ulm university universität
uulm

Universität Ulm | 89069 Ulm | Germany

Fakultät für Ingenieurwissenschaften und Informatik
Institut für Datenbanken und Informationssysteme

Evaluierung eines Konfigurationsmechanismus für eine Java Server Applikation mit einem Java Rich Client

Bachelorarbeit an der Universität Ulm

Vorgelegt von:
Manuel Stegmiller
manuel.stegmiller@uni-ulm.de

Gutachter:
Prof. Dr. Manfred Reichert

Betreuer:
Gregor Grambow, Daniel Golesny

2014

Fassung 24. September 2014

Kurzfassung

In moderner Softwareentwicklung ist es üblich, dass Applikationen nicht nur für einen Kunden programmiert werden, sondern für mehrere. Dies erfordert die Möglichkeit die Applikation auf die einzelnen Kunden anzupassen. Das kann durch Konfigurationsmechanismen erfolgen. Außerdem gibt es in der Softwareentwicklung oft verschiedene Testumgebungen die selbst verschiedene Anforderungen an eine Applikation haben. In dem Projekt ePEP der Daimler TSS wurde die Komplexität der Konfiguration zu einem zeitraubendem Problem. Aus diesem Grund beschäftigt sich die Arbeit mit der Vereinfachung der Konfiguration. Dafür wurden zunächst die Probleme des genannten Projektes untersucht. Darauf folgend wurde mit Hilfe von Experteninterviews eine Handlungsempfehlung erstellt. In dieser sind die Erfahrungen der Experten zusammengefasst welche beschreiben wie man sehr geschickt die Komplexität vereinfachen kann. Die Empfehlungen, welche erarbeitet wurden, werden im vorletzten Kapitel evaluiert. Dazu haben sich die Experten die Empfehlungen angesehen und es wird gezeigt, dass diese Empfehlungen die Probleme, die zu Beginn der Arbeit beschrieben werden, lösen können.

Danksagung

An dieser Stelle möchte ich mich bei allen bedanken, die mich während der Anfertigung dieser Arbeit unterstützt und motiviert haben.

Besonders möchte ich meinen beiden Betreuern, Gregor Grambow und Daniel Golesny, die meine Arbeit und somit auch mich betreut haben, danken. Kritische Fragen, Tipps und Motivation dienten als wertvolle Bereicherung meiner Arbeit.

Auch möchte ich mich bei der Daimler TSS bedanken, für die Möglichkeit bei ihnen zu forschen und zu arbeiten. Vor Allem bei den Mitarbeitern die die Zeit gefunden haben mich als Experten zu beraten und so die Forschung überhaupt erst möglich gemacht haben.

Nicht zuletzt gilt der Dank auch meinen Eltern, die mich während des kompletten Studiums finanziell und vor allem emotional gestärkt haben.

Inhaltsverzeichnis

1

Einleitung

Moderne Softwareentwicklung baut darauf auf, dass Programme wiederverwendbar geschrieben werden. Viele Programme werden demnach so geschrieben, dass man sie mehr als nur einmal einsetzen kann. Jedoch muss ein solches Programm immer wieder mit kleinen Änderungen an die aktuellen Anforderungen angepasst werden. Um dies zu erreichen, werden Konfigurationsmechanismen in die Programme eingebaut. Diese Arbeit evaluiert Konfigurationsmechanismen für Java Web Applications. Dabei orientiert sich die Problemstellung an einem Projekt der Daimler TSS GmbH (kurz: die TSS), dem Projekt ePEP (elektronischer Produktions Einsatz Prozess). Die TSS ist eine IT-Tochterfirma der Daimler AG und befasst sich ausschließlich mit Aufträgen für eben diese. ePEP ist ein Programm um Prozesse in Werken zu verwalten und wurde von Grund auf von einem Entwicklerteam in der Java Abteilung der TSS entwickelt. Um kleine Änderungen schnell, und vor allem ohne Änderungen am Code, vornehmen zu können wurden bestimmte Parameter in eine Konfigurationsdatei ausgelagert. Anfangs

noch für ein Werk gedacht wuchs die Anwendung schnell und wurde stetig erweitert. Ohne den zugrunde liegenden Konfigurationsmechanismus zu überarbeiten wurden über die Zeit hinweg immer mehr Werke in die Anwendung aufgenommen und immer mehr Anforderungen mussten bedient werden. Die Konfigurationsdateien die verwendet werden sind dabei so sehr gewachsen, dass eine schnelle und vor allem fehlerfreie Anpassung jetzt nicht mehr möglich ist. Dieses Problem zugrundeliegend widmet sich diese Arbeit der Evaluation eines Konfigurationsmechanismus, mit dem es möglich ist, Programme (auch Applikationen genannt) zuverlässig zu konfigurieren.

1.1 Motivation und Problemstellung

Das in dieser Arbeit behandelte Problem betrifft die Verwaltung der Konfiguration bzw. der Konfigurationsdateien einer Applikation. In der modernen Softwareentwicklung ist es üblich, dass ein Programm nicht mehr nur auf einem System läuft. Meistens läuft das Programm auf vielen Umgebungen die im Entwicklungsprozess alle ihre eigene Rolle innehaben. So gibt es in dem Projekt, auf dem die Problemstellung beruht, vier Umgebungen: eine Produktions-, eine Integrations-, eine Test- und eine Entwicklungsumgebung (häufig auch Developmentumgebung genannt). Alle Umgebungen haben verschiedene Anforderungen oder Namensbereiche und müssen daher unterschiedlich konfiguriert werden. Zusätzlich läuft das Programm oft nicht nur bei einem Kunden, im folgenden Mandanten genannt, sondern bei mehreren. Bei ePEP entsprechen die Mandanten den verschiedenen Werken. Diese haben verschiedene Anforderungen, sodass für alle Umgebungen im Entwicklungsprozess noch eine Vielzahl an Mandanten verwaltet werden müssen. Jede einzelne Umgebung hat dabei andere Werte für Parameter wie Datenbanknamen oder IP-Adressen. Hinzu kommen verschiedene applikationsspezifische Parameter wie Interface-Ansichten oder Datenstrukturen die sich je nach Einsatzort ändern können. Diese Parameter werden demnach in einen Konfigurationsmechanismus ausgelagert um nicht direkt den Code ändern zu müssen, wenn eine Anpassung erforderlich ist. Mit steigender Anzahl an Umgebungen und Mandanten, steigt auch der Konfigurationsaufwand.

In Abbildung 1.1 ist veranschaulicht, wie viele verschiedene Konfigurationen es geben kann und wie diese zusammen hängen. Dabei gibt es zwei Richtungen. Horizontal sind

Abbildung 1.1: Ansicht der Problematik

	Man. 1	Man. 2	Man. 3	Man. 4	...
DEV					
TEST					
INT					
PROD					

die verschiedenen Mandanten dargestellt. Vertikal sind die unterschiedlichen Entwicklungsumgebungen dargestellt. Das bedeutet, dass jedes Feld unterschiedliche Anforderungen haben kann. Ein Feld steht dabei für eine, nicht in der Größe beschränkte, Menge an Parametern. Einige Werte sind über mehrere Felder hinweg gleich, während andere sich von Feld zu Feld unterscheiden. Weitere sind für Mandant Zwei auf jeder Umgebung gleich (blau dargestellt), weichen davon jedoch bei jedem anderen Mandanten ab. Eine ähnliche Situation (grün dargestellt) kann für Umgebungen eintreten. Diese Forderungen betreffen immer einzelne Werte, und bedeutet nicht von vornherein, dass alle Werte der markierten Umgebungen gleich sein müssen. Durch diese Vielfalt in den Konfigurationsanforderungen entstehen komplexe Konfigurationsdateien, die kaum mehr zu überblicken sind. Diese führen zu zeitaufwändigem Suchen und sind aufgrund ihrer Komplexität sehr fehleranfällig und bieten keinerlei Automatisierungswerkzeuge.

1.2 Zielsetzung

Ziel dieser Arbeit ist es, bestehende Herangehensweisen zur Konfiguration zu evaluieren und zu vergleichen. Zusätzlich zu bestehenden Lösungen sollen eigene Herangehensweisen erarbeitet werden. Dabei soll aus der Erfahrung mehrerer Experten innerhalb der TSS gelernt werden um eine passende Lösung für Java-Projekte zu finden. Diese soll in Form einer Handlungsempfehlung für zukünftige Teams zur Verfügung stehen. Die Anforderungen an diese Handlungsempfehlung werden in Kapitel 4 beschrieben.

1.3 Aufbau der Arbeit

Die Arbeit beginnt mit der Problemanalyse. Darin wird darauf eingegangen, welche Probleme bei der Konfiguration auftreten können. Beispielhafte Szenarien stellen diese Probleme anschaulich dar. Darauf folgend werden Lösungen für die analysierten Probleme gesucht. Um möglichst praxisnahe Lösungswege zu erhalten werden Experten der TSS interviewt. Im Anschluss an die Interviews werden die gewonnenen Erkenntnisse zusammengefasst und ausgewertet. Das Kernstück der Arbeit bildet die darauf folgende Handlungsempfehlung, welche aus den Ergebnissen der Interviews entsteht. Diese wird noch evaluiert und diskutiert. Den Schlussteil bildet eine Zusammenfassung der Ergebnisse.

2

Grundlagen

In diesem Kapitel wird erläutert unter welchen Gegebenheiten das Problem, welches diese Arbeit behandelt, entstanden ist. Um das Arbeitsumfeld genauer zu betrachten wird zunächst die Daimler TSS kurz vorgestellt. Neben der Firmengeschichte, so wie einigen Referenzen, wird genauer auf die Abteilung eingegangen in der das Problem aufkam.

2.1 Umfeld

Wie bereits in der Einleitung erwähnt, wird diese Arbeit in der Firma Daimler TSS geschrieben. Diese liefert neben dem Thema auch das Arbeits- und Forschungsumfeld. Die Daimler TSS ist ein,auf IT-Services spezialisiertes, Unternehmen. Sie fungiert als Tochterfirma von Daimler und versteht sich als Dienstleistungsunternehmen. Gestartet ist das

Unternehmen 1998 als Software Entwicklungsteam. Mittlerweile beschäftigt die TSS über 600 Mitarbeiter. Die TSS arbeitet ausschließlich konzernintern, das heißt die Kunden sind alle aus dem Konzern der Daimler AG. Das Angebot der TSS besteht aus IT Dienstleistungen und umfasst die Beratung durch Fachwissen im IT Bereich, Sicherheitslösungen, Trainings und auch die Entwicklung kundenspezifischer Software. Als Vorzeigeprojekt zählt hierzu vor allem die IT-Lösung für car2go, ein internationales Carsharing Angebot. Viele Teams arbeiten aber auch an kleineren Softwarelösungen für den Konzern. Dabei handelt es sich um Software in Fahrzeugen bis hin zu Prozessoptimierungssoftware. Diese Teams sind zusammengefasst unter der Abteilung Custom-Software. Innerhalb dieser wird zusätzlich zwischen verschiedenen Technologien unterschieden. So gibt es eine Abteilung die hauptsächlich mit der Programmiersprache JAVA programmiert. Ein Projekt dieser Abteilung ist das Projekt ePEP. ePEP ist eine Software um Produktionsprozesse in Werken zu unterstützen. Diese Software wurde von Grund auf von einem Team der TSS entwickelt. Dieses Projekt dient als Fallstudie für die Arbeit. An ihm wird vor allem die Analyse der Probleme durchgeführt um herauszufinden, welche Probleme zu einem großen Aufwand in der Konfiguration führen.

2.2 Definition Konfiguration

Da in der Literatur das Thema Konfiguration im Zusammenhang mit Software allgemein anders aufgefasst wird als in dieser Arbeit, wird an dieser Stelle die Begrifflichkeit noch einmal klar definiert. Mit Softwarekonfiguration bezeichnet diese Arbeit eine Anpassung eines Programmes. Diese Anpassung fordert kein neues kompilieren der Software, sondern kann über vorher definierte Mechanismen erfolgen. Dazu können Parameter geändert werden die vom Programm eingelesen und interpretiert werden. Konfiguration wird also als Möglichkeit gesehen eine fertig programmierte Software im Nachhinein zu ändern und auf bestimmte Gegebenheiten anzupassen. Als Beispiel kann eine Software angesehen werden, welche automatisch nach einem definierten Zeitintervall eine Log-Datei über ihren Status schreibt. Je nach Einsatzzweck kann dieses Zeitintervall durch Konfiguration von außen bestimmt werden, ohne dass es fest im Code verankert ist. Oft werden aber nicht nur Zahlenwerte konfiguriert, sondern auch GUI Elemente

um die Programmansicht beim Nutzer zu ändern. Weiterhin müssen Anbindungen an Datenbanken und eventuelle Schnittstellen geändert werden. Alle Szenarien haben gemeinsam, dass bestimmte Werte entweder zur Entwicklungszeit nicht bekannt sind, oder sich je nach Einsatzzweck ändern. Um diese Werte zu definieren wird ein Konfigurationsmechanismus benötigt.
Die Unterscheidung zwischen Parameter und Wert ist im Verlauf der Arbeit wichtig. Ein Parameter bezeichnet eine Eigenschaft welche konfiguriert wird. Ein Wert bezeichnet die Ausprägung. Beispielsweise hat ein Parameter ”Tag” die Ausprägung ”Mittwoch”.

3

Verwandte Arbeiten

Sucht man nach Literatur zu *Software Configuration* so findet man hauptsächlich Arbeiten über Software Configuration Management (SCM). Dies unterscheidet sich jedoch vom Thema dieser Arbeit. Eine Definition von SCM nach [Ber02] lautet wie folgt: *"Configuration management is a discipline that governs the identification, control, status accounting, audit, and interface of a given software product. It is one of the many processes that occur within an engineering development environment in which several engineering, software, and manufacturing process are performed concurrently."*. Das bedeutet, dass SCM sich vor Allem mit den Prozessen der Software Entwicklung befasst, wie etwa Versionierung oder Dokumentation. Konfiguration im Sinne dieser Arbeit bedeutet allerdings, dass das Softwareverhalten konfiguriert wird. Das klassische SCM beschäftigt sich also mit den Prozessen um die Software herum, während sich diese Arbeit mit der Anpassung der eigentlichen Software über Konfigurationsmechanismen befasst. Ein ähnlich ernüchterndes Ergebnis liefert die Suche nach folgenden Schlüssel-

wörtern: *applicaton configuration, configuration file, Konfiguration Software, Mandant Konfiguration, configuration development*. Die Ergebnisse enthalten die Wörter meist nur in getrenntem Kontext oder sprechen lediglich an, dass bestimmte Teile konfiguriert werden müssen, nicht jedoch wie dies genau funktioniert. Vielversprechende Titel wie der von [PLS+00] (Using QDL to Specify QoS Aware Distributed Application Configuration) stellen sich als unpassend heraus. Das gegebene Beispiel beschäftigt sich mit der Verteilung auf verschiedene Server (Distributed Computing), während es in dieser Arbeit um die Konfiguration mehrerer Mandanten geht, die Software aber nicht zwingend verteilt gehostet wird. Die Arbeit [BAKF04] beschäftigt sich mit *product configuration systems*. Dieses sind allerdings keine Systeme um die Software selbst zu konfigurieren, sondern Softwaresysteme welche genutzt werden um Produkte aus einer Auswahl von Möglichkeiten individuell auszuwählen. Entferntere Suchbegriffe gehen schnell über die Grenzen des Fachbereiches hinaus.
Die folgenden Absätze beleuchten drei verwandte Themen etwas näher. Diese sind bei der Recherche aufgefallen, da sie auch vom Individualisieren von Software handeln. In den Absätzen werden jeweils einige Arbeiten kurz vorgestellt, die möglichst viel mit dem Thema Konfiguration von Software zu tun haben.

3.1 Software Product Lines

Software product lines (Softwareproduktlinien, kurz SPL) ist eine Menge aus Softwaresystemen die eine gemeinsame Funktionsbasis teilen [UI14]. Dabei werden aus einer Basis mehrere individuelle Produkte abgeleitet. Arbeiten aus diesem Umfeld beschreiben vor allem den Entwicklungsprozess und nicht das Konzept der Konfiguration. So werden Stichpunkte wie *aspect-oriented* und *model-driven development* genannt [VG07]. Diese helfen im Prozess des Software Engineering, jedoch nicht bei der eigentlichen Umsetzung der Konfiguration. Cristina Gacek und Michalis Anastasopoulos beschreiben in ihrem Paper *Implementing Product Line Variablilities* [GA01], dass sich die Literatur bisher zu wenig mit dem eigentlichen Umsetzen von SPL beschäftigt hat. Sie geben einen Überblick welche Techniken man zur Implementierung einsetzen kann. Es wird aber nicht beschrieben, wie man einen Mechanismus gestalten soll der die Anforderungen

aus Kapitel 4 erfüllt. Auch die Publikation *A Hierarchical Variability Model for Software Product Lines* [GØS12] wendet sich nicht dem Thema zu wie man zu konfigurierende Parameter festlegen und schnell wieder ändern kann.

3.2 Dependency Injection

Ein weiteres verwandtes Thema ist *dependency injection* (DP), wie es Martin Fowler in [Fow04] beschreibt. Teile davon lassen sich nutzen um Code zu schreiben, der Konfigurationsdateien lädt. Dennoch beschreibt der Autor nicht wie diese Dateien am einfachsten zu handhaben sind, um Übersichtlichkeit und eine geringe Fehlerrate zu garantieren. Frameworks wie Spring [JHD+04] oder Google Guice [Van08] helfen bei der Umsetzung des von Fowler beschriebenen Konzeptes, doch lässt sich damit nicht die Struktur einer Konfigurationsdatei vereinfachen.

3.3 Process Configuration

Eine Suche nach *process configuration* liefert schnell fachfremde Arbeiten wie [AUS01], welche vom Inhalt dem Fachbereich Chemie zuzuordnen ist. Die weitere Suche in dem Gebiet der Prozess Konfiguration lieferte keine Projekte die auf eine ähnliche Weise an einer Konfiguration arbeiten wie diese Arbeit. Dennoch zu erwähnen ist das Projekt SustainHub, welches sich mit der Datensammlung und Kommunikation auf Basis von Prozessen beschäftigt. Vor allem in der Elektro- und Autoindustrie werden Produkte oft aus vielen verschiedenen Komponenten zusammengesetzt [GMSR13], was es schwierig macht eine konsistente Datensammlung zu gewährleisten. Um dies zu bewältigen wird in [GMSR14a] eine automatische und kontextsensitive Herangehensweise an die Prozesskonfiguration aufgezeigt. In [GMSR14b] wird aufgezeigt wie man mithilfe konfigurierbarer Prozesse Daten in den Produktionsprozessen sammeln kann. Ein weiteres Projekt ist das Q-ADVICE Projekt. Dieses beschäftigt sich mit der Unterstützung von Software Engineering (SE) Prozessen. Dazu zählt das automatische Sammeln und Auswerten von Daten während des Prozesses. Die Ausführung dieses Qualitätsmana-

gements erfordert jedoch eine dauernde Anpassung an die dynamischen Aspekte des SE Prozesses. Die Projektpublikation [GOR11] zeigt wie ein deklarativer Ansatz helfen kann automatisiert Prozesse zu konfigurieren. Auch die Arbeit [GOR10] handelt von dynamischen Prozessen. Es geht dabei vor allem darum, Entwickler automatisch mit einem an sie angepasstem Prozess zu unterstützen. Dazu wurde eine kontextsensitive Umgebung entwickelt welche den SE Prozess automatisch anpasst.

3.4 Fazit

Das Problem bei der Literatursuche ist also die Vielseitigkeit des Wortes Konfiguration oder Englisch configuration. Man erhält viele Suchergebnisse, allerdings beschäftigt sich keine gefundene Arbeit mit der Art von Konfiguration, wie sie in Kapitel 2 definiert wurde. Verwandte Themen wie Software Product Lines liefern wichtige Informationen für den Entwicklungsprozess einer variablen Applikation, die Eigentliche Umsetzung wird allerdings nicht behandelt. Aus den genannten Gründen kann keine der gefundenen Arbeiten die Probleme, die dieser Arbeit zugrunde liegen, lösen.

4

Problemanalyse

Dieses Kapitel widmet sich zunächst dem aktuellen Stand des Projektes ePEP. Hierbei wird genau darauf eingegangen, wie es hinsichtlich der Konfiguration aufgebaut ist. Daraufhin werden die Probleme, die während des Betriebs auftreten genauer analysiert und aufgegliedert. An den gegebenen Problemen werden darauf folgend die Anforderungen an eine Lösung oder Handlungsempfehlung aufgestellt.

4.1 Analyse des Ist-Stands

In dem Projekt ePEP gibt es vier differenzierte Umgebungen die konfiguriert werden müssen. Zunächst wird auf der Developmentumgebung gearbeitet, welche lokal auf einem Entwickler-Laptop läuft. Parallel dazu gibt es eine Testumgebung, die aber bereits zentral gehostet wird. Auf diese wird automatisch jede Nacht das aktuellste Build aufgespielt.

Um die aktuelle Software-Version produktionsähnlich zu testen gibt es die Integrationsumgebung. Diese ist der Produktivumgebung nachempfunden und unterscheidet sich nur minimal von dieser. So laufen auf ihr alle Dienste und Anbindungen die es auch auf der Produktivumgebung gibt. Wenn die Software auf der Integrationsumgebung vom Kunden abgenommen wurde, wird diese auf die Produktivumgebung aufgespielt. Es gilt zu beachten, dass jede Umgebung auf einer anderen physischen Hardware ausgeführt wird. Die Konfiguration der Applikation erfolgt pro Umgebung über eine XML-Datei, welche auf den entsprechenden Umgebungsservern liegt. Als zentrale Verwaltung dieser Dateien kommt ein git[1]-Repository zum Einsatz, aus dem die Dateien bei Änderungen herauskopiert werden müssen. Einen Mechanismus um Verwechslungen zu vermeiden gibt es nicht. Soll nun eine Änderung vollzogen werden, muss der Bearbeiter in allen betroffenen Dateien die passenden Stellen manuell suchen und ändern. Momentan haben diese Dateien eine Länge von etwa 8000 Zeilen. Diese Länge kommt zustande, weil jeder Parameter pro Werk einmal aufgeführt ist. Das bedeutet, dass viele Zeilen mehrmals hintereinander fast identisch aufgeführt werden. Ein Beispiel dafür gibt Listing 4.1. Steht eine Änderung an, so muss die richtige Zeile gesucht und angepasst werden. Häufig müssen mehrere Dateien und darin mehrere Zeilen geändert werden, weil nicht nur ein Mandant die Änderung wünscht. Danach müssen die geänderten Dateien aus dem git-Repository auf die Server kopiert und diese dann neu gestartet werden. Solche Änderungen treten in unregelmäßigen Abständen auf, können sich aber leicht aufsummieren. Beispielhaft soll hier die Einführung zweier neuer Werke für das Projekt ePEP genannt werden, bei der sich der Konfigurationsaufwand auf etwa 10 Personentage aufsummiert hat.

Listing 4.1: INT-Konfig-Beispiel

```
1 <cfg:configuration-parameter>
2     <cfg:parameter-name>moduleName</cfg:parameter-name>
3     <cfg:parameter-value name="060">PEMSCOPETLDIIMPORTER</
         cfg:parameter-value>
4     <cfg:parameter-value name="1521">PEMSCOPETLDIIMPORTER_1521<
         /cfg:parameter-value>
```

[1] Eine Software zur Versionsverwaltung

```
5     <cfg:parameter-value name="154">PEMSCOPETLDIIMPORTER_154</
          cfg:parameter-value>
6     <cfg:parameter-value name="1542">PEMSCOPETLDIIMPORTER_1542<
          /cfg:parameter-value>
7     <cfg:parameter-value name="020">PEMSCOPETLDIIMPORTER_020</
          cfg:parameter-value>
8     <cfg:parameter-value name="030">PEMSCOPETLDIIMPORTER_030</
          cfg:parameter-value>
9     <cfg:parameter-value name="069">PEMSCOPETLDIIMPORTER_069</
          cfg:parameter-value>
10    <cfg:parameter-value name="1590">PEMSCOPETLDIIMPORTER_1590<
          /cfg:parameter-value>
11    <cfg:parameter-value name="1575">PEMSCOPETLDIIMPORTER_1575<
          /cfg:parameter-value>
12 </cfg:configuration-parameter>
```

4.2 Aufgliederung der Probleme

Der Aufbau der oben genannten XML Dateien führt zu einigen immer wiederkehrenden Problemen, welche in diesem Abschnitt zusammengefasst werden. Folgende Szenarien sind mit den bisher verwendeten Dateien zwar umsetzbar, jedoch nur sehr aufwändig. Dieser Aufwand entsteht vor allem durch häufig gemachte Fehler bei Änderungen an den Dateien. Das größte Problem, das sich durch die Komplexität der Dateien ergibt, ist also die Fehleranfälligkeit. Des Weiteren werden die Dateien sehr unübersichtlich. In den folgenden tabellarischen Darstellungen der Szenarien ist mit einer Zelle immer nur ein spezieller Konfigurationsparameter gemeint und nicht die komplette Konfiguration des entsprechenden Mandanten auf der entsprechenden Umgebung. Vorstellbar ist aber auch, dass die Probleme für mehrere oder alle Parameter auftreten. Die folgenden Probleme sind zusammen mit den Mitarbeitern am Projekt ePEP erarbeitet worden. Es

sind immer wiederkehrende Situationen, bei denen gehofft wird, durch entsprechende Techniken die Fehleranfälligkeit zu reduzieren.

Ein häufiges Szenario ist, dass ein Mandant (bzw. eine Minderheit an Mandanten) eine andere Anforderung an einen Parameter hat, als alle anderen Mandanten. Dies führt dazu, dass für einen oder wenige Mandanten dieser Parameter explizit geändert werden muss. Man stelle sich hierfür eine Menge an Mandanten vor, welche alle in Amerika operieren und dort einen bestimmten Steuersatz haben. Ein Mandant agiert aber in Deutschland und hat so einen abweichenden Steuersatz. Zusätzlich stellt man sich eine Datei vor welche genaue Angaben über die Steuern des jeweiligen Landes enthält. Diese Datei ist nun bei allen Mandanten an demselben Speicherort, also eine Art Standardkonfiguration. Bei dem Mandanten mit den abweichenden Steuergesetzen hat diese Datei einen anderen Namen. So muss die Konfiguration für diesen einen Mandanten auf allen Umgebungen geändert werden. Anschaulich ist das in Abbildung 4.1 dargestellt. Der in Deutschland operierende Mandant unterscheidet sich auf allen Umgebungen von den amerikanischen Mandanten.

Abbildung 4.1: Das Default-Problem

	US	US	Deu.	US	...
DEV					
TEST					
INT					
PROD					

Ein prinzipiell sehr ähnliches Problem, bei dem die Abweichung aber genau in der orthogonalen Dimension liegt, kann man sich folgendermaßen vorstellen: Hierbei unterscheidet sich eine Umgebung von allen anderen. Gegeben sei ein Backup-Intervall einer Datei mit Prozessdaten. Für alle Umgebungen soll jeden Tag ein Backup erstellt werden. Auf der Test-Umgebung reicht es allerdings, wenn ein Backup nur jede Woche einmal gemacht wird. In Abbildung 4.2 ist dargestellt, dass die Testumgebung bei allen Mandanten einen vom Rest abweichenden Wert hat.

Der Entwicklungsprozess besagt, dass Konfigurationen auf der Integrationsumgebung getestet werden, bevor sie auf die Produktivumgebung aufgespielt werden. Dieses

Abbildung 4.2: Die Abweichende Umgebung

	Man. 1	Man. 2	Man. 3	Man. 4	...
DEV					
TEST					
INT					
PROD					

Vorgehen führt dazu, dass ein Wert von der Integrations- auf die Produktivumgebung kopiert werden soll, und das bei allen Mandanten. Vorstellbar ist, dass am Ende eines Entwicklungszyklus verschiedene Werte kopiert werden müssen. Dies ist anschaulich dargestellt in Abbildung 4.3. Manuell ist das sehr zeitaufwändig da schnell viele Werte zusammen kommen.

Abbildung 4.3: Umgebung Kopieren Problem

	Man. 1	Man. 2	Man. 3	Man. 4	...
DEV					
TEST					
INT	↓	↓	↓	↓	↓
PROD					

Das Problem mit dem Kopieren von Werten kann auch wieder in der anderen Dimension auftreten. Vorstellbar ist, dass ein Mandant in ein anderes Land umzieht, weil er zukünftig in Europa agieren möchte, und daraufhin auch den angepassten Steuersatz benötigt. Ist dieser schon bei einem anderen Mandanten eingerichtet so kann man diese Konfiguration kopieren. In Abbildung 4.4 wird ein solches Szenario aufgezeigt. Hierbei soll der Wert der Testumgebung von Mandant 1 zu den Mandanten 2 und 3 kopiert werden.

Das Kopieren von Werten oder ganzen Konfigurationen ist auch dann nötig, wenn ein neuer Mandant angelegt werden muss. Meist können bestehende Konfigurationen zum größten Teil übernommen werden.

Anstatt die Werte nur zu kopieren besteht manchmal auch die Anforderung die Werte zu vererben, oder anders ausgedrückt, diese von einander abhängig zu machen.

Abbildung 4.4: Mandant Kopieren Problem

	Man. 1	Man. 2	Man. 3	Man. 4	...
DEV					
TEST	$\rightarrow$				
INT					
PROD					

Das bedeutet, sobald Änderungen an dem ”Original” vorgenommen werden, sind diese automatisch auch bei dem Mandant vollzogen der davon abhängt. Analog zum ersten genannten Szenario ließen sich so Standardwerte für eine bestimmte Gruppe von Mandanten einrichten. Wird beispielsweise eine Schnittstelle verändert, die von mehreren Mandanten genutzt wird, müssen nur bei einem Mandanten die entsprechenden Parameter angepasst werden. Die anderen Mandanten erhalten diese Änderung automatisch.

Alle hier genannten Probleme treten auch kombiniert auf. So ist es vorstellbar, dass sich ein Mandant in einigen Parametern von allen anderen Mandanten unterscheidet, gleichzeitig soll seine Konfiguration auf der Testumgebung der neue Umgebungsstandard werden. Durch die Vielzahl an Szenarien entsteht die hohe Fehlerrate. Entwickler übersehen immer wieder Parameter die sie ändern sollten oder ändern Falsche.

In Listing 4.2 sieht man einen Ausschnitt aus einer Konfigurationsdatei. Hier wird ein Exportverzeichnis konfiguriert. Unter ”name” steht jeweils ein Kürzel welches den jeweiligen Mandanten identifiziert. Dahinter der Verzeichnispfad. Man sieht, dass alle Verzeichnispfade gleich sind, dennoch muss für jeden Mandant eine Zeile angelegt werden. Außerdem ist der Ausschnitt nur aus der Datei für die Integrationsumgebung. Das heißt, die gleichen Zeilen stehen noch drei weitere male in den Dateien für die anderen Umgebungen. Sollen die eben genannten Szenarien umgesetzt werden, muss sehr viel Zeit investiert werden um die richtigen Parameter zu finden. Diese müssen dann manuell angepasst werden. Eine Vererbung, wie im letzten Szenario beschrieben ist nicht möglich.

Listing 4.2: INT-Konfig-Ausschnitt

```
1 <cfg:configuration-parameter>
```

```
2     <cfg:parameter-name>exportFileDir</cfg:parameter-name>
3     <cfg:parameter-value name="060">/srv/jas/app/epep/baskets/
         EPEPOUT/</cfg:parameter-value>
4     <cfg:parameter-value name="1521">/srv/jas/app/epep/baskets/
         EPEPOUT/</cfg:parameter-value>
5     <cfg:parameter-value name="154">/srv/jas/app/epep/baskets/
         EPEPOUT/</cfg:parameter-value>
6     <cfg:parameter-value name="1542">/srv/jas/app/epep/baskets/
         EPEPOUT/</cfg:parameter-value>
7     <cfg:parameter-value name="020">/srv/jas/app/epep/baskets/
         EPEPOUT/</cfg:parameter-value>
8     <cfg:parameter-value name="030">/srv/jas/app/epep/baskets/
         EPEPOUT/</cfg:parameter-value>
9     <cfg:parameter-value name="069">/srv/jas/app/epep/baskets/
         EPEPOUT/</cfg:parameter-value>
10    <cfg:parameter-value name="1590">/srv/jas/app/epep/baskets/
         EPEPOUT/</cfg:parameter-value>
11    <cfg:parameter-value name="1575">/srv/jas/app/epep/baskets/
         EPEPOUT/</cfg:parameter-value>
12 </cfg:configuration-parameter>
```

4.3 Anforderungsbeschreibung

Die Hauptanforderung an eine Lösung besteht darin, dass der Aufwand reduziert wird, der für die Wartung eines Projektes in Hinsicht auf die Konfiguration erbracht werden muss. Dazu sollen die im vorausgehenden Abschnitt beschriebenen Probleme berücksichtigt werden. Es ist nicht notwendig, dass für jedes Problem eine perfekte Lösung existiert. Aber es soll möglich sein, die meisten Konfigurationsänderungen in kurzer Zeit zu erledigen. Des Weiteren ist es wichtig, dass die gefundene Vorgehensweise Übersichtlichkeit gewährt, um Änderungen, Unterschiede oder Fehler schnell zu finden.

Tabelle 4.1: Anforderungen

Nr.	Problem	Zusammenfassung	Beispielsituation	Relevanz
1	Default Problem (DP)	Ein Mandant weicht von einem Standardwert ab der für alle anderen Mandanten Gültigkeit hat.	Steuersatz-Konfiguration für einen Mandanten anderer Nationalität	sehr hoch
2	Abweichende Umgebung (AUP)	Ein Wert weicht für eine Umgebung von allen anderen Umgebungen ab und dies über alle Mandanten hinweg.	Backup-Intervall für die Testumgebung	sehr hoch
3	Umgebung Kopieren Problem (UKP)	Bestehende Konfigurationen sollen auf eine andere Umgebung kopiert werden.	Nach dem Test sollen INT-Werte nach PROD kopiert werden.	mittel
4	Mandant Kopieren Problem (MKP)	Konfigurationen sollen von einem Mandanten zu einem anderen kopiert werden.	Angepasster Steuersatz soll übernommen werden	mittel
5	Vererbung (VP)	Eine Konfiguration in Feld B soll immer gleich sein wie Feld A. Auch wenn Feld A geändert wird.	Standard-Werte nur für bestimmte Gruppen von Mandanten.	niedrig

In Tabelle 4.1 sind die Probleme noch einmal kurz zusammengefasst. Es ist nicht gefordert, dass alle Probleme in der Lösung behandelt werden, es sollten jedoch möglichst viele von den genannten Problemen vereinfacht werden. Bei einem Vergleich mehrerer Ansätze werden die Probleme mit verschiedener Relevanz betrachtet, die in der Tabelle angegeben ist. Die Relevanz ergibt sich aus Gesprächen mit Entwicklern. Diese schätzen ein welche Szenarien am häufigsten Auftreten, sodass die Vereinfachung dieser zu einer stärkeren Verringerung des Wartungsaufwandes führt.
Eine weitere Funktion ist das automatische Erneuern der Konfiguration während des laufenden Betriebes. Bisher ist es dem Entwicklerteam nur durch einen Neustart des Servers möglich neue Konfigurationen zu laden. Ein Mechanismus der dies während der Serverlaufzeit ermöglicht ist wünschenswert um die Unterbrechungszeit des Betriebs während einer Konfigurationsänderung zu verkürzen.

5

Informationsbeschaffung

Zunächst wurde versucht mit Hilfe von bestehender Literatur ein Lösungsmuster zu finden. Allerdings hat eine ausgiebige Suche keine nennenswerten Ergebnisse gebracht. Aus diesem Grund wurde entschieden Experteninterviews zu führen. Als Experten werden dazu Mitarbeiter der TSS ausgewählt, die nach Einschätzung der Abteilungsleiter oder ihrer Kollegen Erfahrung auf dem Gebiet der Konfiguration mitbringen. Speziell wurde darauf Wert gelegt, dass diese Experten auch Erfahrung im Umgang mit mehreren Mandanten haben.

5.1 Aufbau der Interviews

Der Aufbau der Interviews orientiert sich an [May06]. Darin wird empfohlen offene Fragen zu stellen, sich aber dennoch an einen Leitfaden zu halten um die Vergleichbarkeit zu

steigern. Zu Beginn des Interviews werden Informationen über den Experten gesammelt. Dabei steht seine Erfahrung mit der Konfiguration im Vordergrund. Anhand der Problemanalyse wird dann dem Mitarbeiter das Problem erklärt. Daraufhin folgen die Fragen zur Lösung. Diese äußern sich als konkrete Beispiele die aus den Problemen hervorgehen. Die Frage ”Wie würden Sie bei diesen Szenarien vorgehen?” steht im Mittelpunkt. Neben dieser Frage ist es wichtig, den Aufbau der Applikation zu erfahren. Das sind alle Informationen bezüglich Anzahl der Mandanten und Umgebungen, so wie die Häufigkeit mit der Konfigurationsänderungen vollzogen werden müssen. Dies hilft dabei abzuschätzen in wie fern die Erfahrungen des Experten für diese Arbeit relevant sind. Die Beispiele aus der Problemanalyse werden als fiktive Szenarien vorausgesetzt, das heißt alle Annahmen über Konfigurationsparameter oder Mandanten sind gegeben. Trifft ein Beispiel nicht genau auf die Applikation des Experten zu, wird dennoch eine naheliegende Lösung gesucht. Die Formulierung der Fragen setzt voraus, dass der Mitarbeiter vorher mit der Problemanalyse vertraut gemacht worden ist. Die Fragen wurden ausgewählt um jedes Teilproblem aus Kapitel 4 abzudecken. Die Beispiele wurden so formuliert, dass die Problemstellung unabhängig vom eigentlichen Zweck der Applikation klar wird. Um dies zu erreichen werden fiktive Parameter eingeführt, die auf eine bestimmte Art und Weise geändert werden sollen. Dadurch können sich die Experten auf ähnliche Szenarien aus ihrem Arbeitsumfeld beziehen, und man erhält dennoch von jedem eine Antwort auf dieselben Beispiele. Diese gestalten sich folgendermaßen:

Das Default-Problem Wie sieht Ihre Konfiguration aus wenn ein Mandant einen anderen Steuersatz hat, wie alle anderen Mandanten?

Abweichende Umgebung Angenommen das Backup der Applikation wird nicht von einem externen Programm übernommen, wie konfigurieren Sie ihre Umgebungen wenn das Backup-Intervall einer Datei auf der Testumgebung bei einer Woche liegen soll, während es auf allen anderen Umgebungen täglich sein soll.

Umgebung Kopieren Problem Wie gehen sie vor, wenn sie bestimmte Werte, die auf der Integrationsumgebung getestet wurden, auf die Produktionsumgebung übertragen wollen?

Mandant Kopieren Problem Ein neuer Mandant wird angelegt und es sollen Konfigurationen eines bestehenden Mandanten kopiert werden. Wie gehen sie vor?

Vererbung Sie haben bei einem Mandanten eine Schnittstelle konfiguriert. Sie wollen diese Konfiguration jetzt für andere Mandanten vererben lassen. Das heißt, sie müssen bei einer Änderung nur den Mandanten verändern, von dem die Anderen geerbt haben. Wird bei diesem eine Änderung vollzogen wirkt sich diese automatisch auf die entsprechenden Mandanten aus.

5.2 Interviews

In diesem Kapitel wird der Verlauf der Interviews beschrieben. Zur Auswahl der Experten wurde am Anfang auf Projekte gesetzt, die bekannte Softwarepakete verwenden, in der Hoffnung, dass diese viele Mandanten und fertige Lösungen beinhalten. Da dies aus verschiedenen Gründen nicht geklappt hat, (diese werden im nächsten Abschnitt beschrieben,) wurde die Expertenauswahl auf das Java Umfeld beschränkt. Diese haben einen besseren Bezug zu dem Problem, da die Projektgröße etwa ähnlich zu der von ePEP ist. Außerdem ist das Meiste davon so genannte Custom-Software, das heißt von Beginn an selbst entwickelte Software. Es gibt also meist einen oder mehrere Experten, die sich mit der Konfiguration auseinandergesetzt haben.

5.2.1 Cognos/SAP

Zuerst wurden Experten von Projekten interviewt die Standardsoftware verwenden. Solche Projekte sind in der TSS unter anderem SAP und Cognos. Für beide Softwarepakete bietet die TSS eine Beratung und Einrichtung an. Die Software wurde also nicht wie im Falle von ePEP selber entwickelt, sondern von SAP bzw. IBM übernommen. Lediglich die Einrichtung und einige kleine Anpassungen werden von der TSS übernommen. In den Interviews hat sich herausgestellt, dass die Anforderungen und Probleme aus dem vorhergehenden Kapitel so bei den Projekten überhaupt nicht eintreten. Die Mandanten sind viel strikter von einander getrennt. Für jeden Mandant wird eine eigene Umge-

bung aufgesetzt und jeder Mandant wird einzeln beraten und betreut. Weiter ist die Art der Konfiguration eine ganz andere. Während im Projekt ePEP auch viele funktionale Anforderungen über die Konfiguration bearbeitet werden können, lassen sich in den genannten zwei Projekten hauptsächlich Schnittstellen konfigurieren, mit denen man das Firmennetz mit dem Softwarepaket verbindet. Diese Schnittstellen ändern sich nur sehr selten, weshalb man in diesen Projekten die Probleme, welche durch häufige Änderungen erst wirklich in den Vordergrund treten, nicht kennt. Des Weiteren werden bei ePEP die Werte auch weitestgehend von den Mandanten vorgeschrieben und ändern sich häufig. Bei den Beratern der zwei großen Projekte werden die Werte mit den Mandanten erarbeitet und dann meist nicht mehr geändert, außer es treten Fehler auf. Aus den drei Interviews mit den Experten, zwei aus dem SAP-Umfeld einer aus dem Cognos-Umfeld, konnte man aus den genannten Gründen keine erwähnenswerten Erkenntnisse zu einem Lösungsansatz ziehen. Dennoch eignen sich diese Interviews um erste Erkenntnisse über den Interviewablauf und die Experten zu gewinnen. Die Problemanalyse wurde noch einmal überarbeitet und die Fragen und Szenarien so formuliert, dass es den Experten leichter fällt diese nachzuvollziehen.

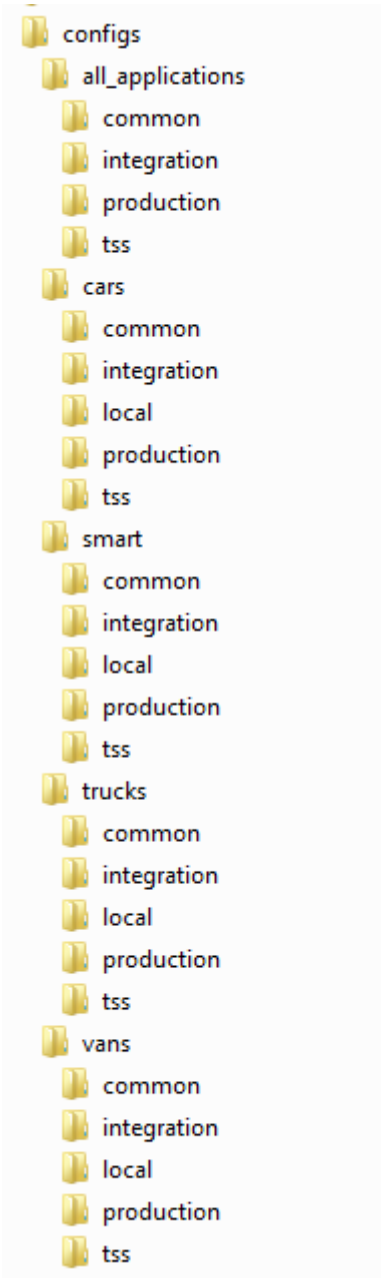
5.2.2 Java-Umfeld

Da die ersten Interviews bei den großen Projekten nicht so liefen wie erwartet wurde die Problemstellung noch einmal überarbeitet und die Fragen anders gewählt. Außerdem wurde der Fokus auf die Java Abteilung innerhalb der TSS gerichtet, da dort alle Software selbst entwickelt wird. Dort entspricht unter Anderem auch die Projektgröße etwa der von ePEP. Weiterhin ist das Ziel der Arbeit eine Handlungsempfehlung für eine javabasierte Umgebung zu entwickeln, das heißt, dass die Erkenntnisse aus den Experteninterviews eins zu eins übertragen werden können ohne technische Anpassungen vornehmen zu müssen.

Erstes Interview

Vor dem Interview wurde dem Experten ein Auszug aus der Problemanalyse zugeschickt. Daraufhin konnte dieser entscheiden, ob er für die Fragen genug Fachwissen besitzt, oder ob man sich doch besser an einen Anderen wendet. Des Weiteren hatte er so Zeit sich auf die Fragen und Probleme vorzubereiten und wurde nicht innerhalb des Interviews mit Fragen überrascht auf die er spontan keine Antwort geben kann. Diese Maßnahme hat sich als sehr wirkungsvoll erwiesen, denn direkt im ersten Interview gab es nützliche Erkenntnisse. Der Experte verwendet in seinen Projekten zwei Systeme, die jeweils auf einer Hierarchie aufbauen. Bei der ersten Variante wird ein Baum aufgebaut, der pro Mandant einen Ordner enthält in denen pro Umgebung ein weiterer Ordner ist. In jedem Ordner ist eine Konfigurationsdatei, die neue Parameter oder Parameter einer höheren Hierarchieebene enthalten kann. Der Screenshot in Abbildung 5.1 zeigt diesen Aufbau. Während dem Build-Prozess des Programmes wird dann ausgewählt für welchen Mandant und welche Umgebung das Programm kompiliert werden soll. Daraus wird dann automatisch eine einzige Konfigurationsdatei erzeugt, welche die jeweiligen Werte aus der Hierarchie wiedergibt. In der zweiten Variante wird nur eine Hierarchiestufe verwendet. Es existiert eine zentrale Default-Datei, die alle Standardwerte enthält. Pro Mandant gibt es eine weitere Datei, welche mandantenspezifisch die Standardwerte ergänzen oder ersetzen kann. Neben diesen Mandanten-Dateien gibt es auch noch Umgebungsdateien in denen umgebungsspezifische Werte eingetragen werden. Die Menge der Werte in den Umgebungsdateien und die Menge der Werte in den anderen Dateien sind jedoch disjunkt. Das heißt, die umgebungsspezifischen Werte kollidieren nicht mit Werten aus der zentralen Datei oder den Mandantendateien. Das liegt daran, dass es in diesem Projekt keine unterschiedlichen funktionalen Anforderungen an die Umgebungen gibt. Es werden nur Parameter wie IP-Adressen oder Verzeichnisnamen konfiguriert, die sich von Umgebung zu Umgebung unterscheiden. Das Programm wird auf jeder Laufzeitumgebung mit der vollständigen Konfiguration ausgeliefert und installiert. Die richtigen Parameter werden mit Hilfe eines beim Start angegebenen Run-Modes ausgewählt. Das hat den Vorteil, dass nicht viele verschiedene Builds existieren, sondern auf jeder Umgebung das absolut gleiche Programm läuft.

Abbildung 5.1: Ordnerstruktur des genannten Projektes



Am Ende des Interviews empfahl der Experte einen weiteren Mitarbeiter, der seiner Meinung nach ein gutes Verständnis für die Thematik hat.

Zweites Interview

Der Empfehlung des ersten Experten folgend wurde das zweite Interview mit dem genannten Mitarbeiter geführt. Dieser beschäftigt sich mit einem Verfahren um speziell umgebungsspezifische Parameter zu konfigurieren. Um dies zu erreichen hat er mit dem Framework X-text eine DSL (Domain Specific Language) entwickelt. Ziel dieser Sprache ist es, dass an einer Stelle alle Werte eines Parameters, bzw. einer Gruppe von Parametern, zusammengefasst sind. Diese werden jeweils mit Schlüsselwörtern versehen welche angeben zu welcher Umgebung diese gehören. Listing 5.1 zeigt einen Ausschnitt des Codes in dem eine solche Konfiguration durchgeführt wird.

Listing 5.1: configDSL

```
1  package configurations start
2     configuration pid com.daimler.map.cq.base.services.
         MapMailProvider start
3       keys [ defaultEmailInt, defaultEmailTic ]
4
5       instance [ daimler_dev, dev-test_general ]
6       values [
7        key defaultEmailInt -> "{String}
            tss_map_internal@daimler.com"
8        key defaultEmailTic -> "{String}
            tss_map_internal@daimler.com"
9       ],
10      instance [ daimler_int ]
11      values [
12       key defaultEmailInt -> "{String}comin.test@daimler.com"
13       key defaultEmailTic -> "{String}comin.test@daimler.com"
14      ],
```

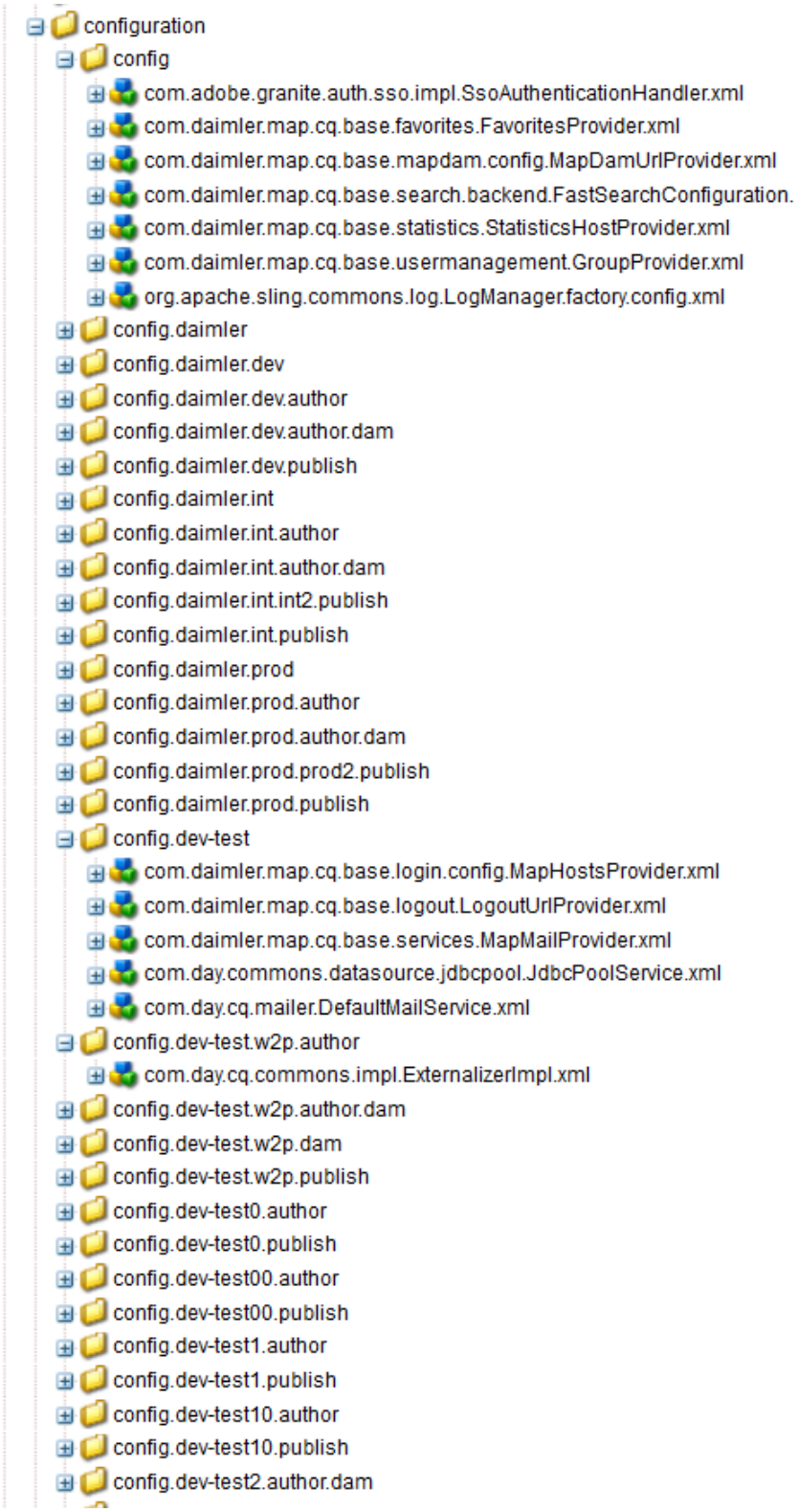
```
15          instance [ daimler_prod ]
16          values [
17           key defaultEmailInt -> ""
18           key defaultEmailTic -> ""
19          ]
20      end
21  end
```

Genauer sieht man einen Ausschnitt der genutzt wird, um einen MailProvider zu konfigurieren. Mit der Hilfe dieser DSL können nun einzelne Werte je nach Umgebung zugewiesen werden. Das Schlüsselwort instance gibt an, für welche Umgebungen die darauf folgenden Werte gültig sind. Die DSL generiert aus diesem Code eine Ordnerstruktur, die dann von dem Programm eingelesen wird. Welche Konfiguration geladen wird, wird durch einen Run-Mode bestimmt, den man beim Start angibt. Abbildung 5.2 zeigt die generierte Ordnerstruktur. Je nach Run Mode, beispielsweise ”author, dev-test, w2p” werden dann die passenden Parameter geladen. Die in der Abbildung aufgeklappten Ordner sind jene, welche für die gegebenen Beispielumgebungen geladen werden. Im oberen Teil der Abbildung sind die Standardwerte, die immer geladen werden. Weiter unten sieht man den Ordner für *dev-test*, so wie die weitere Spezialisierung, welche alle drei Umgebungen repräsentiert.

Drittes Interview

Der Experte im dritten Interview stammt aus einem Projekt, in dem die Konfigurationsdateien größer als 300MB sind. Die Konfigurationsdateien in ePEP sind etwa 2MB groß. Dieser Vergleich lässt die Vermutung zu, dass das Interview auf ähnliche Probleme stoßen wird, wie sie schon beim Interview zu den Themen SAP und Cognos auftraten. Entgegen dieser Vermutung konnte der Experte allerdings sehr genau beschreiben wie der Konfigurationsmechanismus in dem Projekt aufgebaut ist. Das liegt vor allem daran, dass auch dieses Projekt in der TSS komplett entwickelt wurde, ohne auf Fremdsoftware zu setzen. Das Projekt betreut auch mehrere Mandanten und hat ähnlich zu ePEP mehre-

Abbildung 5.2: Screenshot der erzeugten Dateien



re Entwicklungsumgebungen. Das Projekt nutzt themenbezogene Konfigurationsdateien um schneller zusammenhängende Parameter zu finden. Die Übersicht, welche Werte in welcher Datei zu finden sind, kann nur durch eine sehr strikt eingehaltene Dokumentation im Code stattfinden. Es wird zu jedem geladenen Parameter im Code ein Kommentar eingefügt, der genaue Informationen über die Konfigurationsdatei enthält, in der man den Parameter finden kann. In den Dateien sind die Parameter als Standardwerte definiert und direkt unter den Standardwerten werden abweichende Werte notiert. Dies ist in dem Projekt sehr übersichtlich, da der Großteil der Werte sich nicht unterscheidet. Um die verschiedenen Umgebungen zu trennen, werden die Konfigurationsdateien, ähnlich wie bei ePEP, mehrmals nebeneinander gehalten. Allerdings ist es durch die thematische Trennung nicht notwendig alle Dateien nebeneinander zu halten, sondern nur diejenigen, welche Unterschiede zwischen den Umgebungen abbilden. Ein Tool zum Auslesen und Ändern der Konfigurationsdateien verwendet das Projekt nicht.

Viertes Interview

Das Team des Projektes, das Gegenstand des vierten Interviews war, verwendet zur Konfiguration eine Datenbank. Diese enthält eine Standardkonfiguration. Des Weiteren enthält sie die Werte, welche sich mandantenspezifisch unterscheiden. Da alle Werte in derselben Datenbanktabelle liegen, werden Mandanten-IDs verwendet, um diese zu unterscheiden. Pro Mandant wartet das Team ein SQL-Skript in dem alle Werte, die für diesen Mandanten spezifisch sind, aufgeführt werden. Um nun einen Wert zu ändern reicht es in diesem Skript eine Anpassung vorzunehmen und es einmal auszuführen. Die Änderungen werden dann passend erzeugt. Während die Applikation läuft, wird immer erst versucht eine spezifische Konfiguration zu laden, je nachdem welcher Mandant sich einloggt. Ist keine vorhanden, gibt es einen Fallback auf die Standardkonfiguration. Auch die umgebungsspezifischen Parameter werden in dieser Datenbank auf die gleiche Art und Weise gehandhabt. Der Experte betonte jedoch, dass es zwischen mandantenspezifischen und umgebungsspezifischen Werten keine Überschneidung gibt. Dies wurde von Anfang an so gehandhabt um die Komplexität geringer zu halten.

5.3 Diskussion

In diesem Abschnitt werden die aus den Interviews gewonnenen Erkenntnisse zusammengefasst und verglichen. Dabei wird erarbeitet, welche Probleme sich lösen lassen und welche nicht. Da alle Experten im Grunde genommen auf einem hierarchischen Prinzip aufbauen, werden die verschiedenen Umsetzungen davon genauer betrachtet, unter der Fragestellung welche Umsetzung in welcher Situation angebracht ist. Eine Entscheidung für eine Umsetzung sollte nicht willkürlich, sondern gezielt aufgrund des Software-Designs getroffen werden. Im Folgenden wird das grundlegende Prinzip noch einmal erläutert, und es wird betrachtet, welche Probleme und Anforderungen sich damit vereinfachen lassen. Im Anschluss werden die verschiedenen Umsetzungen noch einmal verglichen und evaluiert.

5.3.1 Auswertung

Alle Experten verwenden für ihr Projekt eine Hierarchie zur Konfigurierung. Damit lässt sich vor allem das DP (Default-Problem) recht einfach lösen und je nach Umsetzung auch das AUP (abweichende Umgebung). Verwendet wird dabei ein Mechanismus der zunächst prüft ob es einen speziellen Wert für den aktuellen Mandanten gibt. Erst wenn keiner gefunden wird, werden die Standardwerte verwendet. Dabei unterscheiden sich die Umsetzungen vor allem darin, wie und wann die Werte geladen, und wie die Informationen gespeichert und geordnet werden. Die Kernkomplexität, dass abweichende Werte extra aufgelistet werden müssen, bleibt demnach erhalten. Wenn sich also jeder Wert individuell unterscheidet, gibt es keine bessere Möglichkeit als alle Werte manuell aufzulisten, da die Daten zwingend vorhanden sein müssen. Die folgenden Mechanismen zur Strukturierung der Konfigurationsdaten sind also nur eine Erleichterung für die Übersicht und reduzieren nur dann die Datenmenge, wenn man gleiche Attribute zusammenfassen kann. Da dies häufig vorkommt, und die hohe Fehlerrate bei Änderungen ihren Ursprung vor allem in der Unübersichtlichkeit hat, wird dennoch eine deutliche Verbesserung bei Konfigurationsarbeiten erzielt.

Aus den Interviews lassen sich verschiedene Möglichkeiten der Datenhaltung ableiten. Zur Speicherung der Parameter werden vor allem XML Dateien eingesetzt. Es werden aber auch .properties-Dateien und Datenbanken eingesetzt. Zur Hierarchisierung werden die Daten entweder voneinander getrennt, also in verschiedene Dateien oder Tabellen geschrieben, oder alle in einer Datei gehalten. Dabei muss innerhalb der Datei eine Möglichkeit der Kennzeichnung gefunden werden.

Anders als erwartet konnte kein Experte ein allgemeines Tool nennen, mit dem man die Konfigurationsdaten und Beziehungen visualisieren kann. Die Experten berichteten aber auch, dass sich ihr Konfigurationsaufwand in Grenzen hält und dieser nie ein zentrales Problem war. Das Thema ist innerhalb ihrer Entwicklungsteams nur wenige Male angesprochen worden. Das steht in starkem Gegensatz zu den Erfahrungen, die im Projekt ePEP gemacht wurden. Doch da die Größe der Konfigurationsdateien in den Projekten der Experten jene von ePEP zum Teil stark überschreitet, ist es wahrscheinlich, dass die Anforderungen im Projekt ePEP über die der anderen hinausgehen und das der Grund für die Probleme ist. Die Interviews geben dennoch den Eindruck, dass ein strukturiertes Vorgehen die Fehlerrate deutlich senken kann und für eine bessere Übersicht sorgt. Aus diesem Grund werden an dieser Stelle nun die einzelnen Vorgehensweisen, welche die Experten genannt haben, jeweils unter Zuhilfenahme der Kernprobleme, verglichen.

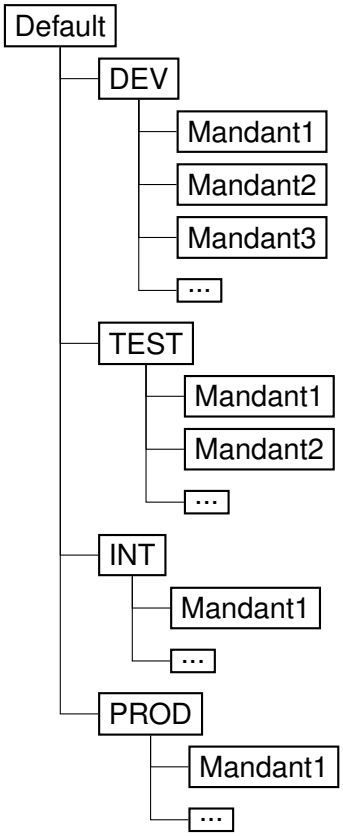
Default Problem

Das DP wird von allen Lösungsansätzen hinreichend gelöst. Dazu verwenden diese eine hierarchische Anordnung von Konfigurationsparametern. Es gibt einen Standardwert und nur die Abweichungen werden notiert. Dieses Vorgehen wurde von allen Experten so beschrieben, was zu der Schlussfolgerung führt, dass dies eine praxisnahe Lösung ist. Da dies weiterhin das Problem mit höchster Priorität ist, erweist sich der hierarchische Ansatz als sehr gute Lösung. Die Tabelle 5.1 zeigt die Antworten, die jeder Experte auf die genannten Beispiele gegeben hat. Darin ist sehr gut zu sehen, dass die einzelnen Projekte prinzipiell dasselbe Vorgehen haben.

Tabelle 5.1: DefaultProblem

Experte	Antwort
1	In der Datei des betreffenden Mandanten werden die Daten für den Steuersatz angepasst. Alle anderen Mandanten greifen auf den Steuersatz in der Standard-Datei zurück.
2	Der Ansatz wurde nur für Unterschiede zwischen den Umgebungen entwickelt, es ist aber klar, dass man diesen sehr ähnlich auch auf Mandanten anwenden kann. Dazu gibt man im Code, so wie später im Run-Mode, statt der abweichenden Umgebungen jeweils die Mandanten an.
3	In jeder Datei stehen direkt unter dem Standardwert die mandantenspezifischen Werte. Um den Steuersatz anzupassen wird der neue Wert mit dem Verweis auf den entsprechenden Mandanten unter den Standard-Steuersatz geschrieben.
4	In der Datenbank wird der abweichende Steuersatz als Wert mit der Mandanten ID abgespeichert. Alle anderen Mandanten greifen auf den Standard Eintrag zurück.

Abbildung 5.3: Baumstruktur



Abweichende Umgebung

Das AUP tritt bei fast allen Experten nicht in Kombination mit dem DP auf. Durch klare Richtlinien, welche auch so in der Dokumentation stehen, sind Entwickler daran gehindert mandantenabhängige Parameter auch umgebungsspezifisch zu verwenden. Solange man diese Trennung klar ausführen kann, ist es möglich zwei Hierarchien nebeneinander aufzubauen. Dies ist aufgrund der Anforderungen einiger Projekte nicht immer möglich, wie beispielsweise im Projekt ePEP. Tabelle 5.2 zeigt die Antworten der Experten für dieses Problem. Obwohl alle Experten eine strikte Trennung empfehlen, ist es vorstellbar, dass man eine Hierarchie mit drei Ebenen aufbaut. Abbildung 5.3 zeigt eine Struktur, mit der Mandanten und umgebungsspezifische Paramter verwaltet werden können, wie sie auch im ersten Projekt des ersten Experten verwendet wird. Hier ergibt sich jedoch ein Problem, wenn ein Mandant von dem Standardwert abweicht, denn dann muss der abweichende Wert pro Umgebung einmal eintragen werden, was wieder sehr fehleranfällig ist. Dem lässt sich entgegenwirken, wenn man auf Höhe der Default-Datei noch, pro Mandant, eine weitere Datei einfügt. In dieser kann man jene Parameter eintragen, in welchen sich die Mandanten vom Standard unterscheiden, die aber nicht umgebungsabhängig sind. Dadurch gibt es aber noch mehr Dateien und je nach Anforderung kann sich die Lage eines Parameters ändern was sehr unübersichtlich und schlecht nachvollziehbar ist. Daher arbeiten alle Experten mit streng getrennten Schlüsseln.

Umgebung Kopieren

Die Experten haben alle erklärt, dass ein Kopieren von Umgebungskonfigurationen bei ihnen nicht gebraucht wird, da nur technische Parameter umgebunsspezifisch konfiguriert werden. Technische Parameter sind beispielsweise Serveradressen oder Namensbereiche. Was jedoch nicht auf Umgebungsebene unterschieden wird, sind funktionale Anforderungen. Diese müssen in den befragten Projekten nicht extra auf den Testumgebungen getestet werden. Die einzige Ausnahme ist das Projekt des zweiten Experten, der die DSL entwickelt hat. Dieser hat genau aus dem Grund diese Sprache entwickelt. Um eine bestehende Konfiguration auf eine andere Umgebung zu übernehmen muss

Tabelle 5.2: AbweichendeUmgebung

Experte	Antwort
1	In den Konfigurationsdateien auf der Hierarchiestufe der Umgebungen wird das Backupintervall eingetragen. Bei dem zweiten Projekt, das der Experte genannt hat, werden die Backupintervalle in die Umgebungsdateien eingetragen.
2	Es wird eine DSL Datei für das Backupintervall angelegt, darauf werden die Umgebungen mit dem Schlüsselwort instance festgelegt bei denen die jeweiligen Werte gelten sollen. (Vergleiche Listing 5.1)
3	Der Parameter wird in die passende Konfigurationsdatei eingetragen. Das bedeutet, es wird eine Datei gesucht die thematisch ähnliche Parameter enthält. Da nicht alle Dateien für jede Umgebung zur Verfügung stehen, wird eine Datei ausgewählt die bereits für alle Umgebungen existiert. Ist dies nicht möglich oder thematisch widersprüchlich, wird eine neue Datei angelegt.
4	In der Datenbank wird der Wert für das Backupintervall auf den Standard gesetzt. Für die Testumgebung wird der Wert noch einmal, mit einer speziellen ID versehen, abgelegt.

er im Code der DSL das Schlüsselwort der neuen Umgebung für den Parameter hinzufügen, der kopiert werden soll. Dieser steht dann auch auf der Zielumgebung zur Verfügung. Die anderen Mitarbeiter gaben an, dass man einige spezielle Parameter mit in die Umgebungs-Dateien schreiben kann, dann fällt aber die Möglichkeit weg, diese noch nach Mandanten zu unterscheiden. Das Projekt ePEP hat jedoch die Anforderung, Parameter sowohl nach Mandanten, als auch nach Umgebungen unterschiedlich zu konfigurieren. Soll das erreicht werden, kann ein Drei-Ebenen-Aufbau, wie im vorherigen Abschnitt (Abweichende Umgebung) erklärt, verwendet werden. Darin erfolgt das Kopieren der Dateien jeweils manuell. Wenn alle Parameter übernommen werden können, so kann einfach die komplette Datei kopiert werden. Ist das allerdings nicht gewünscht, ist manuelles Ersetzen in den Dateien gefordert.

Mandant Kopieren

Das Kopieren von Mandanten kann aufgrund der Hierarchien von allen Experten schnell und problemlos vollzogen werden. Existiert eine Default Datei, so muss man einfach nur eine neue Datei für den neuen Mandanten anlegen. Die speziellen Werte lassen

sich einfach durch kopieren der spezifischen Datei übertragen. Sind die abweichenden Werte nicht in unterschiedlichen Dateien gespeichert, so ist deutlich mehr Aufwand nötig. Dann muss bei jedem Parameter nach einem spezifischen Wert gesucht werden und dieser muss gegebenenfalls kopiert und noch einmal für den neuen Mandanten eingefügt werden.

Vererbung

Eine Vererbung, wie sie im Kapitel 5.1 beschrieben ist, konnte keiner der Experten mit seinem Vorgehen erzeugen. Da der Mandantenkreis aber nie über zehn Mandanten hinaus gewachsen ist, ist es mit einer bestehenden Default-Datei auch gar nicht unbedingt nötig eine solche Vererbung aufzubauen. Die meisten Fälle sind durch den Standardfall abgedeckt. Es bleiben einige wenige Fälle die abweichen. Im schlimmsten Fall sind das 8 Parameter die einer Änderung bedürfen. Denn sobald mindestens zwei den gleichen Wert haben kann man diesen sinnvollerweise als Standard setzen. Es bleiben demnach 8 Mandanten die noch zu konfigurieren sind. Hat man nun je ein Mandantenpaar, die dieselben Werte teilen, also 4 Paare, so muss man mit Vererbung 4 Konfigurationsvorgänge vornehmen, und ohne Vererbung 8. Für den schlimmsten Fall ergibt das eine Ersparnis von 4 Konfigurationsvorgängen, was relativ gering ist. Zwar würde eine Vererbung garantieren, dass entsprechende Mandanten immer denselben Wert haben, und so eventuellen Fehlern vorbeugen, doch würde eine Erweiterung des Hierarchie-Konzeptes auch einen Aufwand darstellen. Theoretisch spricht jedoch nichts dagegen, als Konfigurationswert einen anderen Mandantennamen einzutragen. Der Parser, der die Dateien auswertet, muss diesen Namen erkennen und auf den Wert des genannten Mandanten ausweichen. Bei der gegebenen Mandantenzahl und dem Vorhandensein eines Default-Wertes ist dies jedoch nicht sehr viel effizienter.

Frameworks und Tools

Neben dem Aufbau der Dateien und dem Konzept der Konfiguration wurden die Experten auch nach hilfreichen Tools gefragt, die sie bei der Konfiguration verwenden. Während

keiner ein spezielles Programm zur Verwaltung und zur Ansicht der XML Dateien benutzte waren sich alle Experten einig, was das Laden der Konfiguration im Code an geht. Alle benutzten *Apache Commons Configuration*. Dies ist eine Software Bibliothek, die es Java Programmen erlaubt von verschiedenen Quellen Konfigurationsparameter zu laden [ASF14]. Aus diesem Grund werden die Grundlagen dieses Frameworks in der Handlungsempfehlung erläutert.

Zusammenfassung

Da eine Literaturrecherche aus den in Kapitel 3 genannten Gründen ohne Erfolg blieb, musste auf eine andere Informationsquelle zurück gegriffen werden. Da das Arbeitsumfeld die Möglichkeit bietet Entwickler zu interviewen, welche täglich mit Konfiguration konfrontiert sind, waren Experteninterviews die nächstliegende Lösung. Diese verliefen anfangs nicht wie geplant, doch nach einer Überarbeitung der Fragestellung erbrachten diese die gewünschten Ergebnisse. Im Allgemeinen waren sich alle Experten einig, dass die Konfiguration für die befragten Projekte kein allzu großes Problem darstellt. Während den Interviews sind jedoch deutliche Abweichungen der Anforderungen zum Projekt ePEP aufgefallen. Dennoch lassen sich die Mechanismen, welche die Experten benutzen, auch für das Projekt ePEP nutzen. Der hierarchische Ansatz ist bei richtiger Dokumentation so effektiv, dass der Konfigurationsaufwand deutlich eingeschränkt wird. Die Auswertung zeigt, dass die Problemsituationen mit der höchsten Relevanz deutlich vereinfacht werden. Dennoch muss klar sein, dass die Kernkomplexität nicht entfernt werden kann. Jede Abweichung von einem Standard muss natürlich manuell konfiguriert werden, häufen sich diese Abweichungen, so steigt der Aufwand selbst mit dem hierarchischen Modell schnell an. Neben dem Konzept zum Aufbau der Konfigurationsdateien haben die Interviews noch einige weitere wichtige Informationen zutage gefördert. Es ist sehr wichtig sich vor der Entwicklung eines Programmes Gedanken darum zu machen was alles konfiguriert werden soll und wie sehr man bestimme Parameter einschränkt. Alle Experten haben dazu geraten die mandantenspezifischen und die umgebungsspezifischen Werte strikt zu trennen. Dies sollte aber nur von Beginn an ausgeschlossen, und nicht in einer bestehenden Applikation nachgezogen werden, da dies zu erheblichen

Eingriffen in diese führt. Aber auch die Dokumentation der Parameter im Code ist sehr wichtig, da so die entsprechenden Zeilen in den Konfigurationsdateien deutlich schneller wieder zu finden sind. Die technische Umsetzung dieser Dateien, beziehungsweise die Speicherung der Parameter, erfolgt bei den Experten in unterschiedlicher Art und Weise. Die meisten Projekte verwenden verbreitete Dateiformate wie XML oder Java-Properties-Dateien, es gibt aber auch Projekte welche ihre Konfigurationsparameter in Datenbanken ablegen.
Zusammenfassend kann man sagen, das sich alle Experten über die Grundlagen einig sind, weshalb diese in Form der Handlungsempfehlung noch einmal detailliert erläutert werden.

6

Handlungsempfehlung

In diesem Kapitel werden die Informationen aus der Auswertung zu einer Handlungsempfehlung zusammengefasst. Diese soll zukünftigen Teams als Leitfaden dienen. Im ersten Teil wird auf den Aufbau der Konfigurationsdateien eingegangen. Der zweite Teil ist speziell auf die Programmiersprache Java zugeschnitten und bietet eine Übersicht über mögliche Techniken um die Konzepte umzusetzen.

6.1 Konzept

In diesem Teil der Handlungsempfehlung wird das Konzept beschrieben mit dem, basierend auf den Informationen der Experten, am besten der Konfigurationsaufwand verringert wird. Konzept bedeutet dabei, dass die Umsetzung möglichst detailliert, jedoch auf einer abstrakten und generalisierbaren Ebene, beschrieben wird.

Speicherformat Zunächst muss über das Speicherformat der Konfigurationsparameter entschieden werden. Aus folgenden Gründen hat sich dabei die Verwendung von XML Dateien als effektiv erwiesen: Die Speicherung in Dateien anstatt in einer Datenbank hat den Vorteil, dass man diese Dateien einfach dem ausgelieferten Build beilegen kann, und keinen weiteren Aufwand für die Wartung oder den Betrieb einer Datenbank aufbringen muss. Es ist nicht nötig, zusätzlich zur eigenen Applikation noch eine weitere Datenbank-Software zu betreiben. Dies spart Ressourcen und ist unabhängig von der Server-Umgebung auf der die Applikation einmal laufen wird. Gerade auf der Development und der Test Umgebung fehlt häufig die Infrastruktur um zusätzliche Datenbanken zu betreiben. Des Weiteren lassen sich Dateien einfacher durch ein Versionsverwaltungssystem verwalten, welches es erleichtert auf ältere Versionen zurückzugreifen. Die Entscheidung für das Format XML begründet sich in dessen anpassbarem Aufbau. Im Gegensatz zu .properties Dateien oder .ini Dateien lassen sich in XML Dateien komplexe Strukturen abspeichern. So kann man beispielsweise eigene Typen von Parametern verwenden und ist nicht auf die Nutzung primitiver Datentypen angewiesen. Daneben ist die Codierung einer XML Datei immer am Anfang der Datei festgelegt. Somit beugt man Problemen mit Umlauten oder Sonderzeichen vor. Allerdings ist für sehr kleine Projekte ein einfacheres Dateiformat potentiell besser geeignet, da dieses oft keinen extra Parser benötigt, und für Menschen im Allgemeinen leichter zu lesen ist.

Bewahrung der Dateien Um mit XML-Dateien die geforderten Vereinfachungen zu erhalten, muss man sich über die Verteilung der Parameter in diesen Dateien Gedanken machen. Eine Unterteilung der Parameter auf mehrere Dateien hat sich als sinnvoll erwiesen, da sonst eine einzelne Datei sehr groß und unübersichtlich wird. Zunächst sollte man die Dateien in einer Hierarchie anordnen um Standardwerte setzen zu können. Eine Trennung der spezifischen Werte von den Standardwerten ermöglicht ein leichtes finden oder kopieren von Werten eines bestimmten Mandanten. Ein Ordner pro Mandant, so wie ein zusätzlicher Ordner für die Dateien mit den Standardwerten garantieren ein leicht verständliches System. Innerhalb dieser Ordner ist es nun auch möglich, mehrere Konfigurationsdateien abzulegen, womit sich Parameter nach Themen gruppieren lassen. Die Abbildung 6.1 zeigt ein Beispiel. Die komplett umrandeten Felder stellen Ordner dar,

Abbildung 6.1: Default Ordner

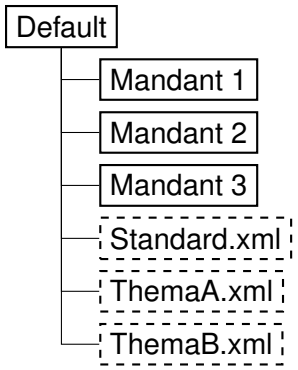
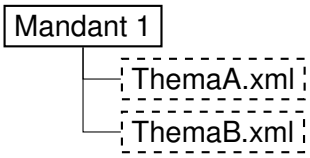


Abbildung 6.2: Mandanten Ordner



die Anderen Dateien. In den Mandantenordnern befinden sich jeweils noch diejenigen Dateien, die je nach Anforderung die Standardwerte ersetzen, diese sind in Abbildung 6.2 dargestellt.

Umgebungen Um die verschiedenen Entwicklungsumgebungen zu konfigurieren, empfiehlt es sich eine zweite Baumstruktur aufzubauen. Diese sieht der aus Abbildung 6.1 ähnlich, hat statt Unterordner für Mandanten eben die Unterordner für die Umgebungen. Voraussetzung für eine zweite Baumstruktur ist aber, dass die Parameter aus der Ersten keinerlei Überschneidung mit denen aus dem zweiten Baum besitzen. Ist dies aufgrund des Projektrahmens nicht möglich so sollten alle Dateien pro Umgebung einmal gepflegt werden. Dies schafft die Möglichkeit, Dateien untereinander recht einfach auszutauschen und individuell Konfigurationen zu testen, bevor man diese auf der Produktivumgebung aufsetzt. Dies ähnelt der Hierarchie wie sie in Kapitel 5, im Abschnitt Abweichende Umgebung, beschrieben ist. Der einzige Unterschied ist, dass es keinen globalen Standard gibt. Dieser ist auf der einen Seite in manchen Situationen sicherlich hilfreich, bringt aber viel Komplexität mit sich, wenn am Ende bestimmte Parameter zwei oder drei mal überschrieben werden.

Thematische Trennung Bei vielen Parametern ist es sinnvoll diese thematisch zu sortieren und in verschiedene Dateien zu schreiben. Das bietet den Vorteil, dass bei der Versionierung dieser Dateien nicht zu einer Datei sehr viele Versionen bestehen, sondern zu jeder Datei einige wenige. So lassen sich alte Zustände besser wiederherstellen. Daneben sind zusammengehörende Parameter auch leichter zu finden. Ist es gefordert, beispielsweise eine ganze Reihe von IP-Adressen zu ändern und alle Adressen sind in einer Datei gruppiert, so kann man zügig alle Adressen ändern ohne jedes mal nach dem nächsten Parameter zu suchen. Dies lässt sich natürlich auch erreichen wenn man die Ordnung innerhalb einer Datei strikt einhält, doch dann fehlt die bessere Möglichkeit der Versionierung.

Laden der Konfiguration Alle gespeicherten Parameter müssen natürlich an geeigneter Stelle im Code geladen werden. Wie dies technisch umgesetzt werden kann, wird genauer in Kapitel 6.2 erläutert. An dieser Stelle soll geklärt werden, wie man sicherstellt, dass die richtigen Parameter geladen werden. Wird die Applikation nur zentral auf einem Server ausgeführt, so hat man das Problem eigentlich nicht, da man alle Parameter laden und auf andere Art und Weise, zum Beispiel über Nutzerberechtigungen, regeln muss wer welche Konfiguration sieht. Stellt man die Applikation jedoch mehrmals online, für jeden Mandanten einmal, so muss man jeweils die richtigen Werte laden. Am einfachsten ist es dabei einen Run-Mode einzuführen, der beim Programmstart festgelegt wird. Dieser gibt an in welcher Umgebung die Applikation läuft und für welchen Mandanten sie ausgeführt wird. Auf der Grundlage des Run-Modes lassen sich dann die richtigen Parameter laden. Festlegen lässt dieser sich entweder über eine zusätzliche kleine Konfigurationsdatei, in der man hinein schreibt welcher Mandant, oder welche Umgebung, geladen werden soll, oder über Start-Parameter beim Programmaufruf. Diese Methode ist zu bevorzugen da dann beim Laden kein weiter Aufwand betrieben werden muss. Kann man nicht alle Konfigurationsdateien mit ausliefern, da die Mandanten Zugriff auf den Server haben, aber die Dateien nicht sehen dürfen, oder aus anderen Gründen, dann kann man vor dem Ausliefern eine neue Datei generieren, die alle entsprechenden Mandanten-Dateien zusammenfasst. Dies erfordert jedoch zusätzlichen Aufwand.

6.2 Technische Aspekte

Da alle Experten ein und dasselbe Framework verwenden um die Konfigurationsdateien zu laden, soll an dieser Stelle ein kurzer Einblick geboten werden. Es handelt sich um das *Apache Commons Configuration*. Dies ist eine Software Bibliothek um Konfigurationen von verschiedenen Ressourcen zu laden [ASF14]. Dieses Framework empfiehlt sich, da es eine sehr gute Dokumentation besitzt, die auf dessen Homepage zu finden ist. Außerdem lassen sich damit sehr bequem die Dateien einlesen, ohne dass ein komplizierter Parser erstellt wurde. Zusätzlich ist es möglich aus verschiedenen Quellen Parameter zu lesen, falls Alternativen zum XML Format gewünscht oder notwendig sind.

7

Diskussion

Zunächst soll in diesem Kapitel die im vorangegangenen Kapitel erarbeitete Handlungsempfehlung evaluiert werden. Zum Einen werden dazu die in Kapitel 4 beschriebenen Problemfälle noch einmal aufgegriffen und untersucht in wie weit die Empfehlungen hierbei weiterhelfen. Zum Anderen werden die befragten Experten darum gebeten die Handlungsempfehlung zu bewerten. Im zweiten Teil des Kapitels wird ein Rückblick über die Arbeit statt finden. In diesem sollen Probleme und Schwierigkeiten aufgegriffen werden, die während der Arbeit aufgetreten sind.

7.1 Evaluierung

An dieser Stelle werden die Probleme aus Kapitel 4 noch einmal aufgegriffen, und diskutiert welche Herangehensweise die Handlungsempfehlung für das jeweilige Problem

vorschlägt und ob es sich so lösen oder vereinfachen lässt. Dazu werden dieselben Fragen beantwortet, die auch die Experten in den Interviews beantworten sollten. Im Anschluss wird das Feedback der befragten Experten ausgewertet.

Das Default Problem

Das DP beschreibt das Problem, dass der Großteil der Mandanten dieselben Werte verwendet, einige Wenige jedoch davon abweichen. Für dieses Problem bietet die Handlungsempfehlung eine gute Lösung. Man initialisiert alle Mandanten und die Standardwerte. Für ein Minimalbeispiel benötigt man zwei Mandanten mit ihren speziellen Dateien, so wie die Standarddatei. In dieser wird der Steuersatz einmalig auf den Wert konfiguriert, der normalerweise herangezogen wird. Dem abweichenden Mandant wird in seiner spezialisierten Datei ein Eintrag hinzugefügt, der den Steuersatz aus der Standard Datei überschreibt.

Abweichende Umgebung

Das Problem der abweichenden Umgebungen ähnelt vom Prinzip dem DP. Es geht dabei um Umgebungsparameter, welche nur für eine Umgebung geändert werden müssen. Je nach Anforderung an die Applikation hat man an dieser Stelle zwei Möglichkeiten. Bei der ersten Möglichkeit überschneiden sich die umgebungsspezifischen und die mandantenspezifischen Werte nicht. Bei der Zweiten können Parameter sowohl von der Umgebung als auch vom Mandanten abhängen. Lässt sich eine Überschneidung nicht vermeiden, werden alle Dateien pro Umgebung einmal gespeichert. Damit kann jede Umgebung individuell konfiguriert werden, aber der Komfort einer Standard-Umgebung geht verloren. Hat man keine Überschneidungen so wird empfohlen eine zweite Baumstruktur aufzubauen, die nur für die Umgebungen gedacht ist. Dies bringt den Vorteil, dass danach auch auf der Umgebungsebene eine Hierarchie vorhanden ist und Standardwerte vergeben werden können.

Umgebung Kopieren Problem

Das Problem beim Kopieren von Umgebungsparametern äußerte sich darin, dass bestimmte Parameter von einer Umgebung auf die Nächste übertragen werden sollen. Beispielsweise von der Integrationsumgebung auf die Produktivumgebung. Um Werte von einer Umgebung auf eine Andere zu kopieren, werden die entsprechenden Dateien kopiert und an der gewünschten Stelle in der Hierarchie wieder eingefügt. Wurde für die Umgebungskonfiguration ein eigener Baum eingerichtet, so muss man sich nur mit den zwei entsprechenden Dateien beschäftigen. Das Problem tritt jedoch fast ausschließlich bei inhaltlichen Parametern auf, das heißt bei Parametern, die auch von den Mandanten abhängig sind. Tritt dieser Fall ein, so speichert man, nach der Empfehlung, alle Konfigurationsdateien pro Umgebung einmal ab. Um nun Werte von einer in die nächste Umgebung zu übertragen, werden die entsprechenden Dateien herausgesucht und in die Hierarchie der neuen Umgebung kopiert.

Mandant Kopieren Problem

Das Mandant-Kopieren-Problem hat den Hintergrund, dass ein Mandant als Vorlage dienen soll. Von Diesem soll entweder ein komplett neuer Mandant abgeleitet werden, oder einzelne Werte zu einem Anderen kopiert werden. Sollen neben den Standardkonfigurationen auch die speziellen Parameter des Vorlage-Mandanten kopiert werden, so kopiert man die Mandantendatei. Ist diese richtig in die Hierarchie eingeordnet, lassen sich Änderungen direkt in der neu angelegten Kopie durchführen.

Vererbung

Die Vererbung beschreibt das Verhalten, dass ein Mandant auf einen Anderen verweist. Das bedeutet, er verwendet immer genau den Wert, den der andere Mandant verwendet, unabhängig davon, wann dieser zuletzt geändert wurde. Das beschriebene Verhalten ist mit der Hierarchie, wie sie in der Handlungsempfehlung beschrieben wird, nicht möglich. Es ist jedoch vorstellbar, einen Parser und eine passende Syntax zu entwickeln, welche

genau dieses Verhalten realisieren. Der Aufwand, die abweichenden Parameter von Hand zu konfigurieren, sollte sich aber in Grenzen halten, da bereits die Möglichkeit vorhanden ist einen Standardwert einzurichten.

Experten Feedback

Um die Meinung der Experten zu der Handlungsempfehlung zu erfahren wurde ihnen diese zugeschickt. Die Experten hatten Zeit sich mit der Empfehlung zu befassen und wurden dann gebeten ein Feedback zu geben. Aufgrund des Feedbacks wurden noch einige Argumente in der Handlungsempfehlung ergänzt. Neben einer allgemeinen Einschätzung und Bewertung der Handlungsempfehlung sollte das Feedback Antworten auf folgende Fragen beinhalten:

1. Passt die Handlungsempfehlung auf das Projekt an dem Sie arbeiten?
2. Ist die Vorgehensweise wie sie beschrieben wird umsetzbar, basierend auf Ihren Erfahrungen?
3. Haben sie für bestimmte Teile einen besseren Vorschlag?

Tabelle 7.1 zeigt die Antworten der Experten auf die erste Frage. Diese zeigen, dass ein nachträgliches Ändern des Konfigurationsmechanismus für die Experten zu aufwändig ist. Sie können sich jedoch vorstellen den empfohlenen Mechanismus einzusetzen. Der Experte, dessen Projekt auf einer Datenbank zur Konfiguration basiert, konnte die Empfehlung zu einer dateibasierten Konfiguration nicht nachvollziehen. Er zieht eine Datenbank vor, da für sein Projekt ohnehin eine Datenbank zur Speicherung von Kundendaten gepflegt werden muss. Der Experte aus dem dritten Interview wies noch einmal darauf hin, dass in ihrem Projekt die spezifischen Werte nicht in extra Dateien gespeichert werden. Dies funktioniert in dem Projekt aber nur, weil sie wenige Abweichungen von den Standardwerten haben.

Die zweite Frage wurde von den Experten sehr knapp beantwortet. Zwei Experten antworteten lediglich mit einem ”Ja”. Ein weiteres Nachfragen war nicht möglich, da sich diese Experten zum Zeitraum des Feedbacks im Urlaub befanden und bis zum Ende der

Tabelle 7.1: Feedback Frage 1

Experte	Passt die Handlungsempfehlung auf das Projekt an dem Sie arbeiten?
1	Die Handlungsempfehlung könnte so umgesetzt werden. Ein Wechsel ist jedoch unpraktikabel. Für ein neues Projekt ist die Empfehlung gut geeignet.
2	Für das Projekt wurde extra eine DSL entwickelt. Diese aufzugeben ergibt keinen Sinn. Des Weiteren sind sehr viele Umgebungen mit dem aufgeführten Mechanismus schwerer zu handhaben als mit der DSL.
3	Unser Projekt trennt die spezifischen Werte nicht in extra Dateien, das funktioniert bisher gut.
4	Da wir ohnehin eine Datenbank verwenden, werden wir diese auch weiter nutzen um damit die Applikation zu konfigurieren.

Tabelle 7.2: Feedback Frage 2

1	Die Handlungsempfehlung ist leicht umzusetzen. Es sollte dennoch darauf geachtet werden, eine gute Dokumentation zu führen und alle Entwickler sollten das System kennen.
2	Ja.
3	Wie bereits in der Handlungsempfehlung selbst angegeben, muss man sich vor Beginn eines Projektes festlegen, auf welche Art man die Umgebungen konfiguriert. Das kann bei einem dynamisch wachsenden Projekt unter Umständen zu Problemen führen.
4	Nach meiner Einschätzung ist die Handlungsempfehlung umsetzbar.

Arbeit nicht wieder erreichbar waren. Die anderen Experten betonten, dass man sich sehr strikt an das System halten muss, und sich unbedingt vor Beginn der Programmierarbeit damit auseinandersetzen muss, welche Parameter wie spezialisiert werden können, um zu entscheiden wie man die Umgebungen konfiguriert. Einer warnte vor der thematischen Trennung, da so zusätzliche Komplexität entsteht, während ein Anderer genau diese Trennung empfiehlt. Zusammenfassend lässt sich sagen, dass die Handlungsempfehlung umsetzbar ist und für die geforderten Probleme eine übersichtliche Lösung zur Verfügung stellt. Tabelle 7.2 zeigt die Antworten der Experten auf die zweite Frage.

Die dritte Frage wurde nur von zwei Experten beantwortet. Der Experte aus dem ersten Interview ergänzte zwei Argumente, die für XML-Dateien sprechen. Er sieht den größten Vorteil an XML-Dateien darin, dass sich die Codierung am Anfang der Datei festlegen lässt. Damit setzt sich das Format erheblich von den sonst üblichen Formaten, wie .properties, ab. Durch das Festlegen der Codierung können keine Probleme mit Umlauten

oder Sonderzeichen auftreten. Ein weiteres Argument, welches grundsätzlich für die Verwendung von Dateien spricht, ist die Versionierung. Dateien lassen sich einfacher mit Versionsverwaltungssystemen verwalten als Datenbanken. So ist es leichter auf ältere Versionen zurück zu springen, oder mehrere Zweige nebeneinander zu halten.
Der Experte aus dem dritten Interview erklärte, dass er es für besser halte, die abweichenden Parameter in dieselbe Datei zu schreiben. Das heißt die Spezialisierungen werden direkt unter die Standardparameter in der Default-Datei geschrieben, um ”räumliche Nähe” zu wahren. Das ist jedoch dann unpraktisch, wenn ganze Blöcke von Mandanten-Konfigurationen kopiert werden sollen. Denn dann ist es nötig, jeden einzelnen Parameter zu bearbeiten. Ein einfaches kopieren und nötigenfalls abändern einer bestehenden Mandantendatei ist dann nicht mehr möglich. Auf Nachfrage äußerte er, dass diese Art von Kopieren bei ihm im Projekt nicht vorkomme. Des Weiteren hat er nur relativ wenige abweichende Parameter, sodass in zusätzlichen Dateien nur wenige Zeilen stünden. Dafür lohnt sich seiner Meinung nach der Aufwand nicht, mehrere Dateien zu pflegen.

Zusammenfassend kann man sagen, dass mit der Handlungsempfehlung die genannten Probleme praxisgerecht vereinfacht würden. Zwar gibt es kein fertiges Framework mit dem gewisse Vorgänge automatisiert werden können, doch bietet die Handlungsempfehlung eine schnell umsetzbare und praktikable Lösung an.

7.2 Rückblick

In diesem Abschnitt soll auf die Arbeit selber eingegangen werden. Dies dient dazu, Probleme näher zu untersuchen die während der Arbeit aufgetreten sind. Da die Arbeit ein praktisches Problem behandelt und in einer Firma durchgeführt wurde, entstehen einige Unwägbarkeiten, die bei einer theoretischen Arbeit nicht auftreten. Dabei sollen alle Arbeitsschritte betrachtet werden, von der Themenfindung über die Forschung bis zum abschließenden Schreiben der Arbeit. Die Probleme, die dabei auftraten sollen kritisch hinterfragt werden und daraus Schlüsse für die Zukunft gezogen werden. Das Thema der Arbeit entstand als Vorschlag des Projektleiters des Projektes ePEP. Dieser

bat um Hilfe bezüglich des in dieser Arbeit behandelten Problems. Da der Titel des Themas vielversprechend klang, wurde das Thema ohne weitere Recherche angenommen. Zunächst war diese Arbeit als Vergleich von bestehenden Frameworks geplant, doch als sich zeigte, dass es für dieses Problem noch keine bestehenden Arbeiten gab, wurde der Schwerpunkt der Arbeit geändert. Die Literaturrecherche war schwierig, da es sich bei dem Problem nicht um eine theoretische Fragestellung handelt, sondern um ein praktisches Problem, das erst in größeren Projekten auftritt. Aus diesem Grund ist es problematisch, nicht nur verwandte Arbeiten zu finden, sondern Arbeiten die speziell dieses Problem behandeln. Für die Zukunft kann man daraus schließen, dass man sich noch während der Themenfindung über vorhandene Quellen informiert und genau herausfindet wie man die Arbeit schreiben möchte. Da es keine Literaturquellen gab, setzte man folglich auf Experteninterviews. Diese Entscheidung kostete viel Zeit, auch weil zunächst vermutet wurde, lediglich nicht ausreichend recherchiert zu haben. Auch bei den Interviews zeigte sich wieder, dass eine gute Vorbereitung sehr wichtig ist, denn die ersten beiden Interviews brachten keine nennenswerten Ergebnisse. Dies lag unter anderem daran, dass die Interviewpartner falsch ausgewählt worden sind und die Problemanalyse zu ungenau war. Die Experten konnten das Problem zunächst nicht nachvollziehen oder Parallelen zu ihren eigenen Projekten finden. Daneben war es schwierig, bei den Experten für ausreichend lange Zeit einen Interviewtermin zu bekommen. Nicht nur der volle Terminplan der Experten war ein Hindernis, zusätzlich fiel die Arbeit in die Sommermonate, in denen die Experten Urlaub hatten. Dies stellte vor allem für das Feedback ein großes Problem dar. Durch die überarbeitete Problemanalyse und die darauf aufbauenden Interviews entstand eine Handlungsempfehlung, die für zukünftige Projekte eine Orientierungshilfe darstellt und nützliche Empfehlungen zur Organisation von Konfigurationsdateien gibt. Zunächst sollten deutlich mehr Experten interviewt werden, da aber alle ausgewählten Experten bereits Erfahrungen aus mehreren Projekten mitbrachten und sich einig waren, was die Umsetzung der Konfiguration betrifft, wurde entschieden, dass die Erfahrungen dieser Experten ausreicht. Interviews mit unternehmensfremden Experten hätten die Erfahrungen ergänzen können. Aufgrund der langjährigen Erfahrung der befragten Experten wurde aber darauf verzichtet weitere hinzuzuziehen. Nachdem die Handlungsempfehlung fertig gestellt war,

wurden die Experten noch einmal kontaktiert und darum gebeten diese zu bewerten. Wie bereits erwähnt stellte es sich zu diesem Zeitpunkt als besonders schwierig heraus weitere Termine für Interviews zu bekommen. So wurden den Experten die Fragen per Email zur Verfügung gestellt, damit sie sich in freien Minuten darauf vorbereiten konnten. Anschließend wurde ein kurzer Termin vereinbart, in dem die Antworten besprochen wurden. Zum Teil wurde dieser Termin auch telefonisch abgehalten. Bei zukünftigen Befragungen sollte direkt ein weiterer Termin vereinbart werden, an dem noch einmal über die Ergebnisse gesprochen werden kann.

8

Fazit

In diesem Kapitel sollen die Ergebnisse noch einmal zusammengefasst werden. Außerdem wird die Vorgehensweise, so wie die Ergebnisse kritisch hinterfragt. Daneben soll ein Ausblick die mögliche Verwendung der Ergebnisse dieser Arbeit in der Zukunft darstellen.

8.1 Zusammenfassung

Am Anfang der Arbeit war ein Problem gegeben, aus welchem eine Empfehlung für zukünftige Projekte entstehen sollte. Die Form des Ergebnisses stand dabei noch nicht fest. Zunächst erwartete man einen Vergleich verschiedener Frameworks und bestehender Lösungsansätze, doch eine gründliche Literaturrecherche erbrachte nur Ergebnisse die nicht direkt mit dem Problem zusammen hängen. Da als Forschungsumfeld die TSS

zur Verfügung stand, wurde die Literaturrecherche durch Experteninterviews innerhalb dieser ersetzt. Mehrere Entwickler, die schon längere Zeit für die TSS tätig sind, und so eine gewisse Erfahrung gesammelt haben, wurden zu dem Thema interviewt. Die gewonnenen Informationen wurden verarbeitet und in Form einer Handlungsempfehlung festgehalten. Das Ergebnis ist eine hierarchische Anordnung von XML Dateien, welche es ermöglicht, Standardwerte zu setzen und diese zu überschreiben. Da alle befragten Experten des Java-Umfeldes einen ähnlichen Ansatz verwenden und damit auch zufrieden sind, wird angenommen, dass das Ergebnis eine valide Lösung zu dem gegebenen Problem ist. In Kapitel 7.1 wurde noch einmal evaluiert ob die gegebenen Empfehlungen eine Lösung des Problems ermöglichen. Zusätzlich wurden die Experten noch einmal gebeten die Handlungsempfehlung zu bewerten. Das positive Feedback hat noch einmal bestätigt, dass die Handlungsempfehlung die Anforderungen aus Kapitel 4 erfüllt.

8.2 Ausblick

Die Handlungsempfehlung dieser Bachelorarbeit wird intern in der TSS zur Verfügung stehen. Dadurch können sich zukünftige Projekte an den Ergebnissen orientieren und Probleme wie sie in Kapitel 4 beschrieben wurden vermeiden. Daneben wird das Projekt ePEP den Konfigurationsmechanismus entsprechend überarbeiten um weiteren Problemen vorzubeugen. Da sich diese Arbeit ausschließlich mit der Technik und dem Konzept hinter dem Konfigurationsmechanismus befasst hat, ist es vorstellbar, für diese Technik eine Unterstützung zur Verfügung zu stellen. Da keiner der Experten ein Tool zur Verwaltung der Konfigurationsdateien verwendet hat, wurde auch keines evaluiert und mit in die Handlungsempfehlung aufgenommen. Eine mögliche Erweiterung des Themas ist das Vergleichen vorhandener oder das Entwickeln eines eigenen Programmes zur Organisation hierarchischer XML Dateien. Mit diesem Programm sollte es möglich sein, Parameter nach eigenen Vorgaben zu filtern und zu verändern. Entwickler müssten sich dann keine Gedanken mehr machen, in welcher Datei ein Parameter konfiguriert ist, sondern ändern diesen einfach direkt im Programm ab, markieren die Mandanten/Umgebungen für die diese Konfiguration gelten soll und das editieren der Dateien übernimmt das Tool. Bei einer entsprechenden Überzahl der Mandanten kann

der Standard automatisch neu gesetzt werden, um die Dateien auch ohne das Tool so übersichtlich wie möglich zu halten. Weitere Funktionen sind die Unterstützung beim Anlegen eines neues Mandanten oder das Kopieren bestimmter Parameter. Mit Hilfe eines solchen Programmes wäre die Umsetzung der Handlungsempfehlung einfacher und ohne Kenntnis derselben möglich.

Abbildungsverzeichnis

Tabellenverzeichnis

Literaturverzeichnis

[ASF14] ASF, The Apache Software F.: *Commons Configurtation - Java Configuration API*. Mai 2014. – `http://commons.apache.org/proper/commons-configuration/index.html`

[AUS01] AZBAR, Nuri ; URSILLO, Pepi ; SPEECE, Richard E.: Effect of process configuration and substrate complexity on the performance of anaerobic processes. In: *Water research* 35 (2001), Nr. 3, S. 817–829

[BAKF04] BLECKER, Thorsten ; ABDELKAFI, Nizar ; KREUTLER, Gerold ; FRIEDRICH, Gerhard: Product configuration systems: state of the art, conceptualization and extensions. (2004)

[Ber02] BERLACK, H. R.: *Encyclopedia of Software Engineering*. John Wiley and Sons, 2002

[Fow04] FOWLER, Martin: *Inversion of control containers and the dependency injection pattern*. 2004

[GA01] GACEK, Critina ; ANASTASOPOULES, Michalis: Implementing product line variabilities. In: *ACM SIGSOFT Software Engineering Notes* Bd. 26 ACM, 2001, S. 109–117

[GMSR13] GRAMBOW, Gregor ; MUNDBROD, Nicolas ; STELLER, Vivian ; REICHERT, Manfred: Challenges of Applying Adaptive Processes to Enable Variability in Sustainability Data Collection. In: *3rd Int'l Symposium on Data-Driven Process Discovery and Analysis (SIMPDA'13)*, CEUR-WS.org, August 2013 (CEUR Workshop Proceedings 1027), S. 74–88

[GMSR14a] GRAMBOW, Gregor ; MUNDBROD, Nicolas ; STELLER, Vivian ; REICHERT, Manfred: Towards Configurable Data Collection for Sustainable Supply Chain Communication. In: *CAiSE'14 Forum*, CEUR-WS.org, June 2014 (CEUR Workshop Proceedings)

[GMSR14b] GRAMBOW, Gregor ; MUNDBROD, Nicolas ; STELLER, Vivian ; REICHERT, Manfred: Towards Process-based Composition of Activities for Collecting Data in Supply Chains. In: *6th Central European Workshop on Services and their Composition (ZEUS 2014)*, 2014

[GOR10] GRAMBOW, Gregor ; OBERHAUSER, Roy ; REICHERT, Manfred: Semantic Workflow Adaption in Support of Workflow Diversity. In: *4th Int'l Conf. on Advances in Semantic Processing (SEMAPRO'10)*, Xpert Publishing Services, October 2010, S. 158–165

[GOR11] GRAMBOW, Gregor ; OBERHAUSER, Roy ; REICHERT, Manfred: Contextual Generation of Declarative Workflows and their Application to Software Engineering Processes. In: *Int J on Advances in Intelligent Systems* 4 (2011), Nr. 3&4, S. 158–179

[GØS12] GUROV, Dilian ; ØSTVOLD, Bjarte M. ; SCHAEFER, Ina: A Hierarchical Variability Model for Software Product Lines. In: *Leveraging Applications of Formal Methods, Verification, and Validation*. Springer Berlin Heidelberg, 2012, S. 181–199

[JHD+04] JOHNSON, Rod ; HOELLER, Juergen ; DONALD, Keith ; SAMPALEANU, Colin ; HARROP, Rob ; RISBERG, Thomas ; ARENDSEN, Alef ; DAVISON, Darren ; KOPYLENKO, Dmitriy ; POLLACK, Mark u. a.: The spring framework, reference documentation. In: *Interface21.(accessed 30.04. 07)* (2004)

[May06] MAYER, Horst O.: *Interview und schriftliche Befragung: Entwicklung, Durchführung und Auswertung*. Oldenbourg Verlag, 2006

[PLS+00] PAL, Partha ; LOYALL, Joseph ; SCHANTZ, Richard ; ZINKY, John ; SHAPIRO, Rich ; MEGQUIER, James: Using QDL to Specify QoS Aware Distributed (QuO) Application Configuration. In: *Third IEEE International Symposi-*

um on Object-Oriented Real-Time Distributed Computing, 2000. (ISORC 2000) Proceedings. (2000)

[UI14] UNIVERSITY, Carnegie M. ; INSTITUTE, Software E.: *Software Product Lines*. September 2014. – `http://www.sei.cmu.edu/productlines/`

[Van08] VANBRABANT, Robbie: *Google Guice: Agile Lightweight Dependency Injection Framework*. Apress, 2008

[VG07] VOELTER, Markus ; GROHER, Iris: Product line implementation using aspect-oriented and model-driven software development. In: *Software Product Line Conference, 2007. SPLC 2007. 11th International* IEEE, 2007, S. 233–242

Name: Manuel Stegmiller Matrikelnummer: 752731

Erklärung

Ich erkläre, dass ich die Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Ulm, den .....

Manuel Stegmiller