Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

Thema:

**Bereitstellung und Test der Software ‚Progress Monitor‘ zum Monitoring und zur Analyse von Dateikonvertierungen für den elektronischen Datenaustausch (EDI)**

Praxisprojekt

Fachbereich I – Wirtschafts‑ und Gesellschaftswissenschaften

Studiengang Wirtschaftsinformatik Online

Betreuender Dozent: Prof. Dr. Ing. Alexander Huber

Betreuer im Unternehmen: Antje Galinsky

vorgelegt von: Michael Wischniewski

Matrikelnummer: 917983

Kemnitzer Chaussee 71

14542 Werder

+49 151 22784419

E‑Mail: s76826@bht‑berlin.de

Abgabetermin: 25. August 2023

Sperrvermerk

Diese Arbeit enthält vertrauliche Daten der Softzoll GmbH & Co. KG. Eine Weitergabe der Arbeit im Ganzen oder in Teilen sowie das Anfertigen von Kopien (auch digital) sind grundsätzlich untersagt. Ausnahmen bedürfen der schriftlichen Genehmigung.

Softzoll GmbH & Co. KG

Abteilung: Software Entwicklung und Integration

Kurfürstenstr. 112

10787 Berlin

Telefon: +49 (0)30 210023‑50

E-Mail: info@softzoll.de

Website: https://www.softzoll.de

Berlin, den 25. August 2023

Danksagung

Im Rahmen meines Praktikums konnte ich viele Erfahrungen im Projektmanagement und in der Softwareentwicklung sammeln und die im Studium angeeigneten Kenntnisse in einem realen Projekt anwenden.

Ohne die Expertise und Unterstützung der Unternehmensleitung und den Mitarbeitenden des Unternehmens hätte ich in dem Projekt nicht so erfolgreich agieren und diese Arbeit in guter Qualität erstellen können.

Mein besonderer Dank gilt der Projektleitung und dem Kollegium aus der Abteilung ‚Softwareentwicklung‘, die mich im Projekt in allen Belangen mit viel Geduld unterstützen. Insbesondere die enge Zusammenarbeit mit der Softwareentwicklung war meiner Ansicht nach ein wichtiger Faktor für die erfolgreiche und termingerechte Umsetzung des Projekts.

Inhaltsverzeichnis

Inhaltsverzeichnis IV

Verzeichnis der Abkürzungen und Akronyme V

Abbildungsverzeichnis VI

Tabellenverzeichnis VII

Definition der Begriffe VIII

1 Einleitung 1

2 Methodik 2

3 Analyse der Anforderungen und Projektrisiken 3

3.1 Anforderungsanalyse 3

3.2 Risikoanalyse 4

4 Projektbeschreibung 5

4.1 Projektbegründung 5

4.2 Projektziel und Projektumfang 6

5 Ist‑Aufnahme 7

6 Spezifikation und Entwurf 10

7 Implementierung 11

8 Software‑Test 12

8.1 Einführung 12

8.2 Testvorbereitung 16

8.3 Testausführung 19

8.4 Testauswertung 23

9 Fazit 25

A Die Anforderungen aus dem Anforderungsdokument ReqSpec 1.001 26

B Auszug aus dem Risikokatalog 29

C Die unterstützten Datenformate und Kommunikationsprotokolle 32

D In der Ist‑Analyse identifizierte Systemkomponenten 33

E Die Entwurfsentscheidungen 37

F Das BPMN‑Diagramm zum Prozess ‚Software‑Test’ 40

G Der Testbericht zum Testfall 6.1.3 41

Literaturverzeichnis 46

Verzeichnis der Abkürzungen und Akronyme

API Application Programming Interface

AS2 Applicability Statement 2

BPMN Business Process Model and Notation

CSV Comma Separated Value

CI Continuous Integration

CD Continuous Deployment

CM Change Management

CSS Cascading Style Sheets

DBMS Datenbankmanagementsystem

DevOps Development and IT Operations

ERP Enterprise-Resource-Planning

GUI Graphical User Interface (Benutzeroberfläche)

EDI Electronic Data Interchange

EDIFACT Electronic Data Interchange for Administration, Commerce and Transport

ETL Extract‑Transform‑Load

Ewk Eintrittswahrscheinlichkeit

FF Fix Format

FTP File Transfer Protocol

HTML Hypertext Markup Language

HTTP Hypertext Transfer Protocol

HTTPS Hypertext Transfer Protocol Secure

IDE Integrated Development Environment

IDoc Intermediate Document

JDBC Java Database Connectivity

JDE JD Edwards

JDK Java Development Kit

JRE Java Runtime Environment

KVP kontinuierlicher Verbesserungsprozess

OFTP Odette File Transfer Protocol

RC Release Candidates

RDBMS Relational Database Management System

RE Requirements Engineering

RM Requirements Management

PDF Portable Document Format

SAP Systeme Anwendungen und Produkt

SFTP Secure File Transfer Protocol

SMTP Simple Mail Transfer Protocol

SQL Structured Query Language

SVN Subversion

TRC Tradacoms

WWW World Wide Web

X12 ANSI X12

X.400 E-Mail-System basierend auf P7-Protokoll

XML Extensible Markup Language

Abbildungsverzeichnis

[**Abb. 3.1:** Ein Auszug aus der User Story Map 3](#_Toc143715815)

[**Abb. 5.1:** Das BPMN‑Diagramm zum Prozess ‚Datenkonvertierung‘ 9](#_Toc143715816)

[**Abb. 8.1:** Die grundlegende Teststrategie 12](#_Toc143715817)

[**Abb. 8.2:** Die vier Prüfebenen des Software‑Tests 14](#_Toc143715818)

[**Abb. 8.3:** Die schematische Darstellung eines Black‑Box‑Tests 15](#_Toc143715819)

[**Abb. 8.4:** Der schematische Testablauf 16](#_Toc143715820)

[**Abb. 8.5:** ‚Progress Monitor‘: Die Benutzeranlage im ‚Progress Monitor‘ 18](#_Toc143715821)

[**Abb. 8.6**: ‚Progress Monitor‘: Die Anlage eines Workflows 19](#_Toc143715822)

[**Abb. 8.7:** Die Testvorschrift zum Testfall 6.1.3 21](#_Toc143715823)

[**Abb. 8.8:** Die Übergabe der zum Testfall 6.1.3 erstellten Testdatei an den Datenkonverter 21](#_Toc143715824)

[**Abb. 8.9:** Die Bildschirmausgabe zum Testfall 6.1.3 22](#_Toc143715825)

[**Abb. D.1:** Das Komponentendiagramm zu den identifizierten Systemkomponenten 34](#_Toc143715826)

[**Abb. F.1:** Das BPMN‑Diagramm zum Prozess ‚Software‑Test’ 40](#_Toc143715827)

[**Abb. G.1:** Der Testbericht zum Testfall 6.1.3, Seite 2 41](#_Toc143715828)

[**Abb. G.2:** Der Testbericht zum Testfall 6.1.3, Seite 3 42](#_Toc143715829)

[**Abb. G.3:** Der Testbericht zum Testfall 6.1.3, Seite 4 43](#_Toc143715830)

[**Abb. G.4:** Der Testbericht zum Testfall 6.1.3, Seite 5 44](#_Toc143715831)

[**Abb. G.5:** Der Testbericht zum Testfall 6.1.3, Seite 6 45](#_Toc143715832)

Tabellenverzeichnis

[**Tab. 8.1:** Eine Übersicht über die zu installierenden Programme 17](#_Toc143715631)

[**Tab. A.1:** Das Anforderungsdokument ReqSpec 1.001 26](#_Toc143715632)

[**Tab. B.1:** Auszug aus dem Risikokatalog 29](#_Toc143715633)

[**Tab. C.1:** Die unterstützten Datenformate und Kommunikationsprotokolle 32](#_Toc143715634)

[**Tab. D.1:** Eine Übersicht zu den in der Ist‑Analyse identifizierten Systemkomponenten 35](#_Toc143715635)

[**Tab. E.1**: Eine Übersicht zu den Spezifikationen und Entwurfsentscheidungen 37](#_Toc143715636)

# Definition der Begriffe

Für das Verständnis der Arbeit und die richtige semantische Einordnung der Fachbegriffe werden diese folgend definiert.

*Apache Tomcat*

Apache Tomcat® ist eine Open Source Software und stellt die Dienste eines World Wide Web (WWW)‑Servers (Web‑Servers) zur Verfügung (Vgl. Alpar et al., (2023), S. 537).

*Application Programming Interface (API)*

Über ein API werden andere Programme an ein Softwaresystem angebunden, um die Systemfunktionen diesen Programmen zur Verfügung zu stellen (Vgl. Kaufmann & Mülder, (2023), S. 241). Das API fungiert somit als Schnittstelle, die alle relevanten Informationen über das Zusammenwirken des Systems mit seiner Umgebung beschreibt (Vgl. Broy, (2023), S. 180).

*Business Process Management Notation (BPMN)*

BPMN ist eine Notation zur Modellierung von Geschäftsprozessen und dient der Analyse und Dokumentation dieser. Über grafische Symbole lassen sich komplexe Geschäftsprozesse übersichtlich darstellen (Vgl. Kaufmann & Mülder, (2023), S. 316, 322).

*Change Management (CM)*

Im CM werden nach Eintreten einer Veränderung in den Geschäftsprozessen oder in den Anforderungen alle Maßnahmen, die für die Berücksichtigung der Veränderung notwendig sind, kontrolliert eingeleitet sowie schnell und effektiv durchgeführt (Vgl. Scheer et al., (2003), S. 5).

*Continuous Delivery (CD)*

Unter CD wird in der Softwareentwicklung die kontinuierliche automatisierte Software‑Auslieferung mittels der ‚Delivery Pipeline’ verstanden (Vgl. Alt et al., (2017), S. 27).

*Cascading Stylesheets (CSS)*

Mittels Cascading Stylesheets (CSS) lässt sich das Layout von HTML‑Seiten, wie z. B. Hintergrund, Schriftgrößen, Rahmen etc. gestalten. CSS ermöglicht die gleichen Styleinformationen für alle Seiten einer Website zu bestimmen, ohne dass einzelne HTML‑Seiten angepasst werden müssen. (Vgl. Kaufmann & Mülder, (2023), S. 224 f.).

*Continuous Integration (CI)*

CI verfolgt das Konzept, Änderungen am Quellcode bzw. Entwicklungsfortschritte von einzelnen Entwicklern eines Teams schnell zusammenzuführen und zu testen. Dadurch werden Fehler frühzeitig erkannt und Softwareprototypen (funktionsfähige Software) in kurzer Zeit entwickelt (Vgl. Alt et al., (2017), S. 29).

*Development and IT Operations (DevOps)*

Der Begriff ‚DevOps’ setzt sich aus den Worten ‚Development’ und ‚IT Operations’ zusammen und betont dabei die Zusammenarbeit dieser beiden Bereiche. Dabei werden die Geschäftsprozesse eingeschlossen und der Kundennutzen in den Mittelpunkt gestellt (Vgl. Halstenberg et al., (2020), S. 1). ‚Development’ (Entwicklung) und ‚IT Operations’ (IT‑Betrieb) sind eigenständige funktionale Organisationseinheiten im IT‑Bereich mit unterschiedlichen Zielsetzungen (Vgl. Alt et al., (2017), S. 23). Beim DevOps‑Ansatz wird auf die ganzheitliche Zusammenarbeit zwischen der Softwareentwicklung, dem IT‑Betrieb und den Geschäftsprozessen fokussiert (Vgl. Halstenberg et al., (2020), S. 1).

*Electronic Data Interchange (EDI)*

EDI ist die Übertragung von Geschäftsdokumenten in einem standardisierten Datenformat zwischen verschiedenen Anwendungssystemen (Vgl. Kaufmann & Mülder, (2023), S. 99).

*Electronic Data Interchange for Administration, Commerce and Transport (EDIFACT)*

Unter EDIFACT wird ein einheitliches Regelwerk für den elektronischen Geschäftsverkehr verstanden. Dieses bezieht sich ausschließlich auf strukturierte Daten (Vgl. Kaufmann & Mülder, (2023), S. 99).

*Extract‑Transform‑Load (ETL)*

Die Übernahme des operativen Datenbestands in den analytischen Datenbestand eines Data Warehouse bzw. eines Relational Database Management System (RDBMS) wird als ETL bezeichnet (Vgl. Kaufmann & Mülder, (2023), S. 380).

*Hypertext Markup Language (HTML)*

Mittels der standardisierten Auszeichnungssprache Hypertext Markup Language (HTML) lassen sich Informationen einer Webseite in einer formatierten und strukturierten Ausgabe präsentieren (Vgl. Kaufmann & Mülder, (2023), S. 216).

*Integrated Development Environment (IDE)*

Eine IDE (integrierte Entwicklungsumgebung) ist eine Sammlung von verschiedenen Werkzeugen, wie Editor, Compiler, Interpreter, Debugger, die für die Softwareentwicklung genutzt werden (Vgl. Kaufmann & Mülder, (2023), S. 503).

*Java Database Connectivity (JDBC)*

JDBC ist ein API zum Zugriff auf tabellarische Daten, insbesondere auf Daten in einem RDBMS (Vgl. ORACLE, (2023)).

*MySQL*

Die Open Source Software MySQL stellt sämtliche Funktionalitäten eines Relational Database Management System (RDBMS) zur Verfügung (Vgl. Burnus, (2008), S. 81).

*Open Source*

Die präzise Bezeichnung ist ‚Open‑Source‑Software’ und bezieht sich auf Software, deren Quellcode (Sourcecode) öffentlich zugänglich ist. Open‑Source‑Software darf ergänzt oder erweitert sowie kopiert und weitergegeben werden (Vgl. Kaufmann & Mülder, (2023), S. 283).

*Release Candidates (RC )*

Ein Software‑Release ist ein (Software)‑Produkt, das an den Kunden ausgeliefert wird. Release Candidate bezeichnet eine Softwareversion, die zur Auslieferung vorgesehen ist, aber noch nicht für den (End)‑Anwender bereitgestellt wurde (Vgl. Kusay-Merkle, (2018), S. 292).

*Relational Database Management System (RDBMS )  
(relationales Datenbankverwaltungssystem)*

In einem RDBMS werden die Daten in zueinander in Relation stehenden Tabellen gespeichert (Vgl. Meier, (2010), S. 6).

*Requirements Engineering (Anforderungsmanagement)*

Das Requirements Engineering (RE) beinhaltet alle Tätigkeiten, die zur Erhebung und Analyse sowie dem Verständnis und zur Dokumentation der Anforderungen erforderlich sind (Vgl. Valentini et al., (2013), S. 9).

*Requirements Management (RM)*

Zum Requirements Management (RM) gehören alle für die Verwaltung und Bereitstellung und der Kommunikation von Anforderungen erforderlichen Tätigkeiten (Vgl. Valentini et al., (2013), S. 9).

*Repository*

Eine Bibliothek zur Verwaltung von Programmcode wird Repository genannt (Vgl. Kaufmann & Mülder, (2023), S. 502).

*Stakeholder*

Stakeholder sind Personen oder Personengruppen, die am Projekt interessiert oder beteiligt oder vom Projekt betroffen sind bzw. das Projekt beeinflussen können (Vgl. Timinger, (2015), S. 314 f.).

*Subversion (SVN)*

SVN ist ein Versionsverwaltungsprogramm, das eine Repository‑Plattform für Programmierer zur Verfügung stellt, um gemeinsam an einem Softwareprojekt zu arbeiten (Vgl. Carle et al., (2013), S. 1).

*Testinfrastruktur*

Zur Testinfrastruktur gehören alle organisatorischen Elemente, wie z. B. Testumgebung, Testwerkzeugen, Büroräumen, Testverfahren, die für die Durchführung des Tests erforderliche sind (Vgl. Droste & Merz, (2019), S. 227).

*Validation*

Die Validation ist die Eignungsprüfung einer Systemkomponente bezogen auf ihren Einsatzzweck (Vgl. Alpar et al., (2023), S. 395).

*Verifizierung*

Das Überprüfen der Übereinstimmung einer Systemkomponente mit ihrer Spezifikation wird als Verifizierung bezeichnet (Vgl. Alpar et al., (2023), S. 398).

*Vorgehensmodell*

Das Vorgehensmodell umfasst alle Aktivitäten und deren Reihenfolge, die zur Durchführung eines Projektes erforderlich sind (Vgl. Aichele & Schönberger, (2014), S. 138).

*Workflow*

In einem Workflow werden die zeitlichen, fachlichen und ressourcenbezogenen Spezifikationen für eine automatische Steuerung der Aufgaben in einem Geschäftsprozess definiert. Die Festlegung und Steuerung der Aufgaben und Arbeitsschritte erfolgt in einem Workflow‑Management‑Systems (Vgl. Gehring & Gabriel, (2022), S. 667 f.).

*Workflow‑Management‑Systems (WFMS)*

Die Aufgaben und Arbeitsschritte eines Workflows werden in einem WFMS automatisiert und gesteuert. Voraussetzung für die automatisierte Steuerung ist, dass die Aufgaben und Arbeitsschritte (Teilprozesse) strukturiert sind (Vgl. Gehring & Gabriel, (2022), S. 394).

*World Wide Web (WWW)*

Das World Wide Web (WWW) ist ein verteiltes Informationssystem und stellt Dienste für die Übertragung von Webseiten über das Hypertext Transfer Protocol (HTTP) zur Verfügung (Vgl. Kaufmann & Mülder, (2023), S. 214 f.).

*WWW‑Server (Web‑Server)*

Ein WWW‑Server (Web‑Server) ist eine Software, die Anfragen über das HTTP‑Protokoll empfängt und beantwortet (Vgl. Alpar et al., (2023), S. 537).

# Einleitung

Im Rahmen der Praxisphase des Bachelor Studiengangs ‚Wirtschaftsinformatik Online’ an der Hochschule für Technik Berlin wurde das Projekt zur Bereitstellung und zum Test der Software ‚Progress Monitor’ im Unternehmen Softzoll GmbH & Co. KG (Softzoll) durchgeführt.

Das operative Geschäft des Unternehmens besteht unter anderem in der Erbringung von Dienstleistungen im elektronischen Datenaustausch (EDI) und der damit verbundenen Datenkonvertierung. Zur Umsetzung der Konvertierung und des Datenaustausches werden die damit verbundenen Arbeitsschritte über Workflows in einem Workflow‑Management‑System (WFMS) organisiert und automatisiert. Für die Datenkonvertierung, den Nachrichtenaustausch und das WFMS kommen von Softzoll selbstentwickelte Softwarekomponenten zum Einsatz.

Zur Festigung bzw. Ausbau der Marktposition und der Wettbewerbsvorteile muss jedes Unternehmen in einem kontinuierlichen Verbesserungsprozess (KVP) die Produktqualität sowie die technischen und organisatorischen Prozesse im Unternehmen überprüfen und durch stetige Veränderungen verbessern (Vgl. Kirner et al., (2006), S. 2). Dazu gehört auch das Qualitätsmanagement, das sicherstellt, dass die Produktqualität erhalten bleibt oder verbessert wird (Vgl. Bechmann & Landerer, (2010), S. 9 ff.).

Als Teil des Qualitätsmanagements wurde zur Fehleranalyse und zur Überprüfung der Softwarequalität sowie zur Verifizierung der Datenkonvertierung bei Softzoll die Software ‚Progress Monitor’ eingeführt.

Folgende Arbeit dokumentiert die Vorbereitung und Durchführung des Projekts.

# Methodik

Ein Projekt ist gemäß DIN 69901‑5 ein neu‑ und einzigartiges Vorhaben, dass wesentlich durch die Einmaligkeit der Bedingungen geprägt ist. Kernaspekt ist hierbei die Erreichung eines konkreten Ziels mit begrenzten zeitlichen, finanziellen und personellen Ressourcen. Die Bearbeitung der komplexen Aufgaben im Projekt erfordern eine methodische Projektorganisation (Vgl. Alam & Gühl, (2020), S. 2).

Die Entwicklung, Integration, Bereitstellung und Tests der Software ‚Progress Monitor‘ erfolgte nach einer standardisierten Methodik des Projektmanagements, die es ermöglichte, das Projektziel mit den für das Projekt begrenzten Ressourcen termingerecht zu erfüllen. Folgend wird die im Projektmanagement angewandte Methodik beschrieben.

Nach Timinger lassen sich im Projektmanagement je nach Anforderung verschiedene Arten von Vorgehensmodellen anwenden. Zum einen gibt es eine Differenzierung bezüglich des Einsatzbereiches. Eine weitere Unterscheidung besteht in der Art des Projektdurchführung bzw. Projektorganisation. Hier wird in sequenziellen, nebenläufigen, wiederholenden und agilen Vorgehensmodellen sowie in wiederverwendungsorientierten Vorgehensmodellen unterschieden. Beim sequenziellen Vorgehen wird das Projekt in einzelnen Phasen, die nacheinander abgearbeitet werden, unterteilt. Jede Phase muss abgeschlossen sein, bevor die folgende Phase begonnen wird. Das Wasserfallmodell ist eines der bekanntesten sequenziellen Vorgehensmodelle. Es beinhaltet die Phasen:

* Anforderungsphase
* Analysephase
* Entwurfsphase / Spezifikation
* Implementierungsphase
* Testphase
* Transferphase

Dieses Vorgehensmodell ermöglicht während des Projektverlaufs den Rücksprung in eine vorausgegangene Phase (Vgl. Timinger, (2015), S. 71 ff.).

In diesem Projekt wurde das Projektmanagement nach dem Wasserfallmodell realisiert.

# Analyse der Anforderungen und Projektrisiken

## Anforderungsanalyse

Die Anforderungsanalyse (Requirements Engineering) ist eine notwendige Grundlage für die Entwickelung von IT‑Systemen. Nach Hermann kann kein anforderungsgerechtes IT‑System ohne Erhebung der Anforderungen erstellt werden (Vgl. Herrmann, (2022), S. 14). Erst die präzise Beschreibung der zu erfüllenden Anforderungen ermöglicht die Erfüllung der Benutzeranforderungen und die Erreichung des Projektziels im geplanten Kostenrahmen und Zeitraum (Vgl. Herrmann, (2022), S. 16 ff.).

Als innovative Technik in der Prozess‑ und Anforderungsanalyse hat sich das Story‑Mapping etabliert. Darin werden unter der Beteiligung aller Stakeholder die Anforderungen der Benutzer in User Stories erfasst (Vgl. Herrmann, (2022), S. 123). Formulierte Anforderungen in den User Stories werden im weiteren Verlauf mit dem Kunden diskutiert bis für den Kunden und den Auftragnehmer ein einheitliches Verständnis und eine Einigung über jede einzelne Anforderung besteht (Vgl. Herrmann, (2022), S. 123). Die Abbildung ‚Abb. 3.1‘ zeigt einen Auszug aus der in diesem Projekt erstellten User Story Map.

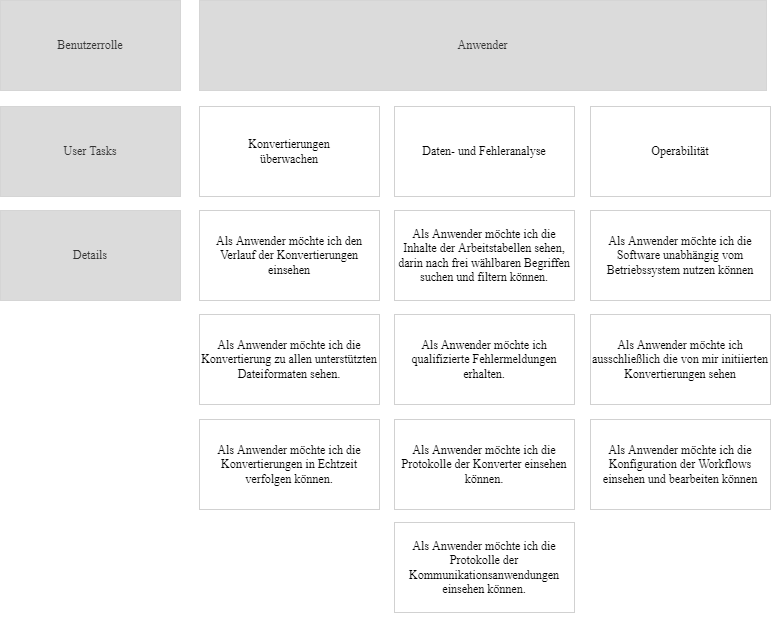


Abb. 3.1: Ein Auszug aus der User Story Map

In der Anforderungsanalyse erfolgte ferner eine Differenzierung der Anforderungen in funktionale und nicht funktionale Anforderungen (Vgl. Herrmann, (2022), S. 37).

Die Entwicklung des ‚Progress Monitors‘ beruhte auf den im Requirements Engineering erhobenen Anforderungen. Darin waren die Plattformunabhängigkeit, die Darstellung des Verlaufs der Konvertierungen in Echtzeit, eine aussagefähige Fehlerausgabe sowie nachvollziehbare Fehlerberichte, die Mehrbenutzerfähigkeit, das Monitoring der Konvertierungen zu allen verwendeten Dateikonvertern wesentliche Vorgaben für die Softwareentwicklung. In der Tabelle ‚Tab. A.1‘ aus dem Anhang A sind die in der Anforderungsanalyse identifizierten Anforderungen übersichtlich zusammengefasst.

## Risikoanalyse

Aus den typischen Merkmalen eines Projekts, wie hohe Komplexität, festgelegter Umfang, Interdisziplinarität, Einmaligkeit sowie zeitliche und finanzielle Einschränkungen, resultieren verschiedene Risiken in der Projektdurchführung (Vgl. Wack, (2007), S. 5). Daher gehören die Identifikation und Bewertung der Projektrisiken sowie die Planung von geeigneten Maßnahmen, um auf die Risiken zu reagieren, zu den wesentlichen Aufgaben im Projekt‑Risikomanagement (Vgl. Wack, (2007), S. 42).

Im Projekt‑Risikomanagement werden potentielle Risiken während des Projektverlaufs fortlaufend identifiziert und bewertet und die Risikoauswirkungen durch geeignete Maßnahmen minimiert. Zur Festlegung von Maßnahmen bei Eintritt eines Risikofalls sind die Risiken hinsichtlich ihres Schadensausmaßes, der Eintrittswahrscheinlichkeit (Ewk) und der Auswirkung zu bewerten. Aus dieser Analyse wird für jedes identifizierte Risiko entschieden, in welchem Umfang Maßnahmen entwickelt werden (Vgl. Wack, (2007), S. 29 ff.).

In diesem Projekt wurden während der Projektdurchführung kontinuierlich die Risiken identifiziert, analysiert und bewertet sowie geeignete Maßnahmen zur Reduzierung des Schadensausmaßes festgelegt. Zu den im Projektverlauf identifizierten Risiken mit enormen Risikopotential gehörten u. a. ein fehlerhaftes Projektmanagement, unpräzise Zielformulierung und unklarer Projektumfang, Fehler in der Anforderungserhebung und sich im Projekt ändernde Anforderungen wie auch fehlende Personalressourcen. Die Bewertung der Risiken und die Entwicklung angemessener Maßnahmen sollten ungeplante Kosten, eine Verzögerung im Projektverlauf oder gar den Abbruch des Projekts verhindern.

Einen Auszug aus dem Risikokatalog enthält die im Anhang B beigefügte Tabelle ‚Tab. B.1‘.

# Projektbeschreibung

## Projektbegründung

Vor dem eigentlichen Start des Projekts erfolgte die Abgrenzung des Untersuchungs‑ und Aufgabenbereich und die Identifizierung der Probleme. Vor der Entwicklung des ‚Progress Monitors‘ konnten die Konvertierungen nicht überwacht werden. Fehler in den Konvertierungen blieben entweder unerkannt oder wurden zu spät bemerkt. Die Fehleranalyse und der Support beanspruchten viel Zeit und Personal. Die zur Projektbeschreibung erstellte Kosten‑Nutzen‑Analyse stellte deutlich heraus, dass der durch die Software erzielte Nutzen, die Projekt‑Aufwendungen überwiegt. In der Kosten‑Nutzen‑Analyse wurden für ein Jahr der zeitliche Aufwand, welcher durch Nacharbeiten und Analysetätigkeiten entsteht mit dem durchschnittlichen Brutto‑Stundensatz der Mitarbeitenden multipliziert. Dabei waren in dieser Erhebung noch keine Folgekosten, wie entgangener Umsatz, Beeinträchtigung der Reputation oder gar Strafzahlungen inkludiert, da diese kaum zu bemessen sind. Die Einsparung dieser Kosten wurden in der Kosten‑Nutzen‑Analyse als Nutzen festgehalten. Im Vergleich dazu wurden die kalkulierten Gesamtkosten des Projekts zuzüglich eines 10‑prozentigen Sicherheitszuschlags gegenübergestellt. Der ermittelte Nutzen war deutlich größer als der Projektaufwand. In der Problemanalyse zeigte sich auch, dass die fehlende Überwachung der Konvertierung zu wiederkehrender Unterbrechung der Arbeitsflusses führte.

Aus der Problembeschreibung ließen sich die konkreten Anforderungen ableiten und die sich daraus ableitenden Aufgaben spezifizieren. Zu den Aufgaben wurden die erforderlichen Ressourcen und der zeitliche Rahmen bestimmt und eine grobe Kostenkalkulation erstellt. Der Abgleich der kalkulierten Kosten mit dem Projektbudget zeigte, dass das Projekt im verfügbaren Budget realisierbar ist. Darauf basierte die daran anschließende Machbarkeitsanalyse zur Einschätzung, ob die Anforderungen mit den verfügbaren Ressourcen und im Zeit‑ und Kostenrahmen erfüllt werden können und das Projektziel erreichbar ist.

Im nächsten Schritt wurde ein grober Projektplan mit Meilensteinen (Termine für Etappenziele im Projekt) erstellt, die im Projekt involvierten Stakeholder, Verantwortlichkeiten und Randbedingungen bestimmt sowie die Ressourcen den Projektaufgaben zugewiesen.

Aus den Ergebnissen der Projektbegründung wurden der Projektantrag und der dazugehörige Projektauftrag erstellt.

## Projektziel und Projektumfang

Das Ziel des unternehmensinternen Projekts ist die Entwicklung der Software ‚Progress Monitor’, die der Überwachung der Datenkonvertierung, der Fehleranalyse und dem Support dient.

Der Projektumfang beschränkte sich auf die Entwicklung der Software, welche die Konvertierungen zu allen verwendeten Datenkonvertern überwachen und in der Konvertierung auftretende Fehler dokumentieren sollte.

Zur Projektdurchführung gehören Tätigkeiten des Projektmanagements, wie die Projektvorbereitung, die Projektorganisation und Projektsteuerung und die Umsetzung. Zur Projektdurchführung gehörten die Anforderungs‑ und Risikoanalyse, die Erhebung der Ist‑Situation und das Erstellen des Soll‑Konzepts sowie die Implementierung der Software.

An dem Projekt beteiligte Stakeholder waren die Mitarbeitenden der Funktionsbereiche ‚Geschäftsleitung’, ‚Change‑ und Qualitätsmanagements’, ‚Controlling’ und ‚Softwareentwicklung’ als Verantwortliche für die Durchführung sowie die Mitarbeitenden der Abteilung ‚EDI‑Projektmanagement’ als Kunden.

# Ist‑Aufnahme

Während der Ist‑Aufnahme gilt es, die für den Untersuchungsbereich relevanten Geschäftsprozesse und bestehende IT‑Systeme sowie die Schnittstellen zu diesen Systemen zu identifizieren und zu analysieren (Vgl. Krallmann et al., (2013). S. 3 f.).

Folgend werden die Ausgangssituation und die Prozesse zur Daten‑Konvertierung bei der Softzoll beschrieben.

Vor der Projektdurchführung konnte keine Software zur Überwachung der Datenkonvertierung genutzt werden. Fehler in der Software, den Konfigurationen und in der für die Konvertierung implementierten Logik wurden nicht rechtzeitig erkannt. Die Fehleranalysen sowie der Support beanspruchten viel Arbeitszeit und personelle Ressourcen.

Zum Verständnis der Umsetzung und daran anschließenden Tests folgt eine kurze Beschreibung der Datenkonvertierung.

Die Datenkonvertierung basiert auf dem Extract‑Transform‑Load‑Ansatz (ETL). ETL ist ein dreistufiger Prozess in welchem die Daten aus einer Datenquelle (in der Regel eine Datei) in eine Datenbank extrahiert, anschließend transformiert und im letzten Schritt in ein Datenobjekt (in der Regel eine Datei) geschrieben (geladen) werden. Im Transformationsprozess erfolgt die Aufbereitung der Daten durch Transformationsregeln entsprechend der durch den Geschäftsprozess spezifizierten Anforderungen. Dabei werden Daten entfernt oder um zusätzliche Informationen angereichert, Datums‑ oder Zahlenwerten formatiert, Werte berechnet, Daten normalisiert, fehlende Einheiten ergänzt oder Einheiten umgerechnet (Vgl. Kaufmann & Mülder, (2023), S. 380 f.).

Ein Konvertierungsprozess besteht aus mehreren Teilprozessen, die eine bestimmte Funktion, wie beispielsweise den Nachrichtenaustausch, die Datenextraktion und Datentransformation, das Schreiben der Daten in ein Datenobjekt und die Überwachung des IT‑Systems, erfüllen. In jedem Teilprozess werden die Aufgaben über eine dedizierte Softwarekomponente bearbeitet. Das erleichtert die Entwicklung neuer Funktionalitäten, da diese spezifisch für einen bestimmten Teilprozess entwickelt werden können. Ferner hat ein Fehler in einer Konvertierung keine Auswirkungen auf andere Konvertierungen.

Die Konvertierungen basieren auf Workflows (automatisierten Geschäftsprozessen). Jeder Workflow ist mit einem Anwendungsfall (Use Case) verknüpft. Im Use Case ist definiert, welcher Kunde mit welchem Geschäftspartner welche Art von Daten austauscht. Daraus resultiert, dass für jeden Anwendungsfall (Use Case) genau ein Workflow eingerichtet ist.

Der Ablauf der Konvertierung ist wie folgt:

Die Daten werden mittels eines Kommunikationsprogramms, Kommunikationsdispatcher genannt, an einen Daten‑Konverter gesandt. Für die Nachrichtenkommunikation werden verschiedene Kommunikationsprotokolle unterstützt. Dabei kommt für jedes unterstützte Protokoll ein dedizierter Kommunikationsdispatcher zur Anwendung. Ein für die Workflowzuordnung zuständiger Prozess liest aus den empfangenen Daten ein bestimmtes Datenmuster aus. Über eine für jeden Datentyp und jede Datenstruktur definierte Konfiguration wird gesteuert, an welcher Position das Datenmuster aus der Datei auszulesen ist. Anhand von im Workflow hinterlegten Parametern wird durch Vergleich des Datenmusters mit den Workflowparametern genau ein Workflow identifiziert. Über Workfloweinstellungen wird der für die Extraktion erforderliche Datenkonverter bestimmt. Hierbei kommt für jeden Datentyp ein dedizierter Konverter zur Anwendung. So wird zum Beispiel zur Konvertierung von XML‑Dateien ein anderer Konverter genutzt als bei der Konvertierung von CSV‑Dateien. Der Transformationsprozess wird durch Transformationsregeln realisiert. Mit dem Schreiben der transformierten Daten in ein Datenobjekt (Load‑Prozess) wird die Konvertierung abgeschlossen. Wie beim Einlesen der Daten, wird auch für das Schreiben für jeden Datentyp ein bestimmter und im Workflow eingetragener Datenkonverter genutzt. Die in der Konvertierung erstellte Datei wird anschließend über einen Kommunikationsdispatcher an den Nachrichtenempfänger gesandt. Auch im Nachrichtenversand werden durch dafür bestimmte Kommunikationsdispatcher diverse Protokolle genutzt. In der im Anhang C beigefügten Tabelle ‚Tab. C.1‘ können die unterstützten Datenformate und Kommunikationsprotokolle eingesehen werden.

Der zur Datenkonvertierung beschriebene Ablauf wird in der Abbildung ‚Abb. 5.1‘ veranschaulicht. Dieser Geschäftsprozess wurde nach dem Modellierungsstandard ‚Business Process Model and Notation‘ (BPMN) modelliert.



Abb. 5.1: Das BPMN‑Diagramm zum Prozess ‚Datenkonvertierung‘

Mit der Betrachtung der Geschäftsprozesse wurden auch die zum Untersuchungsbereich gehörigen Systemkomponenten erfasst und in einem Komponentendiagramm visualisiert. Das Komponentendiagramm zu den identifizierten Systemkomponenten ist in der Abbildung ‚Abb. D.1‘ des Anhangs D dargestellt. Eine detaillierte Aufschlüsselung aller Systemkomponenten enthält die Tabelle ‚Tab. D.1‘ im Anhang D.

# Spezifikation und Entwurf

In der Entwurfsphase werden die in der Anforderungsanalyse identifizierten Anforderungen in einem Lösungskonzept umgesetzt. Die Ergebnisse der Entwurfsphase sind zum Beispiel das Lastenheft und die Systemspezifikation. Darin werden die Eigenschaften der zu entwickelnden Software beschrieben und im weiteren Verlauf systematisch konkretisiert. Die Umsetzung der Ergebnisse aus der Systemspezifikation erfolgt in den Entwurfsentscheidungen (Vgl. Beifuss & Holzbaur, (2020), S. 72).

Die Entwurfsentscheidungen berücksichtigen die Gestaltung der GUI, die Programmiersprache, das Datenbanksystem, mögliche Verteilungskonzepte, Komponentenwiederverwendung und andere Aspekte und dienen als Vorgabe für die Implementierung. Die Entwurfsmodelle müssen in der Implementierung eine signifikante Verfeinerung des Entwurfs ermöglichen (Vgl. Alpar et al., (2023), S. 458).

Im Projekt wurden im Verlauf der Spezifikation, die Eigenschaften, welche die zu entwickelnden Software erfüllen muss, definiert und die Entwurfsentscheidungen erarbeitet.

Zu den wichtigsten Entwurfsentscheidungen gehörten zur Implementierungsphase die Entwicklung in der Programmiersprache ‚Java‘, die Integration des RDBMS MySQL als Datenbasis, die Bereitstellung der Anwendung über einen Apache Webserver sowie die Entwicklung der Benutzerschnittstelle in HTML, Java Script und Cascading Stylesheets. Auch die Entscheidung, die Software über den DevOps‑Ansatz einschließlich der automatisierten Bereitstellung über die ‚Delivery Pipeline zu entwickeln, gehörten zu den Entwurfsentscheidungen.

Die im Projekt getroffenen Entwurfsentscheidungen sind der Tabelle ‚Tab. E.1‘ aus dem Anhang E zu entnehmen.

# Implementierung

Das Projektmanagement sowie die Bereitstellung der Anwendung und die Durchführung der Systemtests stehen in diesem Artikel im Vordergrund. Zur vollständigen Beschreibung der Projektdurchführung und als Einleitung für das folgende Kapitel soll hier die Umsetzung der Implementierung mit wenigen Sätzen zusammengefasst werden.

Die Implementierung der Software erfolgte nach dem DevOps‑Ansatz. ‚DevOps’ ist ein Konzept, welches die Zusammenarbeit der Funktionsbereiche ‚Development’ (Softwareentwicklung) und ‚IT Operations’ (IT‑Betrieb) in den Vordergrund stellt. Es beinhaltet praxiserprobte Lösungsansätze um Lücken zwischen der Softwareentwicklung und dem IT‑Betrieb zu überbrücken (Vgl. Halstenberg et al., (2020), S. 1). In einem DevOps‑Zyklus werden die Phasen ‚Plan’, ‚Code’, ‚Build’, ‚Integrate’, ‚Release’ und ‚Operate’ durchlaufen. In der Release‑Phase sind die Modultests (Unit‑Test) und die Bereitstellung der Software (Deploy) zu realisieren. Die Aktivitäten in den Schritten ‚Build’, ‚Integrate’, ‚Test’ und ‚Deploy’ können weitestgehend automatisiert werden. DevOps basiert auf dem Prinzip der kontinuierlichen Integration (‚Continuous Integration’) und kontinuierliche Bereitstellung (‚Continuous Deployment’) (Vgl. Halstenberg et al., (2020), S. 16 ff.).

Der Programmcode wurde in der Programmiersprache ‚Java’ geschrieben. Die zentrale Verwaltung der Versionierung erfolgte durch SVN. Mittels automatisierter Modultests (Unit Tests) wurde bereits während der Implementierung die Ausführung und Korrektheit des Codes geprüft.

Als RDBMS diente der MySQL‑Datenbankserver und für die Bereitstellung der HTML‑Seiten der Web‑Server Apache Tomcat.

Die Benutzeroberfläche (GUI) wurde mittels Java Script, HTML und Cascading Stylesheets entwickelt.

Das folgende Kapitel beschäftigt sich nun detailliert mit den Systemtests.

# Software‑Test

## Einführung

Zunächst soll die Frage nach dem Zweck von Software‑Tests beantwortet werden. Software‑Tests generieren keinen ökonomischen Ertrag, aber dennoch einen ökonomischen Nutzen. Sie liefern eine Definition der Qualitätskriterien, decken Mängel bzw. Fehler auf und motivieren die Softwareentwickler gute Software zu entwickeln. Durch das frühzeitige Aufdecken von Mängeln bzw. Fehlern werden Folgekosten, die aus Nachbesserung oder Schadensersatzforderungen resultieren, gespart (Vgl. Frühauf et al., (2007), S. 14, 29).

Mit der Implementierung starten auch die Software‑Tests. Diese umfassen analytische Maßnahmen zur Sicherstellung der Softwarequalität. Software‑Tests sind in allen Phasen des Entwicklungsprozesses durchzuführen (Vgl. Hoffmann, (2013), S. 158). Dabei werden im ersten Schritt die einzelnen Softwarekomponenten getestet. Den Komponententest schließen sich die Tests einzelner Teilsysteme und im letzten Schritt die Tests des Gesamtsystems an (Vgl. Alpar et al., (2023), S. 469 f.). Abbildung ‚Abb. 8.1‘ zeigt die grundlegende Teststrategie, die in den Tests involvierten Komponenten bzw. Systeme und die dazugehörige zeitliche Abfolge.



Quelle: Alpar et al. (2023), S. 470

Abb. 8.1: Die grundlegende Teststrategie

In allen Tests werden das Verhalten der Software und die Erfüllung der Spezifikation bzw. des Entwurfs (Verifizierung) sowie die Erfüllung der Anforderungen im Nutzungskontext (Validierung) geprüft. Aus den in der Anforderungsanalyse erhobenen Anforderungen werden die erforderlichen Testaktivitäten abgeleitet.

Die Tests lassen sich nach ihren Merkmalen in die Klassen ‚Prüfebene: In welcher Entwicklungsphase wird getestet?’, ‚Prüfkriterium: Welche Inhalte werden getestet?’ und ‚Prüfmethodik: Wie wird getestet?’ einteilen (Vgl. Hoffmann, (2013), S. 158).

Die Einteilung in die Prüfebenen basiert auf der Programmstruktur der Software und auf der Entwicklungsphase. Darin werden die Tests in Unit‑Tests, Integrationstests, Systemtests und Abnahmetests unterschieden (Vgl. Hoffmann, (2013), S. 159).

*Unit‑Tests*

In einem Unit‑Test, auch als Modultest oder Komponententest bezeichnet, wird eine atomare Programmeinheit getestet. Eine Unit kann aus einzelnen Funktionen, Klassen, oder Paketen und Bibliotheken bestehen (Vgl. Hoffmann, (2013), S. 159).

*Integrationstest*

Die nächsthöhere Abstraktionsstufe in der Kategorie ‚Prüfebene’ bilden die Integrationstests. Darin werden größeren Software‑Komponenten, die sich aus einzelnen Programmmodulen zusammensetzen, getestet. Ein Integrationstest soll sicherstellen, dass die zusammengesetzten Programmkomponenten ein funktionsfähiges System ergeben. In der Praxis werden Integrationstests als ‚Big‑Bang‑Integration’, ‚Strukturorientierte Integration’ oder ‚Funktionsorientierte Integration’ ausgeführt (Vgl. Hoffmann, (2013), S. 163).

*Systemtest*

Sobald alle Teilkomponenten eines Software‑Systems erfolgreich integriert sind, kann mit den Systemtest begonnen werden. Dabei wird das gesamte System getestet und auf die Einhaltung der im Pflichtenheft spezifizierten Eigenschaften überprüft. Im Systemtest wird die Software funktional gemäß den Anforderungen geprüft. Im Gegensatz zum Unit‑ oder Integrationstest bleibt die Code‑Struktur unberücksichtigt. Faktoren, wie unvorhergesehene Fehler, unklare Anforderungen, eingeschränkte Debug‑Möglichkeiten und eingeschränkte Handlungsfähigkeit erschweren die Systemtests (Vgl. Hoffmann, (2013), S. 166 f.).

*Abnahmetest*

Auch der Abnahmetest verifiziert und validiert die Software bezüglich der Vorgaben und Spezifikationen aus dem Pflichtenheft. Im Gegensatz zum Systemtest wird der Abnahmetest unter der realen Einsatzumgebung des Kunden und mit authentischen Daten des Auftraggebers durchgeführt. Beim Abnahmetest gibt der Auftraggeber die Richtlinien des Tests vor oder er führt den Test selbst durch (Vgl. Hoffmann, (2013), S. 168 f.).

Die Abbildung ‚Abb. 8.2‘ stellt die vier genannten Prüfebenen übersichtlich dar.



Quelle: Hoffmann (2013), S. 159

Abb. 8.2: Die vier Prüfebenen des Software‑Tests

Hinsichtlich der Prüfkriterien werden die Software‑Tests nach den inhaltlichen Aspekten eines Testfalls wie folgt klassifiziert.

*Funktionaler Software‑Test*

In den funktionalen Tests wird die Datenverarbeitung geprüft. Zu diesen gehören Funktionstests, Trivialtests, Crashtest, Kompatibilitätstest und Zufallstests (Vgl. Hoffmann, (2013), S. 170 f.).

*Operationale Software‑Tests*

Bei diesen Tests ist das Software‑System unter dem Aspekt des operativen Einsatzes zu validieren und verifizieren. Ergonomietests überprüfen beispielsweise die Benutzbarkeit und Installationstests die reibungsfreie Inbetriebnahme. Sicherheitstests sollen Sicherheitslecks aufzeigen bzw. nachweisen, dass vertrauliche Daten geschützt verarbeitet und gespeichert werden (Vgl. Hoffmann, (2013), S. 172).

*Temporale Software‑Tests*

Hierzu gehören Komplexitätstests, Laufzeittests, Lasttests, Stresstests (Vgl. Hoffmann, (2013), S. 172 f.).

Die in den Tests eingesetzten Methoden und Techniken sind das dritte Merkmal zur Klassifizierung der Software‑Tests. Dabei werden die Software‑Tests in drei Kategorien eingeteilt.

*Black‑Box‑Tests*

In den Black‑Box‑Tests wird ausschließlich das Ein‑ und Ausgabeverhalten der Software geprüft. Die Testfälle werden anhand der Anforderungs‑ und Schnittstellenbeschreibung konstruiert. Die innere Code‑Struktur wird nicht berücksichtigt (Vgl. Hoffmann, (2013), S. 173 f.). Anhand der Abbildung ‚Abb. 8.3‘ ist der Ablauf eines Black‑Box‑Test nachvollziehbar.



Quelle: Frühauf et al. (2007), S. 37

Abb. 8.3: Die schematische Darstellung eines Black‑Box‑Tests

*White‑Box‑Tests*

Für White‑Box‑Tests werden die Testfälle aus der inneren Programmstruktur abgeleitet. Dabei wird, abhängig nach welchem Aspekt der Programm‑Code geprüft werden soll, in kontrollflussorientierten oder datenflussorientierten White‑Box‑Tests unterschieden (Vgl. Hoffmann, (2013), S. 174).

*Gray‑Box‑Tests*

In den Gray‑Box‑Tests sind die Testfälle anhand der Anforderungs‑ und Schnittstellenbeschreibung und der inneren Programmstruktur zu konstruieren (Vgl. Hoffmann, (2013), S. 174).

Der schematische Ablauf von Tests besteht aus den Phasen:

*Testvorbereitung*

In der Testvorbereitung werden die Testfälle ausgewählt, die Testumgebung eingerichtet und Testdaten bereitgestellt (Vgl. Frühauf et al., (2007), S. 36 ff.). Die Spezifikation der Testfälle, die Reihenfolge der auszuführenden Tests und das Vorgehen in den Tests sind in der Testvorschrift dokumentiert. Durch die Vorgaben in der Testvorschrift lassen sich die Aufwendungen reduzieren und die Durchführung der Tests wird transparent. Das ermöglicht den Testfall jederzeit mit denselben Einstellungen zu wiederholen und das dokumentierte Testergebnis nachzuvollziehen (Vgl. Frühauf et al., (2007), S. 36, 72).

*Testausführung*

Während dieser Phase führt der Tester die spezifizierten Testfälle nach den Testvorgaben in der Testvorschrift aus und dokumentiert die Durchführung im Testbericht (Vgl. Frühauf et al., (2007), S. 36 ff.).

*Testauswertung*

Diese Phase dient der Identifizierung von Soll‑Ist‑Abweichungen. Diese werden als Fehler im Testbericht dokumentiert (Vgl. Frühauf et al., (2007), S. 36).

Die Abbildung ‚Abb. 8.4‘ zeigt den schematischen Testablauf.



Quelle: Frühauf et al. (2007), S. 37

Abb. 8.4: Der schematische Testablauf

Nach der theoretischen Einführung wird folgend die Durchführung der Tests zur Software ‚Progress Monitor’ beschrieben. Die Beschreibung beschränkt sich auf die Systemtests, da die Unit‑Tests bereits in der Implementierung entwickelt und automatisiert ausgeführt wurden. Die Durchführung der Systemtests erfolgte als Funktionstest und in Form von Black‑Box‑Tests, in welchen das Ein‑ und Ausgabeverhalten der Software geprüft wird.

## Testvorbereitung

In der Testvorbereitung wurden zunächst die Testfälle und die Durchführung der Tests in der Testvorschrift (Dokument ‚TSpec‑Doc 1/001’ vom 25.07.2023) dokumentiert. Diese beschreibt die Spezifikation der Testfälle gemäß den Anforderungen aus der Anforderungsanalyse, die Reihenfolge der Tests und die Anweisungen zur Vorbereitung der Testinfrastruktur sowie die Testausführung.

Die in der Testvorschrift beschriebenen Testfälle enthalten die Bezeichnung und den Zweck zum Testfall, die Beschreibung des Anfangszustands und der Testumgebung zum Testbeginn sowie die Anweisungen zu den erforderlichen Eingaben und zur Testdurchführung. Ferner sind in der Testvorschrift die erwartete Ausgabe bzw. das erwartete Verhalten des Programms festzuhalten (Vgl. Frühauf et al., (2007), S. 38, 71).

In der Testvorschrift ist jeder Testfall einer Testsequenz zugewiesen. Testsequenzen strukturieren die Testvorschrift, reduzieren den Aufwand und dienen der zielorientierten und effizienten Durchführung der Tests. Eine Testsequenz besteht aus einem oder mehreren Testfällen und ist in der Testvorschrift einem Testabschnitt zugehörig. In einer Testsequenz sind die Testfälle in funktioneller Reihenfolge angeordnet, so dass jeder Testfall die Voraussetzungen für den nachfolgenden erfüllt (Vgl. Frühauf et al., (2007), S. 69 ff.).

Im Kapitel ‚Testausführung‘ ist die Durchführung eines Systemtests anhand der Vorgaben aus der zum Testfall 6.1.3 erstellten Testvorschrift exemplarisch dokumentiert. Der Auszug aus der Testvorschrift veranschaulicht den an der Testvorschrift ausgerichteten Systemtest. Die vollständige Testvorschrift ist dieser Arbeit nicht beigefügt, kann jedoch beim Autor angefragt werden.

Nach der Definition und Beschreibung der Testfälle in der Testvorschrift konnte die Testumgebung eingerichtet und die Software für die Tests bereitgestellt werden. Dazu wurden auf dem Testsystem die in der Tabelle ‚Tab. 8.1‘ genannten Programme installiert.

Tab. 8.1: Eine Übersicht über die zu installierenden Programme

|  |  |  |  |
| --- | --- | --- | --- |
| Nr. | Programm | Version | Zweck |
|  | Apache Tomcat | 7.0 | stellt die Dienste des Webservers für Java‑Programme zur Verfügung |
|  | Java  inklusive JDK, JRE | 8.0 | Bereitstellen des JDK und der JRE |
|  | MySQL | Version 5.0.45‑community‑nt | RDBMS für die Software |
|  | Progress Monitor | Version: 12 Build: dev: 2023-07-06 Development version | Programm zur Überwachung und Kontrolle der Datenkonvertierung |
|  | Daten‑Konverter | Version: 3.3 | Programm für die Datenkonvertierung |

Die Einrichtung der Testumgebung und Bereitstellung der Software ‚Progress Monitor’ erforderten auch die Konfiguration der für den Webserver Apache Tomcat benötigten Umgebungsvariablen und die Installation der Programmdateien. Die Umgebungsvariablen waren als Systemvariablen des Betriebssystems einzutragen.

Für die verschlüsselte Kommunikation wurde im Anschluss das serverseitige SSL‑Zertifikat installiert.

Mit der Installation der Datenbankobjekte und der Softwarekonfiguration war die Einrichtung abgeschlossen (Vgl. Bayer, (2023), S. 1 ff.).

Nach der Installation des Web‑Servers und der Bereitstellung der Software ‚Progress Monitor’ sollten im Programm gemäß den Vorgaben aus der Testvorschrift ein Mandant und jeweils ein Benutzer mit der Benutzerrolle ‚Super‑Administrator’, ‚Administrator’ und ‚Benutzer’ angelegt werden. Diese waren für die Überprüfung der Zugriffsrechte auf Mandanten und Programmkomponenten erforderlich.

Abbildung ‚Abb. 8.5‘ zeigt die Anlage des Benutzers ‚Test‑PM‑Benutzer‘ im Programm ‚Progress Monitor‘



Abb. 8.5: ‚Progress Monitor‘: Die Benutzeranlage im ‚Progress Monitor‘

Anschließend wurden die für die Tests erforderlichen Konten für die Lieferanten und Kunden sowie die dazu verwendeten Workflows angelegt. Die Anlage dieser Objekte erfolgte ebenfalls nach den Vorgaben aus der Testvorschrift.

In der Abbildung ‚Abb. 8.6‘ werden die Anlage und Konfiguration eines Workflows im ‚Progress Monitor‘ dargestellt.



Abb. 8.6: ‚Progress Monitor‘: Die Anlage eines Workflows

Die Aufbereitung der Testdaten beanspruchten einige Zeit in der Testvorbereitung, da für jeden Testfall eine Datei bereitzustellen war.

Mit der Bereitstellung der Testdaten war die Phase der Testvorbereitung abgeschlossen und die Systemtest konnten beginnen.

## Testausführung

Die Systemtests sind funktionsorientiert. Das bedeutet in diesen wird die Erfüllung der funktionalen Anforderungen nach folgenden Kriterien geprüft:

*Funktionsüberdeckung*Es wird jede Funktion in mindestens einem Testfall geprüft (Vgl. Frühauf et al., (2007), S. 45).

*Eingabeüberdeckung*Jedes Eingabedatum ist in mindestens einem Testfall zu prüfen (Vgl. Frühauf et al., (2007), S. 45).

*Ausgabeüberdeckung*Es ist jedes Ausgabedatum in mindestens einem Testfall auszugeben (Vgl. Frühauf et al., (2007), S. 45).

Die Testfälle orientierten sich dabei an der angestrebten Funktion bzw. der definierten Anforderung. Jeder Testfall entspricht dabei einer Anforderung aus der Testvorschrift.

Exemplarisch werden die Tests der funktionalen Anforderungen zur Testsequenz

*TS 6.1 Test der Komponente ‚CSV‑Datenkonverter’; Übergabe über die Funktion ‚Drag an drop’; Test mit englischsprachiger Programmoberfläche*

beschrieben.

In der Testvorschrift sind zu jeder Testsequenz der Zweck des Tests, die Referenzen zu den dazugehörigen Spezifikationen (Anforderungsdokumente, Installations‑ und Konfigurationsvorschriften, Programm‑Dokumentationen etc.) die zu testenden Software‑Komponenten, die erforderlichen Vorbereitungs‑ und Abschlussarbeiten sowie die Testfälle zu dokumentieren (Vgl. Frühauf et al., (2007), S. 69).

Die Testsequenz TS‑6.1 beinhaltet die drei Testfälle

*6.1.1 ‑ Anzeige des Verarbeitungsfortschritts bei fehlerfreier Konvertierung*

In diesem Testfall sind die Anzeige des Verarbeitungsfortschritts und die Ausgaben des Programms bei einer fehlerfreien Konvertierung zu prüfen.

*6.1.2 ‑ Anzeige des Verarbeitungsfortschritts bei fehlerhafter Konvertierung*

Die Testvorschrift zum Testfall ist mit der des Testfalls 6.1.1 identisch. Jedoch muss in diesem Testfall eine fehlerhafte Konvertierung getestet werden.

*6.1.3 ‑ Anzeige der Detailinformationen zu einer Konvertierung*

Zum Testfall 6.1.3 sind die Detailinformationen einer Konvertierung, die fehlerfrei abgeschlossen wurde, zu prüfen.

Die Nummer eines jeden Testfalls setzt sich aus der Nummer der Testsequenz und einer fortlaufenden Nummer zusammen. Testfall 6.1.3 ist somit der 3. Testfall in der Testsequenz 6.1

Die schematische Ausführung der Software‑Tests ist im Anhang F in der Abbildung ‚Abb. F.1‘ als BPMN‑Diagramm dargestellt.

Folgend wird nun die Durchführung des Testfalls 6.1.3 zur Testsequenz 6.1 beschrieben. Dieser Testfall erfolgte gemäß der Vorgaben aus der Testvorschrift mit dem Benutzer ‚Test‑PM‑Benutzer’ und mit einer englischsprachigen Benutzeroberfläche (GUI).

Die Abbildung ‚Abb. 8.7‘ zeigt einen Ausschnitt aus der Testvorschrift zum Testfall 6.1.3.

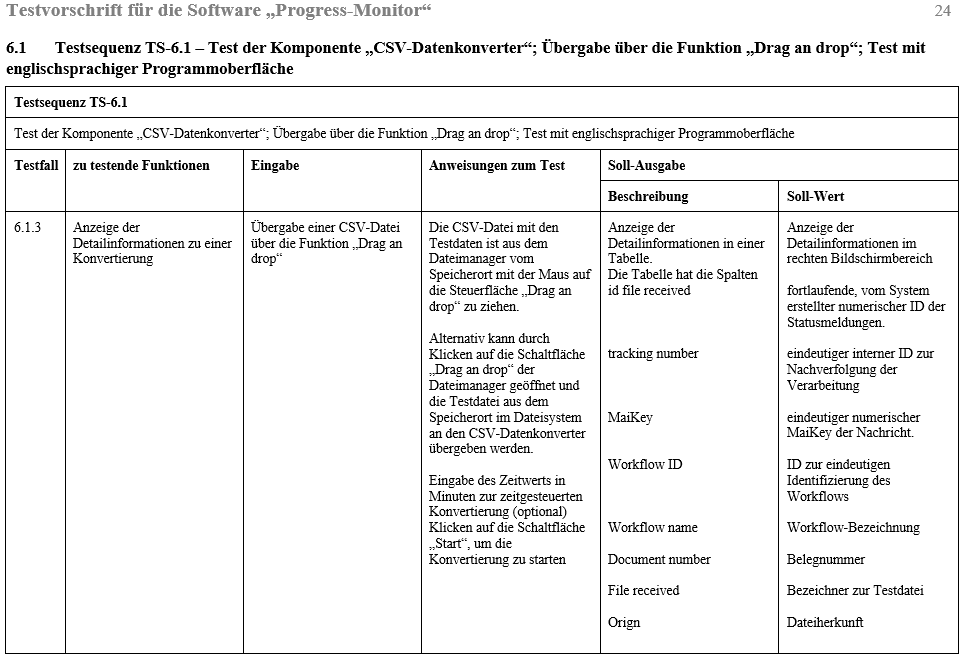


Abb. 8.7: Die Testvorschrift zum Testfall 6.1.3

Der Test beginnt mit der Übergabe der zum Testfall 6.1.3 erstellten Testdatei an den CSV‑Datenkonverter. Die Abbildung ‚Abb. 8.8‘ veranschaulicht diesen Schritt.



Abb. 8.8: Die Übergabe der zum Testfall 6.1.3 erstellten Testdatei an den Datenkonverter

Nach der Konvertierung waren die Detailinformationen zur Konvertierung und die Beschriftung der Textfelder und Steuerelemente im Programm zu prüfen.

In der Abbildung ‚Abb. 8.9‘ wird die Bildschirmausgabe zum ‚Progress Monitor‘ nach der Konvertierung der zum Testfall 6.1.3 erstellten Datei dargestellt.



Abb. 8.9: Die Bildschirmausgabe zum Testfall 6.1.3

Im Test konnte tatsächlich ein Fehler festgestellt werden. Die Tabellenspalte zur ‚Belegnummer’ ist gemäß der Anforderungsspezifikation in der englischsprachigen Benutzeroberfläche mit ‚Document number’ zu bezeichnen. Zu dieser Tabellenspalte wurde jedoch die deutsche Bezeichnung ‚Belegnummer’ angezeigt. Dieser Fehler ist nach den Vorgaben aus der Testvorschrift unkritisch. Das bedeutet, dass anschließend weitere Tests ausgeführt werden konnten. Bei kritischen Fehlern, wie bei einem Ausnahmefehler (Exception), ist gemäß der Testvorschrift der Test zu unterbrechen und der Fehler zu beheben.

Der Test zum Testfall 6.1.3 bewies, wie wichtig Tests sind, auch wenn im Testergebnis lediglich ein unkritischer Fehler festgestellt wurde. Für die methodische und planmäßige Fehlerbeseitigung und für die Wiederholung des Test zu einem späteren Zeitpunkt sind der Test und dessen Ergebnis im Testbericht zu dokumentieren (Vgl. Frühauf et al., (2007), S. 72).

## Testauswertung

Jeder Test wird im Testbericht dokumentiert. Dieser enthält die Beschreibung der Testfälle und die Testresultate und basiert auf der Testvorschrift (Vgl. Frühauf et al., (2007), S. 34). Daher ist eine sorgfältige und detaillierte Testvorschrift für die Testläufe und für die Dokumentation von entscheidender Bedeutung.

Der Testbericht enthält unter anderem folgende Informationen:

*Testzusammenfassung*

Die Testzusammenfassung beinhaltet die Testvorschrift, Sequenz‑Nummer, Testfall‑Nummer, die verwendeten Dokumenten sowie Angaben zum Projekt. Ferner enthält sie die Empfehlung über den Umgang mit identifizierten Fehlern und die dazugehörige Begründung (Vgl. Frühauf et al., (2007), S. 71).

*Testprotokoll*

Im Testprotokoll sind die Durchführung des Tests, die Soll‑Vorgaben, das Testergebnis und die Soll‑Ist‑Abweichungen dokumentiert. Das Testprotokoll muss auch die Fehlermeldungen des Programms und die im Test identifizierten Fehler sowie die Fehlerbeschreibungen enthalten (Vgl. Frühauf et al., (2007), S. 71).

*Software‑Komponenten*

Die Dokumentation der getesteten Software‑Komponenten mit präziser Angabe zur Release‑ und Build‑Version ist für die Verständlichkeit und Transparenz des Tests essentiell und ermöglicht die Wiederholung des Testfalls sowie die Kontrolle zur Anwendung der richtigen Konfiguration (Vgl. Frühauf et al., (2007), S. 72).

Der Testfall 6.1.3 durchgeführte Test erfolgte nach den nach Vorgaben aus der zur Testsequenz 6.1 erstellten Testvorschrift. In diesem Testfall war die Anzeige der Detailinformationen zu einer Konvertierung, die mittels der Komponente ‚CSV‑Datenkonverter‘ und mit einer englischsprachigen Benutzeroberfläche erfolgte, zu testen. Der Test basierte auf den Spezifikationen aus den Anforderungsdokumenten ReqSpec 1.001 und ReqSpec 2.001. In diesem Test sollte die Darstellung der Informationen zur Konvertierung nach dem Einsatzzweck validiert und gemäß den Anforderungsspezifikationen verifiziert werden. Das Testergebnis bestätigte die Eignungsprüfung und die Einhaltung der Spezifikation, ausgenommen der Beschriftung zu einer Tabellenspalte.

Die Daten zur Testvorschrift, Sequenz‑Nummer und Testfall‑Nummer sowie die verwendeten Dokumente, die Testvorbereitung, Testdurchführung und das Testergebnis einschließlich der Soll‑Ist‑Abweichung wurden gemäß der Testvorschrift im Testprotokoll dokumentiert.

Die Abbildungen ‚Abb. G.1‘ bis ‚Abb. G.5’ im Anhang G zeigen den Testbericht mit dem Testergebnis zum Testfall 6.1.3.

Weitere Tests konzentrierten sich auf die Benutzerverwaltung und den damit verbundenen Zugriffsrechten auf Mandanten und Programmkomponenten sowie auf die verschlüsselte Kommunikation mit dem ‚Progress Monitor‘, den Nachrichtenaustausch über die Kommunikationsdispatcher, das Einlesen der Daten und die Datenkonvertierung über weitere Datenkonverter.

In der Testvorschrift wurden zu jedem Test, wie zum Testfall 6.1.3 beschrieben, die Vorgaben für einen Testfall in der dazugehörigen Testsequenz innerhalb des dazugehörigen Testabschnitts dokumentiert. Basierend auf der Testvorschrift erfolgte die Durchführung dieser Tests ebenfalls gemäß der zum Testfall 6.1.3 angewandten Methodik. Dabei wurden in jedem Test Fehler in der Software gefunden. Einige Fehler waren auch kritisch und führten gemäß der Vorgaben aus der Testvorschrift zum Testabbruch. Jeder Test wurde so oft wiederholt, bis die Testausführung fehlerfrei war. Durch die entwickelten Testroutinen ließen sich die Tests einfach wiederholen und als Vorlage für weitere Tests adaptieren. Das methodische Testen auf Grundlage der Testvorschrift ermöglichte ein standardisiertes Vorgehen, welches den Testaufwand reduzierte sowie die Testdurchführung beschleunigte und vereinheitlichte. Ferner konnten die dokumentierten Testresultate einfach miteinander verglichen werden.

# Fazit

Auf Grund fehlender Werkzeuge für die Überwachung der Datenkonvertierungen und die Datenanalyse sowie für die Unterstützung im Support war für das Unternehmen Softzoll GmbH & Co. KG der Einsatz einer Software, welche diese Anforderungen erfüllt, erforderlich. Im EDI‑Bereich sind der Nachrichtenaustausch und die Konvertierung unterschiedlicher Datenformate für verschiedenen Branchen von spezifischen Charakteristiken geprägt. Die auf dem Markt verfügbare Standardsoftware konnte die Anforderungen, welche das Unternehmen an die Software stellt, nicht erfüllen. Daher fiel in der Anforderungsanalyse und in der Machbarkeitsstudie die Entscheidung, die fachliche und technische Expertise der Mitarbeiter im Unternehmen zu nutzen und die Software ‚Progress Monitor‘ selbst zu entwickeln.

Die Software ermöglicht zukünftig Fehler in den Datenkonvertern, in der Workflow‑Konfiguration sowie in der für die Daten‑Transformation implementierten Logik schneller zu finden. Ein weiterer Nutzwert der Software ist die Unterstützung in der Datenanalyse. Häufig enthalten die zu konvertierenden Daten inhaltliche und/oder strukturelle Fehler. Durch den ‚Progress Monitor‘ lassen sich diese in viel kürzerer Zeit als bisher finden und mittels der qualifizierten Fehlerausgabe einfacher dokumentieren. Basierend auf der Unterstützung in der Datenanalyse kann auch der Support für die Kunden verbessert werden. Kundenanfragen lassen sich nun durch die Software schneller als zuvor beantworten. Durch die Unterstützung in der Fehler‑ und Datenanalyse verbessert sich auch die Qualität des Kundensupports. Ferner stellt die Standardisierung zukünftiger Integrations‑, System‑ und Abnahmetests einen wesentlichen Nutzen der Software dar. Mit Hilfe des ‚Progress Monitors‘ lassen sich Testroutinen konzipieren und standardisieren. Dies gewährleistet die konsistente Durchführung der Tests und ermöglicht die systematische Dokumentation der Testergebnisse. Die Testergebnisse werden somit nachvollziehbar und lassen sich einfacher miteinander vergleichen. Mittels der Testszenarien können die Tests (teil)‑automatisiert und beschleunigt sowie einfach wiederholt werden. Die entwickelten Testszenarien lassen sich für nachfolgende Tests adaptieren. Daraus resultieren, wie in der Kosten‑Nutzenanalyse kalkuliert, eine signifikante Kosten‑ und Zeitersparnis.

Das Projekt konnte durch das methodische Vorgehen termingerecht und innerhalb des kalkulierten Kostenrahmens abgeschlossen werden. Mit Hilfe des DevOps‑Konzepts ließ sich die Zeit der Auslieferung funktionsfähiger Release Candidates (RC) verkürzen und die Softwarequalität erhöhen. Künftige Projekte werden daher auf die in diesem Projekt angewandte Methodik und das DevOps‑Konzept aufbauen.

Anhang

Die Anforderungen aus dem Anforderungsdokument ReqSpec 1.001

Die an die Software ‚Progress Monitor’ gestellten Anforderungen sind in der Tabelle ‚Tab. A.1‘ zusammengefasst.

Tab. A.1: Das Anforderungsdokument ReqSpec 1.001

|  |  |  |  |
| --- | --- | --- | --- |
| Nr. | Anforderung | Beschreibung | Kategorie |
|  | Die Anwendung soll unabhängig vom Betriebssystem nutzbar sein. | Die Anwender arbeiten im Home‑Office bzw. mobile Office und nutzen Endgeräte mit verschiedenen Betriebssystemen. | nicht  funktional |
|  | Die Konvertierungen sind in Echtzeit darzustellen. | Die Anwender müssen die Datenkonvertierung und evtl. auftretende Fehler in Echtzeit verfolgen können. | nicht  funktional |
|  | Es sind die Konvertierungen zu allen verwendeten Dateikonvertern anzuzeigen. | Die Dateikonverter sind die Basis‑Softwarekomponenten für die Datenkonvertierung. Derzeit werden Konverter für die Dateiformate Comma Separated Value (CSV), Electronic Data Interchange for Administration, Commerce and Transport (EDIFACT), Fix Format (FF), Hypertext Markup Language (HTML), Intermediate Document (IDoc), Infor (Infor), JD Edwards (JDE), Tradacoms (TRC), ANSI X.12 (X.12) und Extensible Markup Language (XML) verwendet. In der Software sind die Konvertierungen zu allen genannten Dateiformaten anzuzeigen. | funktional |
|  | Sämtliche unterstützte Kommunikationsanbindungen sollen für den Anwender einzusehen sein. | Der Nachrichtenaustausch erfolgt über die Netzwerkprotokolle AS2, FTP, HTTP, HTTPS, SFTP, SMTP, X.400. In der Software soll der Benutzer alle genannten Protokolle einsehen können. | funktional |
|  | Der Benutzer muss die Inhalte der in der Datenbank verwendeten Arbeitstabellen einsehen können. | Die Daten werden in der Konvertierung in verschiedenen Tabellen einer SQL‑Datenbank gespeichert. Diese Tabellen werden Arbeitstabellen genannt. Die Inhalte aus diesen Tabellen sind für den Anwender über das Frontend der Anwendung anzuzeigen. Dabei hat der Benutzer die Möglichkeit nach frei wählbaren Suchbegriffen in den Tabellen zu suchen und zu filtern. | funktional |

|  |  |  |  |
| --- | --- | --- | --- |
| Nr. | Anforderung | Beschreibung | Kategorie |
|  | Der Fortschritt der Konvertierung ist grafisch und als Prozentwert darzustellen. | Die Konvertierung erfolgt in mehreren Phasen nach der Methode Extract‑Transform‑Load (ETL). In der Software ist der Fortschritt zu jeder der Phasen als Balkendiagramm und als prozentualer Wert dazustellen. | funktional |
|  | Fehler in der Konvertierung sind anzuzeigen. | Bei auftretenden Fehlern ist eine qualifizierte Fehlermeldung auszugeben. diese Fehlermeldung soll über die betreffende Software‑Komponente und den Fehlertyp informieren. | funktional |
|  | Die Protokolle der Konverter und der für die Kommunikation verwendeten Softwarekomponenten sollen einsehbar sein. | Der Anwender soll zu Analysezwecken alle Protokolle der Konverter und Kommunikationsanwendungen einsehen können. | funktional |
|  | Die Konfiguration der für die Konvertierung verwendeten Workflows muss einzusehen sein und bearbeitet werden können. | Die Einstellungen zu jeder Konvertierung werden in Workflows vorgenommen. Für eine Analyse sollen die Workflowkonfigurationen einzusehen sein und durch den Benutzer bearbeitet werden können. | funktional |
|  | Der Benutzer soll mehrere zeitgleich ausgeführte Konvertierungen beobachten können. | In der täglichen Arbeit werden mehrere Konvertierungen zeitgleich ausgeführt. Diese sollen, gekapselt in einem dedizierten Prozess, parallel dargestellt werden.  Im Programm werden für jede Konvertierung werden die dazugehörigen Informationen separat visualisiert. | funktional |
|  | Das Programm muss mehrbenutzerfähig sein. | Die Anwendung soll von mehreren Benutzern parallel ausgeführt werden können. Dabei muss sich jeder Benutzer über ein eigenes Benutzerkonto am Programm anmelden können. | nicht funktional |
|  | Die Benutzer müssen sich im Programm authentifizieren. | Zur Wahrung des Datenschutzes und der Datensicherheit muss sich jeder Benutzer über einen Benutzernamen und Kennwort am Programm anmelden. | nicht funktional |
|  | verschlüsselte Kommunikation über HTTPS | Über HTTPS soll die verschlüsselte Kommunikation gewährleistet werden. | nicht funktional |
|  | autorisierte Zugriffe über dedizierte Benutzerrechte und -rollen | Die Benutzeraktivitäten sollen durch dedizierte Benutzerrechte und ‑rollen autorisiert werden | nicht funktional |
|  | Sämtliche Benutzeraktivitäten sind detaillierte zu protokollieren. | In Protokolldateien (Logdateien) sollen die Benutzeraktivitäten zur Anmeldung am Programm zu Änderungen der Konfigurationen an den Workflows, Kommunikationsverbindungen und den Benutzerkonten und zur Durchführung der Konvertierungen protokolliert werden. Die Protokolldateien werden in TXT‑Format und UTF‑8 Zeichenkodierung erstellt. Jeder Eintrag in der Logdatei wird mit einem Datums‑Zeit‑Stempel versehen. Jeden Tag um 00:00 Uhr wird die am Tag zuvor erstellte Datei archiviert und eine neue Protokolldatei erstellt. | nicht funktional |
|  | Anzeige des Verarbeitungsfortschritts bei fehlerfreier Konvertierung | in grüner Schrift: Anzeige der Meldung ‚100%’  Beispielwert:  ‚**100%**’ | nicht funktional |
|  | Anzeige des Verarbeitungsfortschritts bei fehlerhafter Konvertierung in der deutschsprachigen Programmoberfläche | in roter Schrift: Anzeige des Fortschritts zur Konvertierung als Prozentwert und die Meldung  {**Prozentwert**}‚**mit Fehler**’  Beispielwert:  ‚**80% mit Fehler**’ | nicht funktional |
|  | Anzeige des Verarbeitungsfortschritts bei fehlerhafter Konvertierung in der englischsprachigen Programmoberfläche | in roter Schrift: Anzeige des Fortschritts zur Konvertierung als Prozentwert und die Meldung  {**Prozentwert**}‚**with errors**’  Beispielwert:  ‚**80% with errors**’ | nicht funktional |
|  | Beschriftung der Textfelder und Steuerelemente im Programm | Beschriftung der Textfelder und Steuerelemente im Programm erfolgt gemäß Anforderungs‑Dokument Referenz: ReqSpec 2.001 vom 12.04.2019 | nicht funktional |

Auszug aus dem Risikokatalog

Die folgende Tabelle ‚Tab. B.1‘ enthält einen Auszug aus dem Risikokatalog.

Tab. B.1: Auszug aus dem Risikokatalog

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Nr. | Risiko | Wirkung | Ewk | Schadensausmaß | Maßnahme |
|  | Fehlerhaftes Projektmanagement | Nichterreichen der Projektziels  Terminüberschreitung | Hoch | Schwer: kein Projektabschluss  Überschreitung des Finanz‑ und Zeit‑Budgets | Effektives Projektmanagement |
|  | unpräzise Formulierung des Projektziels | Nichterreichen der Projektziels | gering | Schwer: kein Projektabschluss | Erstellen eines detaillierten Lastenhefts  Erstellen eines detaillierten Pflichtenhefts  sorgfältig ausgearbeiteter Projektantrag und der Projektauftrag  sorgfältige und präzise Anforderungsanalyse |
|  | Unklarer Projektumfang |  | Hoch | Schwer: kein Projektabschluss  Überschreitung des Finanz‑ und Zeit‑Budgets  Nichterfüllen der Anforderungen | gut organisierter und strukturiertes Projektmanagement  Erstellen eines detaillierten Lastenhefts  Erstellen eines detaillierten Pflichtenhefts  sorgfältig ausgearbeiteter Projektantrag und der Projektauftrag  sorgfältige und präzise Anforderungsanalyse |
|  | Fehlende Personalressourcen | Terminüberschreitung | Hoch | Mittel: Überschreitung des Finanz‑ und Zeit‑Budgets | Pufferzeiten in den Arbeitspaketen  Ermitteln der Abhängigkeiten zu den Aufgaben  Planung der Substitution von Mitarbeitenden |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Nr. | Risiko | Wirkung | Ewk | Schadensausmaß | Maßnahme |
|  | Fehlende Festlegung der Funktionen und Rollen | Verzögerungen im Projektablauf  Terminüberschreitung  Nichterreichen der Projektziels  Überschreitung des finanziellen Budgets | Hoch | Schwer: kein Projektabschluss  Überschreitung des Finanz‑ und Zeit‑Budgets | präzise Bestimmung der Funktionen und Rollen und Zuweisung dieser an die Ausführenden |
|  | Fehler in der Anforderungserhebung | Annahme falscher Anforderungen; Anforderungen werden nicht erfüllt | Hoch | Schwer: Anforderungen des Kunden werden nicht erfüllt | Bestimmen der kompetenten Ansprechpartner und Entscheidungsträger  Gründliche Vorbereitung auf die Interviews  Führen und Dokumentieren der Interviews  Pflichtenheft  Entwickeln von Mockups und Prototypen  Intensive Kommunikation mit dem Kunden und den Shareholdern  sorgfältige und präzise Anforderungsanalyse |
|  | Änderung der Anforderungen | Nichterreichen der Projektziels  Terminüberschreitung  Verzögerungen im Projektablauf  fehlende bzw. unzureichende Funktionalitäten in der Software | Hoch | Mittel: unzureichende/fehlerhafte Software  Nachbesserung erforderlich | sorgfältige Durchführung des Requirements Engineering und Change Managements |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Nr. | Risiko | Wirkung | Ewk | Schadensausmaß | Maßnahme |
|  | Fehler in der Implementierung | Fehler in der Ausführung der Software; fehlerhafte Software‑Funktionalitäten  geforderte Funktionalitäten werden nicht oder fehlerhaft umgesetzt | Hoch | Schwerkalkulierbar  Kosten = Personalstunden \* Stundensatz | Unit‑Test,  Integrationstests,  Systemprüfungs‑Test,  Abnahmeprüfung  Dokumentation der Tests |
|  | Fehlende bzw. mangelhafte Durchführung der Qualitätssicherung | mangelhafte bzw. fehlerhafte Software  fehlende Funktionalitäten in der Software | Mittel: | Mittel: fehlerhafte Software  Nachbesserung erforderlich | Planung, Konzeption und Definition der Qualitätssicherung  Einbeziehung des Kunden in die Qualitätssicherung  agile Softwareentwicklung mit kontinuierlichen Tests und frühzeitiger Entwicklung von Prototypen |
|  | Fehlendes Bewusstsein für die Produktqualität | mangelhafte bzw. fehlerhafte Software  fehlende Funktionalitäten in der Software | Mittel | Mittel: fehlerhafte Software  Nachbesserung erforderlich | Motivation der Mitarbeitenden zu qualitätsorientiertem Handeln |

Die unterstützten Datenformate und Kommunikationsprotokolle

In der Tabelle ‚Tab. C.1‘ sind die durch die Datenkonverter konvertierbaren Datenformate und die von den Kommunikationsdispatchern unterstützten Kommunikationsprotokolle übersichtlich aufgelistet.

Tab. C.1: Die unterstützten Datenformate und Kommunikationsprotokolle

|  |  |  |
| --- | --- | --- |
| eingehend | Comma Separated Value (CSV)  Electronic Data Interchange for Administration, Commerce and Transport (EDIFACT)  Fix Format (FF)  Hypertext Markup Language (HTML)  Intermediate Document (IDoc)  JD Edwards (JDE)  Portable Document Format (PDF)  Tradacoms (TRC)  ANSI X12 (X12)  Extensible Markup Language (XML) | Applicability Statement 2 (AS2)  File Transfer Protocol (FTP)  Hypertext Transfer Protocol (HTTP)  Hypertext Transfer Protocol Secure (HTTPS)  Odette File Transfer Protocol (OFTP)  Secure File Transfer Protocol (SFTP)  Simple Mail Transfer Protocol (SMTP)  X.400 |
| ausgehend | Comma Separated Value (CSV)  Electronic Data Interchange for Administration, Commerce and Transport (EDIFACT)  Fix Format (FF)  Hypertext Markup Language (HTML)  Intermediate Document (IDoc)  JD Edwards (JDE)  Portable Document Format (PDF)  Tradacoms (TRC)  ANSI X12 (X12)  Extensible Markup Language (XML) | Applicability Statement 2 (AS2)  File Transfer Protocol (FTP)  Hypertext Transfer Protocol (HTTP)  Hypertext Transfer Protocol Secure (HTTPS)  Odette File Transfer Protocol (OFTP)  Secure File Transfer Protocol (SFTP)  Simple Mail Transfer Protocol (SMTP)  X.400 |

In der Ist‑Analyse identifizierte Systemkomponenten

Mit der Ist‑Analyse der Prozesse wurden gemäß Krallmann auch die darin involvierten Systemkomponenten sowie die Schnittstellen, die für den Datenaustausch zwischen diesen Komponenten erforderlich sind, identifiziert (Vgl. Krallmann et al., (2013), S. 145 ff.). Zu den wesentlichen Komponenten des betrachteten IT‑Systems gehören die Kommunikationsdispatcher, die Datenkonverter und die MySQL‑Datenbank.

Im in der Abbildung ‚Abb. D.1‘ dargestellten Komponenten‑Diagramm sind die in der Ist‑Aufnahme identifizierten Systemkomponenten übersichtlich dargestellt.

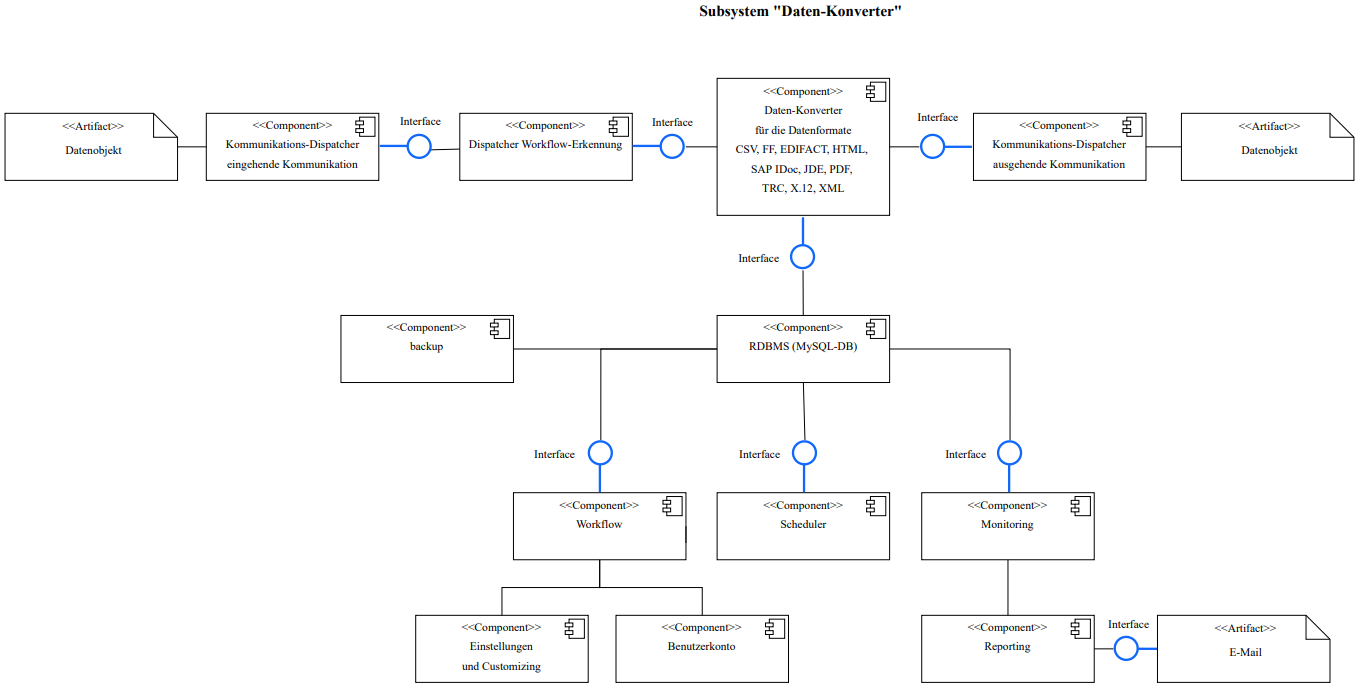


Abb. D.1: Das Komponentendiagramm zu den identifizierten Systemkomponenten

Die Tabelle ‚Tab. D.1‘ listet alle in der Ist‑Analyse bestimmten Systemkomponenten detailliert auf.

Tab. D.1: Eine Übersicht zu den in der Ist‑Analyse identifizierten Systemkomponenten

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Komponente | Beschreibung | | Informationen zur Implementierung | | Schnittstelle |
| AS2‑Kommunikations‑  dispatcher | Komponente für die ein‑ und ausgehende AS2‑Kommunikation | | entwickelt in C++  basiert auf QT‑Framework | | Daten-Konverter |
| FTP‑Kommunikations‑  dispatcher | Komponente für die ein‑ und ausgehende FTP‑Kommunikation | | entwickelt in C++  basiert auf QT‑Framework | | Daten-Konverter |
| HTTP‑Kommunikations‑  dispatcher | Komponente für die ein‑ und ausgehende HTTP‑Kommunikation | | entwickelt in C++  basiert auf QT‑Framework | | Daten-Konverter |
| HTTPS‑Kommunikations‑  dispatcher | Komponente für die ein‑ und ausgehende HTTPS‑Kommunikation | | entwickelt in C++  basiert auf QT‑Framework | | Daten-Konverter |
| OFTP‑Kommunikations‑  dispatcher | Komponente für die ein‑ und ausgehende OFTP‑Kommunikation | | entwickelt in C++  basiert auf QT‑Framework | | Daten-Konverter |
| SFTP‑Kommunikations‑  dispatcher | Komponente für die ein‑ und ausgehende SFTP‑Kommunikation | | entwickelt in C++  basiert auf QT‑Framework | | Daten-Konverter |
| SMTP‑Kommunikations‑  dispatcher | Komponente für die ein‑ und ausgehende SMTP‑Kommunikation | | entwickelt in C++  basiert auf QT‑Framework | | Daten-Konverter |
| X.400‑Kommunikations‑  dispatcher | Komponente für die ein‑ und ausgehende X.400‑Kommunikation | | entwickelt in C++  basiert auf QT‑Framework | | Daten-Konverter |
| CSV‑Converter | | Komponente für die Konvertierung von CSV‑Dateien | | entwickelt in C++  basiert auf QT‑Framework | ODBC (MySQL-DB) |
| FF‑Converter | | Komponente für die Konvertierung von FF‑Dateien | | entwickelt in C++  basiert auf QT‑Framework | ODBC (MySQL-DB) |
| EDIFACT‑Converter | | Komponente für die Konvertierung von EDIFACT‑Dateien | | entwickelt in C++  basiert auf QT‑Framework | ODBC (MySQL-DB) |

|  |  |  |  |
| --- | --- | --- | --- |
| Komponente | Beschreibung | Informationen zur Implementierung | Schnittstelle |
| HTML‑Converter | Komponente für die Konvertierung von HTML‑Dateien | entwickelt in C++  basiert auf QT‑Framework | ODBC (MySQL-DB) |
| IDoc‑Converter | Komponente für die Konvertierung von SAP IDoc‑Dateien | entwickelt in C++  basiert auf QT‑Framework | ODBC (MySQL-DB) |
| JDE‑Converter | Komponente für die Konvertierung von JDE‑Dateien | entwickelt in C++  basiert auf QT‑Framework | ODBC (MySQL-DB) |
| PDF‑Converter | Komponente für die Konvertierung von PDF‑Dateien | entwickelt in C++  basiert auf QT‑Framework | ODBC (MySQL-DB) |
| TRC‑Converter | Komponente für die Konvertierung von TRC‑Dateien | entwickelt in C++  basiert auf QT‑Framework | ODBC (MySQL-DB) |
| X.12‑Converter | Komponente für die Konvertierung von X.12‑Dateien | entwickelt in C++  basiert auf QT‑Framework | ODBC (MySQL-DB) |
| XML‑Converter | Komponente für die Konvertierung von XML‑Dateien | entwickelt in C++  basiert auf QT‑Framework | ODBC (MySQL-DB) |
| Scheduler | Komponente für die zeitgesteuerte Konvertierung | entwickelt in C++  basiert auf QT‑Framework | ODBC (MySQL-DB) |
| Dispatcher‑Check‑System | Komponente für die Systemüberwachung | entwickelt in C++  basiert auf QT‑Framework | ODBC (MySQL-DB) |
| Dispatcher‑Workflow‑Erkennung | Komponente für die Identifizierung der Workflows | entwickelt in C++  basiert auf QT‑Framework | ODBC (MySQL-DB) |
| MySQL‑DB | RDBMS | Datenbank für die Datenhaltung |  |

Die Entwurfsentscheidungen

Aus der Tabelle ‚Tab. E.1‘ lassen sich die für die Implementierung getroffenen Entwurfsentscheidungen ablesen.

Tab. E.1: Eine Übersicht zu den Spezifikationen und Entwurfsentscheidungen

|  |  |  |
| --- | --- | --- |
| Entwurfsentscheidungen / Spezifikationen zur Implementierung der Software ‚Progress Monitor’ | | |
| Zweck:  Überwachung der Datenkonvertierungen  Analyse der Datenkonvertierungen  Fehleranalyse  Unterstützung im fachlichen und technischen Support  Qualitätssicherung, Qualitätsverbesserung  Schulung von Kunden und Mitarbeitenden | | |
| Bereich: Software | | |
| Nr. | Entwurfsentscheidung | Anforderung |
|  | Die Software wird für die aktuelle Version der Java Runtime Environment (JRE) entwickelt. | Die Software muss unabhängig vom Betriebssystem nutzbar sein. |
|  | Das zugrundeliegende RDBMS ist MySQL, Version 5.0.45-community-nt | Nutzung einer gemeinsamen Datenbasis für den Daten‑Konverter und die Software  ‚Progress Monitor’ |
|  | Als Webserver wird Apache Tomcat eingesetzt. | Für den Webserver soll etablierte Open‑Source‑Software eingesetzt werden |
|  | Die Schnittstelle zwischen der Software und der Datenbank wird über Java Database Connectivity, Version 4.3 (JDBC) realisiert. | Nutzung einer gemeinsamen Datenbasis für den Daten‑Konverter und die Software ‚Progress Monitor’ |
|  | Die Ergebnisse der Konvertierung werden in der Software über Tabellen dargestellt. | Der Anwender muss die Ergebnisse zur Konvertierung filtern und sortieren können. |
|  | In der Konvertierung und Kommunikation aufgetretene Fehler werden mit Fehlerursache und Referenz auf die den Fehler auslösende Komponente am Bildschirm ausgegeben.  Zusätzlich müssen alle Fehler in einer Datei protokolliert werden. Dabei sind die Fehler durch Fehlercodes, welche den Fehlertyp klassifizieren, eine Fehlerbeschreibung und eine Referenz auf die Komponenten, welche den Fehler ausgelöst hat, zu dokumentieren. | Für die Fehleranalyse und den Support sind Fehler, die während der Kommunikation und des Verarbeitungsprozesses auftreten, mit Fehlerursache und Lokalisation der Fehlerursache anzuzeigen. |
|  | Die Tabellen in der Software sind sortierbar und filterbar. | Der Anwender muss die Ergebnisse zur Konvertierung filtern und sortieren können. |
|  | Entwicklung einer Benutzerkontenverwaltung | Verwaltung von Zugriffsberechtigung aus Sicherheitsgründen |
|  | Entwicklung von Benutzerrollen und einer Zugriffssteuerung mit dedizierten Benutzerrechten | Steuerung von dedizierten Benutzerrechten aus Sicherheitsgründen |
|  | Implementierung einer Mandantenverwaltung | individuelle Zugriffssteuerung pro Mandant und Benutzer |
|  | Implementierung zur Protokollierung der Benutzeraktionen | Sämtlich Benutzeraktionen müssen zur Nachvollziehbarkeit protokolliert werden. |

|  |  |  |  |
| --- | --- | --- | --- |
| Bereich: Software | | | |
| Nr. | Entwurfsentscheidung | | Anforderung |
|  | Die Anwendung ist für die Anwender über die Browser Google Chrome (ab Version ), Firefox (ab Version ), Microsoft Edge (ab Version ) nutzbar. | | Die Software muss unabhängig vom Betriebssystem nutzbar sein |
|  | Die Kommunikation mit dem Webserver, welcher den Progress Monitor bereitstellt, wird mittels SSL verschlüsselt und über das Anwendungsprotokoll HTTPS realisiert.  Für den Web‑Server darf nur in von einer beglaubigten Zertifizierungsstelle ausgestelltes Zertifikat bereitgestellt werden. | | Aus Sicherheitsgründen muss die Kommunikation zwischen Endgerät und Anwendung über SSL verschlüsselt werden. |
|  | Die im Frontend präsentierten Webseiten werden mittels HTML erstellt.  Die Gestaltung (Formatierung) dieser Webseiten erfolgt ausschließlich über Cascading Stylesheets (CSS).  Im Frontend zu realisierende Funktionalitäten werden mittels Java‑Script umgesetzt. | | Für die Präsentation der Websites sind etablierte und standardisierte Auszeichnungssprachen und Werkzeuge zu verwenden. |
| Bereich: Softwareentwicklung | | | |
| Nr. | Entwurfsentscheidung | Anforderung | |
|  | Die Software wird nach dem DevOps‑Konzept mit den darin integrierten Techniken  Kontinuierliche Optimierung,  Releaseplanung (Release)  Kontinuierliche Integration (Continuos Integration)  Kontinuierliche Bereitstellung (Continuos Deployment)  entwickelt. | Durch die Anwendung des DevOps‑Konzepts ist die Zusammenarbeit der Funktionsbereiche ‚Softwareentwicklung’ und in ‚IT‑Operations’ zu intensivieren und infolgedessen die Effektivität und Effizienz der Arbeit zu verbessern.  Ziel der Umsetzung des DevOps‑Konzepts ist die Erhöhung des Betriebsergebnisses und die Qualitätssicherung in der Softwareentwicklung als Baustein des kontinuierlichen Verbesserungsprozesses. Durch das DevOps‑Konzept sollen in der Implementierungsphase frühzeitig funktionsfähige Prototypen bereitgestellt werden. Das Feedback der Anwender zu den Prototypen soll in neue Release‑Versionen einfließen. | |
|  | Die Software wird in der integrierten Entwicklungsumgebung (Integrated Development Environment (IDE)) Eclipse entwickelt. | Die Software soll in einer etablierten Open‑Source Entwicklungsumgebung, in welcher die Anbindung an das Versionsverwaltungssystem ‚Subversion’ (SVN) integriert ist, und die den Softwareentwicklern vertraut ist, entwickelt werden. | |

|  |  |  |
| --- | --- | --- |
| Bereich: Softwareentwicklung | | |
| Nr. | Entwurfsentscheidung | Anforderung |
|  | Änderungen an der Software werden als Builds zentral in dem Versionsverwaltungssystem SVN verwaltet. | Software‑Builds und ‑Releases sind in der Versionsverwaltung (Repository‑Verwaltung) SVN zentral zu organisieren und zu archivieren.  Die zentrale Verwaltung verschiedener Versionsstände in Forks und Branches sichert jeden Versionsstand und ermöglicht, etwa im Fehlerfall, die Wiederherstellung früherer Versionsstände in der Versionshistorie. Durch die Protokollierung und Archivierung lassen sich Änderungen nachvollziehen. Die Implementierung basierend auf SVN ist ein zentraler Bestandteil der Qualitätssicherung bei Softzoll. |
|  | In der Softwareentwicklung werden zur Qualitätssicherung Metrik‑Werkzeuge wie FindBugs, JDepend, Checkstyle integriert.  Metrikregeln, die der Sicherstellung der Quellcodequalität dienen, werden vor jedem Build automatisiert ausgeführt. Entspricht der Quellcode nicht den in den Regeln definierten Anwendungen wird der Build‑Prozess abgebrochen. | Für die Gewährleistung der Software‑Qualität sind in der Software‑Entwicklung geeignete Metrik‑Werkzeuge wie FindBugs, JDepend, Checkstyle regelmäßig zu nutzen.  Der Sicherstellung der Code‑Qualität, wie die Bezeichnung von Klassen, Methoden, Variablen, Konstanten, die Kommentierung und Formatierung des Quellcodes, ist dabei die größte Aufmerksamkeit zu widmen. |
|  | In der Softwareentwicklung wird zur Qualitätssicherung regelmäßig das Refactoring zur Quellcode‑Refrakturierung angewandt. | Für die Gewährleistung der Software‑Qualität ist der Quellcode regelmäßig zu refrakturieren. |
|  | Zur Überprüfung der Ausführung und Korrektheit wird der Quellcode kontinuierlich und automatisiert durch Modultests (Unit Tests) validiert und verifiziert. | Der Quellcode ist kontinuierlich und automatisiert nach den Vorgaben im Validation‑ und Verification‑Plan durch Modultests (Unit Tests) zu validieren und verifizieren.  Der Validation‑ und Verification‑Plan ist ein essentieller Bestandteil der Qualitätssicherung. |
|  | Die Ausführung und korrekte Umsetzung der geforderten Funktionalitäten in der Software wird kontinuierlich durch Produkttests und Akzeptanztests verifiziert. | Die korrekte Umsetzung der geforderten Funktionalitäten ist regelmäßig durch Produkttests und Akzeptanztests, die nach den Vorgaben im Validation‑ und Verification‑Plan ausgeführt werden, nachzuweisen. Der Validation‑ und Verification‑Plan ist ein essentieller Bestandteil der Qualitätssicherung. |

Das BPMN‑Diagramm zum Prozess ‚Software‑Test’

Der Geschäftsprozess ‚Software‑Test‘ wird in der nachstehenden Abbildung ‚Abb. F.1‘ als BPMN‑Diagramm veranschaulicht.



Abb. F.1: Das BPMN‑Diagramm zum Prozess ‚Software‑Test’

Der Testbericht zum Testfall 6.1.3

Die folgenden Abbildungen ‚Abb. G.1‘ bis ‚Abb. G.5‘ dokumentieren das Testergebnis zum Testfall 6.1.3.

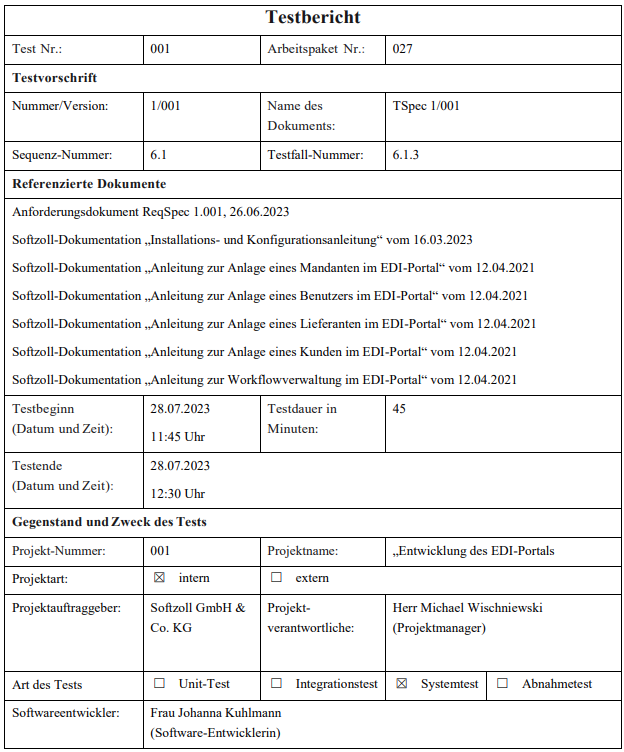


Abb. G.1: Der Testbericht zum Testfall 6.1.3, Seite 2

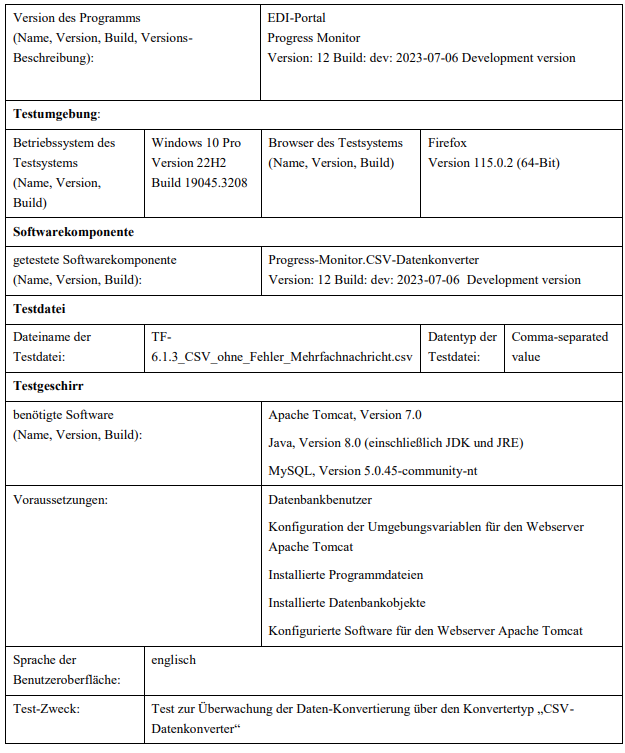


Abb. G.2: Der Testbericht zum Testfall 6.1.3, Seite 3

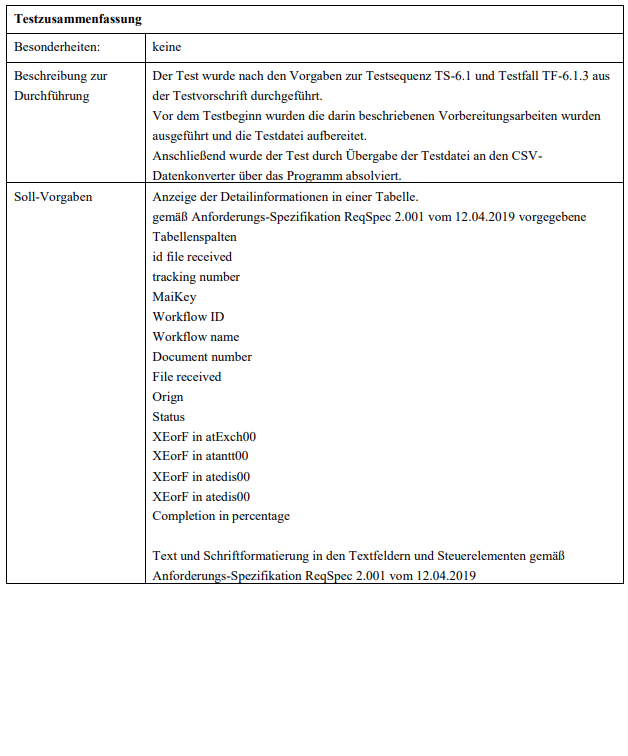


Abb. G.3: Der Testbericht zum Testfall 6.1.3, Seite 4

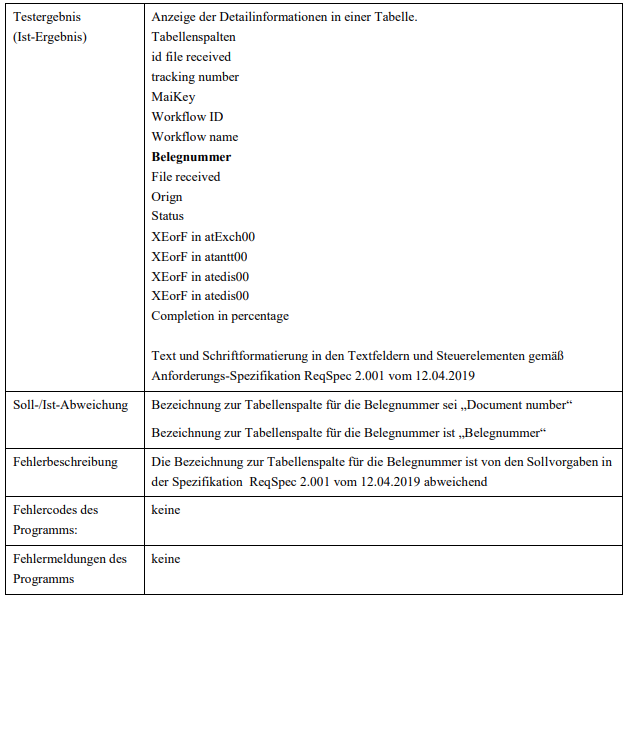


Abb. G.4: Der Testbericht zum Testfall 6.1.3, Seite 5

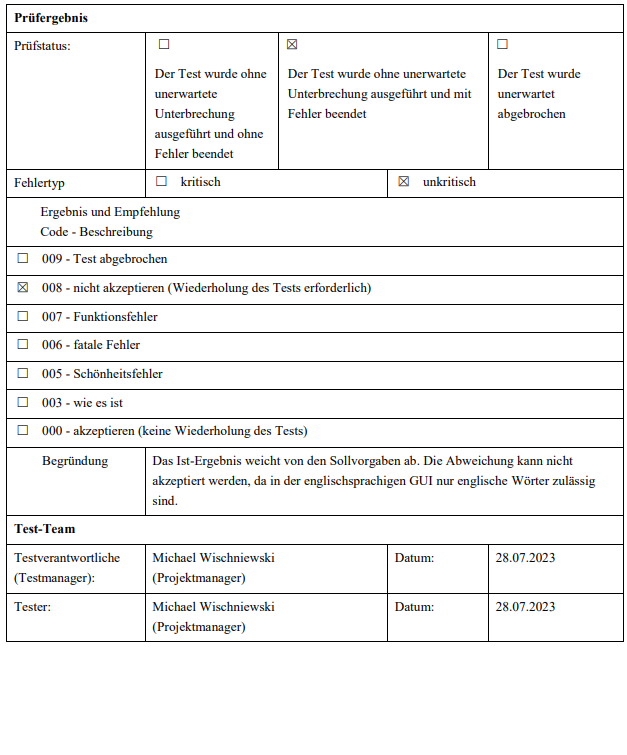


Abb. G.5: Der Testbericht zum Testfall 6.1.3, Seite 6

Literaturverzeichnis

Aichele, C., & Schönberger, M. (Hrsg.). (2014). *App4U: Mehrwerte durch Apps im B2B und B2C*. Springer Fachmedien Wiesbaden. https://doi.org/10.1007/978-3-8348-2436-3

Alam, D., & Gühl, U. (2020). *Projektmanagement für die Praxis: Ein Leitfaden und Werkzeugkasten für erfolgreiche Projekte*. Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-662-62170-7

Alpar, P., Alt, R., Bensberg, F., & Czarnecki, C. (2023). *Anwendungsorientierte Wirtschaftsinformatik: Strategische Planung, Entwicklung und Nutzung von Informationssystemen*. Springer Fachmedien Wiesbaden. https://doi.org/10.1007/978-3-658-40352-2

Alt, R., Auth, G., & Kögler, C. (2017). *Innovationsorientiertes IT-Management mit DevOps*. Springer Fachmedien Wiesbaden. https://doi.org/10.1007/978-3-658-18704-0

Bayer, M. (2023). *Installation von Tomcat und EDI-Portal*. Softzoll GmbH & Co. KG.

Bechmann, R., & Landerer, S. (2010). *Qualitätsmanagement und kontinuierlicher Verbesserungsprozess*. Bund-Verl.

Beifuss, A., & Holzbaur, U. (2020). *Projektmanagement für Studierende: Strategie und Methode für ein erfolgreiches Studium*. Springer Fachmedien Wiesbaden. https://doi.org/10.1007/978-3-658-32664-7

Broy, M. (2023). *Logische und Methodische Grundlagen der Entwicklung verteilter Systeme: Unter Mitarbeit von Alexander Malkis*. Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-662-67317-1

Burnus, H. (2008). *Datenbankentwicklung in IT-Berufen: Eine praktisch orientierte Einführung mit MS Access und MySQL ; [mit Online-Service zum Buch]* (1. Aufl.) Vieweg.

Carle, G., Günther, S., Herold, N., & Posselt, S. (2013). *Was\_ist\_Subversion\_TU\_Munich.pdf*. Technische Universität München. https://www.net.in.tum.de/pub/grnvs/2013/svnintro.pdf

Droste, O., & Merz, C. (2019). *Testmanagement in der Praxis*. Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-662-49653-4

Frühauf, K., Ludewig, J., & Sandmayr, H. (2007). *Software-Prüfung: Eine Anleitung zum Test und zur Inspektion* (6.). vdf Hochschulverlag AG.

Gehring, H., & Gabriel, R. (2022). *Wirtschaftsinformatik*. Springer Fachmedien Wiesbaden. https://doi.org/10.1007/978-3-658-37702-1

Halstenberg, J., Pfitzinger, B., & Jestädt, T. (2020). *DevOps: Ein Überblick*. Springer Fachmedien Wiesbaden. https://doi.org/10.1007/978-3-658-31405-7

Herrmann, A. (2022). *Grundlagen der Anforderungsanalyse: Standardkonformes Requirements Engineering*. Springer Fachmedien Wiesbaden. https://doi.org/10.1007/978-3-658-35460-2

Hoffmann, D. W. (2013). *Software-Qualität*. Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-35700-8

Kaufmann, J., & Mülder, W. (2023). *Grundkurs Wirtschaftsinformatik: Eine kompakte und praxisorientierte Einführung*. Springer Fachmedien Wiesbaden. https://doi.org/10.1007/978-3-658-37937-7

Kirner, E., Armbruster, H., & Kinkel, S. (2006). *Kontinuierlicher Verbesserungsprozess-Baustein zur Prozessinnovation in KMU: Nutzung und Effekte von KVP im Verarbeitenden Gewerbe*. Mitteilungen aus der ISI-Erhebung-Modernisierung der Produktion.

Krallmann, H., Schönherr, M., & Trier, M. (2013). *Systemanalyse im Unternehmen* (6. Aufl.). Oldenbourg Verlag München Wien.

Kusay-Merkle, U. (2018). *Agiles Projektmanagement im Berufsalltag: Für mittlere und kleine Projekte*. Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-662-56800-2

Meier, A. (2010). *Relationale und postrelationale Datenbanken*. Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-05256-9

ORACLE. (2023). *ORACLE Java Tutorials*. https://docs.oracle.com/javase/tutorial/jdbc/overview/index.html

Scheer, A.-W., Abolhassan, F., Jost, W., & Kirchmer, M. (Hrsg.). (2003). *Change Management im Unternehmen*. Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-19020-9

Timinger, H. (2015). *Wiley-Schnellkurs Projektmanagement*. Wiley-VCH Verlag GmbH & Co. KGaA.

Valentini, U., Weißbach, R., Fahney, R., Gartung, T., Glunde, J., Herrmann, A., Hoffmann, A., & Knauss, E. (2013). *Requirements Engineering und Projektmanagement* (A. Herrmann, E. Knauss, & R. Weißbach, Hrsg.). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-29432-7

Wack, J. (2007). *Risikomanagement für IT-Projekte* (1. Aufl.). Dt. Univ.-Verl.

Abschließende Erklärung

Ich versichere hiermit, dass ich die vorliegende Arbeit selbstständig, ohne unzulässige Hilfe Dritter und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe. Die aus fremden Quellen direkt oder indirekt übernommenen Gedanken sind als solche kenntlich gemacht.

Berlin, den 25. August 2023