (z.B. CICS) ein solcher Job. In diesem werden zunächst einige zur Laufzeit benötigten Bibliotheken und Dateien eingebunden, unter anderem die CICS spezifischen Dateien[4]. Außerdem werden hier die SIT [5] Parameter definiert. Zunächst wird festgelegt welche Standard SIT verwendet werden soll. Anschließend können diese Standardwerte überschrieben werden. Zu diesen Parametern zählen unter anderem der eindeutige Name der CICS-Instanz, der Speicherort der dazugehörenden CSD und die Information, ob eine Verbindung zu einem Db2 Datenbankmanagementsystem und/oder zu einem Queue Manager hergestellt werden soll.

Nach der Durchführung dieser Schritte wird das CICS gestartet und es steht dem Entwickler eine neue CICS-Instanz zur Verfügung. Der komplette Ablauf dauert, unter der Annahme, dass alle Beteiligten verfügbar sind, sie nur diese Anforderung umsetzen müssen und für die Beratung ein Arbeitstag veranschlagt wird, circa zwei Arbeitstage.

4.2.2. Bereitstellungsprozess einer Db2 Datenbank

Wie in Abbildung 4.2 zu erkennen ist, ist der Bereitstellungsprozess einer neuen Db2 Datenbank mit vielen Vorarbeiten im Entwicklerteam verbunden. Zunächst müssen sogenannte Projektinformationen, unter anderem Daten der Voruntersuchung, vom Entwicklerteam bereitgestellt werden. Das Projektkürzel[6], der Datenbank- und Projektname sowie die Projektbezeichnung müssen mit den involvierten Abteilungen besprochen werden. Über den sogenannten „Datenbankänderungsantrag“ wird ein Genehmigungsprozess angestoßen. Wenn

[4] Beschreibung in Absatz 4.2.1
[5] CICS system initialization table
[6] eindeutige Kennung von DATEV Projekten

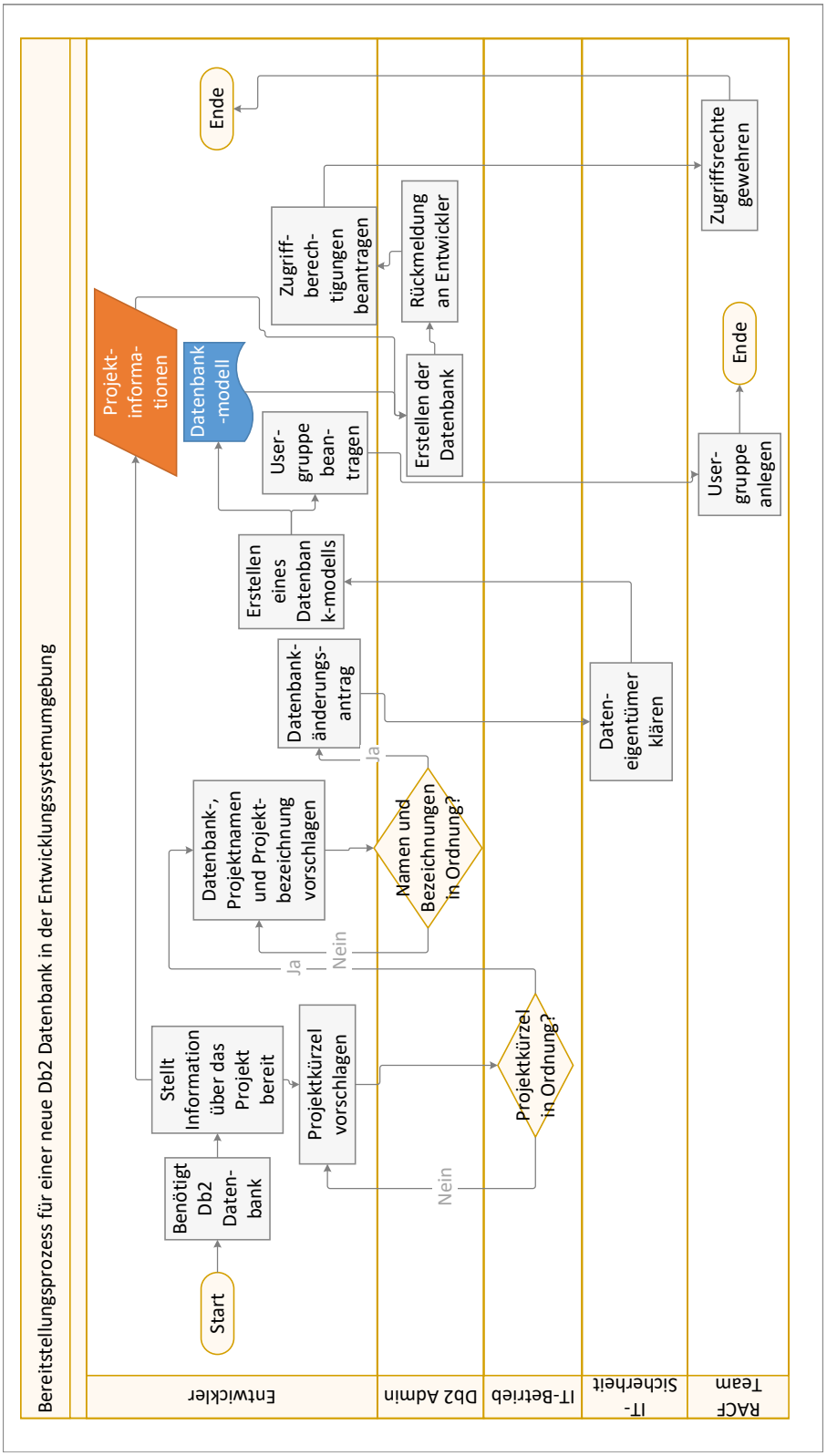


Abbildung 4.2.: Bereistellungsprozess einer Db2 Datenbank

alle Genehmigungen erteilt wurden, kann ein Dateneigentümer festgelegt werden. Anschließend muss die Datenbank mittels eines Datenbankmodells vom Entwicklerteam beschrieben werden und eine Usergruppe, die im späteren Verlauf die Datenbankzugriffsrechte benötigt, beantragt und angelegt werden. Die eigentliche manuelle Erstellung der Datenbank wird mittels des Datenbankmodells und den Projektinformationen im Anschluss dazu durchgeführt.

Die Zugriffsrechte für die zuvor angeforderte Usergruppe auf die neue Datenbank werden beantragt. Schließlich steht dem Entwicklerteam die neue Db2 Datenbank zur Verfügung. Wird die Db2 Datenbank in Verbindung mit einem CICS verwendet, wie im Fallbeispiel dieser Arbeit, der DATEV-Rechnungsschreibung, so muss der entsprechende Eintrag in die CSD Datei manuell aufgenommen werden. Der komplette Ablauf dauert, unter der Annahme, dass alle Beteiligten verfügbar sind, sie nur diese Anforderung umsetzen müssen und für die Beratung ein Arbeitstag veranschlagt wird, circa zwei Arbeitstage.

4.2.3. Bereitstellungsprozess einer IBM MQ Queue

Auch bei dem Bereitstellungsprozess, siehe Abbildung 4.3, einer IBM MQ Queue ist das Entwicklerteam der Initiator.

Die Grundlage diese Prozesses ist ein Antrag auf Erstellung einer neuen IBM MQ Queue. Zuvor findet eine Beratung via Telefon, Email oder Terminen statt. Die Queues werden anschließend manuell eingerichtet und stehen dem Entwicklerteam zur Verfügung. Wird die Queue in Verbindung mit einem CICS verwendet, so muss der Queue Manger auf dem die Queue angelegt wurde dem CICS mittels eines entsprechenden Eintrags in der CSD Datei mitgeteilt werden. Trotz des scheinbar schmalen Prozesses dauert der Ablauf unter der Annahme, dass alle Beteiligten verfügbar sind, sie nur diese Anforderung umsetzen müssen und für die Beratung ein Arbeitstag veranschlagt wird, circa zwei Arbeitstage.

4.2.4. Zusammenfassung aktueller Bereitstellungsprozess

Wie in den drei Diagrammen, Abbildungen 4.1, 4.2 und 4.3, zu erkennen ist, ist der aktuelle Bereitstellungsprozess mit vielen manuellen Schritten verbunden. Der Hauptaufwand liegt bei den Administratorenteams, die in verschiedenen organisatorischen Einheiten angesiedelt sind. Das Entwicklerteam ist Initiator des Ablaufs. Folglich kümmert es sich um die erforderlichen Formulare und die erste Kontaktaufnahme zum jeweiligen Administratorenteam.

Zusätzlich zu den zahlreichen manuellen Schritten sind die vielen Absprachen zwischen mehreren Abteilungen zu nennen. Steht ein beteiligtes Team nicht zu Verfügung, oder sind andere Projekte höher priorisiert, kommt es zu Verzögerungen, das Entwicklungsteam muss warten, der komplette Zeitplan kann sich dadurch nach hinten verschieben. Der Prozess für

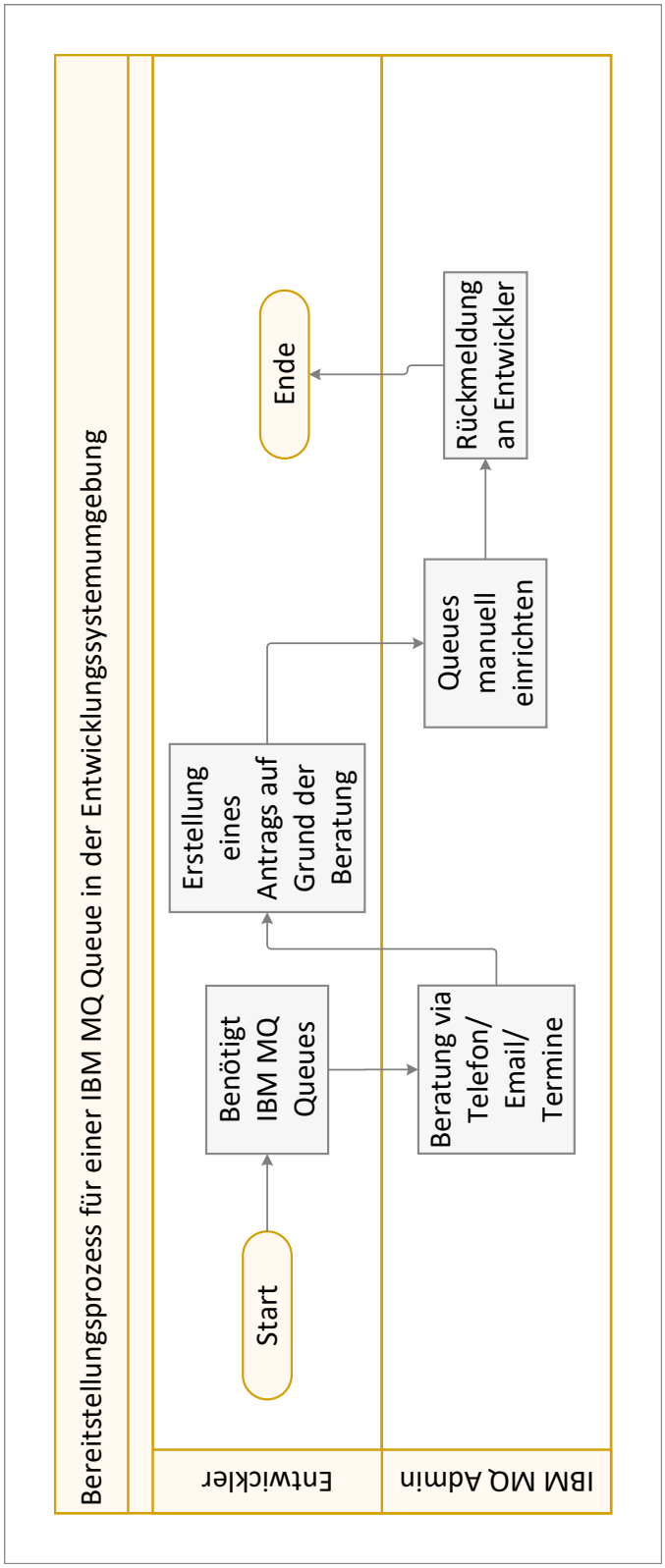


Abbildung 4.3.: Bereistellungsprozess einer IBM MQ Queue

die Bereitstellung einer neuen 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 Arbeitstagen gerechnet. Natürlich ist ein parallelisierter Ablauf der einzelnen Teilprozesse möglich, so kann die Gesamtdauer im besten Fall auf circa zwei bis drei Arbeitstage verkürzt werden.

Der Start des gesamten Prozesses findet meist auf „Zuruf“ statt. So existiert für die erste Kontaktaufnahme kein Formular, keine Automation oder ähnliches. Es wird auf E-Mail, Telefon oder mittels Terminen zurückgegriffen.

Wird jetzt der Vergleich zu dem Bereitstellungsprozess im cloud native-Umfeld[7] in folgenden Punkten gezogen, so steht dem automatisierten und dadurch effizienten cloud native Prozess, ein durch viele Absprachen langsamer und fehleranfälliger Prozess auf z/OS gegenüber. Das betrifft insbesondere die Bereiche

- Produktautonomie
- Entwicklungsdeployment
- Architektur
- Isolation

Produktautonomie

Die Autonomie eines Product-Teams ist hier nicht gegeben, die Administratorenteams sind für den Betrieb der Subsysteme verantwortlich. Daran wird sich auch durch automatisierte Provisionierung in Zukunft nichts ändern, da hier hohes Spezialistenwissen benötigt wird, das nicht von jedem Product-Team bereitgestellt werden kann. Dazu kommt, dass die Provisionierung im Rahmen dieser Arbeit auch nur die Bereitstellung von Entwicklungs-Instanzen der Middleware Systeme betrachtet. Die Administratorenteams sind aber insbesondere für die produktiven Instanzen im Betrieb verantwortlich, die nach wie vor von Anwendungen geteilt werden. Der Cloud-Aspekt des automatischen Deployments einer Anwendung inklusive Runtime zwischen Stages wird hier noch nicht umgesetzt.

Entwicklungsdeployment

Da für das Deployment in die bislang geteilte Entwicklungsumgebung einer z/OS CICS Anwendung, wie in Absatz 2.4.1.3 beschrieben, auch geteilte Programm-Bibliotheken zum Einsatz kommen, muss beim Testen einer Anwendung immer geprüft werden, ob tatsächlich der von dem Entwickler zu testende Stand der Anwendung in dieser Bibliothek abgelegt ist, ob die Bibliothek der CICS Instanz zur Verfügung steht und ob nicht eine andere Bibliothek mit anderem Stand von der CICS Instanz angezogen wird. Diese Probleme verschärfen

[7] Siehe Absatz 4.1

sich, wenn an der gleichen Anwendung mehrere Mitarbeiter parallel entwickeln. Lt. den Erfahrungen der Entwickler sind hier viele Absprache notwendig, um beispielsweise nicht den Stand eines Kollegen zu überschreiben.

Architektur

Neben den Problemen im Bereitstellungsprozess ist die grundlegende Architektur einer z/OS Anwendung zu betrachten. Im Vergleich zur „Microservices“-Architektur, die im cloud native Umfeld häufig zum Einsatz kommt[8], sind klassische z/OS Anwendungen monolithisch aufgebaut. Insbesondere aus Performancegründen haben sich über die Jahre Abhängigkeiten und Verzahnungen zwischen einzelnen Anwendungen, auch Fachdomänen-übergreifend, gebildet. Diese starke Koppelung erhöht die Komplexität und erschwert das Testen dieser Anwendungen.

Isolation

Wird der Architekturaspekt der geteilten Subsystem Ressourcen betrachtet, kristallisiert sich ein weiteres Problem heraus. Es ist möglich, dass eine Anwendung durch einen Fehler (beispielsweise Speicherüberschreibung) eine komplette CICS Instanz zum Absturz bringt. Ist dies der Fall, sind auf Grund einer Anwendung alle anderen Anwendungen dieser CICS Instanz nicht mehr verfügbar, bzw können nicht mehr getestet werden. Mit Hilfe von isolierten Testumgebungen, d.h. anwendungsspezifisch und fallabhängig provisionierten Laufzeitumgebungen kann dieser Problem gelöst werden.

Nach diesem Vergleich sieht die DATEV e.G. bei dem hier zu untersuchenden automatisierten Prozess vor allem bzgl. Entwicklungseffizienz einen Vorteil. Dabei wird der Fokus auf die Effizienz bei der Bereitstellung isolierter Testumgebungen, ohne Wartezeit und mit möglichst wenig manueller Interaktion gelegt. Dadurch verspricht sich die DATEV e.G. kürzere Entwicklungszyklen und weniger Fehleranfälligkeit durch hohen Abstimmungsaufwand.

Hierzu wird in dieser Arbeit das Tool „IBM Cloud Provisioning and Management for z/OS“ für die automatisierte Bereitstellung von Laufzeitumgebungen für die Entwicklungsstage untersucht. Als Beispielanwendung dient ein Bereich der z/OS Anwendung „DATEV-Rechnungsschreibung“.

4.3. DATEV-Rechnungsschreibung

Für diese Arbeit wurde die DATEV-Rechnungsschreibung als Beispielanwendung herangezogen, weil sie folgenden Anforderungen entspricht. Es handelt sich zum einem um eine

[8] Siehe Absatz 2.1

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 IBM MQ als Messaginglösung. Somit kann ein umfangreicher Bereitstellungsmechanismus untersucht werden.

Bei dem Gesamtablauf handelt es sich um einen Batch-Ablauf auf dem Großrechner der DATEV e.G. Dieser setzt sich aus folgenden Teilen zusammen:

- Sammeln von Berechnungssätzen
- Tägliche Bewertung
- Rechnungsaufbereitung

Für die Beantwortung der Forschungsfragen ist nur der Teil der „Tägliche Bewertung“ relevant, die Preisermittlung.

4.3.1. 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[9]-Db2-Datenbank. Dort werden die Berechnungsdaten der letzten 36 Monate aufbewahrt. Dabei handelt es ich um insgesamt circa 3,8 Milliarden Datensätze mit einer Gesamtgröße von circa 400 GB mit Indizes. Diese Datensätze beinhalten alle Informationen für die endgültige Erzeugung der Rechnungen der DATEV e.G.

4.3.2. Preisermittlung

Die Preisermittlung ist für die Berechnung der Preise mit den dazugehörigen kundenindividuellen Abhängigkeiten, beispielsweise Rabatten, zuständig. Der Inpute kommt an Lasttagen von bis zu 180.000 Geschäftspartnern. Im DATEV e.G. Umfeld ist ein Geschäftspartner entweder eine Kanzlei oder ein einzelner Mandant. Aufgrund dieser Last wird die Berechnung auf dem IBM Mainframe der DATEV e.G. in CICS durchgeführt und ist in folgende zwei Bereiche aufgeteilt:

- Bereitstellen der Preisinformationen
- Berechnung der Preise

[9] Abrechnungs- und Umsatz-Basis

Bereitstellen der Preisinformationen

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 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 einem sogenannten „SHARED GETMAIN“-Bereich gespeichert. Dabei handelt es sich im Prinzip um einen Hauptspeicherbereich, der dem CICS Subsystem zur Verfügung steht. Die Adresse dieses Bereiches wird den Transaktionen zur Verfügung gestellt. Somit greifen die einzelnen Transaktionen nicht mehr direkt auf die Datenbank zu, sondern stattdessen auf den schnelleren Hauptspeicher. Diese Vorarbeit ist notwendig, da die notwendige Performance aufgrund von bis zu 60 Millionen Datenbankzugriffen sonst nicht erreicht werden würde.

Berechnung der Preise

Um die Berechnungsdaten der 180.000 Geschäftspartner an CICS-Instanzen zu übertragen, stehen dem System weitere Queues zur Verfügung. Darunter ist eine allgemeine Queue, in der alle Aufträge, die für die Weiterverarbeitung zur Verfügung stehen, geschrieben werden. Pro Geschäftspartner wird ein Auftrag angelegt. In diesem Auftrag befinden sich die Namen vier weiterer Queues. Eine dieser Queues beinhaltet alle Informationen, die für die Preisermittlung des dazugehörigen Geschäftspartners notwendig sind. Hierzu zählt unter anderem die Adresse des vorher beschriebenen Hauptspeicherbereichs. In den restlichen drei Queues sind die Ergebnisse der Preisermittlung gespeichert. Die Ergebnisse stehen somit dem 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-Transaktion 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. Eine solche Festlegung wird in Abstimmung zwischen Entwicklung und CICS-Administration abhängig von den Lastanforderungen getroffen.

Kapitel 5.

Realisierung

In diesem Kapitel wird die Implementierung eines Templates zur Beantwortung der Forschungsfragen der Arbeit[1] beschrieben. Dazu wird nach der im Kapitel 3 beschriebenen Reihenfolge der Arbeitsschritte vorgegangen. Es ist noch einmal zu erwähnen, dass zunächst abzuwägen ist, welche Implementierungsvariante des „IBM Cloud Provisioning and Management for z/OS“besser geeignet ist: z/OSPT oder z/OSMF. Die Provisionierung einer CICS-Instanz wird vorerst auf dem Testplex untersucht. Danach wird in weiteren Schritten zuerst eine Db2 Datenbank und schließlich IBM MQ Queues dem Bereitstellungsprozess hinzugefügt. Um einen Testablauf der DATEV-Rechnungsschreibung mit der so generierten Laufzeitumgebung durchführen zu können, muss das Template auch in der Entwicklungsstage verfügbar sein. Ist dies sichergestellt wird der dadurch ermöglichte Bereitstellungsprozess anhand von drei use-cases aufgezeigt. Es folgt ein Fazit zu dieser Implementierung. Zuletzt folgt eine Bewertung der implementierten Provisionierungslösung durch die Stakeholder bei DATEV e.G. (Entwickler, Administration, Technologiestrategie).

5.1. Vergleich zwischen z/OSPT und z/OSMF

Bei beiden Varianten steht sowohl eine schnelle Provisionierung, als auch Deprovisionierung von Instanzen durch den Entwickler im Vordergrund. Mittels Templates läuft beides automatisiert, verlässlich, und ohne manuelle Abstimmungen ab. Die Bereitstellung einer Instanz ist wiederholbar, so kann die Instanz z.B. im Fehlerfall sicher neu erstellt werden und muss nicht langlebig gepflegt werden.

[1] Siehe Absatz 1.2

Kriterium	z/OSPT	z/OSMF
Schnittstelle	Kommandozeile	browserbasierende Oberfläche
Verwaltung von Templates	Zuweisung von Domains und Tenants nicht intuitiv	Alle Arbeitsschritte intuitiv
Verwaltung von Instanzen bzw. Container	möglich	möglich
Einsatz von Images	Ja	Nein

Tabelle 5.1.: Vergleich zwischen z/OSPT und z/OSMF

Es folgt ein Vergleich der beiden Tools an Hand folgender Kriterien:

- Schnittstelle
- Verwaltung von Templates
- Verwaltung von Instanzen bzw. Containern
- Einsatz von Images

Aus der Tabelle 5.1 ergibt sich folgendes Fazit:
z/OSPT ist durch den Einsatz von Images deutlich flexibler bezüglich der Konfigurationsmöglichkeiten der Templates. Jedoch ist die browserbasierende Schnittstelle von z/OSMF intuitiver als ein Kommandozeileninterface, dadurch fällt die Einarbeitung in automatisierte Bereitstellungsmechanismen leichter. Hinzu kommt, dass innerhalb von z/OSPT für die Zuweisung von Domains und Tenants eine weitere externe Konfigurationsdatei gepflegt werden muss.

Ausschlaggebender Grund für das Nutzen von z/OSMF für die Beantwortung der Forschungsfragen ist die browserbasierende Oberfläche. Die damit ermittelten Forschungsergebnisse sind aber auch als Basis für eine Bewertung von z/OSPT nutzbar.

5.2. Testplex

Der Zugriff auf Ressourcen und Tools bei DATEV e.G. wird über ein Rechtekonzept über RACF verwaltet. Um die Forschungsfragen beantworten zu können, mussten vor Beginn der eigentlichen Untersuchung 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. Beispielsweise benötigt „IBM Cloud Provisioning and Management for z/OS“ lesenden Zugriff auf den Speicherpfad der Template Dateien. Auf dem Testplex ist es möglich, die Rechte für das Erstellen der CICS Dateien, das Starten einer CICS Instanz und die Administration von

Db2 und IBM MQ einer persönlichen UserID zu geben, was in der Entwicklungsstage nicht ohne weiteres umsetzbar ist.

Schließlich konnte, wie in Kapitel 3 beschrieben, mit dem ersten Versuch, das bei der Installation von z/OSMF mitgelieferte „cics_getting_started“ Template zu provisionieren, begonnen werden. Ziel war es, mit dem Tool vertraut zu werden und die grundsätzliche Lauffähigkeit zu prüfen.

5.2.1. „cics_getting_started “-Template

Da es sich, wie in Kapitel 3 beschrieben, um ein mitgeliefertes Template handelt, sind alle benötigten Workflowdefinitionsfiles und Template Dateien vorhanden. Wie in Absatz 2.5.1 aufgezeigt, muss das Template in die Software Services von z/OSMF aufgenommen werden. Hierzu müssen die Template-und Workflow-Dateien in einem Unix Dateisystem auf dem Großrechner abgelegt sein. Nach dem Aufnehmen werden dem Template noch eine Domain und ein Tenant zugewiesen.[2]

Folgende Variablen sind in dem Variableinputfile nur mit Platzhaltern versehen und müssen für eine erfolgreiche Provisionierung ersetzt werden:

Variablenname	Kurzbeschreibung
DFH_REGION_APPLID	Applikations ID der zu provisionierenden CICS-Instance.
DFH_REGION_HLQ	High-level qualifier für die CICS Dateien.
DFH_STC_ID	User ID mit dem die CICS-Instanz startet.
DFH_REGION_VTAMNODE	Name des VTAM Knotens, wenn die CICS-Instanz hochfährt.
DFH_CICS_USSHOME	Homeverzeichnes des Unix System Services
DFH_CICS_HLQ	High-level qualifier von dem CICS Installationsort.

Tabelle 5.2.: Zu verändernde Variablen im „cics_getting_started “-Template

Als nächster Schritt wurde ein Testlauf und somit ein erster Versuch, das Template zu provisionieren, durchgeführt. Dabei kam es anfangs trotz Testplex-Umgebung zu Rechteproblemen, 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 wurde das „cics_getting_started “-Template provisioniert und die definierten Aktionen aus dem Actiondefinitionfile getestet. Dabei wurde sich auf das Nutzen der z/OSMF Oberfläche fokussiert und nicht auf die eigentliche Funktionsfähigkeit der CICS-Instanz.

[2] Siehe 2.5

5.2.2. „cics_54“-Template

Wie in Kapitel 3 bereits beschrieben, ermöglicht das „cics_54“-Template komplexere Konfigurationsmöglichkeiten. Es mussten neben den in Tabelle 5.2 genannten Variablen noch folgende, in Tabelle 5.3 genannte Variablen gesetzt werden. Die Kurzbeschreibungen und die Beschreibungen aller weiteren Variablen, die im Standard Template vorhanden sind, sind unter [http 20p] zu finden.

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 Prefix bei Authentificationanfragen für Ressourcen fest.
DFH_LE_HLQ	High-level qualifier[3] für die Sprachumgebung[4]
DFH_REGION_LOGSTREAM	Legt fest, wie die Log Dateien für die provisionierte CICS-Instanz erstellt werden sollen.
DFH_REGION_DFLTUSER	Default User ID für die CICS-Instanz.
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 abgegeben werden.

Tabelle 5.3.: Zu verändernde Variablen im „cics_54“-Template

Es wurden die gleichen Änderungen bezüglich der DATEV Job Vorgaben und der spezifischen Anforderungen der CICS-Start-Mechanismen wie in Absatz 5.2.1 durchgeführt. Nach Aufnahme in z/OSMF konnte die Provisionierung und Deprovisionierung erfolgreich durchgeführt werden. Dadurch wurden erste Erfahrungen mit z/OSMF und die grundsätzliche technische Funktionsweise in der echten DATEV e.G. Entwicklungsumgebung erprobt, was vor allem für die CICS-Administratoren einen wichtigen Schritt darstellte. Dennoch handelt es sich erst einmal um eine IBM Standard CICS Instanz, die nicht mit einer DATEV e.G. spezifischen CICS Instanz zu vergleichen ist.

5.2.2.1. DATEV e.G. spezifisches CICS Template

Ziel dieses Schritts war die Provisionierung einer funktionsfähigen DATEV e.G. spezifischen CICS Instanz. In der im letzten Schritt provisionierten Standard IBM CICS-Instanz sind

z.B. keine DATEV e.G. internen Transaktionen verfügbar. Dabei handelt es sich um interne Management-Transaktionen, mit denen z.B. Entwickler ihre fachlichen Transaktionen verwalten. Beispielhaft sei hier die Transaktion XMON zu erwähnen, ein DATEV proprietäres Auskunftssystem über CICS-Transaktionen, Programme, Dateien usw. Dies ist auch in einer automatisiert provisionierten CICS-Instanz notwendig.

Um dieses Template an die DATEV Umgebung anzupassen und letztendlich eine „DATEV CICS Instanz“ zu provisionieren, wurden folgende Schritte durchgeführt.

- Analyse des bestehenden Templates und der darauffolgenden Überarbeitung
- Umgang und Anpassung der CSD Datei
- Anpassung der Jobs und Skripte mit Schwerpunkt auf der „createCICS.jcl“-JCL-Datei[5].

Die Analyse ergab, dass das mitgelieferte „cics_54“-Template mit insgesamt 76 verwendeten Dateien sehr komplex und 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, festzulegen, ob dies mit Skripten oder mit der REST-API geschieht.

Die Grundstruktur des „cics_54“-Templates wurde beibehalten, allerdings wurden nach dem „YAGNI“-Prinzip[6] alle für eine DATEV e.G. spezifische CICS Instanz nicht benötigten Steps, die dazugehörigen Variablen und Dateien entfernt. Wie in Tabelle 5.4 zu sehen ist, konnten dadurch circa die Hälfte der Dateien gelöscht werden und bei der provision.xml konnte ein Drittel an Quellcode eingespart werden. Somit gewinnt das Template an Übersichtlichkeit und kann dadurch einfacher angepasst und gewartet werden. Das überarbeitete Template dient dem weiteren Vorgehen als Grundlage.

	IBM Standard CICS Template	DATEV e.G. spezifisches Template
Verwendete Dateien	76	36
provision.xml	circa 583 Codezeilen	circa 199 Codezeilen.

Tabelle 5.4.: Vergleich der beiden Templates in Bezug auf deren Umfang

Wie in Absatz 4.2.1 beschrieben, benötigt eine CICS Instanz spezifische Dateien. Die Namen dieser CICS Dateien wurden im dafür zuständigen Job an die DATEV e.G. internen Namenskonventionen angepasst.

[5] Quellcode siehe Anhang A.4
[6] „You aren’t gonna need it„-Prinzip

Die nächste Voraussetzung für die Bereitstellung einer CICS Instanz ist das Erstellen einer CSD-Datei[7]. Folgende Entscheidung wurde in Zusammenarbeit mit dem CICS Administratorenteam getroffen. Um Auswirkungen auf bereits im Einsatz befindliche Instanzen zu verhindern, wird die von den Kollegen gepflegte CSD-Datei bei jeder Provisionierung kopiert und mit bestimmten Namenskonventionen gespeichert. So besitzt jede automatisch bereitgestellte CICS Instanz eine eigene CSD Datei. Durch dieses Vorgehen wird zudem sichergestellt, dass bei jeder Provisionierung die aktuellste Version der CSD Datei verwendet wird. Im Falle einer gemeinsamen CSD Datei für alle provisionierten Instanzen müssten Änderungen, wie beispielsweise die Verfügbarkeit einer neueren CICS Version, an einer weiteren Stelle angepasst werden. Wird als Basis aber die aktuell schon verwaltete CSD Datei verwendet, genügt eine Neuprovisionierung des Templates um die Änderung auch für die provisionierten Instanzen zu übernehmen. Neue Ressourcen, wie zum Beispiel eine Verbindung zu einem Db2 Subsystem, können so ohne Nebenwirkungen zu anderen CICS Instanzen in die CSD aufgenommen werden. Ein weiterer Vorteil ist, dass bei der Deprovisionierung der CICS-Instanz diese Kopie der Standard Datei ohne Nebenwirkungen gelöscht werden kann.

Um dies umzusetzen, wurden zunächst zwei JCL Jobs geschrieben und in das entsprechende Workflowdefinitionfile eingebunden. Einer für die Implementierung des Kopiervorgangs und einer zum Löschen dieser Kopie. Für die Anpassung der CSD der zu provisionierenden Instanz ist ein weiterer Job, der „InitAdditionalCSD.jcl“-Job, notwendig. Es mussten bestimmte Gruppen (Zeilen sieben, neun und zehn in Abbildung 5.1) zu der CSD Liste der CICS Instanz hinzugefügt werden. Dabei handelt es sich um Ressourcen, die jede DATEV e.G. CICS Instanz benötigt. Die Reihenfolge ist relevant, da sie der Initialisierungsreihenfolge beim Startvorgang der CICS Instanz entspricht. Diese beiden Jobs wurde jeweils als neuer Workflowdefinitionfile-Step in den z/OSMF Workflow eingebunden.

[7] Beschreibung siehe Absatz 4.2.1

```
1  //INIT   EXEC PGM=DFHCSDUP
2  //STEPLIB  DD DSN=CICS.TS54.SDFHLOAD,DISP=SHR
3  //DFHCSD   DD DSN=CICS.DFHCSD.XPROV.TCICS42,DISP=SHR
4  //SYSPRINT DD SYSOUT=V
5  //*Reihenfolge ist WICHTIG!!
6  //SYSIN    DD *
7  ADD LIST(TCICS42) GROUP(TESTPCT)
8  ADD LIST(TCICS42) GROUP(DB0C)
9  ADD LIST(TCICS42) GROUP(RCTTEST)
10 ADD LIST(TCICS42) GROUP(FCTT1)
11 ADD LIST(TCICS42) GROUP(MQPROV01)
12 //
```

Listing 5.1: Hinzufügen weiterer CSD Gruppen zur Liste der provisionierten CICS-Instanz mittels des Jobs „InitAdditionalCSD.jcl“

Der abschließende Schritt zur Bereitstellung einer CICS Instanz ist das Erzeugen des STC Jobs. Die Definition bzw. das Script zur Erzeugung dieses Jobs ist in der „createCICS.jcl“-JCL-Datei zu finden. Im „cics_54“-Template beinhaltet diese ein Makro für die Validierung der SIT Parameter. Zusätzlich werden alle aus der Datei für die Eingabevariablen benötigten Variablenwerte 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

Für das Einbinden der benötigten Bibliotheken und der zuvor angelegten CICS spezifischen Dateien ist nur das Hinzufügen weiterer DD-Statements notwendig.

In Abbildung 5.2 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 e.G. spezifischen CICS wurde das ursprünglich in dem Template verwendete Makro für die Validierung von SIT Parametern beibehalten. Alles danach wurde zunächst durch eine zur Verfügung gestellten DATEV e.G. Standard JCL, für die Erzeugung eines CICS, ersetzt. Nach und nach wurde damit die für die DATEV e.G. spezifische CICS Provisionierung notwendige Logik, (siehe Abbildung 5.2), hinzugefügt. Damit wurde die vorher statische DATEV e.G. Standard JCL für das Erstellen von CICS Instanzen durch die Verwendung von Template Variablen flexibilisiert.

```
#set ($value5 = $!{instance_DFH_REGION_SITPARMS})
#set ($multipart = "NO")
#set ($tempStr = "")
#if($value5 != "")
#foreach( $sit in $value5.split(","))
#if($multipart == "YES")
#if( $sit.indexOf(')') > 0 )
## Validate SIT
#validateSit($tempStr.concat($sit.trim()))
#set ($multipart = "NO")
#else
#set ($tempStr = $tempStr + $sit.trim() + ",")
#end
#else
#if( $sit.indexOf('(') > 0 && $sit.indexOf(')') == -1 )
#set ($multipart = "YES")
#set ($tempStr = $sit.trim() + ",")
#else
#validateSit($sit.trim())
#end
#end
#end
#end
```

Listing 5.2: Setzen der SIT Parameter durch Auslesen der „DFH_REGION_SITPARAMS“ Variablen. (Zeile 379 bis 401 aus der „createCICS.jcl“-JCL-Datei, siehe Anhang A.4)

Es wurden nur die wirklich benötigten SIT Parameter aufgenommen. Die anzunehmenden Werte wurden einzeln mit dem 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 existiert 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 einer DATEV e.G. spezifischen CICS Instanz umgesetzt. Getestet wurde die Nutzbarkeit der Instanz mit einem Anmeldevorgang an dieses CICS, wie in Abbildung 5.1 zu sehen ist. Darüber hinaus

wurde erfolgreich nachgewiesen, dass Standard Transaktionen der DATEV e.G. in dieser Instanz funktionsfähig sind. Die Deprovisionierung verlief nach Plan.

Abbildung 5.1.: Login Bildschirm der provisionierten DATEV spezifischen CICS-Instanz

5.2.2.2. Bereitstellung Db2

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

Für die Erstellung einer Db2 Datenbank existiert innerhalb der DATEV e.G. bereits ein „Self Service“. Dieser stellt für die Provisionierung einer Db2 Datenbank auch eine REST-API zur Verfügung. Wie im Absatz 2.5 beschrieben, ist es möglich, innerhalb eines Workflow Steps einen REST-Request abzusenden. Der Code ist in Abbildung A.8 im Anhang zu finden. 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 dem DB0C Sandbox Datenbankmanagement-system[8] 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 5.1 in Zeile acht bereits zu sehen ist, das Hinzufügen einer weiteren CSD Gruppe notwendig, sowie die Aufnahme weitere Bibliotheken in die „createCICS.jcl“-JCL-Datei. Dieser Aufruf wurde mittels neuer Variablen im Template möglichst dynamisch gestaltet und diese Variablen mussten in dem Variableinputfile gesetzt werden.

[8]Siehe Absatz 2.4.2

5.2.2.3. Bereitstellung IBM MQ

In diesem Absatz wird die Provisionierung einer IBM MQ Queue im Testplex beschrieben. Es ist prinzipiell auch möglich, einen IBM MQ Queue Manager zu provisionieren, der Fokus dieser Arbeit liegt aber auf der Bereitstellung von Queues. Hintergrund ist, dass für IBM MQ Queue Manger bei DATEV e.G. laut IBM MQ-Administration vorerst keine automatische Bereitstellung vorgesehen werden soll, gegebenenfalls kann dies aber 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 vorerst nicht umzusetzen. Der Fokus lag damit 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 5.2 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 IBM 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 Message auf die Queue geschrieben und wieder abgeholt. Der Job für das Löschen der Queues ist analog aufgebaut.

Ähnlich wie in Absatz 5.2.2.2 für die Datenbank-Provisionierung beschrieben, muss der CSD Datei eine weitere Gruppe für den Queuemanager angegeben werden. Zu sehen in Abbildung 5.1 in Zeile 11. Dadurch hat eine Anwendung in dieser CICS-Instanz Zugriff auf alle Queues, die sich innerhalb dieses Managers befinden. Des Weiteren ist die Aufnahme weiterer IBM MQ-System-Bibliotheken in der „createCICS.jcl“-JCL-Datei notwendig.

5.3. Entwicklungsstage

Innerhalb der Entwicklungsstage sind die Sicherheits- und Rechtevorschriften schärfer als auf dem Testplex. So wäre es zwar möglich, alle für die administrativen Aufgaben notwendigen Rechte einer persönlichen UserID zu geben. Dies würde aber bedeuten, dass auch alle Anwender (Entwickler) dieses Templates diese Rechte benötigen würden. 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 IBM MQ festgelegt, hierfür jeweils einen technischen User[9] 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

[9] User ID mit zunächst keinen Berechtigungen

```
****** ***************************** Top of Data ******************************
000100 //P$PRVMQ1 JOB (0000,00000,00000,00000),KRUG,
000200 //             CLASS=V,MSGCLASS=V,TIME=(4,3),
000300 //             MSGLEVEL=(1,1),REGION=0M,NOTIFY=XXXXXXX,USER=YYYYYYY
000400 //*-------------------------------------------------------------------*
000500 //*                                                                   *
000600 //* DISPLAY QUEUES                                                    *
000700 //*                                                                   *
000800 //*-------------------------------------------------------------------*
000900 //STEPDISQ  EXEC  PGM=CSQUTIL,PARM='M00I',REGION=0M
001000 /*
001100 //STEPLIB    DD DSN=MQS.TEST.LLT.SCSQAUTH,DISP=SHR
001200 //            DD DSN=MQS.TEST.LLT.SCSQANLE,DISP=SHR
001300 //SYSPRINT DD SYSOUT=*
001400 //SYSIN      DD     *
001500   COMMAND DDNAME(MQSCIN) FAILURE(STOP)
001600 /*
001700 //MQSCIN     DD     *
001800   DEFINE QLOCAL(AWTB.PABHGMSHARE) -
001900                 REPLACE -
002000                 MAXDEPTH(6) -
002100                 PROCESS(AWTP.AWTP) -
002200                 TRIGGER -
002300                 MAXMSGL(1300) -
002400                 INITQ(SERVICE.TCICS42.INITQ) -
002500                 TRIGTYPE(EVERY) -
002600                 QDEPTHHI(80) -
002700                 QDEPTHLO(40)
002720 /*
002800 //PUT       EXEC PGM=CSQ4BCS2,
002900 //             PARM=('AWTB.PABHGMSHARE M00I')
003100 //STEPLIB    DD DSN=MQS.TEST.LLT.SCSQAUTH,DISP=SHR
003200 //            DD DSN=MQS.TEST.LLT.SCSQANLE,DISP=SHR
003300 //            DD DSN=MQS.TEST.LLT.SCSQLOAD,DISP=SHR
003400 //STDOUT    DD SYSOUT=*
003500 //STDERR    DD SYSOUT=*
003600 //SYSPRINT DD SYSOUT=*
003700 //SYSIN     DD *
003800 TEST
003900 /*
004000 //GET       EXEC PGM=CSQ4BCJ1,
004100 //             PARM=('M00I AWTB.PABHGMSHARE 1 D N')
004300 //STEPLIB    DD DSN=MQS.TEST.LLT.SCSQAUTH,DISP=SHR
004400 //            DD DSN=MQS.TEST.LLT.SCSQANLE,DISP=SHR
004500 //            DD DSN=MQS.TEST.LLT.SCSQLOAD,DISP=SHR
004600 //SYSDBOUT DD   SYSOUT=*
004700 //SYSABOUT DD   SYSOUT=*
004800 //SYSPRINT DD   SYSOUT=*
004900 //SYSOUT   DD   SYSOUT=*
005000 //
****** ***************************** Bottom of Data ***************************
```

Abbildung 5.2.: Define IBM Queue, am Beispiel einer Trigger Queue

diesen technischen Usern ausführen zu dürfen. Für Db2 ist ein solcher User nicht notwendig, da es sich beim Datenbankmanagementsystem hinter der REST-API um DB0C[10] handelt und jeder Entwickler in dieser „DB2-Sandbox“ seine Datenbanken verwalten darf.

Bei der Übertragung des Templates vom Testplex in die Entwicklungsstage waren Anpassungen in allen drei Bereichen des Templates notwendig.

5.3.1. CICS Anpassung

Damit der CICS spezifische technische User zum Einsatz kommt, musste der „Job“ Baustein jeder JCL in jedem Step modifiziert werden. Dafür bietet z/OSMF die Möglichkeit, beim Zuweisen des Tenants eine Standard Jobkarte[11], die vor jedem Job des Templates eingefügt wird, zu hinterlegen. Die CICS spezifischen Dateien können von der täglichen Datensicherung der Entwicklungsstage ausgeschlossen werden, da diese bei der Deprovisionierung gelöscht werden. Um dies zu gewährleisten, musste der Messageclass Parameter mit dem Wert „NONE“ angegeben werden.

[10] Siehe Absatz 2.4.2
[11] Beschrieben im Glossar Eintrag Batch Job

Als Vorlage für die spezifische CSD Datei, die im Template genutzt wird, wird die Standard Entwicklunsgstage CSD genutzt. In der Entwicklungsstage kommen im Vergleich zum Testplex andere Db2 und IBM MQ Bibliotheken zum Einsatz. Dahingehend wurde die „createCICS.jcl“-JCL-Datei angepasst. Zusätzlich musste ein SIT Parameter angepasst werden, so dass die Log Dateien in der Entwicklungsstage funktionsfähig sind. Eine weitere CSD Gruppe für die Verkettung der Anwendungsbibliotheken musste hinzugefügt werden. Siehe Zeile 16 im Codeabschnitt 5.3. Diese sorgt dafür, dass die Bibliotheken, die die kompilierten Programme der kompletten Entwicklungsstage beinhalten, zur Verfügung stehen. Dazu kam noch eine neue Bibliothek. Diese dient später als Ablageort der kompilierten Programme, die explizit nur in dieser CICS-Instanz vorhanden sind und dort getestet werden sollen. Diese Verkettung ist ein Standardvorgehen innerhalb der DATEV e.G., mit dem es möglich ist, neue Programmversionen und Schnittstellen zu testen und Zugriff auf notwendige, in Entwicklung bereitstehende sonstige Module zu haben.

5.3.2. Db2 Anpassung

Um die Datenbankstruktur der DATEV-Rechnungsschreibung im DB0C Datenbankmanagementsystem nachzustellen, d.h. ein isoliertes Sandbox-Datenbank-System, zu provisionieren, wurde eine genaue technische Analyse der DATEV-Rechnungs-schreibungsdatenbank durchgeführt. Dabei stellte sich heraus, dass die Komplexität der benötigten Tabellen sehr hoch ist. 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[12]. Bei den meisten werden innerhalb der View noch weitere Tabellen, teilweise aus anderen Datenbanken, gejoint. Insgesamt besteht das System aus vierzehn Tabellen, die auf vier Datenbanken aufgeteilt sind, und zwölf Views für den Zugriff auf diese Tabellen.

Um dies in einer Db2-Sandbox-Instanz[13] nachzubilden, sind sog. DDL Skripte notwendig. Mit der Erstellung wurde von der Db2 Administration begonnen. Es stellte sich heraus, dass bereits für einen kleinen Teil an Tabellen circa 600 Zeilen DDL Code[14] notwendig sind. Eine vollständige Provisionierung eines so komplexen Datenbanksystems übersteigt den zeitlichen Rahmen dieser Arbeit. Sollte sich die Provisionierung generell als zielführend erweisen wird dieser Einmalaufwand erbracht werden. Im weiteren Lebenszyklus der Anwendung kann auf diese Db2 jederzeit zugegriffen werden, Änderungen sind ein erheblich geringerer Aufwand als die Initialleistung.

Für die weiteren Schritte der hier vorliegenden Arbeit wurden deshalb die bereits im Datenbankmanagementsystem DB0T vorhandenen Datenbanken der DATEV-Rechnungsschreibung

[12] Alias einer Datenbankabfrage, auf die wie auf eine normale Tabelle zugegriffen werden kann
[13] Siehe Absatz 2.4.2
[14] Im Anhang A.9 zufinden

genutzt. Hierfür mussten die dafür vorgesehenen Variablen in der Eingabedatei des Templates angepasst werden. Dadurch ändert sich die Gruppe in Zeile acht im Codeabschnitt 5.1 von „DB0C“ auf „DB0T“. Außerdem wurden sowohl in der Provisionierungs- als auch in der Deprovisionierungsdatei die Datenbanksteps auskommentiert, da diese für den weiteren Test in der Entwicklungs-Stage nicht mehr zum Einsatz kommen.

5.3.3. IBM MQ Anpassung

Da für die DATEV Rechnungsschreibung, wie im Absatz 4.3.2 beschrieben, sehr viele gleichartige Queues benötigt werden, wurde für die Erstellung dieser von den IBM MQ Administratorenteam ein REXX Skript angefertigt. Dies geschah unabhängig dieser Arbeit zum Zeitpunkt der Einführung des aktuellen DATEV Rechnungsschreibungsprozesses. Dieses Skript steht dieser Arbeit zur Verfügung. Für die Provisionierung IBM MQ Queues waren folgende Arbeitsschritte notwendig.

- Anpassung des zur Verfügung stehenden Skriptes
- Implementierung von Jobs für restliche Queues (d.h. nicht im REXX Skript enthalten)
- 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 Queue Manager 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 die maximale Queuetiefe und die maximale Länge einer einzelnen Nachricht festgelegt. Im Original-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 im Rahmen der DATEV Rechnungsschreibung nur für den Datenaustausch zwischen der CICS Transaktion für die Preisermittlung und dem Batch Ablauf zuständig. Wie in Absatz 4.3.2 beschrieben, benötigt der Ablauf noch weitere Queues. Da es sich hierbei um spezielle Queues handelt, wurde auf die im Absatz 5.2.2.3 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 Processes[15] 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 IBM MQ CSD Gruppe. Somit müsste für jede provisionierte CICS-Instanz eine solche CSD Gruppe angelegt werden. In Absprache mit der IBM MQ-Administrations wurde entschieden, die Verwaltung der IBM MQ CSD Gruppe komplett im Template durchzuführen. Diese Entscheidung hatte eine Änderung des in Abbildung 5.1 gezeigten Codes der „InitAdditionalCSD.jcl“, der durch einen Workflowdefinitionfile-Step dem Template zugeordnet ist, zur Folge. So wird, wie in Abbildung 5.3 in Zeile sieben bis zehn dargestellt, zunächst eine Gruppe angelegt und anschließend wird diese Gruppe dem CSD (Zeile fünfzehn) hinzugefügt.

```
1  //INIT   EXEC PGM=DFHCSDUP
2  //STEPLIB DD DSN=CICS.TS54.SDFHLOAD,DISP=SHR
3  //DFHCSD  DD DSN=CICS.DFHCSD.XPROV.TCICS42,DISP=SHR
4  //SYSPRINT DD SYSOUT=V
5  //*Reihenfolge ist WICHTIG!!
6  //SYSIN    DD *
7  DEFINE MQCONN(M00I)
8         G(MQPROV01)
9         MQNAME(M00I)
10        INITQ(SERVICE.TCICS42.INITQ)
11 ADD LIST(TCICS42) GROUP(TESTPCT)
12 ADD LIST(TCICS42) GROUP(DB0T)
13 ADD LIST(TCICS42) GROUP(RCTTEST)
14 ADD LIST(TCICS42) GROUP(FCTT1)
15 ADD LIST(TCICS42) GROUP(MQPROV01)
16 ADD LIST(TCICS42) GROUP(RPL)
17 //
```

Listing 5.3: Erstellung einer neuen CSD Gruppe mit Hilfe eines Jobs am Beispiel des „InitAdditionalCSD.jcl“-Jobs

[15] Siehe Absatz 2.4.3.3

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

5.3.4. Testablauf

Für die Prüfung der Funktionsfähigkeit der so generierten Laufzeitumgebung steht dieser Arbeit ein Testablauf zur Verfügung. Dieser wurde von den Mitarbeitern der DATEV Rechnungsschreibung bereitgestellt. Dabei handelt es sich um einen Teilablauf des gesamten DATEV Rechnungsschreibungsprozesses. In diesem Ablauf wird nur die Preisermittlung, die die Laufzeitumgebung CICS benötigt, getestet. Als Eingabe dienen vordefinierte Dateien mit Testdaten, die Ergebnisse werden zur Prüfung 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 in der provisionierten Laufzeitumgebung zu starten, musste lediglich der verwendete Queue Manager angepasst werden. Über die Queues und das verwendete Triggering wird die Transaktion im provisionierten CICS gestartet. Um das Ergebnis der Anwendung zu prüfen, wurde dies mit dem gleichen Testablauf und den gleichen Eingabedateien in der aktuell existierenden Testumgebung der DATEV e.G. Rechnungsschreibung durchgeführt. Bei einem anschließenden Vergleich der Ausgabedateien beider Läufe waren keine Abweichungen festzustellen und der Test wird somit als erfolgreich bewertet.

5.4. Bereitstellungsprozess aktuelles Template

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

1. Use-Case: Neue Template Instanz
   Ein Entwickler möchte eine Programmänderung auf einer isolierten Laufzeitumgebung testen. Dadurch wird eine neue Instanz eines Templates benötigt. Dem Entwicklerteam steht das Template in z/OSMF zur Verfügung und es wurde noch keine Instanz dieses Templates provisioniert.

2. Use-Case: Zusätzliche Template Instanz
   Ein weiterer Entwickler möchte parallel zu einem anderen Entwickler eine weitere Programmänderung (anderer „Git-Branch“) auf einer isolierten Laufzeitumgebung testen. Nur diese Änderung soll getestet werden, unabhängig von anderen parallelen Aktivitäten in der Anwendung. Dem Entwicklerteam steht dieses Template in z/OSMF zur Verfügung aber es wurde bereits eine Instanz dieses Templates bereitgestellt. Um isoliert von dieser existierenden Instanz und den darin vorhandenen Änderungen zu testen, wird eine weitere Instanz benötigt.

3. Use-Case: Änderungen durch Administratorenteam
   Die Provisionierung einer IBM MQ Instanz soll z.B. von REXX Skripten auf eine REST-API umgestellt werden. Dies hat Änderungen an Workflowdefinitionfiles zur Folge. Hier ist zwischen zwei weiteren Fällen zu unterscheiden:

   a) Das Template wurde noch nicht veröffentlicht.

   b) Das Template wurde veröffentlicht.

5.4.1. Use-Case: Neue Template Instanz

Der Entwickler meldet sich an der z/OSMF Oberfläche an und klickt auf den Menüleistenpunkt „Cloud Provisioning“. Anschließend öffnet er die „Software Services“ und wählt dort das oben genannte Template aus. Nun kann er mit einem Klick auf „Run“ (Siehe Abbildung 5.3) eine neue Instanz erzeugen. Mit dieser Instanz kann er seine Programmabläufe testen.

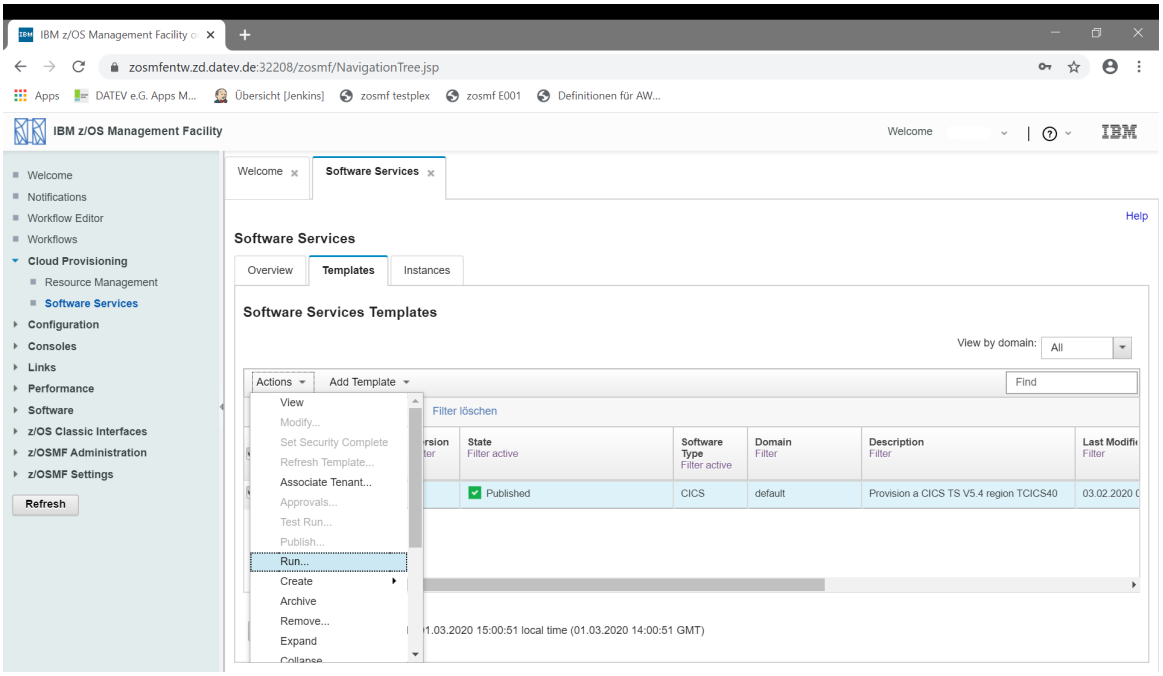


Abbildung 5.3.: Erzeugen einer Instanz mit dem „Run“-Befehl auf einem veröffentlichten Template in der z/OSMF Oberfläche

5.4.2. Use-Case: Zusätzliche Template Instanz

Mit dem aktuellen Stand des implementierten Templates muss der Entwickler wissen, an welchem Speicherort das Template abgelegt ist, da er die Template-Dateien - nicht die Workflowdateien - kopieren muss. Es sind Änderungen am Variableinputfile notwendig. Unter anderem ist eine andere CICS Application ID zu wählen, der Entwickler muss dafür wissen welche CICS Application ID von noch keiner Instanz genutzt wird. Um die Queues und IBM MQ Prozesse aus Fall eins nicht zu überschreiben, muss ein anderer Queue Manager vom Entwickler gesetzt werden. Dieser Queue Manager muss von dem zuständigen Administratorenteam manuell bereitgestellt werden. Die Erzeugung einer von Fall eins unabhängigen Instanz setzt die Aufnahme eines neuen Templates, welches die veränderten Dateien beinhalten, in z/OSMF voraus. Anschließend kann wie in Abbildung 5.3 dargestellt, eine weitere Instanz erzeugt werden.

5.4.3. Use-Case: Änderungen durch Administratorenteam

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

5.4.3.1. nicht veröffentlichtes Template

Hier kann das Template in der z/OSMF Oberfläche per Mausklick aktualisiert werden.

5.4.3.2. veröffentlichtes Template

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

5.5. Fazit Realisierung

Am Ende der Realisierung steht ein funktionsfähiges Template. Dieses Template provisioniert ein für die Rechnungsschreibung individualisiertes CICS und die benötigten IBM MQ Queues. Wie in Absatz 5.3.2 beschrieben, wurde die Provisionierung einer spezifischen Db2 Datenbank wegen hoher Komplexität für diese Arbeit außen vorgelassen Auf dem Testplex wurde bewiesen, dass die Provisionierung einer Datenbank möglich ist. Die Provisionierung der anwendungsspezifischen Db2-Tabellen wäre mit hohem einmaligen Arbeitsaufwand ebenfalls möglich. Ein Testablauf der Beispielanwendung DATEV-Rechnungsschreibung in einer provisionierten, isolierten CICS-Laufzeitumgebung konnte korrekt durchgeführt werden.

Folgende Probleme wurden im Rahmen der Implementierung erkannt:

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

Als erstes Problem sind nicht sprechenden Fehlermeldungen von z/OSMF, Abbildung 5.4, zu nennen. Z.B. wird bei dem Hinzufügen und Aktualisieren eines Templates in z/OSMF

Abbildung 5.4.: Beispiel einer Fehlermeldung von zOSMF

das Template und damit alle davon benötigten Dateien (Actiondefinitionfile, Workflowdefinitionfiles, Variableinputfile, JCL-Dateien und REXX-Skripte) auf Syntaxfehler geprüft. Die in Abbildung 5.4 gezeigte Meldung tritt dann ein, wenn einer dieser Dateien ein Syntaxfehler vorhanden ist. Beispielsweise wenn in einer JCL-Datei das Zeichenlimit von 80 Zeichen pro Zeile überschritten wird, dafür genügt schon ein „Blank“. Wie zu erkennen ist, zeigt

die Fehlermeldung weder 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 zur Laufzeit eines Workflow-Steps immer der Fehlercode und der genaue Ort des Fehlers ausgegeben. Beispielsweise wird bei einem Workflow-Step, in dem ein REST Aufruf durchgeführt wird, und ein Fehler auftritt, der Requestcode und die hinterlegte Fehlermeldung an der z/OSMF Oberfläche angezeigt.

Ein weiteres Problem ist eine nicht genau identifizierbare Programmiersprache, die für die dynamische Generierung von Skripten genutzt wird. Sie kommt unter anderem in den JCLs und REXX-Skripten, die dem Template über Workflowdefinitionfile-Steps zugeordnet werden, vor. Um diese Sprache einzusetzen muss am Zeilenanfang ein „#“ angegeben werden. Diese Sprache bietet einige Features. So ermöglicht diese die dynamische Wertzuweisung von zum Beispiel REXX-Variablen durch Variablen des Templates. Außerdem besteht eine Art von String Verarbeitung. In Abbildung 5.4 ist ein Beispiel zu sehen. Dort werden die Queuenamen, die als kommaseparierte Liste in der Templatevariable „DFH_MQ_QUEUENAMES“ 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 stehen außerdem zur Verfügung.

```
1  i=0
2  #set ($names = ${instance.DFH_MQ_QUEUENAMES})
3  #set ($tempStr = "")
4  #if($names != "")
5  #foreach( $queue in $names.split(","))
6  i=i+1
7  names.i="$queue"
8  #end
9  #end
10 names.0=i
```

Listing 5.4: Auslesen der „DFH_MQ_QUEUENAMES“ Variablen und schreiben in REXX Variablen (Zeile 52 bis 61 aus „defineQsRexx.jcl“, siehe Anhang A.5)

In Abbildung 5.5 wird das Ergebnis, welches beim Run erzeugt 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 eine Aussage, um welche Sprache es sich genau handelt gefunden. Somit liegt dem Wissen über diese Sprache nur der Code aus Beispielen der IBM zugrunde.

```
1  i=0
2  i=i+1
3  names.i="L1.GPNRBERINFO"
4  i=i+1
5  names.i="L1.KLAMMERINFOLIST"
6  i=i+1
7  names.i="L1.KUNDENPREISINFOLIST"
8  i=i+1
9  names.i="L1.PABHREFERENZLIST"
10 names.0=i
```

Listing 5.5: Zur Laufzeit erzeugtes Skript, der Grundlage aus Codeabschnitt 5.4

Ein weiterer Problempunkt ist das mit z/OSMF und dem Template einhergehende Zugriffsrechtekonzept. Die z/OSMF Berechtigungsgruppen passen nicht zu de DATEV e.G. internen Richtlinien. Die Aufnahme in eine solche Gruppe, um zum Beispiel die z/OSMF Oberfläche nutzen zu dürfen, geschieht auf Zuruf und manuelles Hinzufügen einer User ID durch einen Mitarbeiter, des RACF-Teams. Generell ist der Einsatz einer für das ganze Template gültigen Standard Jobkarte, um technische User verwenden zu können, nicht optimal. z/OSMF bietet hier eigentlich eine Möglichkeit in der Stepdefinition einen „runAsUser“ anzugeben. Unter diesem User würde der Step dann ausgeführt werden. Folglich ist das die Stelle, an der zum Beispiel für CICS Steps der technische User für administrative CICS Aufgaben angegeben werden müsste. Damit würde das Gewähren der expliziten Rechte zum Starten eines Jobs mit der technischen User ID entfallen und damit die manuelle Arbeit des „Gewährens“. Um jedoch einen „runAsUser“ in der Workflow-Stepdefinition angeben zu können, muss in der dem Template zugewiesenen Domain ein sogenannter „Cloud Security Admin“ hinterlegt sein. Dieser würde sicherstellen, dass nur die für ein Template zugelassenen User dieses Template auch provisionieren dürfen. D.h. es kann beispielweise sichergestellt werden, dass nur DATEV-Rechnungsschreibungs-Entwickler das DATEV-Rechnungsschreibungs Template provisionieren. 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 z/OSMF bei der DATEV e.G. noch in einem Untersuchungs-Stadium befindet, wird von der Erstellung einer eigenen Domain abgesehen. Dadurch ist es notwendig, die nicht optimale Lösung mit der Jobkarte[16] einzusetzen. An diesen beiden Fällen ist zu erkennen, dass das Rechtekonzept noch nicht für einen firmenweiten Einsatz bei DATEV e.G. ausgelegt ist und noch überarbeitet und angepasst werden muss. Dies ist jedoch explizit nicht Bestandteil dieser Arbeit.

[16] Siehe Absatz 5.3.1

5.6. Interviews

Um ein allgemeines Meinungsbild zu „IBM Cloud Provisioing and Management for z/OS“ und einen möglichen Nutzwert bei den Stakeholdern, also den Administratorenteams, den Entwicklern und dem Technologiestrategieteam zu erfassen, wurden Interviews mit einzelnen Vertretern dieser Teams durchgeführt. Sowohl der Fragenkatalog, als auch die ausgefüllten und digitalisierten Fragebögen sind im Anhang A.7 zu finden. Diese Fragebögen werden im Folgenden zunächst nach Gruppen ausgewertet. Schließlich wird daraus ein allgemeines Stimmungsbild abgeleitet.

5.6.1. CICS Administratoren

Der momentan etablierte Bereitstellungsprozess wird von der CICS Administration mit einem hohen manuellen Aufwand verbunden. Dies, kombiniert mit viel Abstimmungsbedarf zwischen den Administratoren- und Entwicklerteams, führt dazu, dass der Prozess als langsam und verbesserungswürdig angesehen wird. In der Umsetzung mit z/OSMF sieht die CICS Administration trotz des vermuteten, hohen Einarbeitungsaufwandes bereits einen Mehrwert. Der Hauptvorteil des vorgestellten z/OSPT Lösungsansatzes sei dessen Flexibilität. Jedoch schreckt bei z/OSPT die Komplexität des zu erstellenden dynamischen Templates ab. Dieser Effekt wird durch fehlende Toolunterstützung und dem dadurch fehlenden Syntaxhighlighting beim Editieren der Templatedateien bzw. der Workflowdefinitionfiles verstärkt. Ist die Hürde des Einarbeitungsaufwandes und der Eingewöhnung in das Editieren von Template Dateien und der Workflowdefinitionsdateien genommen, stehe einer aufwandssparenden Provisionierung mittels „IBM Cloud Provisioning and Management for z/OS“ nichts im Wege.

5.6.2. Db2 Administratoren

Das ganze „IBM Cloud Provisioing and Management for z/OS“-Tool wird als sehr mächtig, aber komplex beschrieben. Im Vergleich dazu funktioniere der momentan etablierte Bereitstellungsprozess sehr gut, da dieser bereits lange eingesetzt werde und damit ausreichend bekannt ist. Jedoch könnten lange Wartezeiten, die durch die vielen Abhängigkeiten von Personen und Abteilungen zustande kommen, durch einen automatisierten Ablauf mittels des Tools eliminiert werden. Der durch z/OSMF ermöglichte Prozess zeige zwar, dass eine Automatisierung in diesem Bereich möglich ist, aber auch noch viel Forschungsaufwand und Weiterentwicklung in diesem Bereich notwendig sei, um die Provisionierung wirklich nutzen zu können. z/OSPT diene dabei als Hilfsmittel, den Bereitstellungsprozess in eine CI/CD-Pipeline aufzunehmen und so weiter zu automatisieren. Das durch „IBM Cloud Provisioing

and Management for z/OS“ ermöglichte automatisierte Provisionieren von z/OS Middleware wird als notwendiger Schritt, um den Mainframe weiterhin erfolgreich zu betreiben, betrachtet.

5.6.2.1. IBM MQ Administratoren

Die IBM MQ Administratoren stimmen überein, dass der momentan etablierte Bereitstellungprozess mit einem hohen manuellen Arbeitsaufwand verbunden ist. Durch Arbeiten auf Zuruf und Kommunikation über Telefon, Email oder Terminen entstehen häufig Rückfragen. Die Meinungen über die Lösungen mit z/OSMF und z/OSPT sind jedoch unterschiedlich. So biete der z/OSMF Ablauf zwar einen Mehrwert durch Abbau von manuellen Eingriffen, allerdings sei dieser bezogen auf die Queues noch sehr spezifisch. Um einen größeren Mehrwert zu generieren, sei das automatische Provisionieren eines Queuemanagers mit in das Template aufzunehmen. Hier sei der zusätzliche Arbeitsaufwand bei der Erstellung eines Templates nicht zu vernachlässigen. Beim z/OSPT Lösungsansatz wird kritisiert, dass es sich um „pseudo“Docker Terminologie handle. Die hier von der IBM gewählte Namensgebung führe zu Verwirrung, da ein z/OSPT Container zwar ein Container im Sinne von einem Behältnis für Middleware ist, jedoch nicht im Sinne eines Docker Containers, der in unterschiedlichen Systemumgebungen lauffähig ist. Ein weiterer Kritikpunkt ist, dass das gesamte „IBM Cloud Provisioing and Management for z/OS“-Tool im Vergleich zur momentanen Cloud Foundry Lösung nicht einfach genug zu verwenden sei. Wenn diese Probleme behoben werden können, könnten sich die IBM MQ Administratoren vorstellen, IBM MQ Ressourcen mittels „IBM Cloud Provisioing and Management for z/OS“ zu verwalten und zur Verfügung zu stellen. Dabei sei zu beachten, dass erst noch eigene Erfahrungen mit dem Tool gesammelt werden sollten, bevor eine endgültige Bewertung möglich ist.

5.6.2.2. Entwicklerteam der DATEV Rechnungsschreibung

Der Entwicklerfragebogen wurde zusammen mit zwei Entwicklern ausgefüllt.

Aus Sicht des Entwicklers wird vor allem für den in Absatz 5.4.2 beschriebenen Fall viel Wissen über die z/OSMF Oberfläche und das Template selbst benötigt, da sie Änderungen direkt in dem Variableinputfile vornehmen müssen. Dieses Wissen müsse auch bei nicht häufiger Nutzung über einen längeren Zeitraum erhalten werden. Deshalb sei der z/OSPT Lösungsansatz, mit dem auf die z/OSMF Oberfläche verzichtet werden, und stattdessen mittels Jenkins Build Pipeline oder dem DATEV „Marktplatz“ gearbeitet werden könne, besser geeignet. Ist diese Integration möglich, wird ein Hauptvorteil darin gesehen, dass der Bereitstellungsprozess mehr in den Händen des eigenen Teams läge. So sei eine Steigerung der Effizienz zu erreichen. Es stehe dem Sammeln von Erfahrungen mit dem Prozess und mit „IBM Cloud Provisioing and Management for z/OS“ nichts im Wege. Für die Zukunft

könne man sich die Nutzung auch für die Qualitätsicherungs- und Produktionsstage, um dort beispielsweise CICS-Instanzen horizontal zu skalieren, vorstellen.

5.6.2.3. Fachberaterin im Bereich Technologiestrategie

Laut der Fachberaterin im Bereich Technologiestrategie ist der gezeigte Ablauf beziehungsweise die z/OSMF Oberfläche für die Aufgabe des Provisionierens von z/OS Middleware geeignet. Jedoch sei es besser, wenn z/OSMF in den bereits existierenden „Marktplatz“ für DATEV Cloud Lösungen integriert wäre. Der Prozess, der mit Hilfe von z/OSPT ermöglicht wird, wird als gut angesehen, da durch ihn die Entwicklung von z/OS Anwendungen an die Vorgehensweise der Cloud Native Entwicklung angenähert werden könne. Hier komme die Rolle des Build Engineers auch für solche Anwendungen ins Spiel. Dieser kümmert sich um die Erstellung und Pflege der Build-Pipeline. Ein große Nachteil im momentan etablierten Bereitstellungsprozess sei vor allem, dass eine Anzahl von Entwicklern, die parallel an einem Produkt arbeiten, sich die gleiche Entwicklungssystemumgebung teilen. So arbeiten alle mit der gleichen CICS-Instanz, der gleichen Test-Datenbank und mit den gleichen IBM MQ Queues. Dadurch beeinflussen Änderungen des einen Entwicklers die Tests der anderen Kollegen, es entsteht Koordinationsaufwand. Falls Änderungen an der Umgebung notwendig sind, kann während dieser Zeit kein Entwickler weiterarbeiten. Hier liege der Vorteil von „IBM Cloud Provisioing and Management for z/OS“. Es ermögliche aus Entwicklersicht eine sehr einfache, schnelle Möglichkeit eine isolierte Umgebung bereitzustellen, unabhängig von den 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 das „IBM Cloud Provisioing and Management for z/OS“-Tool die Entwicklung beziehungsweise den Bereitstellungsprozess deutlich verbessern könne. So sei für den Entwickler ein an die Cloud Native Welt angenäherter Entwicklungsprozess möglich. Dadurch werde der Wechsel zwischen beiden Umgebungen immer fließender.

5.6.3. Meinungsbild

Über alle befragten Mitarbeiter-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 Abstimmungsaufwand zeitaufwändig ist. Sie würden einen automatisierten und dadurch schnelleren und weniger fehleranfälligen Prozess begrüßen.

Jedoch muss dieser Prozess aus Entwicklersicht mit minimalem Konfigurationsaufwand verbunden sein. Dies könnte durch eine Provisionierung mittels z/OSPT und einer Integration in eine Jenkins Build Pipeline. Aus Sicht der Administratoren wären mit dieser Umsetzung nur wenige allgemeine Templates zu verwalten, da die Entwickler mit z/OSPT Images arbeiten und keine weiteren Templates erzeugen würden. Der dadurch komplexere Aufbau, der entstehende Erstaufwand bei der Erstellung solcher Templates und die damit verbundene steile Lernkurve hat eine abschreckende Wirkung.

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

Kapitel 6.

Ausblick

Je weiter sich das Projekt der vorliegenden Arbeit dem Abschluss näherte, desto mehr kristallisierte sich ein Hauptproblem für den praktischen Einsatz heraus. Das erstellte Template ist sehr auf die DATEV Rechnungsschreibung spezialisiert, das heißt es ist funktionsfähig, kann aber nicht ohne zeitaufwändige Eingriffe in das Template, in die Workflowdefinitionsdateien und die eigentlichen REXX Skripte und Jobs, an eine andere Anwendung angepasst werden. Folglich müsste das Template dynamischer implementiert sein. Um dies zu verdeutlichen, wird als Beispiel die Provisionierung von IBM MQ Queues herangezogen. Momentan werden die Prozesse und die Trigger Queues statisch angelegt. Das heißt, dass sowohl Namen, als auch die damit verknüpften Queueparameter, fest hinterlegt sind, um nur ein Beispiel zu nennen. Besser wäre es, alle Parameter in der Eingabedatei des Templates anzugeben. Aus IBM MQ Sicht ist hinzuzufügen, dass die fehlende automatisierte Bereitstellung eines Queue Managers den gewünschten Effekt einer weitgehenden Automatisierung und Unabhängigkeit von der Administration, abschwächt.

Während der Realisierung stellte sich ebenfalls heraus, dass ein Template, das mehrere Subsysteme beinhaltet und dadurch sehr anwendungsspezifisch ist, nicht für einen firmenweiten Einsatz geeignet ist. So ist zu empfehlen, dass für jedes Subsystem, also CICS, Db2 und IBM MQ, ein separates „Basis“-Template realisiert wird.

Angenommen es besteht für jedes Subsystem ein Template und das IBM MQ Template beinhaltet die Provisionierung eines Queue Managers, so könnte jeder Entwickler seine eigenen Instanzen der Templates besitzen und für eigene Tests nutzen. Dennoch wäre der ermöglichte Bereitstellungsprozess nicht optimal. So müsste für jede kleine Änderung an der Konfigurationsdatei ein neues Template erzeugt werden, siehe zweiter use-case im Abschnitt 5.4.2. Das dort genannte Beispiel einer CICS-Instanz und der notwendigen eindeutigen Application ID wird hier aufgegriffen. Eine Möglichkeit dieses Problem zu lösen, wäre einen Pool mit verfügbaren Application IDs bereitzustellen und dann mittels eines Programms eine ungenutzte Application ID zu ermitteln. Dieses Programm könnte dann als Step in das Template aufgenommen werden. Jedoch müsste immer noch für jede Änderung an der Konfigurationsdatei ein neues Template erzeugt werden.

Hier schafft z/OSPT Abhilfe. Damit kann, wie in Absatz 2.5.2 beschrieben, mit Hilfe der z/OSPT-Datei und dem Konzept der Images das Template von außen konfiguriert werden. Dadurch fällt das Kopieren des Templates für den Mitarbeiter weg, dieser kann mittels des Kommandozeileninterfaces ein z/OSPT-Image bauen und daraus einen z/OSPT-Container erzeugen. Das Kommandozeileninterface hat einen weiteren Vorteil. Mit dessen Hilfe können Arbeitsschritte für die Provisionierung der Middleware über groovy in einen Jenkins-Ablauf aufgenommen werden. Somit läuft der Prozess automatisiert ab und nähert sich modernen Entwicklungsabläufen wie denen aus der Cloud Native Entwicklung an.

Angenommen es existieren jeweils ein CICS, ein Db2 und ein IBM MQ Template, und diese sind so realisiert, dass sie firmenweit eingesetzt werden können. Dann wäre der nächste Schritt, die Aufnahme in den „DATEV Marktplatz“, möglich. Der „DATEV Marktplatz“ 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 Zukunft könnten hier auch Dienste wie CICS, Db2 und IBM MQ zur Auswahl stehen. Dabei ist in Betracht zu ziehen, ob für den User nur bestimmte vorgefertigte Profile, wie „klein“, „mittel“ und „groß“, auswählbar sind. Dies ist in public Clouds bzw. auch bei der DATEV PaaS Lösung ein übliches Vorgehen. Die im Hintergrund verbundenen Templates und Images müssten dahingehend angepasst werden. Um einen solchen „Service Broker“ zu verwirklichen, könnte die von z/OSMF zur Verfügung gestellte REST-API verwendet werden. Diese ermöglicht den Zugriff auf fast alle z/OSMF Funktionalitäten mittels Http-Requests. Für die Tenant Zuweisung zu einem Template würde weiterhin die z/OSMF Oberfläche benötigt werden. Insgesamt ist zu erkennen, dass mit z/OSMF beziehungsweise IBM hier noch Verbesserungen machbar sind.

Diese technische Umsetzung ermöglicht in Zukunft den in Diagramm 6.1 dargestellten Bereitstellungsprozess. Es ist zu erkennen, dass Verantwortung für die Bereitstellung einer Laufzeitumgebung in der Entwicklungsstage von den Administratorenteams an die Entwicklerteams übertragen wird. Dadurch wird Kommunikationsaufwand eingespart und einem Entwickler steht binnen weniger Minuten eine funktionsfähige Laufzeitumgebung in der Entwicklungsstage für seine legacy z/OS Anwendung zur Verfügung. Bei Problemen oder Beratungswunsch unterstützen die Administratorenteams weiterhin. Der Hauptaufwand der Umsetzung dieser Lösung liegt bei den Administratorenteams. Der Effizienzgewinn stellt sich ein, wenn für die wichtigsten Anwendungen Templates existieren, die von den Entwicklern jederzeit genutzt werden können, um in isolierten, individuellen Umgebungen zu entwickeln und zu testen.

Ist all dies umgesetzt, kann in einem nächsten Schritt der cloud native Aspekt des automatisierten Deployments einer Anwendung inklusive Laufzeitumgebung zwischen Stages bis hin zur Produktion auch für z/OS Anwendungen in Betracht gezogen werden.

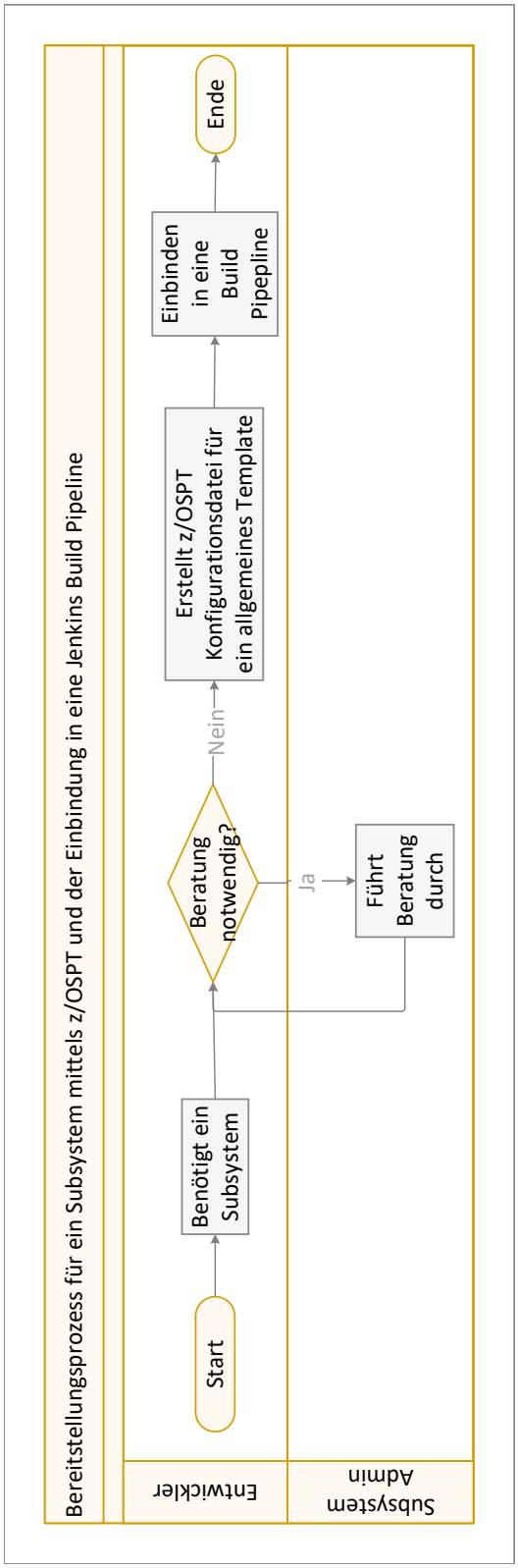


Abbildung 6.1.: Bereistellungsprozess eines Subsystems mittels einer z/OSPT Konfigurationsdatei

Kapitel 7.

Zusammenfassung und Fazit

Zusammenfassend lässt sich sagen, dass es generell möglich ist mit „IBM Cloud Provisioning and Management for z/OS“ Laufzeitumgebungen für legacy z/OS Anwendungen automatisiert bereitzustellen. Wie aus Tabelle 7.1 zu erkennen ist, ist es in einem gewissen Grad auch möglich damit den Bereitstellungsprozess für z/OS Anwendungen bei DATEV e.G. an den cloud native Prozess anzunähern.

	„IBM Cloud Provisioning and Management for z/OS“	cloud native
Produkt-Teams	nein	ja
automatisierte Bereitstellung von Laufzeitumgebungen in der:		
Entwicklungsstage	ja	ja
Qualitätssicherungsstage	nein	ja
Produktionsstage	nein	ja
CI/CD-Pipeline Unterstützung	ja, mit z/OSPT	ja

Tabelle 7.1.: Vergleich von „IBM Cloud Provisioning and Management for z/OS“ und cloud native in Bezug auf ihren Bereitstellungsprozess

Die Annäherung beschränkt sich jedoch nur auf eine automatisierte Provisionierung von Laufzeitumgebungen in der Entwicklungsstage. Es kann kein klassisches Product-Team-Vorgehen umgesetzt werden, sondern auch mit den Einsatz von „IBM Cloud Provisioning and Management for z/OS“ bleibt die Verwaltung und Überwachung der Middleware bei den Administratorenteams.

Wird der z/OSMF Lösungsansatz genauer betrachtet, dann ist zu erkennen, dass dieser durch den Abbau der Kommunikation zwischen den Abteilungen und nur einmaligem Erstellen der Skripte weniger fehleranfällig und effizienter als der momentan etablierte Prozess ist. Nachdem ein zeitaufwändiger Initialisierungsaufwand um die Erstellung eines Templates zu implementieren erbracht wurde, ermöglicht schon die Lösung mit z/OSMF eine Provisionierung einer anwendungsspezifischen Laufzeitumgebung binnen circa einer Stunde. Jedoch

ist es noch nicht perfekt. Der Bereitstellungsprozess ist noch immer mit einigen manuellen Schritten verbunden. So muss das Template manuell kopiert werden und Änderungen an der Konfiguration müssen innerhalb des Templates stattfinden.

Hierfür wurde in der Arbeit eine Lösung mit Hilfe von z/OSPT 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 z/OSPT das Einbinden in den DATEV e.G. internen „Marktplatz“ für Cloud Lösungen ermöglichen. Um diese Ziele zu erreichen, muss noch viel Aufwand in die Gestaltung des Templates gesteckt und Best Practices erarbeitet werden. Zusätzlich müsste ein sogenannter „Service Broker“ für die Einbindung der einzelnen Subsysteme in den „Marktplatz“ implementiert werden. z/OSPT ist dadurch dem cloud native Prozess näher als z/OSMF und ermöglicht die Provisionierung einer anwendungsspezifischen Laufzeitumgebung binnen weniger Minuten.

Beide Lösungsansätze erzeugen bei den Stakeholdern, also den Entwicklerteams und den Administratorenteams einen Mehrwert und werden akzeptiert. Hier wird vor allem aus Sicht des Entwicklerteams an Effizienz gewonnen, da im Vergleich zu den circa zwei Tagen im bestcase des etablierten Bereitstellungsprozesses, circa eine Stunde mit z/OSMF und wenige Minuten mit z/OSPT eine deutlich Zeiteinsparung nach sich zieht. Aus Sicht der Administratorenteams ist der hohe Initialisierungsaufwand für die erste Erstellung der Templates abschreckend, danach sinken die notwendigen Absprachen mit den Kollegen deutlich und erhöhen auch hier die Effizienz. Allgemein lässt sich sagen, dass die Nutzung von „IBM Cloud Provisioning and Management for z/OS“ ein kleiner Baustein in Richtung von DevOps und einer CI/CD-Pipeline für den Mainframe sein kann. Dieser Baustein bezieht sich nur auf den Aspekt von automatisierter Bereitstellung von isolierten Testumgebungen. Dieser Aspekt ist jedoch essenziell für den Einsatz einer Build Pipeline. So kann dieser kleine aber wichtige Baustein zu einer besseren Entwicklungseffizienz beitragen. Außerdem hilft dieser Schritt, um dem Image eines veralteten Systems mit veralteten langsamen Prozessen zu entkommen.

Anhang A.

Anhang

A.1. Verwendete Versionen der z/OS Komponenten im Rahmen dieser Arbeit

- CICS Transaction Server 5.4
- IBM MQ Version 9.1.0.0
- Db2 v11 item z/OS 2.3 (inkl. IBM Cloud Provisioning and Management for z/OS, z/OSMF, z/OSPT)

A.2. Agenda der neunzehnten Acadamic Mainframe Consortium e.V. Tagung vom 16.01.2020 bis 17.01.2020

IBM-Tag am Donnerstag, 16.01.2020

Die Sprecher werden erst im Januar festgelegt

10:00 – 10:20 Uhr
Begrüßung Wolfram Greis, AMC
Yvette A LaMar, Director, IBM Z Influencer Ecosystem
Roland Trauner, IBM System Z Academic Initiative, Europe

10.20 – 11:00 Uhr
IBM Z15 News / Update Roland Trauner
IBM System Z Academic Initiative, Europe

11.00 – 12:30 Uhr
Linux Container on IBM Z and LinuxONE Wilhelm Mild
IBM Executive IT Architect - Integration Architectures for Mobile, IBM Z and Linux
Yulia Gaponenko, Software Developer

12:30 – 13:30 Uhr
Mittagspause

13:30 – 14:30 Uhr
Containers for zOS, Applications and Container Orchestration **Wilhelm Mild, IBM**
*** Kubernetes / OpenShift**

14:30 – 15:00 Uhr
Middleware Provisionierung mit zOSMF - eine Bachelorarbeit David Krug, DATEV eG

15:00 – 15:15 Uhr
Pause

15:15 – 16:00 Uhr
HyperProtect Update Stefan Liesche
IBM Distinguished Engineer - IBM Hyper Protect Services
Stefan Schmitt
STSM Hyper Protect Services

16:00 – 16:30 Uhr
Offene Punkte, Feedback, weitere Planung NN, IBM & Wolfram Greis, AMC

16:30 – 17:30 Uhr
History@IBM oder Chiptest Lab Fläche (wahlweise) für alle Interessierten

Ab 18:00
Netzwerken im IBM Clubheim für alle Interessierten

Agenda für die Tagung des Academic Mainframe Consortium e.V.

am 16./17.01.2020

AMC-Tagung am Freitag, 17.01.2020

10:00 – 10:15 Uhr **Begrüßung und Vorstellungsrunde**	Wolfram Greis, AMC
10:15 – 10:45 Uhr **IBM System Z Academic Initiative 2020**	Roland Trauner, IBM
10:45 –16:00 Uhr **News vom Academic Mainframe Consortium**	Wolfram Greis, AMC
Berichte und Diskussionen zu den Arbeitsgruppen	Arbeitsgruppenleiter
Verschiedenes	Alle
Weiteres Vorgehen / Nächstes Treffen	Wolfram Greis, AMC

Die Reihenfolge der Punkte ist noch nicht endgültig festgelegt

A.3. IT workload distribution worldwide in 2018 and 2020, by cloud type

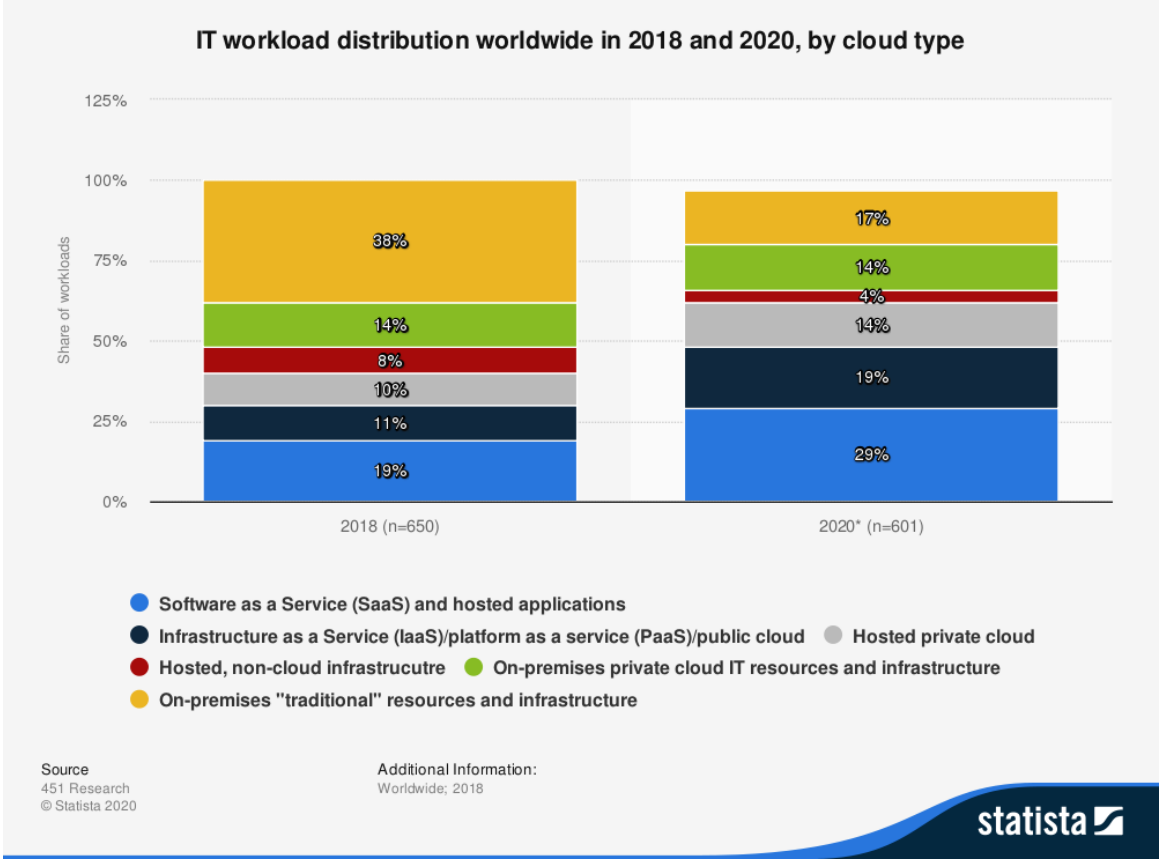


Abbildung A.1.: Weltweiter IT Workload im Jahr 2018 und als Vorhersage im Jahr 2020 bei Cloudtyp

A.4. Code createCICS.jcl

```
1  ##*************************************************************
2  ##Velocity macro to
3  ##validate SIT parameters. The following SIT parms
4  ##will be commented out if the user supplies them
5  ##    APPLID
6  ##    CICSSVC
7  ##    CPSMCONN
8  ##    CSDACC
9  ##    CSDBKUP
10 ##    CSDRECOV
11 ##    CSDRLS
```

```
12 ##    DFLTUSER
13 ##    JVMPROFILEDIR
14 ##    KEYRING
15 ##    RLS (when needed for shared CSD)
16 ##    SEC
17 ##    SECPRFX
18 ##    SIT
19 ##    SYSIDNT
20 ##    TCPIP
21 ##    USSHOME
22 ##    USSCONFIG
23 ##    XAPPC
24 ##    XCMD
25 ##    XDB2
26 ##    XDCT
27 ##    XFCT
28 ##    XHFS
29 ##    XJCT
30 ##    XPCT
31 ##    XPPT
32 ##    XPSB
33 ##    XPTKT
34 ##    XRES
35 ##    XTRAN
36 ##    XTST
37 ##    XUSER
38 ##*************************************************************
39 #macro(validateSit $sit)
40 #if($sit.toUpperCase().startsWith("APPLID") ||
41     $sit.toUpperCase().startsWith("CICSSVC") ||
42     $sit.toUpperCase().startsWith("CPSMCONN") ||
43     $sit.toUpperCase().startsWith("CSDACC") ||
44     $sit.toUpperCase().startsWith("CSDRLS") ||
45     ($sit.toUpperCase().startsWith("CSDRECOV")
46     && $!{instance-DFH_REGION_LOGSTREAM} == "DUMMY") ||
47     ($sit.toUpperCase().startsWith("CSDBKUP")
48     && $!{instance-DFH_REGION_LOGSTREAM} == "DUMMY") ||
49     $sit.toUpperCase().startsWith("DFLTUSER") ||
50     $sit.toUpperCase().startsWith("JVMPROFILEDIR") ||
51     $sit.toUpperCase().startsWith("KEYRING") ||
```

```
52      ($sit.toUpperCase().matches("^RLS[^A-Z].*$")
53     && $!{instance-DFH_REGION_CSD_TYPE} == "SHAREDRLS") ||
54      $sit.toUpperCase().startsWith("SEC") ||
55      $sit.toUpperCase().startsWith("SECPRFX") ||
56      $sit.toUpperCase().startsWith("SIT") ||
57      $sit.toUpperCase().startsWith("SYSIDNT") ||
58      $sit.toUpperCase().startsWith("TCPIP") ||
59      $sit.toUpperCase().startsWith("USSHOME") ||
60      $sit.toUpperCase().startsWith("USSCONFIG") ||
61      $sit.toUpperCase().startsWith("XAPPC") ||
62      $sit.toUpperCase().startsWith("XCMD") ||
63      $sit.toUpperCase().startsWith("XDB2") ||
64      $sit.toUpperCase().startsWith("XDCT") ||
65      $sit.toUpperCase().startsWith("XFCT") ||
66      $sit.toUpperCase().startsWith("XHFS") ||
67      $sit.toUpperCase().startsWith("XJCT") ||
68      $sit.toUpperCase().startsWith("XPCT") ||
69      $sit.toUpperCase().startsWith("XPPT") ||
70      $sit.toUpperCase().startsWith("XPSB") ||
71      $sit.toUpperCase().startsWith("XRES") ||
72      $sit.toUpperCase().startsWith("XTRAN") ||
73      $sit.toUpperCase().startsWith("XTST") ||
74      $sit.toUpperCase().startsWith("XUSER"))
75  ## Comment out the SIT
76  *The following SIT parameter is commented out as it clashes with
77  *a property in the provisioning template.
78  *$sit
79  #elseif($sit.toUpperCase().startsWith("GRPLIST") &&
80    !$sit.toUpperCase().contains("${csdlist}"))
81  ## Take what they have and append after CICS region specific list
82  ## Obtain the data after the =
83  #set ($index = $sit.indexOf('='))
84  #set ($index = $index + 1)
85  #set ($sitvalue = $sit.substring($index))
86  ## Check the first char if its ( then more work is required
87  #if($sitvalue.charAt(0) == "(")
88  #set ($endsit = $sitvalue.length() - 1)
89  #set ($sitvalue = $sitvalue.substring(1,$endsit))
90  #end
91  #if($sitvalue.toUpperCase().startsWith("DFHLIST,"))
```

```
92  #set ($sitvalue = $sitvalue.substring(8))
93  *The list DFHLIST has been removed from the provided GRPLIST
94  *specified in the DFH_REGION_SITPARMS
95  *property. It is not required.
96  *The groups from DFHLIST have been added to the list ${csdlist}.
97  #end
98  #if(!$sitvalue.toUpperCase().equals("DFHLIST"))
99  GRPLIST=(${csdlist},$sitvalue)
100 #else
101 *The GRPLIST SIT parameter only contains DFHLIST.
102 *This list is not
103 *required. The groups from DFHLIST have been added to the list
104 *${csdlist}.
105 *$sit
106 #end
107 ## End of GRPLIST processing
108 #else
109 ## Output the SIT as is
110 $sit
111 #end
112 #end
113 ##*************************************************************
114 #macro(validateEyuparm $eyuparm)
115 #if($eyuparm.toUpperCase().startsWith("CMASSYSID") ||
116     $eyuparm.toUpperCase().startsWith("NAME") ||
117     $eyuparm.toUpperCase().startsWith("CICSPLEX"))
118 ## Comment out the EYUPARM
119 *The following EYU parameter is commented out as it clashes with
120 *a property in the provisioning template.
121 *$eyuparm
122 #else
123 ## Output the EYUPARM as is
124 $eyuparm
125 #end
126 #end
127 ##*************************************************************
128 //*************************************************************
129 //**            CICS START PROCEDURE                           *
130 //**            YYYY                                           *
131 //*************************************************************
```

```
132 #set ($applid = "${instance-DFH_REGION_APPLID}")
133 ## csdlist auf leer gesetzt um groessere Aenderungen zu vermeiden
134 #set ($csdlist = "")
135 //COPY1 EXEC PGM=IEBGENER,MEMLIMIT=0M
136 //SYSUT2 DD DISP=SHR,
137 //         DSN=
138 //      ${instance-DFH_ZOS_PROCLIB}(${instance-DFH_REGION_APPLID})
139 //SYSPRINT DD SYSOUT=*
140 //SYSIN  DD *
141 //SYSUT1 DD DATA,DLM='@@'
142 //DFHSTART  PROC CICSAPPL='${applid}',
143 //               CICSREL='TS54',
144 //               CPSMREL='TS54',
145 //               START='AUTO',
146 //               PARMLIB='CICS.STARTUP.PARM'
147 //**
148 //**
149 //RMUTL     EXEC PGM=DFHRMUTL,PARM='SYSIN',REGION=1M
150 //**
151 //STEPLIB   DD   DISP=SHR,DSN=CICS.&CICSREL..SDFHLOAD
152 //SYSPRINT DD   SYSOUT=M
153 //DFHGCD    DD   DISP=SHR,DSN=TCICS.&CICSAPPL..DFHGCD
154 //SYSIN     DD   DISP=SHR,DSN=&PARMLIB(&START)
155 //**
156 //**
157 //CICS      EXEC PGM=DFHSIP,PARM='SYSIN',REGION=0M,
158 //             MEMLIMIT=50G
159 //**
160 //** Include STEPLIB Libraries ******************************
161 #set($stladded = 0)
162 #set ($value1 = $!{instance-DFH_REGION_STEPLIB})
163 #if($value1 != "")
164 #foreach( $dataset in $value1.split(","))
165 #if($stladded == 0)
166 //STEPLIB  DD DISP=SHR,DSN=$dataset.trim()
167 #set($stladded = 1)
168 #else
169 //         DD DISP=SHR,DSN=$dataset.trim()
170 #end
171 #end
```

```
172 #end
173 //**
174 //**CICS DATASETS
175 //**
176 //** Include the DFHRPL Libraries **********************
177 #set($rpladded = 0)
178 #set ($value2 = $!{instance-DFH_REGION_RPL})
179 #if($value2 != "")
180 #foreach( $dataset in $value2.split(","))
181 #if($rpladded == 0)
182 //DFHRPL   DD DISP=SHR,DSN=$dataset.trim()
183 #set($rpladded = 1)
184 #else
185 //         DD DISP=SHR,DSN=$dataset.trim()
186 #end
187 #end
188 #end
189 //**
190 //**          DFHRPL-USER-DATEIEN WERDEN DYNAMISCH VERKETTET
191 //**
192 //**CICS PARAMETERS
193 //**
194 //EYUPARM  DD  DISP=SHR,
195 //                DSN=CICS.&CPSMREL..CPSM.SEYUPARM.TEST(TESTPLEX)
196 //DFHINTRA DD  DISP=SHR,DSN=TCICS.&CICSAPPL..DFHINTRA
197 //DFHTEMP  DD  DISP=SHR,DSN=TCICS.&CICSAPPL..DFHTEMP
198 //DFHDMPA  DD  DISP=SHR,DSN=TCICS.&CICSAPPL..DFHDMPA
199 //DFHDMPB  DD  DISP=SHR,DSN=TCICS.&CICSAPPL..DFHDMPB
200 //DFHGCD   DD  DISP=SHR,DSN=TCICS.&CICSAPPL..DFHGCD
201 //DFHLCD   DD  DISP=SHR,DSN=TCICS.&CICSAPPL..DFHLCD
202 //DFHLRQ   DD  DISP=SHR,DSN=TCICS.&CICSAPPL..DFHLRQ
203 //CIGSSUBM DD  SYSOUT=(A,INTRDR)
204 //**
205 //**EXTRAPARTION DATA SETS
206 //**
207 //LOGUSR   DD  SYSOUT=M
208 //MSGUSR   DD  SYSOUT=M
209 //AUDITLOG DD  SYSOUT=M
210 //JVMOUT   DD  SYSOUT=M                          * JVMSERVER STDOUT
211 //JVMERR   DD  SYSOUT=M                          * JVMSERVER STDERR
```

```
212 //JVMTRACE DD  SYSOUT=M                          * JVMSERVER TRACE
213 //SYSUDUMP DD  DUMMY
214 //SYSABEND DD  DUMMY
215 //TRACEOUT DD  SYSOUT=M
216 //PRINTER  DD  SYSOUT=M
217 //DFHAUXT  DD  DUMMY
218 //DFHBUXT  DD  DUMMY
219 //COUT     DD  DUMMY
220 //DFHCXRF  DD  SYSOUT=M
221 //**
222 //**LE/370-DESTINATIONS
223 //**
224 //CEEMSG   DD  SYSOUT=M
225 //CEEOUT   DD  SYSOUT=M
226 //**
227 //**CICS REXX DATASETS
228 //**
229 //CICAUTH  DD  DISP=SHR,DSN=CICS.&CICSREL..REXX.SCICCMDS.TEST
230 //CICEXEC  DD  DISP=SHR,DSN=CICS.REXX.EXEC.TEST
231 //           DD  DISP=SHR,DSN=CICS.&CICSREL..REXX.SCICEXEC.TEST
232 //CICUSER  DD  DISP=SHR,DSN=CICS.&CICSREL..REXX.SCICUSER.TEST
233 //           DD  DISP=SHR,DSN=CICS.&CICSREL..REXX.SCICPNL.TEST
234 //**
235 //**DEBUGTOOL
236 //**
237 //EQADPFMB DD  DISP=SHR,DSN=CICS.DEBUG.PROFILES.&CICSAPPL
238 //**-------------------------------------------------------------------
239 //**END OF CICS-START PROCEDURE
240 //**-------------------------------------------------------------------
241 //SYSIN    DD *
242 * Abend the region if any SIT parameters are invalid
243 PARMERR=ABEND
244 ************************************************************
245 * The SIT parameters in this first section are auto-generated *
246 * based on the configured provisioning template properties    *
247 * GRPLIST=(${csdlist}) unter CICSSVC rausgenommen wegen csdlist*
248 ************************************************************
249 SIT=${instance-DFH_REGION_SIT}
250 #if(${!instance-DFH_REGION_CICSSVC}
251 # && ${!instance-DFH_REGION_CICSSVC} !="")
```

```
252 CICSSVC=${instance-DFH_REGION_CICSSVC}
253 #end
254 #if(${!instance-DFH_ZFS_MOUNTPOINT}
255 # && ${!instance-DFH_ZFS_MOUNTPOINT} != "")
256 JVMPROFILEDIR=${instance-DFH_ZFS_MOUNTPOINT}/
257 ${instance-DFH_REGION_APPLID}/JVMProfiles
258 #end
259 #if(${!instance-DFH_REGION_LOGSTREAM} == "DUMMY")
260 START=INITIAL
261 #else
262 START=AUTO
263 #end
264 #if(${!instance-DFH_REGION_CSD_TYPE} == "NORLSREADWRITE")
265 CSDRLS=NO
266 #else
267 #if(${!instance-DFH_REGION_CSD_TYPE} == "SHAREDRLS")
268 CSDRLS=YES
269 #else
270 CSDRLS=NO
271 #if(${!instance-DFH_REGION_LOGSTREAM} == "DUMMY")
272 CSDRECOV=NONE
273 CSDBKUP=STATIC
274 #end
275 #end
276 #end
277 #if(${!instance-DFH_REGION_CSD_TYPE} == "SHAREDREADONLY")
278 CSDACC=READONLY
279 #else
280 CSDACC=READWRITE
281 #end
282 TRANISO=YES
283 IRCSTRT=YES
284 ISC=YES
285 APPLID=${instance-DFH_REGION_APPLID}
286 SYSIDNT=${instance-DFH_REGION_SYSIDNT}
287 DFLTUSER=${instance-DFH_REGION_DFLTUSER}
288 USSHOME=${instance-DFH_CICS_USSHOME}
289 USSCONFIG=${instance-DFH_CICS_USSCONFIG}
290 TCPIP=YES
291 ## Security Settings
```

```
292 #if(${instance-DFH_REGION_SEC} == "YES")
293 SEC=YES
294 #if($!{instance-DFH_REGION_XAPPC}
295 # && $!{instance-DFH_REGION_XAPPC} != "")
296 XAPPC=${instance-DFH_REGION_XAPPC}
297 #end
298 #if($!{instance-DFH_REGION_XCMD}
299 # && $!{instance-DFH_REGION_XCMD} != "")
300 XCMD=${instance-DFH_REGION_XCMD}
301 #end
302 #if($!{instance-DFH_REGION_XDB2}
303 # && $!{instance-DFH_REGION_XDB2} != "")
304 XDB2=${instance-DFH_REGION_XDB2}
305 #end
306 #if($!{instance-DFH_REGION_XDCT}
307 # && $!{instance-DFH_REGION_XDCT} != "")
308 XDCT=${instance-DFH_REGION_XDCT}
309 #end
310 #if($!{instance-DFH_REGION_XFCT}
311 # && $!{instance-DFH_REGION_XFCT} != "")
312 XFCT=${instance-DFH_REGION_XFCT}
313 #end
314 #if($!{instance-DFH_REGION_XHFS}
315 # && $!{instance-DFH_REGION_XHFS} != "")
316 XHFS=${instance-DFH_REGION_XHFS}
317 #end
318 #if($!{instance-DFH_REGION_XJCT}
319 # && $!{instance-DFH_REGION_XJCT} != "")
320 XJCT=${instance-DFH_REGION_XJCT}
321 #end
322 #if($!{instance-DFH_REGION_XPCT}
323 # && $!{instance-DFH_REGION_XPCT} != "")
324 XPCT=${instance-DFH_REGION_XPCT}
325 #end
326 #if($!{instance-DFH_REGION_XPPT}
327 # && $!{instance-DFH_REGION_XPPT} != "")
328 XPPT=${instance-DFH_REGION_XPPT}
329 #end
330 #if($!{instance-DFH_REGION_XPSB}
331 # && $!{instance-DFH_REGION_XPSB} != "")
```

```
332 XPSB=${instance.DFH_REGION_XPSB}
333 #end
334 #if(${!instance.DFH_REGION_XPTKT}
335 # && ${!instance.DFH_REGION_XPTKT} != "")
336 XPTKT=${instance.DFH_REGION_XPTKT}
337 #end
338 #if(${!instance.DFH_REGION_XRES}
339 # && ${!instance.DFH_REGION_XRES} != "")
340 XRES=${instance.DFH_REGION_XRES}
341 #end
342 #if(${!instance.DFH_REGION_XTRAN}
343 # && ${!instance.DFH_REGION_XTRAN} != "")
344 XTRAN=${instance.DFH_REGION_XTRAN}
345 #end
346 #if(${!instance.DFH_REGION_XTST}
347 # && ${!instance.DFH_REGION_XTST} != "")
348 XTST=${instance.DFH_REGION_XTST}
349 #end
350 #if(${!instance.DFH_REGION_XUSER}
351 # && ${!instance.DFH_REGION_XUSER} != "")
352 XUSER=${instance.DFH_REGION_XUSER}
353 #end
354 #else
355 SEC=NO
356 #end
357 ##
358 ## Is an MQCONN required
359 ##
360 #if(${!instance.DFH_MQ_SSID} && ${!instance.DFH_MQ_SSID} != "")
361 MQCONN=YES
362 #end
363 ## Is a DB2CONN required
364 ##
365 #if(${!instance.DFH_DB2_HLQ} && ${!instance.DFH_DB2_HLQ} != "")
366 DB2CONN=YES
367 #end
368 ##
369 ## Setup the SIT PARAMETERS
370 ##
371 ************************************************************
```

```
372 * The SIT parameters below this line were provided by the      *
373 * DFH_REGION_SITPARMS property of the provisioning template.   *
374 * Some SIT parameters are not allowed to be specified in this  *
375 * property, as they would overwrite the auto-generated         *
376 * properties above. Those properties are provided below but    *
377 * are commented out.                                           *
378 ***************************************************************
379 #set ($value5 = $!{instance-DFH_REGION_SITPARMS})
380 #set ($multipart = "NO")
381 #set ($tempStr = "")
382 #if($value5 != "")
383 #foreach( $sit in $value5.split(","))
384 #if($multipart == "YES")
385 #if( $sit.indexOf(')') > 0 )
386 ## Validate SIT
387 #validateSit($tempStr.concat($sit.trim()))
388 #set ($multipart = "NO")
389 #else
390 #set ($tempStr = $tempStr + $sit.trim() + ",")
391 #end
392 #else
393 #if( $sit.indexOf('(') > 0 && $sit.indexOf(')') == -1 )
394 #set ($multipart = "YES")
395 #set ($tempStr = $sit.trim() + ",")
396 #else
397 #validateSit($sit.trim())
398 #end
399 #end
400 #end
401 #end
402 .END
403 /*
404 //
405 @@
406 /*
```

Listing A.1: createCICS.jcl JCL Quellcode, der mittels eines Workflowdefinitionfile-Steps in das Template aufgenommen wurde

A.5. Code defineQsRexx.jcl

```
1  //P$PRVMQ1 JOB (6709,00000,30193,00000),KRUG,
2  //              CLASS=V,MSGCLASS=V,TIME=(4,3),
3  //              MSGLEVEL=(1,1),REGION=0M,
4  //              USER=TU11998,NOTIFY=${_step-stepOwnerUpper}
5  //*************************************************************
6  //*          COPY STEP - IEBGENER -                           *
7  //*************************************************************
8  //*
9  //GENER    EXEC PGM=IEBGENER
10 //*
11 //SYSPRINT DD SYSOUT=*
12 //*
13 //SYSIN    DD DUMMY
14 //*
15 //SYSUT2   DD DISP=(NEW,PASS),DSN=&&TMP(TSOREXX),
16 //            SPACE=(1024,(1000,500,1))
17 //*
18 //SYSUT1  DD *,DLM=$$
19 /*REXX*/
20 /*************************************************************/
21 /* Erzeugen AWTB-Queues aus Template (c)NorbertPfister 05/2012 */
22 /* Ermitteln QSGDISP Template Anpassung NorbertPfister 12/2013 */
23 /* Wartezeit sleep eingebaut          Norbert Pfister 08/2014  */
24 /*                                    ..............  MM/20JJ  */
25 /*************************************************************/
26 /*  Aufruf : "DEFQAWTB" +                                    */
27 /*             QM(mx00)                        |Ziel-QMGR     */
28 /*             QN(AWTB.xxxxxxxxxxxxx)          |Queue-Prefix  */
29 /*             ANZ(NNN)                        |Anzahl Queues */
30 /*                                                          */
31 /*  Erzeugen Queues per "DEFINE REPLACE QL(.) LIKE(.)" .     */
32 /*  REXX connectet mit ME1C und uebergibt                    */
33 /*  Commands an Ziel-QMGR QM(.)                              */
34 /*  Der uebergebene Queue-Prefix dient                       */
35 /*  1.) zum Bilden des Namens der LIKE-Queue: QN+".TEMPLATE" */
36 /*  2.) zum Bilden der neuen Queues: QN+".nnnnnn",           */
37 /*    eine sechstellige                                      */
38 /*    laufende Nummer beginnend mit 000001 bis Parameter ANZ . */
```

```
39 /*************************************************************/
40 /* ------------------------------------------------- */
41 /* interaktiv                                        */
42 /* ------------------------------------------------- */
43 target_qm='${instance-DFH_MQ_TARGET_QUEUEMANAGER}'
44 source_qm=${instance-DFH_MQ_SOURCE_QUEUEMANAGER}
45 q_hlq=${instance-DFH_MQ_HLQ}
46 #if(${instance-DFH_MQ_ANZAHL})
47 Loop=${instance-DFH_MQ_ANZAHL}
48 #else
49 Loop=03
50 #end
51
52 i=0
53 #set ($names = ${instance-DFH_MQ_QUEUENAMES})
54 #set ($tempStr = "")
55 #if($names != "")
56 #foreach( $queue in $names.split(","))
57 i=i+1
58 names.i="$queue"
59 #end
60 #end
61 names.0=i
62
63 /* ------------------------------------------------- */
64 /* Commands generieren                               */
65 /* ------------------------------------------------- */
66    k     = 0
67    rcci = RXMQVC('INIT')
68
69     Do i=1 to names.0
70
71       q_name=q_hlq||'.'||names.i||'.'
72        select
73         when names.i="L1.GPNRBERINFO" then do
74         maxdepth=150000
75         maxlength=128
76         end
77         when names.i="L1.KLAMMERINFOLIST" then do
78         maxdepth=10000
```

```
79          maxlength=100
80          end
81          when names.i="L1.KUNDENPREISINFOLIST" then do
82          maxdepth=150000
83          maxlength=1300
84          end
85          when names.i="L1.PABHREFERENZLIST" then do
86          maxdepth=100000
87          maxlength=100
88          end
89          otherwise do
90          maxdepth=100
91          maxlength=100
92          end
93        end
94        rccg = generate()
95
96      End
97
98     rcct = RXMQVC('TERM')
99
100    csqout.0 = k
101
102    /* aus Batchjob ? */
103    if datatype(Loop,'N') then do
104       do ix = 1 to k
105          say csqout.ix
106       end
107    end
108    /* interaktiv   ? */
109    else
110       address $datev "browse stem csqout. 200"
111
112    Exit rccg
113 /* ---------------------------------------------------------------------*/
114 generate:
115 /* ------------------------------------ */
116 /* Template abfragen                    */
117 /* ------------------------------------ */
118    if rcg > 0 then return rcg
```

```
119
120     Do l=1      to Loop
121
122         q_new= q_name||Right(l,6,'0')
123         command    = "DEFINE REPLACE QL("q_new")",
124                      "MAXDEPTH("maxdepth")",
125                      "NOTRIGGER",
126                      "MAXMSGL("maxlength")",
127                      "SHARE",
128                      "DEFSOPT(SHARED)",
129                      "QDEPTHHI(80)",
130                      "QDEPTHLO(40)"
131         /* ---------------------------------------- */
132         /* alle 10 Schleifen kurz warten            */
133         /* ---------------------------------------- */
134         if l//10 = 0 then ,
135            Call Wait 1
136         cmd_qm=target_qm
137         rcg = command_send()
138         if rcg > 0 then leave
139
140         rcg = q_test()
141         if rcg > 0 then leave
142
143     End
144 return rcg
145 /* ------------------------------------------------------------------------*/
146 command_send:
147 /* ---------------------------------------- */
148 /* execute commands (only on mainframe) */
149 /* ---------------------------------------- */
150 k=k+1;csqout.k = Time('l') command
151 CONN.QM    = "ME1C"
152 CONN.CQ    = "ADMIN.PCF."cmd_qm
153 CONN.RQ    = "SYSTEM.DEFAULT.MODEL.QUEUE"
154 CONN.TO    = 5000
155 rcc = RXMQVC('COMMAND', 'CONN.', command, 'Reply.')
156 rcc_RXMQVC = word(rcc,1)
157 rcc_QMGR   = word(rcc,2)
158 rcc_REASON = word(rcc,3)
```

```
159
160 do fw = 1 to Reply.0
161    k=k+1;csqout.k = Reply.fw
162 end
163
164 if (rcc_RXMQVC <> 0) | (rcc_QMGR <> 0) then do
165   k=k+1;csqout.k = ,
166        'Command' command 'zu' CONN.QM 'fehlgeschlagen:' rcc
167   return 99
168 end
169 return 0
170 /* ---------------------------------------------------------------- */
171 Wait: Procedure
172    call syscalls 'ON'
173    Address syscall 'sleep 'Arg(1)
174    return 0
175 /* ---------------------------------------------------------------- */
176 q_test:
177 /* ------------------------------------- */
178 /* einfacher Put und Get auf Queue      */
179 /* ------------------------------------- */
180 /*---- OPEN ----*/
181 say RXMQCONN( target_qm )
182 oo = mqoo_inquire+mqoo_output+mqoo_input_as_q_def
183 say RXMQOPEN(q_new, oo, 'hobj', 'ood.')
184 /*---- PUT ----*/
185 d.1       = q_new||' Put Get OK'
186 d.0       = Length(d.1)
187 imd.PER   = MQPER_PERSISTENT
188 ipmo.opt = MQPMO_SYNCPOINT  + MQPMO_LOGICAL_ORDER
189 ipmo.VER = 2
190 imd.ver   = MQMD_CURRENT_VERSION
191 imd.form = MQFMT_STRING
192 imd.MF    = MQMF_MSG_IN_GROUP
193 rcc = RXMQPUT( hobj ,'d.', 'imd.', 'omd.', 'ipmo.', 'opmo.')
194 rcc_RXMQPUT = word(rcc,1)
195 rcc_QMGR    = word(rcc,2)
196 rcc_REASON = word(rcc,3)
197
198 if (rcc_RXMQPUT <> 0 | rcc_QMGR <> 0) then do
```

```
199   k=k+1;csqout.k=
200       'RXMQPUT ist Fehlgeschlagen: ' rcc
201   return 98
202 end
203 /*---- GET ----*/
204 immd.ver    = 2
205 immo.opt = MQGMO_FAIL_IF_QUIESCING +,
206              MQGMO_WAIT,
207              MQGMO_ALL_MSGS_AVAILABLE +,
208              MQGMO_LOGICAL_ORDER+,
209              MQGMO_VERSION_2
210 data.0      = 300
211 data.1      = ''
212
213 rcc = RXMQGET(Hobj,'data.','immd.','ommd.','immo.','ommo.')
214 rcc_RXMQGET = word(rcc,1)
215 rcc_QMGR    = word(rcc,2)
216 rcc_REASON = word(rcc,3)
217
218 if (rcc_RXMQGET <> 0 | rcc_QMGR <> 0) then do
219   k=k+1;csqout.k=
220       'RXMQGET ist Fehlgeschlagen: ' rcc
221   return 97
222  end
223 else say data.1
224 /*---- CLOSE ----*/
225 say RXMQCLOS(hobj,mqco_none)
226 say RXMQDISC()
227 return 0
228 $$
229 //*
230 //*****************************************************************
231 //*          RUN  STEP - IKJEFT01 -                               *
232 //*****************************************************************
233 //*
234 //RUNIT     EXEC PGM=IKJEFT01
235 //*
236 //SYSEXEC   DD DISP=(OLD,DELETE),DSN=&&TMP
237 //*
238 //SYSTSPRT DD SYSOUT=*
```

```
239 //*
240 //SYSTSIN   DD *
241   %ISOREXX
242 /*
243 //*
244 //*****************************************************************
```

Listing A.2: defineQsRexx.jcl JCL Quellcode (beinhaltet Inline REXX Skript Zeile neunzehn bis 228), der mittels eines Workflowdefinitionfile-Steps in das Template aufgenommen wurde

A.6. Fragen an die IBM

Krug, David

Von: Krug, David
Gesendet: Freitag, 28. Februar 2020 10:36
An: 'Marcel Amrein'
Betreff: AW: Bachelorarbeit im Rahmen von IBM z/OS Cloud Provisioning

Hallo Herr Amrein,

vielen Dank für Ihre Antwort.

Mit freundlichem Gruß

David Krug
Accounting Datenmanagement und Archivservices | T1331

DATEV eG
Standort: Fürther Straße 111
Telefon: +49(911)319-42873
E-Mail: david.krug@datev.de
Postanschrift: DATEV eG, 90329 Nürnberg

-----Ursprüngliche Nachricht-----
Von: Marcel Amrein <marcel.amrein@de.ibm.com>
Gesendet: Donnerstag, 27. Februar 2020 17:21
An: Krug, David <David.Krug@DATEV.DE>
Cc: Tobias Leicher <tobias.leicher@de.ibm.com>
Betreff: Re: Bachelorarbeit im Rahmen von IBM z/OS Cloud Provisioning

Hallo Herr Krug,
gerne nehme ich zu Ihren Fragen kurz Stellung:

Ich fange mit der zweiten an:
>> 2. Was ist die Motivation der IBM, hier ein Tool bereitzustellen? />>

IBM ist dabei, den Mainframe Cloud-fähig zu machen. Dies bedeutet vor allem und zuerst, dass Mainframe Komponenten sich in eine - hybride, Plattform-neutrale bzw. -übergergreifende- Cloud-Architektur "nahtlos" einfügen lassen müssen, was "von Haus aus" nicht der Fall ist. Zu den ersten Ansätzen gehört dabei, Entwicklern das Provisionieren von Anwendungs-Runtime-Umgebungen in Eigenregie ("Self-Service") zu ermöglichen, und hierbei stehen CICS Regions als die wichtigste und am weitesten verbreitete traditionelle Mainframe-Runtime an vorderster Stelle.

die andere Frage:
>> 1. Wie ist aus Ihrer IBM Sicht und Ihre Erfahrungen mit z/OS Kunden die Einschätzung zum IBM z/OS Cloud Provisioning? />>

Ich persönlich habe in diesem Zusammenhang fast nur mir CICS Kunden zu tun,

und halte "CICS" auch für das interessanteste Anwendungsgebiet für Middleware Cloud Provisioning- das schlägt sich übrigens auch darin nieder, dass das CICS Entwicklungslabor die Führung bei der Gestaltung und Weiterentwicklung des "Cloud Provisioning Toolkits" übernommen hat, welches eine wichtige Zwischenschicht in der Gesamtarchitektur darstellt.
Traditionell sind die meisten IBM Mainframe Betreiber eher konservativ aufgestellt, und die Systemadministratoren, die über Verprobung und Einsatz dieses Tools befinden, sind diesem zunächst überwiegend skeptisch begegnet, "weil man so etwas bisher nicht vermisst hat und auch mit konventionellen Mitteln in sehr kurzer Zeit z.B. eine CICS Region neu erstellen kann".
Mittlerweile ist jedoch zu beobachten, dass in der Mehrzahl der Unternehmen, bei denen IBM Z als strategische Plattform gesetzt ist, starkes Interesse an der Cloud ProvisioningThematik besteht, weil nun eben die "Botschaft" angekommen ist, dass es darum geht, als Mainframe-Plattform im künftigen (Hybrid) Cloud Geschehen mitzuwirken - und nicht etwa darum, die Anforderungen der Vergangenheit mit einem neuen Tool zu bedienen,
Dies schlägt sich darin nieder, dass bei Kundenveranstaltungen wie zuletzt der Z Technical University (Mai 2019, Berlin) die Cloud ProvisioningThematik stark nachgefragt war, auch das entsprechende Hands-On Lab, welches ich bereit gestellt hatte, war gut gebucht, und seither haben verschiedene weitere Z Kunden begonnen, sich mit der Thematik zu beschäftigen.

Mit freundlichen Grüßen / Kind regards

Marcel Amrein

Senior Technical Sales Professional (MQ and CICS)
Certified IT Specialist
IBM Global Markets - Cloud Sales
B462

Phone: +49-7034-643 0038
Mobile: +49-160 901 77 458
Email: marcel.amrein@de.ibm.com

IBM Deutschland GmbH
IBM-Allee 1
71139 Ehningen
Germany

From: "Krug, David" <David.Krug@DATEV.DE>
To: "'marcel.amrein@de.ibm.com'" <marcel.amrein@de.ibm.com>
Date: 27/02/2020 15:42
Subject: [EXTERNAL] Bachelorarbeit im Rahmen von IBM z/OS Cloud Provisioning

Hallo Herr Amrein,

ich schreibe eine Bachelorarbeit mit dem Thema „Automatisierte Provisionierungsmechanismen für Laufzeitumgebungen von Legazy z/OS Anwendungen mit „IBM Cloud Provisioning and Managment for z/OS" am Beispiel der „Rechnungsschreibung" bei DATEV e.G.".
Dies bezüglich habe ich folgende Fragen an Sie:

1. Wie ist aus Ihrer IBM Sicht und Ihre Erfahrungen mit z/OS Kunden die Einschätzung zum IBM z/OS Cloud Provisioning?
2. Was ist die Motivation der IBM, hier ein Tool bereitzustellen?

Ich bedanke mich im Voraus für Ihre Antworten.

Mit freundlichem Gruß
David Krug
Accounting Datenmanagement und Archivservices | T1331
DATEV eG
Standort: Fürther Straße 111
Telefon: +49(911)319-42873
E-Mail: david.krug@datev.de
Postanschrift: DATEV eG, 90329 Nürnberg

Krug, David

Von: Krug, David
Gesendet: Freitag, 28. Februar 2020 10:36
An: 'Tobias Leicher'
Betreff: AW: Bachelorarbeit im Rahmen von IBM z/OS Cloud Provisioning

Hallo Herr Leicher,

vielen Dank für Ihre Antwort.

Mit freundlichem Gruß

David Krug
Accounting Datenmanagement und Archivservices | T1331

DATEV eG
Standort: Fürther Straße 111
Telefon: +49(911)319-42873
E-Mail: david.krug@datev.de
Postanschrift: DATEV eG, 90329 Nürnberg
-----Ursprüngliche Nachricht-----
Von: Tobias Leicher <tobias.leicher@de.ibm.com>
Gesendet: Donnerstag, 27. Februar 2020 17:02
An: Krug, David <David.Krug@DATEV.DE>
Betreff: Re: Bachelorarbeit im Rahmen von IBM z/OS Cloud Provisioning

Hallo Herr Krug,

Gerade im Umfeld von hochtransaktionalen Enterprise Systemen wie IBM Z sind Kunden weltweit sehr zurückhaltend Cloud Provisionierungsmodelle ein zu setzen. Da aber auch hier haben Entwickler in agilen Projekten einen hohen bedarf an automatisierter Bereitstellung von Umgebungen, nicht nur in einem Developer Marketplace, vor allem in automatisierten Test und Build Pipelines. Viele Kunden haben daher im Moment Interesse, jedoch warten viele hier auf die ersten Erfahrungen von anderen.

IBMs Strategie für Kunden ist eine hybride Multicloud und diese beinhaltet nicht nur x86 Systeme, sondern je nach Anwendungszweck die bestmögliche Hard- und Software. IBM Z ist für hochtransaktionale und hoch regulierte Aufgaben nach wie Vor IBMs premium Plattform und daher auch ein essezieller Teil unserer Cloud Strategie. IBM Z findet hierin sowohl als private Cloud Offering mit zOSMF, zospt und IBM Z Cloud Broker, als auch mit dem hyperprotect Portfolio in der public Cloud einen Platz.

Viele Grüße,
Tobias Leicher

Mit freundlichen Grüßen / Kind regards

Tobias Leicher

zClient Architect
zChampion for Modernization
Senior IT Specialist for CICS and zAPI
IBM Systems Software Technical Sales

Phone:
+49-151-15 16 24 89
IBM Deutschland

E-Mail:
tobias.leicher@de.ibm.com
IBM Allee 1

71139 Ehningen

Germany

From: "Krug, David" <David.Krug@DATEV.DE>
To: "'tobias.leicher@de.ibm.com'" <tobias.leicher@de.ibm.com>
Date: 27/02/2020 15:43
Subject: [EXTERNAL] Bachelorarbeit im Rahmen von IBM z/OS Cloud Provisioning

Hallo Herr Leicher,

ich schreibe eine Bachelorarbeit mit dem Thema ?Automatisierte Provisionierungsmechanismen für Laufzeitumgebungen von Legazy z/OS Anwendungen mit ?IBM Cloud Provisioning and Managment for z/OS? am Beispiel der ?Rechnungsschreibung? bei DATEV e.G.?.
Dies bezüglich habe ich folgende Fragen an Sie:

1. Wie ist aus Ihrer IBM Sicht und Ihre Erfahrungen mit z/OS Kunden die Einschätzung zum IBM z/OS Cloud Provisioning?
2. Was ist die Motivation der IBM, hier ein Tool bereitzustellen?

Ich bedanke mich im Voraus für Ihre Antworten.

Mit freundlichem Gruß
David Krug
Accounting Datenmanagement und Archivservices | T1331
DATEV eG
Standort: Fürther Straße 111
Telefon: +49(911)319-42873
E-Mail: david.krug@datev.de
Postanschrift: DATEV eG, 90329 Nürnberg

A.7. Interview Fragebögen

Vorlage

1. **Es wurde ein Template für die DATEV-Rechnungsschreibung, welches ein CICS, die benötigten IBM MQ Queues und theoretisch die benötigte Db2 Datenbanken beinhaltet, vorgestellt. Der Ablauf, der damit einhergeht, beschränkt sich zunächst auf z/OSMF. Bewerten Sie diesen, begründen Sie Ihre Bewertung.**

   1 2 3 4 5
   ☐ ☐ ☐ ☐ ☐

2. **Es wurde eine Ergänzung mit z/OSPT, zu oben genannten Ablauf, erläutert. Bewerten Sie diese, begründen Sie Ihre Bewertung.**

   1 2 3 4 5
   ☐ ☐ ☐ ☐ ☐

3. **Bewerten Sie folgende Punkte bezüglich der Benutzerfreundlichkeit der Oberfläche:**
   a. **Verwaltung der Templates in z/OSMF (nicht für Entwickler relevant)**

      1 2 3 4 5
      ☐ ☐ ☐ ☐ ☐

   b. **Verwaltung der Instanzen in z/OSMF (nicht für Entwickler relevant)**

      1 2 3 4 5
      ☐ ☐ ☐ ☐ ☐

4. **Bewerten Sie die gezeigte Arbeitsweise für Änderungen an den Workflow Definitionsdateien. (nicht für Entwickler relevant)**

   1 2 3 4 5
   ☐ ☐ ☐ ☐ ☐

5. **Wie ist Ihr erster persönlicher Eindruck zum Tool? (nicht für Entwickler relevant)**

6. **Wie würden Sie den aktuellen Bereitstellungsprozess beurteilen?**

7. **Können Sie sich vorstellen, mit dem Tool täglich zu arbeiten?**

8. **Wenn 7. Mit ja beantwortet wurde, begründen Sie ihre Meinung.**

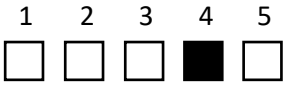
9. **Wenn 7. Mit nein beantwortet wurde, was müsste sich ändern, dass dem so wäre?**

10. **Freitext für sonstiges und Anmerkungen:**

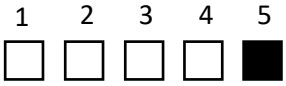
CICS Administrator 1

1. **Es wurde ein Template für die DATEV-Rechnungsschreibung, welches ein CICS, die benötigten IBM MQ Queues und theoretisch die benötigte Db2 Datenbanken beinhaltet, vorgestellt. Der Ablauf, der damit einhergeht, beschränkt sich zunächst auf z/OSMF. Bewerten Sie diesen, begründen Sie Ihre Bewertung.**

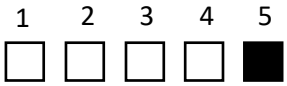
   + flexibel Versionierung und Publish
   - Startschwierigkeiten viele verschiedene Sprachen und Dokumentarten
2. **Es wurde eine Ergänzung mit z/OSPT, zu oben genannten Ablauf, erläutert. Bewerten Sie diese, begründen Sie Ihre Bewertung.**



   + APIfizierung
   + Container-Gedanke
   + konfigurierbar über PT-File von außerhalb der Templates
   -Template muss sehr dynamisch sein
3. **Bewerten Sie folgende Punkte bezüglich der Benutzerfreundlichkeit der Oberfläche:**
   a. **Verwaltung der Templates in z/OSMF (nicht für Entwickler relevant)**



   b. **Verwaltung der Instanzen in z/OSMF (nicht für Entwickler relevant)**

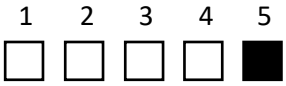


4. **Bewerten Sie die gezeigte Arbeitsweise für Änderungen an den Workflow Definitionsdateien. (nicht für Entwickler relevant)**

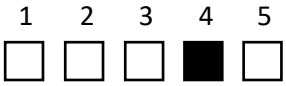
   + lediglich editieren der Files
   -Fehlendes Highlighting für „mixed JCL“
5. **Wie ist Ihr erster persönlicher Eindruck zum Tool? (nicht für Entwickler relevant)**
   - Hoher Ersteinrichtungsaufwand
   - Einarbeitung
   - Abschreckende Wirkung (verschiedene Sprachen usw.)

CICS Administrator 2

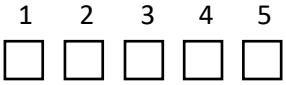
1. **Es wurde ein Template für die DATEV-Rechnungsschreibung, welches ein CICS, die benötigten IBM MQ Queues und theoretisch die benötigte Db2 Datenbanken beinhaltet, vorgestellt. Der Ablauf, der damit einhergeht, beschränkt sich zunächst auf z/OSMF. Bewerten Sie diesen, begründen Sie Ihre Bewertung.**



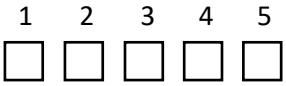
   Das Template ist aus CICS-Sicht ablauffähig und mehrfach einsetzbar.
2. **Es wurde eine Ergänzung mit z/OSPT, zu oben genannten Ablauf, erläutert. Bewerten Sie diese, begründen Sie Ihre Bewertung.**



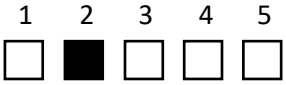
   Großer Vorteil ist die Flexibilität durch den Einsatz von Variablen. Nachteil ist die damit verbundene Komplexität des dahinterliegenden Templates.
3. **Bewerten Sie folgende Punkte bezüglich der Benutzerfreundlichkeit der Oberfläche:**
   Die Oberfläche wurde vom CICS-Team bisher nicht genutzt, daher ist keine Bewertung möglich. Zudem ist wenig Erfahrung mit vergleichbaren Tools wie Cloud Foundry vorhanden.
   a. **Verwaltung der Templates in z/OSMF (nicht für Entwickler relevant)**



   b. **Verwaltung der Instanzen in z/OSMF (nicht für Entwickler relevant)**



4. **Bewerten Sie die gezeigte Arbeitsweise für Änderungen an den Workflow Definitionsdateien. (nicht für Entwickler relevant)**



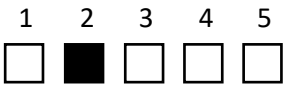
   Keine Toolunterstützung. Editieren per Notepad ohne Highlighting. Keine sprechenden Fehlermeldungen. Dokumentation der JCL-Skriptsprache nicht vorhanden.
5. **Wie ist Ihr erster persönlicher Eindruck zum Tool? (nicht für Entwickler relevant)**
   Prinzipiell kann mit dem Tool alles erreicht werden, allerdings ist sehr viel Anpassungsarbeit notwendig, um es auf die Firmengegebenheiten anzupassen. Hilfreich wären mehr Beispiele, bessere Dokumentation, Step by Step Anleitung oder ein Wizard.
6. **Wie würden Sie den aktuellen Bereitstellungsprozess beurteilen?**
   Hoher manueller Aufwand zu erbringen. Kein SelfService für den Entwickler vorhanden. Vorherige Abstimmung zwischen Sysprog und Entwicklung notwendig.
7. **Können Sie sich vorstellen, mit dem Tool täglich zu arbeiten?**
   Jein, man könnte es sich vorstellen, aber es gibt auch viele Hürden.

Db2 Administrator 1

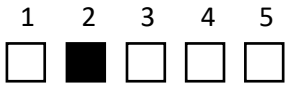
1. **Es wurde ein Template für die DATEV-Rechnungsschreibung, welches ein CICS, die benötigten IBM MQ Queues und theoretisch die benötigte Db2 Datenbanken beinhaltet, vorgestellt. Der Ablauf, der damit einhergeht, beschränkt sich zunächst auf z/OSMF. Bewerten Sie diesen, begründen Sie Ihre Bewertung.**

   Der Ablauf zeigt, dass eine Automatisierung möglich ist. Diese Erkenntnis ist sehr wertvoll. Um die Provisionierung aber wirklich nutzen zu können ist noch viel Weiterentwicklung notwendig.
2. **Es wurde eine Ergänzung mit z/OSPT, zu oben genannten Ablauf, erläutert. Bewerten Sie diese, begründen Sie Ihre Bewertung.**

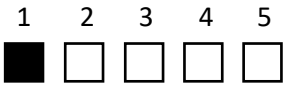
   Für den aktuellen Stand der Untersuchung halte ich z/OSPT noch nicht für relevant. In der Endausbaustufe (Automatisiertes Deployment innerhalb einer CI/DC-Pipeline z.B. mit Jenkins) wird ein CLI-Interface wie z/OSPT aber sehr wichtig und vereinfacht die Nutzung für Entwickler
3. **Bewerten Sie folgende Punkte bezüglich der Benutzerfreundlichkeit der Oberfläche:**
   a. **Verwaltung der Templates in z/OSMF (nicht für Entwickler relevant)**



   b. **Verwaltung der Instanzen in z/OSMF (nicht für Entwickler relevant)**



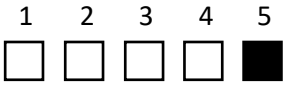
4. **Bewerten Sie die gezeigte Arbeitsweise für Änderungen an den Workflow Definitionsdateien. (nicht für Entwickler relevant)**



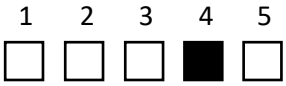
   (wenn es ohne automatisches Deployment gemacht wird)
5. **Wie ist Ihr erster persönlicher Eindruck zum Tool? (nicht für Entwickler relevant)**
   Mächtiges Tool aber auch sehr komplex
6. **Wie würden Sie den aktuellen Bereitstellungsprozess beurteilen?**
   Er funktioniert sehr gut aber man ist von anderen Personen abhängig und hat dadurch natürlich Wartezeiten, die man mit einem automatischen Prozess eliminieren würde.
7. **Können Sie sich vorstellen, mit dem Tool täglich zu arbeiten?**
   Ja
8. **Wenn 7. Mit ja beantwortet wurde, begründen Sie ihre Meinung.**
   Nur wenn man es kapselt (jenkins)

Db2 Administrator 2

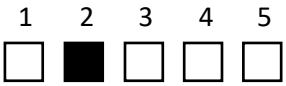
1. **Es wurde ein Template für die DATEV-Rechnungsschreibung, welches ein CICS, die benötigten IBM MQ Queues und theoretisch die benötigte Db2 Datenbanken beinhaltet, vorgestellt. Der Ablauf, der damit einhergeht, beschränkt sich zunächst auf z/OSMF. Bewerten Sie diesen, begründen Sie Ihre Bewertung.**



   Ablauf ist für mich nachvollziehbar. Siehe auch Anmerkungen (Frage 10)
2. **Es wurde eine Ergänzung mit z/OSPT, zu oben genannten Ablauf, erläutert. Bewerten Sie diese, begründen Sie Ihre Bewertung.**



   Die Nutzung von Z/OSPT macht Sinn, denn die Commands (z.B. build und run) sind meiner Meinung nach einfach in einen Quellcode zu integrieren und geläufig. Problematisch sehe ich jedoch die Verwendung der Begriffe Container und Image, da hier Begriffe vertauscht und synonym verwendet werden.
3. **Bewerten Sie folgende Punkte bezüglich der Benutzerfreundlichkeit der Oberfläche:**
   a. **Verwaltung der Templates in z/OSMF (nicht für Entwickler relevant)**



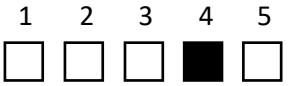
   b. **Verwaltung der Instanzen in z/OSMF (nicht für Entwickler relevant)**

4. **Bewerten Sie die gezeigte Arbeitsweise für Änderungen an den Workflow Definitionsdateien. (nicht für Entwickler relevant)**

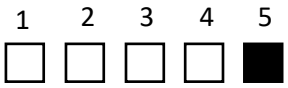
   Besser wäre es, wenn zur Bearbeitung eine IDE verwendet werden würde (Syntax)
5. **Wie ist Ihr erster persönlicher Eindruck zum Tool? (nicht für Entwickler relevant)**
   Zukunft des modernen Deployments auf dem Mainframe. Ähnlich der offenen Welt. Bringt die Plattform z nach vorne!
6. **Wie würden Sie den aktuellen Bereitstellungsprozess beurteilen?**
   Umständlich und wenig intuitiv.
   Es sind viele kleine bürokratischen Hürden nehmen.
   Die Mentalität des einfachen Provisionierens funktioniert hier leider nicht.
7. **Können Sie sich vorstellen, mit dem Tool täglich zu arbeiten?**
   Ja.
8. **Wenn 7. Mit ja beantwortet wurde, begründen Sie ihre Meinung.**

IBM MQ Administrator 1

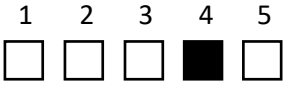
1. **Es wurde ein Template für die DATEV-Rechnungsschreibung, welches ein CICS, die benötigten IBM MQ Queues und theoretisch die benötigte Db2 Datenbanken beinhaltet, vorgestellt. Der Ablauf, der damit einhergeht, beschränkt sich zunächst auf z/OSMF. Bewerten Sie diesen, begründen Sie Ihre Bewertung.**



   - Mehrwert dadurch, dass mehr Verantwortung bei dem Entwickler ist
   - Weniger händische Eingriffe
2. **Es wurde eine Ergänzung mit z/OSPT, zu oben genannten Ablauf, erläutert. Bewerten Sie diese, begründen Sie Ihre Bewertung.**

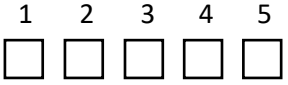


   - Wird das von IBM noch weiterentwickelt?
   - Features, die bereits vorhanden sind, sind schon gut
   - Flexibilität ist höher als mit dem Ablauf aus 1.
3. **Bewerten Sie folgende Punkte bezüglich der Benutzerfreundlichkeit der Oberfläche:**
   a. **Verwaltung der Templates in z/OSMF (nicht für Entwickler relevant)**



   b. **Verwaltung der Instanzen in z/OSMF (nicht für Entwickler relevant)**

4. **Bewerten Sie die gezeigte Arbeitsweise für Änderungen an den Workflow Definitionsdateien. (nicht für Entwickler relevant)**

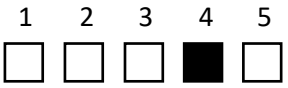


   Da noch nicht selbst damit gearbeitet wurde, kann es nicht beurteilt werden.
5. **Wie ist Ihr erster persönlicher Eindruck zum Tool? (nicht für Entwickler relevant)**
   - Gezeigtes ist gut, aber Zeitaufwand ist mit einzubeziehen und die zu leistenden Vorarbeiten
   - MQ Queue Manager ist komplexer → noch mehr Zeitaufwand und notwendige Vorarbeiten werden mehr
6. **Wie würden Sie den aktuellen Bereitstellungsprozess beurteilen?**
   - Deutlich manueller Arbeitsablauf
   - Viele Rückfragen und viel Arbeiten auf Zuruf, Kommunikation über Email, Telefon oder Termine

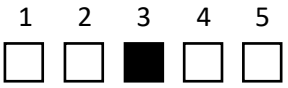
IBM MQ Administrator 2

1. **Es wurde ein Template für die DATEV-Rechnungsschreibung, welches ein CICS, die benötigten IBM MQ Queues und theoretisch die benötigte Db2 Datenbanken beinhaltet, vorgestellt. Der Ablauf, der damit einhergeht, beschränkt sich zunächst auf z/OSMF. Bewerten Sie diesen, begründen Sie Ihre Bewertung.**

   - Fehlender Umfang (speziell für MQ)
   - Nur spezifische Queues mit speziellen Parametern
2. **Es wurde eine Ergänzung mit z/OSPT, zu oben genannten Ablauf, erläutert. Bewerten Sie diese, begründen Sie Ihre Bewertung.**



   - Weiterhin Abstimmung mit Dritten (RACF, IP, Storage) notwendig
   - Nur „pseudo“ Docker Container
3. **Bewerten Sie folgende Punkte bezüglich der Benutzerfreundlichkeit der Oberfläche:**
   a. **Verwaltung der Templates in z/OSMF (nicht für Entwickler relevant)**



   b. **Verwaltung der Instanzen in z/OSMF (nicht für Entwickler relevant)**

4. **Bewerten Sie die gezeigte Arbeitsweise für Änderungen an den Workflow Definitionsdateien. (nicht für Entwickler relevant)**

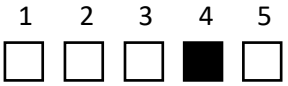
   - Automation nicht vorhanden
     - Zum Beispiel keine Analogie zur Jenkins-Replay-Funktion
5. **Wie ist Ihr erster persönlicher Eindruck zum Tool? (nicht für Entwickler relevant)**
   - Viele gute Ansätze
   - Nicht einfach genug zu verwenden im Vergleich zu Jenkins und einer PaaS Lösung
6. **Wie würden Sie den aktuellen Bereitstellungsprozess beurteilen?**
   - Deutlich manueller Arbeitsablauf
   - Viele Rückfragen und viel Arbeiten auf Zuruf, Kommunikation über Email, Telefon oder Termine

7. **Können Sie sich vorstellen, mit dem Tool täglich zu arbeiten?**
   Ja und Nein

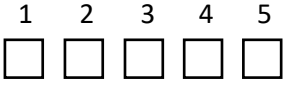
Entwickler 1 & 2

1. **Es wurde ein Template für die DATEV-Rechnungsschreibung, welches ein CICS, die benötigten IBM MQ Queues und theoretisch die benötigte Db2 Datenbanken beinhaltet, vorgestellt. Der Ablauf, der damit einhergeht, beschränkt sich zunächst auf z/OSMF. Bewerten Sie diesen, begründen Sie Ihre Bewertung.**

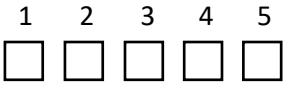
   - Es wird viel Wissen (bezüglich der Oberfläche und des Templates) benötigt und dieses muss auch bei geringer Nutzung über einen längeren Zeitraum erhalten werden.
   - Außerdem ist weiterhin viel Zuarbeit der Administration notwendig.
   - Es stellt sich die Frage, wer die DDL für Datenbanken erstellt.
   - Ansonsten schon ganz gut.
2. **Es wurde eine Ergänzung mit z/OSPT, zu oben genannten Ablauf, erläutert. Bewerten Sie diese, begründen Sie Ihre Bewertung.**



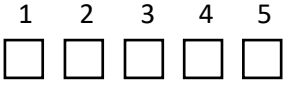
   - Vor allem ein Ausblick für eine Nutzung in der QS und Produktionsstages
   - Ablauf mehr in den Händen des eigenen Teams → höhere Effizienz, aber auch höhere Verantwortung.
   - Unkomplizierte Nutzung mittels Jenkins und den „DATEV Marktplatz“
3. **Bewerten Sie folgende Punkte bezüglich der Benutzerfreundlichkeit der Oberfläche:**
   a. **Verwaltung der Templates in z/OSMF (nicht für Entwickler relevant)**



   b. **Verwaltung der Instanzen in z/OSMF (nicht für Entwickler relevant)**



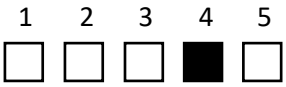
4. **Bewerten Sie die gezeigte Arbeitsweise für Änderungen an den Workflow Definitionsdateien. (nicht für Entwickler relevant)**



5. **Wie ist Ihr erster persönlicher Eindruck zum Tool? (nicht für Entwickler relevant)**
   ---
6. **Wie würden Sie den aktuellen Bereitstellungsprozess beurteilen?**
   - Zeitaufwändig durch viele Absprachen bezüglich Erstaufwand.
   - Wenns läuft, läufts.
   - Abhängigkeit von dritten (z.B. Admins)
7. **Können Sie sich vorstellen, mit dem Tool täglich zu arbeiten?**
   Ja

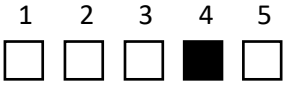
Mitarbeiter des Technologiestrategieteams

1. **Es wurde ein Template für die DATEV-Rechnungsschreibung, welches ein CICS, die benötigten IBM MQ Queues und theoretisch die benötigte Db2 Datenbanken beinhaltet, vorgestellt. Der Ablauf, der damit einhergeht, beschränkt sich zunächst auf z/OSMF. Bewerten Sie diesen, begründen Sie Ihre Bewertung.**



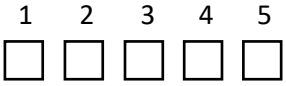
   z/OSMF stellt sich für mich als eine Oberfläche dar, die ich für eine solche „Task“ nutzen kann. Sieht einfach aus. Besser würde es mir gefallen, wenn ich den bereits existierenden „Marketplace“ unserer DATEV Cloud Lösung nutzen könnte.

2. **Es wurde eine Ergänzung mit z/OSPT, zu oben genannten Ablauf, erläutert. Bewerten Sie diese, begründen Sie Ihre Bewertung.**

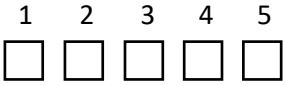


   Unkomplizierte Nutzung mittels Jenkins und den „DATEV Marktplatz“ Die Konfiguration eines Skriptes mit z/OSPT ist für die Einbindung der Provisionierung in automatische Build-Prozesse hilfreich, und damit wichtig. Es muss sich jemand um den Aufbau der Build-Pipeline kümmern, (Build Engineer) die Rolle haben wir aktuell in den z/OS Projekten noch nicht. Aber es macht Sinn und bringt die z/OS Anwendungen näher an die Vorgehensweise der Cloud Native Entwicklung.

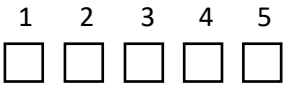
3. **Bewerten Sie folgende Punkte bezüglich der Benutzerfreundlichkeit der Oberfläche:**
   a. **Verwaltung der Templates in z/OSMF (nicht für Entwickler relevant)**



   b. **Verwaltung der Instanzen in z/OSMF (nicht für Entwickler relevant)**



4. **Bewerten Sie die gezeigte Arbeitsweise für Änderungen an den Workflow Definitionsdateien. (nicht für Entwickler relevant)**



5. **Wie ist Ihr erster persönlicher Eindruck zum Tool? (nicht für Entwickler relevant)**
   ---
6. **Wie würden Sie den aktuellen Bereitstellungsprozess beurteilen?**
   Aktuell teilen sich n Entwickler an einem Produkt die gleiche Entwicklungsumgebung, arbeiten im gleichen CICS und auf der gleichen Test-Datenbank. Änderungen beeinflussen auch die Tests der anderen Kollegen, und müssen koordiniert werden. Die

A.8. Workflow Step mit REST-Call

```
<step name="db2_create_database">
        <title>db2_create_database.</title>
        <description>Erzeugt mit der Hilfe des DB2
       Service Brokers eine Datenbank</description>
        <instructions substitution="false">
       Erzeugt DB2 Datenbank.</instructions>
        <weight>10</weight>
        <skills>REST</skills>
       <autoEnable>true</autoEnable>
       <rest>
       <httpMethod>PUT</httpMethod>
       <schemeName substitution="false">http</schemeName>
       <hostname substitution="false">hostname</hostname>
       <uriPath>uriPath</uriPath>
       <requestBody substitution="true">
{
  "service_id": "${instance-DFH_DB2_SERVICEID}",
  "plan_id": "${instance-DFH_DB2_PLANID}",
 "organization_guid": "DUMMY",
  "space_guid": "DUMMY",
  "parameters": {
            "GROUP": "UGZTAL",
            "VUSERID": "${_step-stepOwnerUpper}",
            "DBNAME": "${instance-DFH_DB2_DATABASENAME}",
            "UserID": "${_step-stepOwnerUpper}",
            "Passwort": "DUMMY"
  }
}
       </requestBody>
       <expectedStatusCode>201</expectedStatusCode>
       <requestHeaders substitution="false">
       {"Authorization":"Basic␣VDMwMTkzQTpYZjN1I2RJNA=="}
       </requestHeaders>
       </rest>
   </step>
```

Listing A.3: Workflow Step mit REST-Call

A.9. Produktstammdaten Tabellen Data Definition Language

```
1  CREATE TABLE PSSSCHEMA.TPAUSPSSBASELBART
2          (PARTITIONID INTEGER
3                                                          NOT NULL
4  WITH DEFAULT 1
5          ,PID INTEGER
6                                                          NOT NULL
7          ,AUSPRAEGUNG SMALLINT
8                                                          NOT NULL
9          ,GUELTIGAB DATE
10                                                         NOT NULL
11         ,LFDNR INTEGER
12                                                         NOT NULL
13 WITH DEFAULT
14         ,GUELTIGBIS DATE
15                                                         NOT NULL
16 WITH DEFAULT "9999-12-31"
17         ,PSSID INTEGER
18 WITH DEFAULT NULL
19         ,ZUSATZID CHARACTER(4) FOR SBCS DATA
20 WITH DEFAULT NULL
21         ,BEZEICHNUNG VARCHAR(100) FOR SBCS DATA
22 WITH DEFAULT NULL
23         ,MWSTANTEILFREI DECIMAL(5, 2)
24 WITH DEFAULT NULL
25         ,MWSTANTEILREDUZIERT DECIMAL(5, 2)
26 WITH DEFAULT NULL
27         ,MWSTANTEILVOLL DECIMAL(5, 2)
28 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                                                          NOT NULL
92          ,MAXNUM INTEGER
93                                                          NOT NULL
94          ,MAXNUBEZ CHARACTER(42) FOR SBCS DATA
95                                                          NOT NULL
96  WITH DEFAULT "X"
97          ,BEZEICHNUNG VARCHAR(100) FOR SBCS DATA
98                                                          NOT NULL
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␣fuer␣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␣fuer␣maximale␣Nummer";
134
135
136 SET CURRENT SQLID = "DB2SADM";
137
138
139     COMMENT ON COLUMN PSSSCHEMA.TMAXNUM.BEZEICHNUNG
140             IS "Bezeichnung␣fuer␣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                                                          NOT NULL
365  WITH DEFAULT 1
366          ,PID INTEGER
367                                                          NOT NULL
368          ,ARTNR INTEGER
369                                                          NOT NULL
370  WITH DEFAULT
371          ,ANDAT DATE
372                                                          NOT NULL
373  WITH DEFAULT "1966-02-14"
374          ,OPDATBEN CHARACTER(1) FOR SBCS DATA
375                                                          NOT NULL
376  WITH DEFAULT "J"
377          ,AUSDIENSTREL CHARACTER(1) FOR SBCS DATA
378                                                          NOT NULL
379  WITH DEFAULT "N"
380          ,VERTRIEBSREL CHARACTER(1) FOR SBCS DATA
381                                                          NOT NULL
382  WITH DEFAULT "N"
383          ,VERTRELDAT DATE
384  WITH DEFAULT NULL
385          ,KOMMASTELLEN INTEGER
386  WITH DEFAULT NULL
387          ,ERTRNR INTEGER
388                                                          NOT NULL
389  WITH DEFAULT
390          ,GFEDNR INTEGER
391                                                          NOT NULL
392  WITH DEFAULT
393          ,UPLONR INTEGER
394                                                          NOT NULL
395  WITH DEFAULT
396          ,MWSTEUERSATZ INTEGER
397  WITH DEFAULT NULL
398          ,POLINR INTEGER
```

```
399                                                          NOT NULL
400 WITH DEFAULT
401         ,EXPGNR INTEGER
402                                                          NOT NULL
403 WITH DEFAULT
404         ,EXPONR INTEGER
405                                                          NOT NULL
406 WITH DEFAULT
407         ,ARTIKELTYPID INTEGER
408                                                          NOT NULL
409 WITH DEFAULT
410         ,ARTIKELTYPALT CHARACTER(4) FOR SBCS DATA
411                                                          NOT NULL
412 WITH DEFAULT "0000"
413         ,BERBESTEINHID INTEGER
414                                                          NOT NULL
415 WITH DEFAULT
416         ,BERBESTEINHALT CHARACTER(4) FOR SBCS DATA
417                                                          NOT NULL
418 WITH DEFAULT "0000"
419         ,LEISTGRUPID INTEGER
420                                                          NOT NULL
421 WITH DEFAULT
422         ,LEISTGRUPALT CHARACTER(4) FOR SBCS DATA
423                                                          NOT NULL
424 WITH DEFAULT "0000"
425         ,BERFREQID INTEGER
426                                                          NOT NULL
427 WITH DEFAULT
428         ,NUTZERID INTEGER
429 WITH DEFAULT
430         ,LEISTARTID INTEGER
431                                                          NOT NULL
432 WITH DEFAULT
433         ,BERFREQALT CHARACTER(4) FOR SBCS DATA
434                                                          NOT NULL
435 WITH DEFAULT "0000"
436         ,LEISTARTALT CHARACTER(4) FOR SBCS DATA
437                                                          NOT NULL
438 WITH DEFAULT "99"
```

```
439         ,NUTZERALT CHARACTER(1) FOR SBCS DATA
440                                                 NOT NULL
441 WITH DEFAULT "K"
442         ,BARTBEZ_20 CHARACTER(20) FOR SBCS DATA
443                                                 NOT NULL
444 WITH DEFAULT "X"
445         ,ARTIKELTYPBEZ CHARACTER(50) FOR SBCS DATA
446                                                 NOT NULL
447 WITH DEFAULT "Keine␣Zuordnung"
448         ,BERBESTEINHBEZ CHARACTER(50) FOR SBCS DATA
449                                                 NOT NULL
450 WITH DEFAULT "Keine␣Zuordnung"
451         ,LEISTGRUPBEZ CHARACTER(50) FOR SBCS DATA
452                                                 NOT NULL
453 WITH DEFAULT "Keine␣Zuordnung"
454         ,BERFREQBEZ CHARACTER(50) FOR SBCS DATA
455                                                 NOT NULL
456 WITH DEFAULT "Keine␣Zuordnung"
457         ,NUTZERBEZ CHARACTER(50) FOR SBCS DATA
458                                                 NOT NULL
459 WITH DEFAULT "Keine␣Zuordnung"
460         ,LEISTARTBEZ CHARACTER(50) FOR SBCS DATA
461                                                 NOT NULL
462 WITH DEFAULT "Keine␣Zuordnung"
463         ,BARTBEZ_100 VARCHAR(100) FOR SBCS DATA
464                                                 NOT NULL
465 WITH DEFAULT "X"
466         ,ERTRBEZ VARCHAR(100) FOR SBCS DATA
467                                                 NOT NULL
468 WITH DEFAULT "X"
469         ,GFEDBEZ VARCHAR(100) FOR SBCS DATA
470                                                 NOT NULL
471 WITH DEFAULT "X"
472         ,UPLOBEZ VARCHAR(100) FOR SBCS DATA
473                                                 NOT NULL
474 WITH DEFAULT "X"
475         ,POLIBEZ VARCHAR(100) FOR SBCS DATA
476                                                 NOT NULL
477 WITH DEFAULT "X"
478         ,EXPGBEZ VARCHAR(100) FOR SBCS DATA
```

```
479                                                          NOT NULL
480 WITH DEFAULT "X"
481         ,EXPOBEZ VARCHAR(100) FOR SBCS DATA
482                                                          NOT NULL
483 WITH DEFAULT "X"
484         ,HAKONR INTEGER
485 WITH DEFAULT NULL
486         ,HAKOBEZ VARCHAR(100) FOR SBCS DATA
487 WITH DEFAULT NULL
488         ,INPGNR INTEGER
489                                                          NOT NULL
490 WITH DEFAULT
491         ,INPGBEZ VARCHAR(100) FOR SBCS DATA
492                                                          NOT NULL
493 WITH DEFAULT "X"
494         ,BEZ035 CHARACTER(35) FOR SBCS DATA
495                                                          NOT NULL
496 WITH DEFAULT "X"
497         ,CONSTRAINT PPAUSPSSBART PRIMARY KEY
498         (PARTITIONID
499         ,PID
500         )
501         ,CONSTRAINT UPAUSPSSBART UNIQUE
502         (PARTITIONID
503         ,ARTNR
504         )
505         )
506        IN DATABASE PSSBAPRV
507  APPEND NO
508  NOT 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;
```

Listing A.4: Produktstammdaten Tabellen Data Definition Language

Abbildungsverzeichnis

Tabellenverzeichnis

Quellcodeverzeichnis

Literaturverzeichnis

[Adam 16] R. Adams. *SQL: Der Grundkurs für Ausbildung und Praxis : mit Beispielen in MySQL/MariaDB*. 2016.

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

[Aran 13] C. Aranha. *IBM WebSphere MQ V7.1 and V7.5 features and enhancements*. 2013.

[Cass 07] P. Cassier. *System programmer's guide to Workload manager*. 2007.

[Ceru 03] P. E. Ceruzzi. *A history of modern computing*. 2003.

[Ebbe 11] M. Ebbers, J. Kettner, W. O'Brien, and B. Ogden. *Introduction to the new mainframe: Z/OS basics*. 2011.

[Fosd 05] H. Fosdick. *Rexx programmer's reference*. 2005.

[http 19a] "https://searchengineland.com/google-now-handles-2-999-trillion-searches-per-year-250247". 02.12.2019.

[http 19b] "https://www.datev.de/web/de/m/ueber-datev/das-unternehmen/geschichte/". 25.11.2019.

[http 19c] "https://www.ibm.com/it-infrastructure/z/cics". 23.11.2019.

[http 20a] "https://blog.infotelcorp.com/blog/articles/the-mainframe-skills-gap-is-widening-automation-can-save-us". 25.2.2020.

[http 20b] "https://medium.com/jorgeacetozi/continuous-integration-vs-continuous-delivery-vs-continuous-deployment-d5839a85a959". 25.2.2020.

[http 20c] "https://neuhandeln.de/im-ueberblick-die-groessten-anbieter-fuer-cloud-computing/". 27.2.2020.

[http 20d] "https://pepgotesting.com/continuous-integration/". 25.2.2020.

[http 20e] "https://www.alliedmarketresearch.com/mainframe-market". 25.2.2020.

[http 20f] "https://www.bsi.bund.de/DE/Themen/DigitaleGesellschaft/ CloudComputing/Grundlagen/Grundlagen_node.html". 23.2.2020.

[http 20g] "https://www.cloudcomputing-insider.de/was-ist-cloud-foundry-a-615766/". 23.2.2020.

[http 20h] "https://www.cloudcomputing-insider.de/was-ist-cloud-native-a-669681/". 23.2.2020.

[http 20i] "https://www.computerweekly.com/de/definition/Private-Cloud". 23.2.2020.

[http 20j] "https://www.datev.de/web/de/m/ueber-datev/das-unternehmen/kurzprofil/". 27.2.2020.

[http 20k] "https://www.egovernment-computing.de/was-ist-ein-legacy-system-a-802283/". 22.2.2020.

[http 20l] "https://www.fintechfutures.com/2018/11/ibm-shows-off-three-new-bank-clients/". 25.2.2020.

[http 20m] "https://www.gartner.com/smarterwithgartner/7-options-to-modernize-legacy-systems/". 1.3.2020.

[http 20n] "https://www.hochschulkompass.de/home.html". 09.02.2020.

[http 20o] "https://www.ibm.com/downloads/cas/AGW2EEP0". 29.02.2020.

[http 20p] "https://www.ibm.com/support/knowledgecenter/SSLTBW_2.4.0/com.ibm.zos.v2r4.izua700/izuprog_PrepareforCloudProvisioning.htm". 26.2.2020.

[http 20q] "https://www.ibm.com/support/knowledgecenter/SSXH44_1.1.0/zospt/cics/zospt-cics-properties.html". 26.2.2020.

[Kara 08] S. Karan, F. Rui, L. Oerjan, M. Bob, P. Rita, and R. Paul. *ABCs of z/OS system programming, Volume 6.* 2008.

[Keit 16] W. Keith, P. Gary, and S. Hiren. *IBM Cloud Provisioning and Management for z/OS: An Introduction.* 2016.

[Kim 14] G. Kim. *The Phoenix project: A novel about IT, DevOps, and helping your business win.* 2014.

[Last 17] B. Laster. *Continuous Integration vs. Continuous Delivery vs. Continuous Deployment.* 2017.

[Love 13] M. Lovelace. *VSAM demystified.* 2013.

[Rayn 11] C. Rayns. *CICS transaction server from start to finish*. 2011.

[Rott 18] R. J. T. Rotthove. *IBM z/OS Management Facility V2R3*. 2018.

[Stee 03] B. Steegmans. *DB2 for z/OS and OS/390: Ready for Java*. 2003.

[Vohr 16] D. Vohra. *Pro Docker*. 2016.

Glossar

B

Batch Batch beziehungsweise Batch-Verarbeitung bezeichnet in der IT die sog. „Stapelverarbeitung“. Das heißt, dass Programme mit minimalem menschlichen Eingreifen nacheinander abgearbeitet werden. Beispielsweise 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[1] der DATEV Rechnungsschreibung durchgeführt. Die auszuführenden Programme laufen in sogenannten „Batch-Jobs“. [Ebbe 11, S. 274]. 13

Batch Job In einem BatchJob, 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 Folgenden beschrieben.

Zunächst ist „JOB“ zu nennen, auch Jobkarte genannt. Hier werden der Name des Jobs, Berechnungsinformationen, maximal zur Verfügung stehende CPU-Zeit und weitere Job-weite Parameter gesetzt. Im Beispiel 1 Zeilennummer eins bis drei.

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. Im Beispiel 1 Zeilennummer zehn.

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. Im Beispiel 1 Zeilennummer 12 bis 15 und 19. Hier ist auch zu sehen, dass Daten Inline an ein DD-Statement übergeben werden können. [Ebbe 11, S. 274] . 5, 19, 51

C

[1] Beschreibung in Absatz 4.3.1

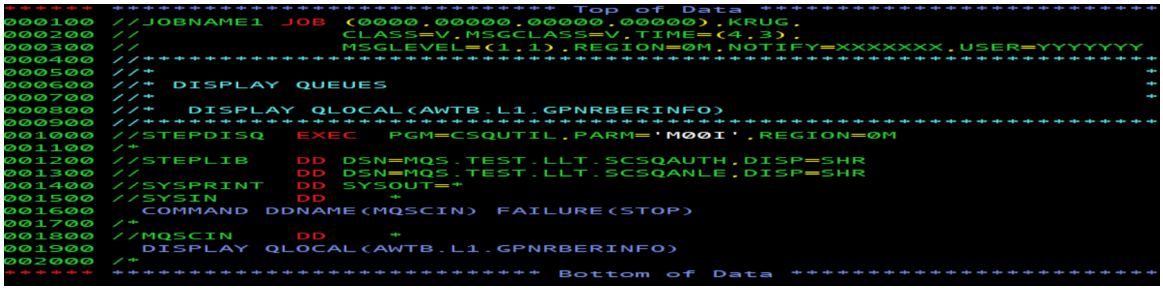


Abbildung 1.: Job Beispiel, Display einer IBM MQ

Cloud-Computing „Cloud Computing bezeichnet das dynamisch an den Bedarf angepasste Anbieten, Nutzen und Abrechnen von IT-Dienstleistungen über ein Netz. Angebot und Nutzung dieser Dienstleistungen erfolgen dabei ausschließlich über definierte technische Schnittstellen und Protokolle. Die Spannbreite der im Rahmen von Cloud Computing angebotenen Dienstleistungen umfasst das komplette Spektrum der Informationstechnik und beinhaltet unter anderem Infrastruktur (z. B. Rechenleistung, Speicherplatz), Plattformen und Software.“[http 20g]. 9

D

DDL Data Definition Language: Befehlssatz zum Anlegen, Ändern und Löschen von Datenbanken, Tabellen usw. und ihren Strukturen. [Adam 16, S. 61]. 52

Docker Docker ist eine offene Plattform für die Entwicklung, Paketierung und Ausführen von Anwendungen auf verteilten Systemen. Zu diesen Systemen zählen unter anderem die Cloud, Datenzentren oder virtuelle Maschinen. [Vohr 16, Kap. „Setting the Environment“]. 9

R

RACF 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. [Kara 08, S. 18-21]. 31

REXX Restructured Extended Executor versteht sich als Skriptsprache und kommt nicht nur am Mainframe zum Einsatz. So kann es als Bindeglied für Betriebssystembefehle, grafischen Interfaces, Objekten, Funktionen und Serviceroutinen gesehen werden. So wird es, wie auch in dieser Arbeit, unter anderem für die Automation von sich wiederholenden Systemadministrationsaufgaben eingesetzt. [Fosd 05, S. 3-4]. 53

V

VSAM Virtual Storage Access Method, spezielle z/OS Dateiart, die schnelle I/O-Zugriffe ermöglicht. [Love 13, S. 2-4]. 13