

Referenzprofil

# Software Developer

**Dr. Jörg Caumanns**

**Marleen Kiral**

Dieses Referenzprofil wurde im Rahmen des bmb+f geförderten Projekts „Arbeitsprozess-orientierte Weiterbildung in der IT-Branche“ erarbeitet von:

**Fraunhofer**   
Institut  
Software- und  
Systemtechnik

Fraunhofer ISST



Dekra Akademie

Bildungspartner



Software AG Systemhaus

Unternehmenspartner

## Danksagung

---

Diese Profilbeschreibung entstand auf Basis eines Praxisprojekts der *Software AG*, deren Projektleiter Herrn Tillenburg wir herzlich für seine fachkundige und umfassende Hilfe danken. Fachlich beratend mitgewirkt haben Herr Helmut Kerschbaumer (*Dekra Akademie*), Herr Karl-Heinz Hageni (*Software AG*) und Frau Cornelia Wachter (*Software AG*). Ohne ihre Hilfe hätte dieses Dokument nicht entstehen können.

# Inhalt

---

<b>1</b>	<b>EINFÜHRUNG: REFERENZPROZESSE ALS CURRICULA .....</b>	<b>4</b>
1.1	EREIGNIS-PROZESS-KETTEN: SYMBOLIK .....	4
1.2	REFERENZPROZESS UND TEILPROZESSE .....	6
<b>2</b>	<b>DAS PROFIL: SOFTWARE DEVELOPER (SOFTWAREENTWICKLER/IN).....</b>	<b>9</b>
2.1	TÄTIGKEITSBESCHREIBUNG .....	9
2.2	PROFILTYPISCHE ARBEITSPROZESSE .....	9
2.3	PROFILPRÄGENDE KOMPETENZFELDER.....	10
2.4	QUALIFIKATIONSERFORDERNISSE.....	11
2.5	EINORDNUNG INS SYSTEM UND KARRIEREPFADE .....	12
<b>3</b>	<b>REFERENZPROZESSE .....</b>	<b>13</b>
3.1	KOOPERATIVE KOMPONENTENENTWICKLUNG .....	13
3.1.1	Referenzprozess: Kooperative Komponentenentwicklung.....	14
3.1.2	Das Beispielprojekt: Entwicklung eines Piloten zur Anbindung eines vorhandenen E-Shops an ein vorhandenes ERP-System .....	15
3.1.3	Prozesskompass: Kooperative Komponentenentwicklung .....	16
3.1.4	Teilprozesse: Kooperative Komponentenentwicklung .....	16
3.1.4.1	Unterstützung von Systemanalyse und -design .....	17
3.1.4.2	Mitwirken bei der Festlegung des Entwicklungsrahmens .....	19
3.1.4.3	Überprüfen des Systemdesigns .....	21
3.1.4.4	Verfeinern der Systementwürfe .....	23
3.1.4.5	Abstimmen der internen Schnittstellen und Datenformate .....	25
3.1.4.6	Ableiten von Testdaten und -szenarien .....	27
3.1.4.7	Implementieren von Testprogrammen .....	29
3.1.4.8	Implementieren der Nutzerschnittstelle.....	30
3.1.4.9	Spezifizieren und Kapseln der Datenbankzugriffe .....	32
3.1.4.10	Spezifizieren und Kapseln von entfernten Aufrufen.....	35
3.1.4.11	Implementieren der Systemfunktionalität.....	38
3.1.4.12	Kapseln von Fremdsystemen .....	40
3.1.4.13	Durchführen von Unit-Tests .....	42
3.1.4.14	Erstellen und Anbinden der Online-Hilfe.....	44
3.1.4.15	Implementieren von Werkzeugen zur Installation und Konfiguration .....	46
3.1.4.16	Mitarbeiten bei Nutzerschulungen .....	48
3.1.4.17	Unterstützen von Systemintegration und -test.....	50

# 1 Einführung: Referenzprozesse als Curricula

---

Das Referenzprojekt des Software Developer verdeutlicht paradigmatisch die diesem Tätigkeitsfeld zugrunde liegenden Arbeitsprozesse, die mit ihnen verbundenen Ansprüche sowie die daraus resultierenden Anforderungen an Inhalt und Durchführung einer qualitativ hochwertigen Weiterbildung.

Das Referenzprojekt erfüllt mehrere Funktionen:

## **Aus der Praxis für die Praxis**

Als Abstraktion tatsächlich stattgefundener Projekte und Prozesse bieten die Referenzprozesse eine realistische und leicht nachvollziehbare Abbildung dessen, was die Tätigkeiten eines Software Developer sind.

## **Prozessorientierung als innovatives „Curriculum“**

Als vollständige Darstellung aller wichtigen Arbeitsprozesse sowie der dazugehörigen Qualifikationen, Tätigkeiten und Werkzeuge bieten die Referenzprozesse die Grundlage für die Weiterbildung zum Software Developer. Alle diese Prozesse müssen – entsprechend den Vorgaben – einmal oder mehrfach durchlaufen werden und ermöglichen dadurch den Weiterzubildenden den arbeitsplatznahen, integrativen Erwerb von relevanten Kompetenzen. Durch den Verbleib im Arbeitsprozess wird nicht nur für die Weiterzubildenden eine hohe Motivation (Arbeit an echten Projekten/Aufgaben) und Nachhaltigkeit erreicht, sondern auch – aus Sicht des Unternehmens – die Kontinuität und Qualität der laufenden Arbeiten gesichert (keine Ausfallzeit durch Seminartage, kein mühsamer Transfer).

## **Qualitätsstandard für die Weiterbildung**

Als Referenz bieten insbesondere die Teilprozesse und die mit ihnen verbundenen Tätigkeits- und Qualifikationsziele einen Qualitätsmaßstab für die arbeitsprozessorientierte Weiterbildung und die resultierenden Abschlüsse. Vollständige Transparenz und klare Zielvorgaben ermöglichen die qualitativ hochwertige Absicherung auch komplexer Kompetenzen sowie den systematischen Erwerb des notwendigen Erfahrungswissens.

## **Transferprozesse**

Die Generalisierung des Referenzprojekts aus der Praxis und seine didaktische Anreicherung ermöglichen eine leichte Auswahl angemessener Transferprozesse, deren Bearbeitung die Grundlage der Weiterbildung ist. Transferprozesse sind reale Prozesse, die Referenzprojekte in einer lernförderlichen Umgebung abbilden. Abgeschlossene Transferprozesse auf Basis der hier dargestellten Anforderungen und Qualitätsmaßstäbe sind nicht nur Qualifikationsnachweis des Einzelnen, sondern bilden auch die Basis eines angemesseneren und zielgerichteteren Umgangs mit Geschäfts- und Arbeitsprozessen im Unternehmen.

## 1.1 Ereignis-Prozess-Ketten: Symbolik

---

Die Darstellung der Referenzprozesse in Form von Ereignis-Prozess-Ketten<sup>1</sup> ermöglicht einen schnellen Überblick. Vollständigkeit kann leicht überprüft werden, Anpassungen und Modifikationen im Hinblick auf das eigene Unternehmen sind problemlos möglich und Anknüpfungspunkte an andere Prozesse, aber auch zu weiter führenden Informationen ergeben sich automatisch.

---

<sup>1</sup> Vgl. A.-W. Scheer, *Wirtschaftsinformatik*, Springer 1998.

Die bei der Darstellung der Referenz- und Teilprozesse verwendete Modellierungssprache stellt eine Anpassung und Weiterentwicklung der klassischen EPK-Modellierung dar:

Referenz- wie Teilprozesse sind aus der Sicht des jeweiligen Spezialisten, also als Arbeitsprozesse einer Person dargestellt.

Referenz- wie Teilprozesse stellen in der Regel keinen Geschäftsprozess dar.

Die EPK-Symbole werden hier wie folgt verwendet:

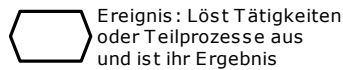
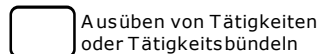
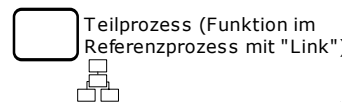


Abbildung 1: Grundlegende Symbole der Referenz- und Teilprozessmodelle.

Die wichtigsten Symbole sind:

- die Tätigkeiten bzw. Tätigkeitsbündel oder Teilprozesse, die mit dem Funktionssymbol dargestellt werden
- die Ereignisse, die Tätigkeiten bzw. Teilprozesse auslösen und Ergebnisse von Teilprozessen sind

Grundsätzlich gilt: Auf ein Ereignis folgt immer ein Teilprozess bzw. eine Tätigkeit.

Ergebnisse von Tätigkeiten sind sehr oft Dokumente; diese werden dann zusätzlich durch das Dokumentsymbol dargestellt.

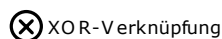
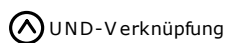


Abbildung 2: Konnektoren.

Wenn Alternativmöglichkeiten bestehen, werden Ereignisse und Teilprozesse/Tätigkeiten über Konnektoren (AND, OR, XOR) verbunden. Dabei steht AND für ein verbindendes

„Und“, OR für ein „Oder“, das alle Möglichkeiten offen lässt, und XOR für ein „ausschließendes Oder“, welches nur einen der angegebenen Pfade ermöglicht.

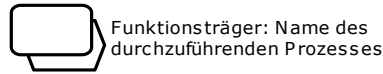


Abbildung 3: Schnittstelle.

Da die Prozesse aus der Sicht des jeweiligen Spezialisten formuliert werden, sind Schnittstellen zu Prozessen anderer Spezialisten oder zu Entscheidungsprozessen auf höherer Ebene notwendig. Dazu wird das Schnittstellensymbol verwendet. Es steht für Prozesse, die der Spezialist nicht selber durchführt, auf deren Durchführung er aber angewiesen ist. Parallel zu jeder Schnittstelle wird die Tätigkeit dargestellt, die der Spezialist selbst in diesem Zusammenhang ausübt, wie „Beraten bei ...“, „Unterstützen bei ...“ oder „Informieren des ...“.

Alle Prozesse werden durch die Verwendung dieser Symbole klar und einfach strukturiert dargestellt und sind offen für die Übertragung in konkrete Transferprozesse.

## 1.2 Referenzprozess und Teilprozesse

Der hier vorgestellte Referenzprozess und seine Teilprozesse stellen das Curriculum des Spezialistenprofils Software Developer dar.

Der Referenzprozess erhebt nicht den Anspruch eines Vorgehensmodells, sondern bildet beispielhaft den möglichen Arbeitsprozess und Verlauf eines Projekts auf Spezialistenebene ab.

Er bildet die Grundlage für Weiterbildungen und damit einen Qualitäts-, Niveau- und Komplexitätsmaßstab. Die zugehörigen Teilprozesse sind hier beispielhaft modelliert und stellen eine Möglichkeit der Durchführung dar. Einzelheiten zu den unverzichtbaren Prozessen und Kompetenzfeldern sind im Referenzprojekt festgelegt. Die Reihenfolge und die Inhalte der Teilprozesse sind abhängig vom jeweils auszuwählenden Transferprojekt und werden in diesem Zusammenhang festgelegt.

Die Darstellung der Prozesse erfolgt systematisch:

Jeder Prozess wird mithilfe von Ereignis-Prozess-Ketten dargestellt. Einem auslösenden Ereignis folgt eine Funktion, die wiederum ein oder mehrere Ereignisse zum Ergebnis hat. Ereignisse und Funktionen können mit AND, OR oder XOR, den Konnektoren, verbunden sein.

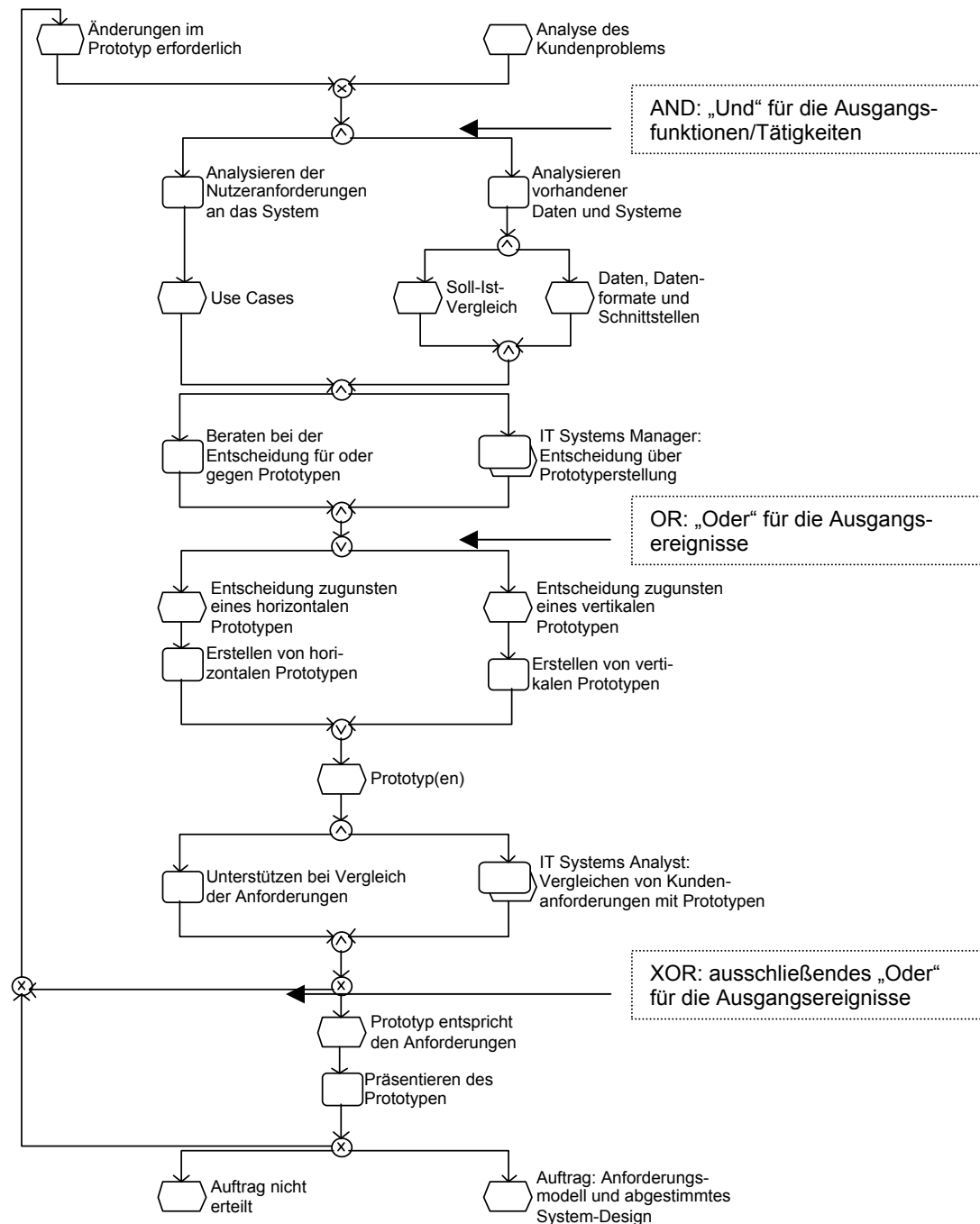


Abbildung 4: Beispielprozess (Teilprozess "Unterstützen von Systemanalyse und -design") mit unterschiedlicher Verwendung von Konnektoren.

Die Verbindung von Referenzprozess und Teilprozessen erfolgt über die Funktionen des Referenzprozesses:

Jede Funktion im Referenzprozess steht für einen Teilprozess.

Ereignisse, die dem jeweiligen Teilprozess direkt vor- oder nachgeordnet sind, sind Anfangs- und Endereignisse der jeweiligen Teilprozesse. Damit stellen die Teilprozesse die Funktionen des Referenzprozesses ausführlich dar und ein Hin- und Herbewegen zwischen Referenz- und Teilprozessen ist jederzeit problemlos möglich.

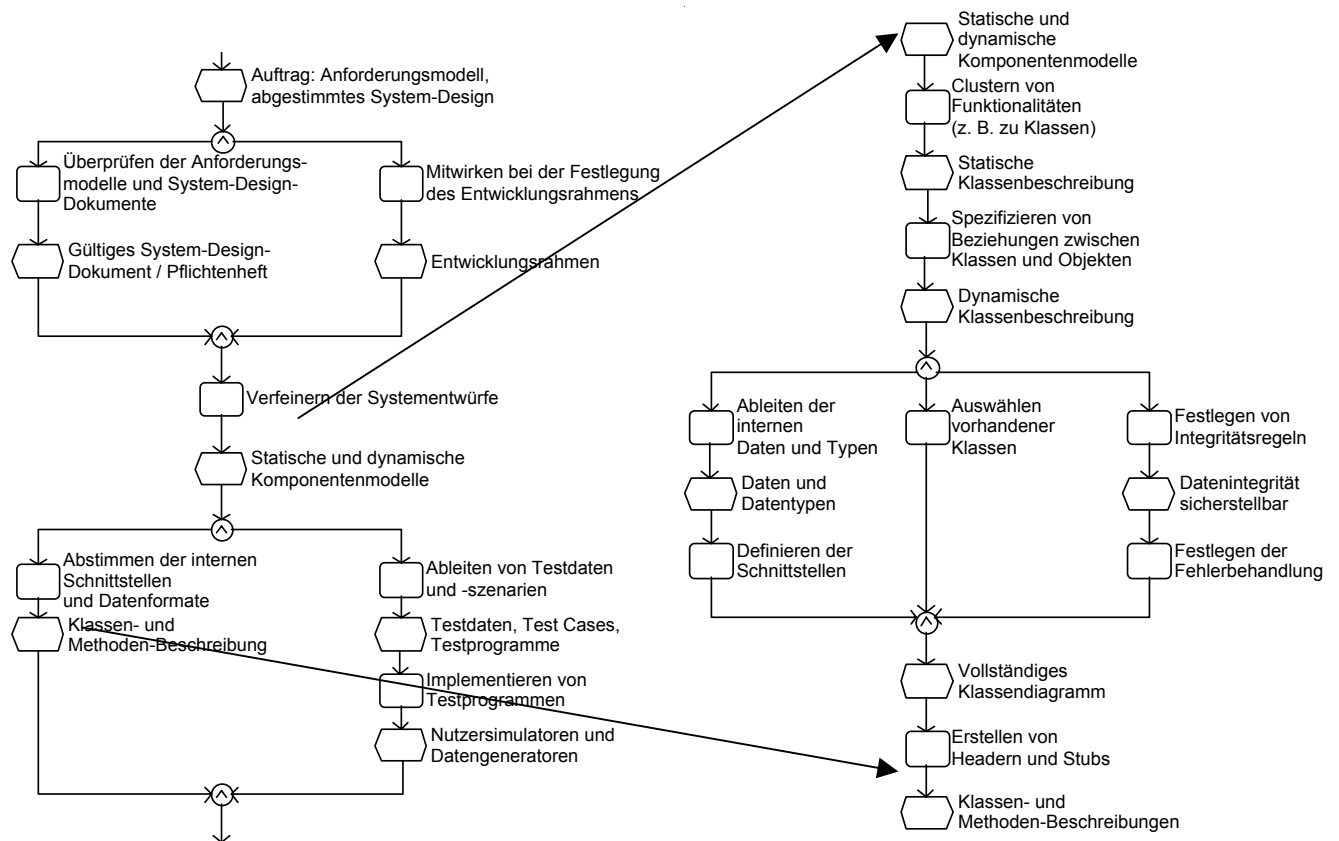


Abbildung 5: Ausschnitt aus dem Referenzprozess des Software Developer (links) und Teilprozess des Software Developer "Abstimmen der internen Schnittstellen und Datenformate" (rechts).

Die Teilprozesse stellen so die wesentlichen Teile eines Projekts dar und lassen sich entsprechend auf Transferprojekte übertragen. Den Teilprozessen sind die jeweils wesentlichen Tätigkeiten und Kompetenzfelder zugeordnet.



## 2 Das Profil: Software Developer (Softwareentwickler/in)

---

Software Developer<sup>2</sup> konzipieren und implementieren einzelne Softwarebausteine bedarfsgerecht und wirtschaftlich auf der Basis vorliegender System-, Datenbank- und GUI-Designs. Sie spezifizieren softwaretechnische Details von Softwarebausteinen und definieren Schnittstellen zu anderen Komponenten des Systems.

Software Developer entwerfen Algorithmen, definieren Datenstrukturen und setzen Programme in einer höheren (3GL- bzw. 4GL-) Programmiersprache um. Sie erstellen Testspezifikationen, Testdaten und Testumgebungen und führen Unit-Tests auf der Ebene der Softwarebausteine durch. Sie lösen Probleme im Entwicklerteam und in Kooperation mit Fachleuten aus dem Anwendungsumfeld.

### 2.1 Tätigkeitsbeschreibung

---

Die zentrale Tätigkeit des Software Developer ist das Programmieren von Softwaresystemen und deren Bausteinen, insbesondere in der industriellen Form der Softwareproduktion mit standardisierten Prinzipien, Methoden, Sprachen und Werkzeugen. Hier muss der Software Developer auf eine hohe Benutzerakzeptanz und deutliche Rationalisierung des Erstellungsprozesses achten. Dazu arbeitet er eng mit dem Konfigurationsmanagement und der Qualitätssicherung zusammen, um bei den von ihm erstellten Bausteinen und Anwendungen Zuverlässigkeit, Benutzerfreundlichkeit, Effizienz, Wartungsfreundlichkeit, Testbarkeit und Flexibilität zu sichern. Dies erfordert die Fähigkeit, mit den rasanten technologischen Weiterentwicklungen Schritt halten zu können und Technologien, Werkzeuge und Bibliotheken bezüglich ihrer Eignung für die eigene Aufgabenstellung auswählen und bewerten zu können.

Um diese zentralen Tätigkeiten herum gruppieren sich eine Reihe von vorwiegend unterstützenden Aufgaben, die von der Erstellung von Prototypen und Installationsprogrammen über verschiedene Tests bis zur Installation beim Kunden reichen. Mit Ausnahme von Großprojekten ist es üblich, dass Software Developer in die Analyse und das Systemdesign eingebunden werden. Durch ihre Erfahrung und Fachkenntnis sind sie in der Lage, die Realisierbarkeit von Lösungen und die damit verbundenen Aufwände unter Berücksichtigung von Implementierungsrisiken abzuschätzen sowie schon bei der Ausgestaltung von Systemdesigns durch Muster (Patterns) und Standardbibliotheken die Grundlagen für eine effiziente Umsetzung zu legen.

In Europa und insbesondere in Deutschland spielt aufgrund der Dominanz von sekundärer IT vor allem die kundennahe Anwendungsentwicklung eine starke Rolle. Dies bedeutet, dass die Einbindung von existierenden Fremdsystemen und die Nutzung von vom Kunden vorgegebenen Datenbasen keine Ausnahme darstellen, sondern vielmehr den Alltag vieler Softwareentwickler bestimmen. Software Developer sollten daher fachlich und methodisch in der Lage sein, Schnittstellen von existierenden Systemen zu bewerten und zu nutzen bzw. existierende Systeme so zu kapseln, dass sie als Komponenten an andere Systeme angebunden werden können. Hierzu ist ein schnelles Eindenken in Geschäftsabläufe und ein Bewusstsein für die Sicherheitsanforderungen des Kunden notwendig.

### 2.2 Profiltypische Arbeitsprozesse

---

Die im Folgenden beschriebenen Teilprozesse dokumentieren den gesamten profiltypischen Arbeitsprozess der IT-Spezialisten. Die Beherrschung dieses Arbeitsprozesses in Verbin-

---

<sup>2</sup> Kapitel 2 gibt – mit Ausnahme des Abschnitts 2.1 „Tätigkeitsbeschreibung“ – den offiziellen Text der „Vereinbarung über die Spezialistenprofile im Rahmen des Verfahrens zur Ordnung der IT-Weiterbildung“ vom 25.05.2002 (Bundesanzeiger 105, ausgegeben am 12.06.2002) wieder.

derung mit den Kompetenzen in den jeweiligen Kompetenzfeldern und der Berufserfahrung bilden die Grundlage für die berufliche Handlungskompetenz.

1. Unterstützen von IT-Systemanalytikern und IT-Systemplanern bei der Systemanalyse und dem Systemdesign, z. B. durch Erstellen von Prototypen
2. Mitwirken bei der Festlegung des Entwicklungsrahmens und der Entwicklungsumgebung, der Abschätzung von Aufwänden, der Festlegung von Meilensteinen und der Identifizierung von Implementierungsrisiken
3. Überprüfen von Anforderungsmodellen und Systemdesign-Dokumenten auf Korrektheit, Eindeutigkeit und Vollständigkeit sowie auf Realisierbarkeit der Systemanforderungen bzgl. Sicherheit und Performance; Abstimmen von funktionalen Änderungen und Erweiterungen mit IT-Systemanalytikern, IT-Systemplanern und weiteren Spezialisten aus dem Bereich Entwicklung
4. Verfeinern von Systementwürfen durch Abbilden der spezifizierten, umfangreicheren Systemkomponenten auf kleinere Softwarebausteine wie z. B. Klassen und Objekte; Spezifizieren der Interaktionen und Beziehungen dieser Softwarebausteine in Form geeigneter Diagramme
5. Abstimmen von konkreten Schnittstellen und Datenformaten innerhalb des Teams
6. Ableiten von Testfällen und -szenarien aus den Spezifikationen für die Softwarebausteine und Bereitstellen von Testdaten für den Unit-Test
7. Entwerfen von Datenbanktabellen und Mechanismen für entfernte Aufrufe, u. a. unter Verwendung von Code-Generatoren
8. Kapseln von existierenden Systemen, Konvertieren von Daten, Abbilden von komplexen Kommunikations- und Abfrageprotokollen auf Klassen und Methoden
9. Implementieren der Softwarebausteine und Durchführen der Unit-Tests, Festhalten der Testergebnisse
10. Implementieren von Installationsprogrammen
11. Unterstützen der Systemintegration und der Systemtests bzw. bei kleineren Projekten Durchführen der Systemintegration mit Unterstützung der am Projekt beteiligten Entwickler
12. Mitarbeiten bei der Erstellung von Handbüchern, Installationsanleitungen und Trainingsmaterialien

## 2.3 Profilprägende Kompetenzfelder

---

Die Beherrschung der profiltypischen Arbeitsprozesse setzt Kompetenzen unterschiedlicher Reichweite in den nachstehend aufgeführten beruflichen Kompetenzfeldern<sup>3</sup> voraus. Den Kompetenzfeldern sind Wissen und Fähigkeiten sowie typische Methoden und Werkzeuge unterschiedlicher Breite und Tiefe zugeordnet.

---

<sup>3</sup> Die Kompetenzfelder werden in der nachfolgenden Auflistung jeweils durch ein zusammenfassendes Stichwort benannt. Da die Weiterbildung zum Spezialisten auf die erfolgreiche Bewältigung zunehmend offener beruflicher Handlungssituationen sowie ganzheitlichen Kompetenzerwerb abzielt, bildet der Kompetenzerwerb einen integralen Bestandteil der Arbeits- und Weiterbildungsprozesse und lässt sich nur im Zusammenhang mit diesen operationalisieren.

Grundlegend zu beherrschende, gemeinsame Kompetenzfelder<sup>4</sup>:

- Unternehmensziele und Kundeninteressen
- Problemanalyse, -lösung
- Kommunikation, Präsentation
- Konflikterkennung, -lösung
- fremdsprachliche Kommunikation (englisch)
- Projektorganisation, -kooperation
- Zeitmanagement, Aufgabenplanung und -priorisierung
- wirtschaftliches Handeln
- Selbstlernen, Lernorganisation
- Innovationspotenziale
- Datenschutz, -sicherheit
- Dokumentation, -standards
- Qualitätssicherung

Fundiert zu beherrschende, gruppenspezifische Kompetenzfelder:

- Methoden und Werkzeuge der Softwareentwicklung
- Engineering-Prozesse
- Systemanalyse
- Entwicklungsstandards (Leistungsfähigkeit, Sicherheit, Verfügbarkeit, Innovation)
- Qualitätsstandards
- Datenbanken, Netzwerke

Routiniert zu beherrschende, profilspezifische Kompetenzfelder:

- Moduldesign, Designmuster
- Programmier- und Darstellungssprachen
- Programmbibliotheken
- Algorithmen und Datenstrukturen
- Schnittstellen
- Datenmodelle, -formate, -typen

## 2.4 Qualifikationserfordernisse

---

Im Regelfall wird ein hinreichendes Qualifikationsniveau auf der Basis einschlägiger Berufsausbildung oder Berufserfahrung vorausgesetzt.

---

<sup>4</sup> Jeder Spezialist muss in den in diesem Abschnitt genannten „weichen“ Kompetenzfeldern wie „Kommunikation, Präsentation“, „Konflikterkennung, -lösung“ usw. ein Niveau erreichen, das über dem einer Fachkraft liegt. Das heißt, er muss auch in diesen Feldern zu eigenständigem Handeln in der Lage sein und zum Erreichen des Ziels in dem jeweiligen Feld ggf. über den Rahmen bekannter Verfahren und Lösungen hinausgehen können.

## 2.5 Einordnung ins System und Karrierepfade

---

Das neue IT-Weiterbildungssystem gibt auf Basis der vier neuen IT-Ausbildungsberufe drei Ebenen für die Weiterqualifizierung vor:

1. die Spezialistenebene, auf der der Software Developer angesiedelt ist
2. die Ebene der operativen Professionals und
3. die Ebene der strategischen Professionals

Selbstverständlich kann sich der Software Developer sukzessive zu einem Professional weiterqualifizieren.

### Verwandte Profile

Die Tätigkeiten des Software Developer decken den zentralen Ausschnitt des Softwareentwicklungs-Zyklus vom Design über die Implementierung bis zum Test ab. Weitere Profile, die entlang des Softwareentwicklungs-Zyklus angesiedelt werden können, sind erheblich stärker spezialisiert, da sie jeweils nur Tätigkeiten in einem kleinen Ausschnitt des Entwicklungszyklus umfassen. Dazu gehören

- der IT Systems Analyst (mit Fokus auf Analyse und Design der Softwareprodukte) und der
- Database Developer, User Interface Developer und Multimedia Developer (mit Fokus auf die jeweilige spezielle Programmierung)

Die Tätigkeitsprofile des IT-Weiterbildungssystems können Schnittstellen zu anderen Profilen aufweisen. Der Software Developer weist Schnittstellen zu einigen Profilen auf. Dazu gehören insbesondere die Koordinatoren von Softwareentwicklungs-Prozessen wie

- der IT Configuration Coordinator (mit Fokus auf das Konfigurationsmanagement für Entwicklungsprozesse) und der
- IT Quality Management Coordinator (mit Fokus auf die Qualitätssicherung auch von Softwareprodukten)

Die Berührungspunkte der Arbeitsprozesse werden in der Prozessdarstellung ausdrücklich aufgezeigt.

Software Developer haben vielfältige Aufgaben (z. B. Programmierung von Testumgebungen oder Simulationen) und Spezialisierungsmöglichkeiten (z. B. auf bestimmte Entwicklungsmodelle und/oder Programmiersprachen). Diese können und sollen in diesem einen Profil nicht berücksichtigt werden. In der Praxis stehen diese Möglichkeiten selbstverständlich jedem Software Developer offen und sollten auch genutzt werden, um Expertisen aufzubauen.

### Aufstiegsqualifizierung

Der Übergang vom Spezialisten zum operativen Professional geschieht für Software Developer am nächsten liegenden über die fachliche Schiene zum IT Systems Manager oder durch die Leitung größerer Softwareprojekte zum IT Business Manager. Für Software Developer, die über einen längeren Zeitraum Systementwicklung für bestimmte Branchen betrieben haben, kommt auch eine Weiterqualifizierung zum (branchenspezifischen) IT Business Consultant in Frage.

### 3 Referenzprozesse

---

Bezüglich der Tätigkeiten des Software Developer (SWD) lassen sich zwei Szenarien unterscheiden: die Erstellung neuer Software(-module) und die Weiterentwicklung existierender Software(-module).

Während der Schwerpunkt im ersten Szenario auf der oftmals relativ freien Umsetzung eines Systemdesigns liegt, bietet das zweite Szenario im Normalfall sehr wenig eigene Gestaltungsfreiräume, da nicht nur die Entwicklungsumgebung, sondern auch die statische und dynamische Ausgestaltung der Komponenten vorgegeben sind. Die Schwierigkeit im zweiten Szenario liegt daher vor allem im Umgang mit oftmals unzureichend dokumentiertem Code und unvorhergesehenen Seiteneffekten.

Beide Szenarien wurden zu einem Referenzprozess "Kooperative Komponentenentwicklung" zusammengefasst, der sich von der Struktur her am ersten Szenario orientiert, durch eine starke Betonung von Systemtests und die Integration existierender Systeme jedoch auch wesentliche Aspekte des zweiten Szenarios erfasst.

#### 3.1 Kooperative Komponentenentwicklung

---

Die Entwicklung von Komponenten eines Anwendungssystems als Referenzprozess des Software Developer besteht – kurz zusammengefasst – aus folgenden ineinander greifenden Teilen:

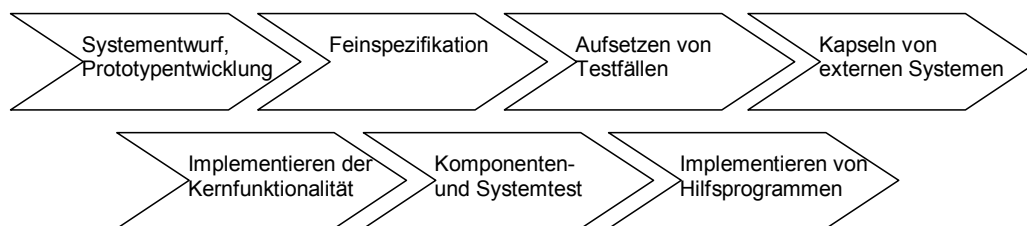


Abbildung 6: Zusammenfassung des Referenzprozesses "Kooperative Komponentenentwicklung".

Diese Prozesse werden im Folgenden ausführlich dargestellt:

Der Referenzprozess gibt den gesamten Anpassungsprozess auf hohem Abstraktionsniveau wieder und ermöglicht so einen Überblick.

Mit den Teilprozessen wird in den Referenzprozess hineingezoomt. Die Teilprozesse entsprechen damit in etwa der Abbildung von Arbeitsprozessen; sie stellen einen konkreten Tätigkeitsverlauf einschließlich auslösendem Ereignis und Ergebnis dar.

Die zur Durchführung der Teilprozesse notwendigen Tätigkeiten und Kompetenzfelder werden jeweils in einem separaten Abschnitt aufgelistet.

Das Praxisprojekt dient als Beispiel zur Konkretisierung und Veranschaulichung. Es ist ein echtes, bereits durchgeführtes Projekt, auf dessen Grundlage die hier dargestellten Referenz- und Teilprozesse entwickelt wurden.

### 3.1.1 Referenzprozess: Kooperative Komponentenentwicklung

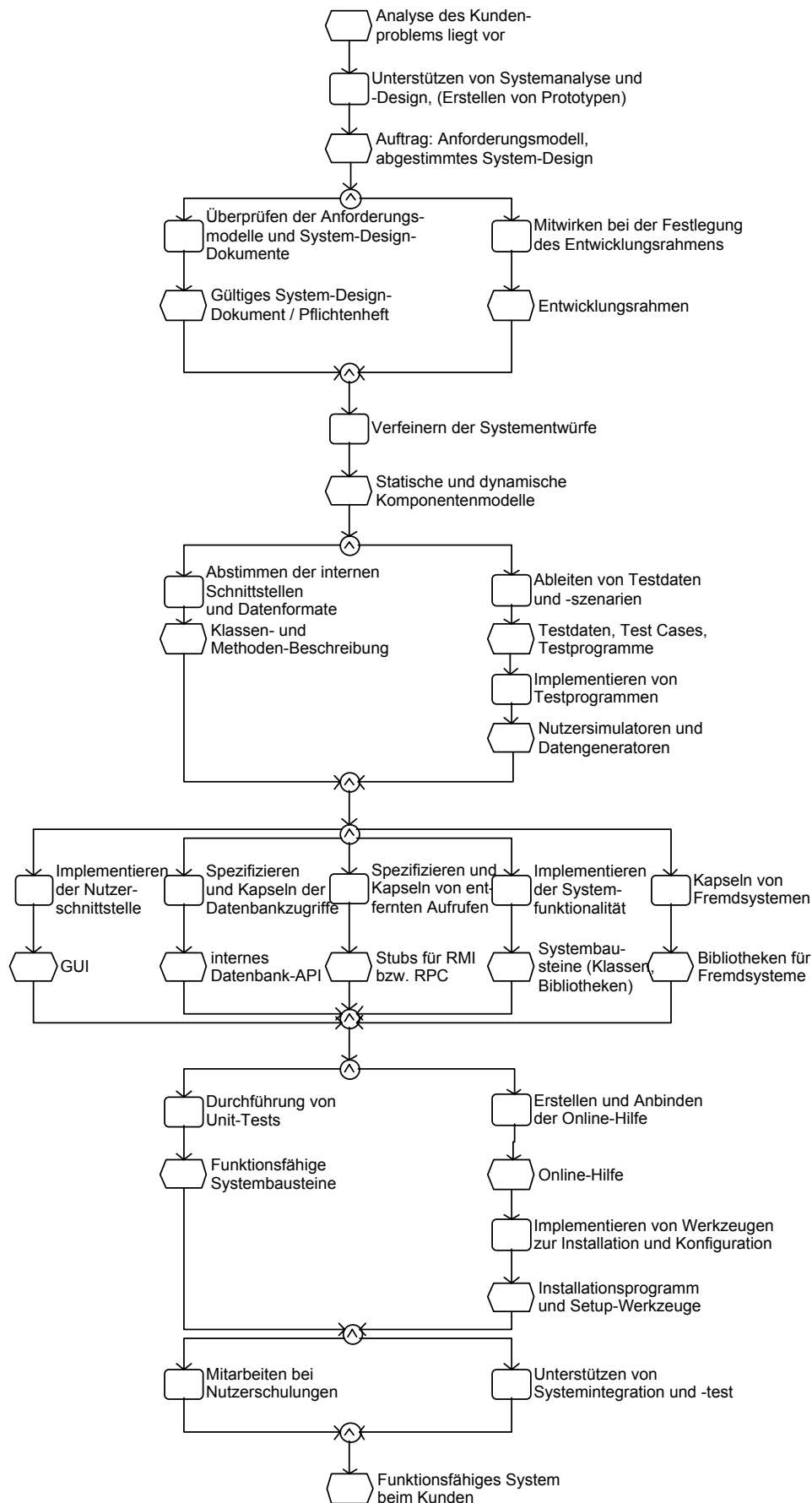


Abbildung 7: Referenzprozess des Software Developer.

Der hier dargestellte Entwicklungsprozess ist typisch für ein Developer-Profil: Nach der fachlichen und formalen Festlegung des Systemdesigns geht es mit der Feinspezifikation um die Spezifika der Komponente bzw. des Anwendungssystems. Sind auch die Testfälle festgelegt, die einzubindenden Datenbanken und anderen Fremdsysteme geeignet gekapselt sowie die Entwicklungsumgebung eingerichtet, erfolgt die Programmierung im engeren Sinne, selbstverständlich in Abstimmung mit dem Team. Nach den diversen Tests und der Systemintegration sowie der Unterstützung der Systemeinführung (u. a. durch Installations- und Konfigurationsprogramme) endet das Projekt damit, dass der Kunde ein funktionierendes System hat.

Der Referenzprozess ist prototypisch für Projekte mittlerer und größerer Komplexität, in denen mehrere Entwickler unter einer Projektleitung zusammenarbeiten. Das zu erstellende System ist auf Clients und Server verteilt (z. B. durch ein webbasiertes User Interface) und muss in eine vorhandene Systemumgebung mit Datenbanken und potenziellen weiteren Fremdsystemen integriert werden.

Der Referenzprozess basiert auf der Grundannahme, dass Softwaresysteme modular aus einzelnen, miteinander interagierenden Bausteinen (Komponenten, Klassen etc.) aufgebaut werden, wobei Funktionalität und Schnittstellen der einzelnen Bausteine aus den vorgegebenen Systemdesign-Dokumenten abgeleitet werden können.

### **3.1.2 Das Beispielprojekt: Entwicklung eines Piloten zur Anbindung eines vorhandenen E-Shops an ein vorhandenes ERP-System**

Viele Baustoffhändler verkaufen ihre Waren nicht nur direkt an Endkunden, sondern vor allem an Baugesellschaften, über die die Waren direkt an Baustellen überall in Deutschland geliefert werden. Um Lagerkapazitäten auf den Baustellen und in Zwischenlagern zu sparen, sind insbesondere für Baugesellschaften webbasierte B2B- und B2C-Lösungen attraktiv, die eine zeitnahe Anforderung von Baumaterialien bei einem Baustoffhändler ermöglichen. Das heißt, die Ware wird im Voraus (online) von der Baugesellschaft bestellt und kann anschließend vor Ort und online vom Bauleiter beim Baustoffhändler abgerufen werden, wenn sie benötigt wird.

Anlass für das diesem Referenzprojekt zugrunde liegende Praxisprojekt war die Ankündigung einer Baugesellschaft, in Zukunft wesentliche Geschäftsvorgänge primär nach dem oben beschriebenen Modell abzuwickeln. Für den an diese Gesellschaft liefernden Baustoffhändler und seinen IT-Dienstleister bedeutete dies, dass sie innerhalb kurzer Zeit eine Online-Bestellung und einen Online-Abruf sämtlicher Waren bereitstellen mussten, um den für das Unternehmen wichtigen Kunden zu halten.

Beim Baustoffhändler wurde schon lange ein Warenwirtschaftssystem auf einem Mainframe-System (BHS auf Basis von Adabas und Natural auf OS/390) eingesetzt. Der Baustoffhändler musste jetzt entscheiden, ob er das vorhandene Warenwirtschaftssystem an einen dazuzukaufenden E-Shop anbindet oder ob er eine Komplettlösung von ERP-System und E-Shop kauft und sein altes Warenwirtschaftssystem aufgibt. Um diese Entscheidung treffen zu können, wurde ein Pilotprojekt bei der Software AG Systemhaus GmbH in Auftrag gegeben. Auf der Grundlage der Erfahrungen im Pilotprojekt sollte dann über die Machbarkeit der Anbindung des vorhandenen Warenwirtschaftssystems an den auch im Pilotprojekt eingesetzten E-Shop entschieden werden. Als Gegenstand des Pilotprojekts wurde die Realisierung einer B2C-Lösung für die Anbindung des E-Shops an das ERP-System festgelegt, um so den Online-Abruf von bestellten Waren zu ermöglichen.

### 3.1.3 Prozesskompass: Kooperative Komponentenentwicklung

1. Unterstützen von Systemanalyse und -design; Entwicklung von Prototypen
2. Überprüfen der Anforderungsmodelle und Systemdesign-Dokumente
3. Mitwirken bei der Festlegung des Entwicklungsrahmens
4. Verfeinern der Systementwürfe
5. Abstimmen der internen Schnittstellen und Datenformate
6. Ableiten von Testszenarien und Testdaten
7. Implementieren von Testprogrammen
8. Implementieren der Nutzerschnittstelle
9. Spezifizieren und Kapseln der Datenbankzugriffe
10. Spezifizieren und Kapseln von entfernten Aufrufen
11. Kapseln von Fremdsystemen
12. Implementieren der Systemfunktionalität
13. Durchführen von Unit-Tests
14. Erstellen und Anbinden der Online-Hilfe
15. Implementieren von Werkzeugen zur Installation und Konfiguration
16. Mitarbeiten bei Nutzerschulungen
17. Unterstützen von Systemintegration und -test

### 3.1.4 Teilprozesse: Kooperative Komponentenentwicklung

Die Teilprozesse geben den Entwicklungsprozess eines modular aufgebauten Anwendungssystems ausführlich und detailliert wieder. Sie entsprechen so einem realen Kundenprojekt, welches als Grundlage für den Referenz- und die Teilprozesse gedient hat und als Beispiel zur Veranschaulichung beschrieben wird.

Sicherlich werden nicht in jedem Softwareentwicklungs-Projekt alle diese Teilprozesse vorkommen. Insbesondere Prozesse wie das Mitwirken bei der Festlegung des Entwicklungsrahmens oder bei den Nutzerschulungen sind sehr stark von der konkreten Einbettung des Projekts abhängig. Ein Software Developer auf der Spezialistenebene sollte allerdings alle diese Prozesse beherrschen. Die Betonung liegt dennoch auf den Prozessen, die für die Softwareentwicklung zentral sind, angefangen vom Überprüfen der Anforderungsmodelle bis hin zum Kapseln von Fremdsystemen.



### 3.1.4.1 Unterstützung von Systemanalyse und -design

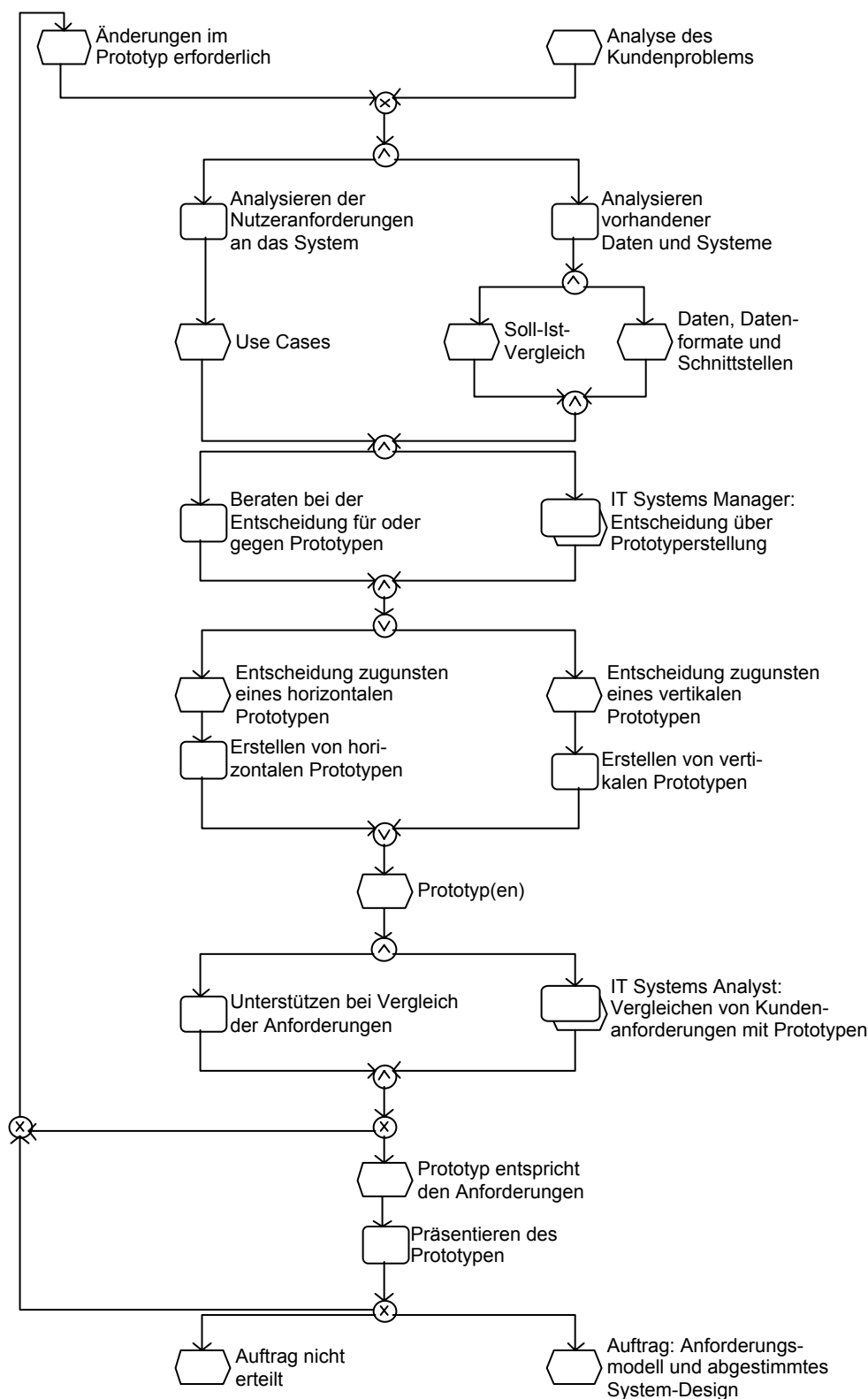


Abbildung 8: Unterstützen von Systemanalyse und -design.

Anmerkung: Die Hauptaufgabe des SWD liegt in der technologischen Beratung von Analysten und Designern sowie in der Teilnahme an der Ausarbeitung realistischer Zeit- und Kosteneinschätzungen.

#### **3.1.4.1.1 Tätigkeiten: Unterstützen von Systemanalyse und -design**

- Analysieren der Nutzeranforderungen an das System
- Analysieren vorhandener Daten und Systeme
- Beraten bei der Entscheidung für oder gegen Prototypen (Anmerkung: horizontale oder vertikale Prototypen werden erstellt, um die Akquise zu unterstützen und technische Risiken zu minimieren)
- Erstellen von horizontalen Prototypen
- Erstellen von vertikalen Prototypen
- Unterstützen beim Vergleich der Anforderungen
- Präsentieren des Prototypen

#### **3.1.4.1.2 Kompetenzfelder: Unterstützen von Systemanalyse und -design**

##### *Fähigkeiten/Fertigkeiten*

- Nutzeranforderungen verstehen und in Modelle umsetzen können
- Aufwand und technische Risiken der Realisierung von Prototypen einschätzen können
- horizontale und vertikale Prototypen entwerfen und unter Verwendung geeigneter Programmiersprachen implementieren können

##### *Wissen*

- Arten von Prototypen und deren Eigenschaften sowie Vor- und Nachteile

##### *Werkzeuge/Methoden*

- Methoden der Aufwandsschätzung
- Präsentationstechniken
- Methoden zur Modellierung von Daten und Vorgängen

#### **3.1.4.1.3 Beispiel: Unterstützen von Systemanalyse und -design**

Der IT-Dienstleister (im Folgenden als Kunde bezeichnet) des Baustoffhändlers bevorzugte eine Webshop-Lösung, bei der das vorhandene – von ihm betriebene – Warenwirtschaftssystem beibehalten werden kann, um zeitaufwändige Datenmigrationen zu vermeiden und den Schulungsaufwand seiner Mitarbeiter zu reduzieren. Die dem Kunden aus früheren gemeinsamen Projekten bekannte Software AG wurde gebeten, ein Angebot für die Umsetzung des Piloten zu unterbreiten.

Da die Schnittstellen zur Datenhaltung bekannt waren und die Schnittstellen zu Client-Systemen vorgegeben wurden, beschränkte sich die Analyse im Wesentlichen auf die Möglichkeiten der schnellen Realisierbarkeit. Da dem Baustoffhändler von seinem Kunden ein festes Datum für die Systemumstellung vorgegeben wurde, war Zeit die kritische Resource. Da nicht auszuschliessen war, dass auch andere Baustoffhändler zukünftig Online-Abrufe von Materialien verlangen würden, sollte das System möglichst modular aufgebaut sein, um verschiedene Warenwirtschaftssysteme unterstützen zu können.

Ausgehend von diesen Anforderungen wurde vom Professional Service der Software AG eine Architektur skizziert, die aus den vorhandenen Komponenten Webshop, EntireX und Tamino aufgebaut werden konnte. Zur Abschätzung der Aufwände wurde ein vertikaler Prototyp für einen kompletten Datendurchlauf vom Mainframe zum Webshop implementiert. Auf Grundlage des Prototypen und einer Architekturpräsentation wurde der Auftrag an die Software AG vergeben.

### 3.1.4.2 Mitwirken bei der Festlegung des Entwicklungsrahmens

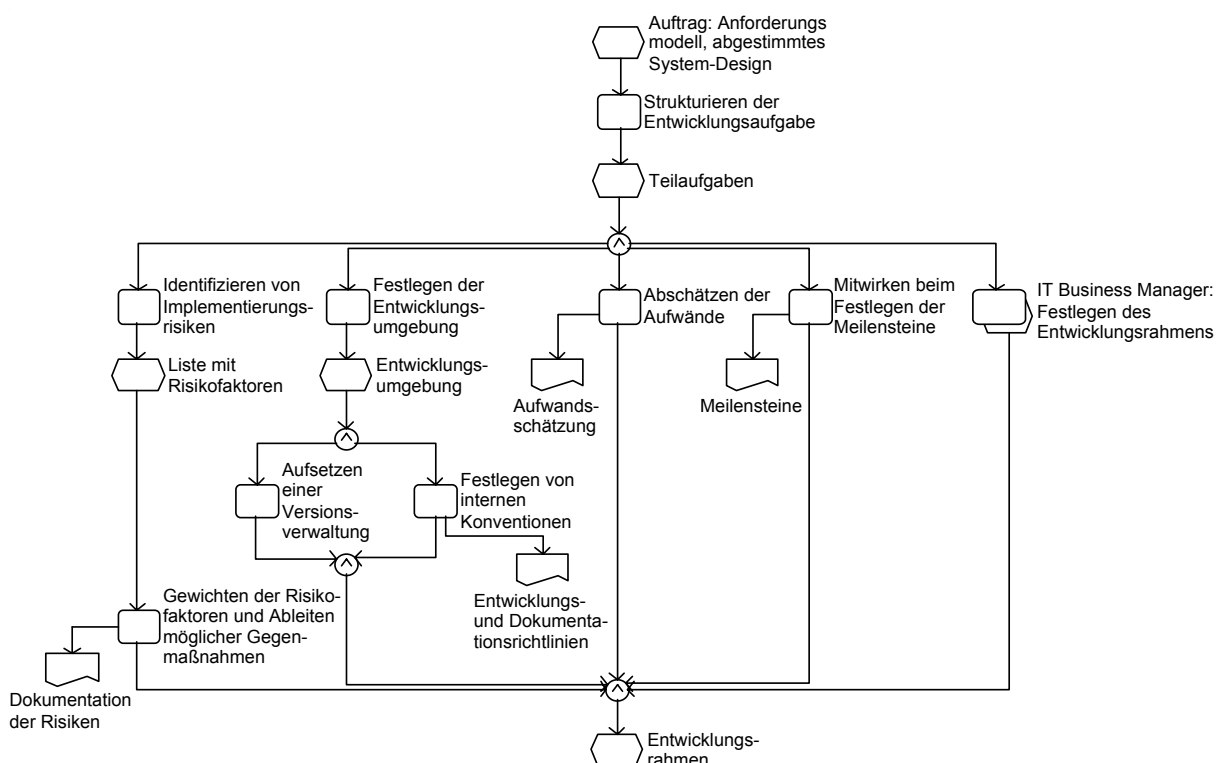


Abbildung 9: Mitwirken bei der Festlegung des Entwicklungsrahmens.

Anmerkung: In diesem Teilprozess hat der SWD eine beratende, mitwirkende oder zuliefernde Funktion.

#### 3.1.4.2.1 Tätigkeiten: Mitwirken bei der Festlegung des Entwicklungsrahmens

- Identifizieren von Implementierungsrisiken
- Festlegen der Entwicklungsumgebung
- Abschätzen der Aufwände
- Mitwirken beim Festlegen der Meilensteine – Anmerkung: der SWD strukturiert seine Aufgabe und setzt sich Teilziele für die Bearbeitung der ihm zugewiesenen Komponente; dabei muss er inhaltliche, technische und organisatorische Abhängigkeiten von anderen Teammitgliedern oder Komponenten berücksichtigen
- Gewichten der Risikofaktoren und Ableiten möglicher Gegenmaßnahmen – Anmerkung: der SWD beurteilt die Kritikalität der identifizierten Risikofaktoren, priorisiert sie und arbeitet Vorschläge für Präventiv- und Gegenmaßnahmen aus; diese dienen als Entscheidungsvorlage für den IT Systems Manager/IT Business Manager
- Aufsetzung einer Versionsverwaltung – Anmerkung: handelt es sich um ein kleineres Projekt, kann der SWD selbst eine Versionsverwaltung aufsetzen; in einem größeren Projektumfang fordert er lediglich benötigte Rechte an einem bestehenden Versionsverwaltungs-System an; die Verwaltung eines solchen obliegt dem IT Configuration Coordinator
- Festlegen von internen Konventionen – Anmerkung: bei "internen Konventionen" handelt es sich um haus- oder projektinterne Programmierstandards, Naming-Konventionen und Ähnliches

### **3.1.4.2.2 Kompetenzfelder: Mitwirken bei der Festlegung des Entwicklungsrahmens**

#### *Fähigkeiten/Fertigkeiten*

- Entwicklungsumgebungen darstellen, bewerten und auswerten können
- Implementierungsrisiken darstellen und priorisieren können
- kritische Elemente ausmachen können
- eigene Fähigkeiten sowie die Fähigkeiten von Teammitgliedern realistisch einschätzen können
- Abhängigkeiten identifizieren und darstellen können
- entwicklungsrelevante Sachverhalte abstrakt und verständlich darstellen können
- begründete Präferenzen erfolgreich kommunizieren können

#### *Wissen*

- Firmen- und Kundenstandards
- Erfahrungswerte im Risikomanagement

#### *Werkzeuge/Methoden*

- Schätzmethoden (z. B. Multiplikatormethode)
- Projektplanungs-Werkzeuge
- Methoden des Risikomanagements
- Versionsverwaltungswerkzeuge (z. B. CVS)

### **3.1.4.2.3 Beispiel: Mitwirken bei der Festlegung des Entwicklungsrahmens**

Zunächst wurden der Entwicklungsserver und die Entwicklungs-PCs aufgebaut. Entwickelt wurde auf einer NT-Plattform mit Bolero, Tamino, Java, EntireX, Adabas Natural (MVS), Apache und Jserv. Notwendig war die Einrichtung eines Webserver mit Apache und Jserv und eines gemeinsamen Repositorys. Die Entwicklungs-PCs wurden in ein lokales Netzwerk eingebunden und auf ihnen die Webshop-Sourcen installiert. Außerdem wurde der Zugriff zum Entwicklungsserver sichergestellt. Der Entwicklungsserver diente auch als Testumgebung, auf dem Testdaten bereitgestellt waren. Alle Daten wurden in einem Projektordner auf dem Server abgelegt, auf dem ein Versionierungssystem eingerichtet wurde.

Während der Entwicklung wurden abendliche Backups geplant, die aber in der Regel nicht lauffähig waren. Lauffähige Versionen wurden zusätzlich lokal gespeichert.

Das Projektteam wurde auf Basis der zu erwartenden Aufwände in einzelnen technischen Bereichen zusammengestellt. Für die Anbindung einzelner Komponenten wurden temporär Mitarbeiter anderer Abteilungen/Teams in das Projekt eingebunden.

Eine gesonderte Risikoanalyse wurde nicht durchgeführt, da durch den bereits erstellten vertikalen Prototyp eine relative Sicherheit herrschte, die gestellten Anforderungen erfüllen zu können. Auch die vorhandenen Daten waren semantisch und syntaktisch konsistent, sodass die Migration als unkritisch eingeschätzt wurde.

Um die Einhaltung des Zeitplans überwachen zu können, wurden mehrere interne Meilensteine definiert, die in einem Abstand von ca. zwei Wochen erreicht werden sollten. Zu jedem Meilenstein wurde ein Review-Plan festgelegt, in dem u. a. beschrieben wurde, welche Testszenarien zu diesem Zeitpunkt fehlerfrei zu durchlaufen sein sollten.

### 3.1.4.3 Überprüfen des Systemdesigns

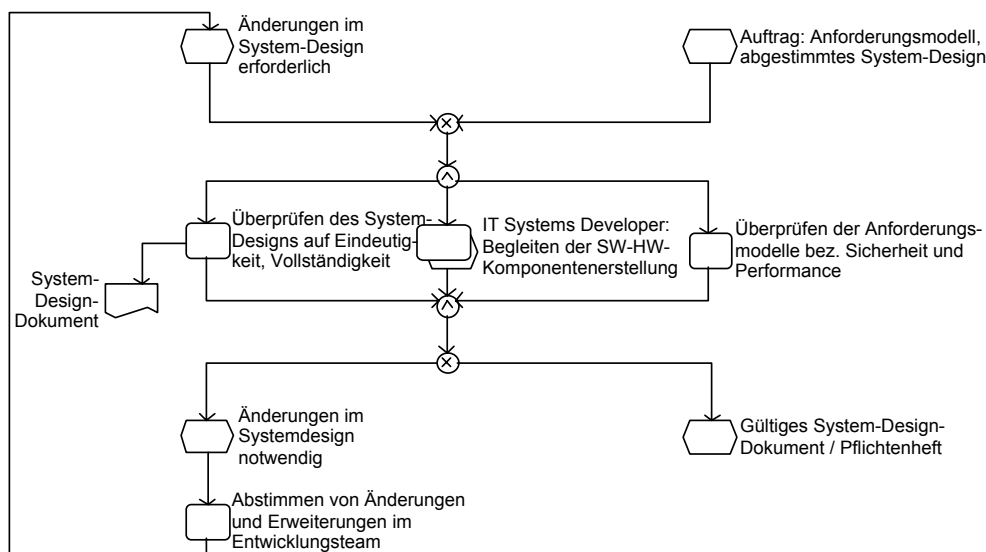


Abbildung 10: Überprüfen des Systemdesigns.

#### 3.1.4.3.1 Tätigkeiten: Überprüfen des Systemdesigns

- Überprüfen des Systemdesigns auf Eindeutigkeit, Vollständigkeit
- Überprüfen der Anforderungsmodelle bzgl. Sicherheit und Performance
- Abstimmen von Änderungen und Erweiterungen im Entwicklungsteam

#### 3.1.4.3.2 Kompetenzfelder: Überprüfen des Systemdesigns

##### Fähigkeiten/Fertigkeiten

- von abstrakten Modellen auf das spätere fertige System schließen können
- Modelle der Systemanalyse und des -designs auf Eindeutigkeit und Erfüllung der funktionalen und nichtfunktionalen Nutzeranforderungen untersuchen können
- sich mit dem Team auseinander setzen und einigen können

##### Wissen

- Design Patterns
- Sicherheitstechnologie

##### Werkzeuge/Methoden

- beschreibende Sprachen (z. B. UML)
- Methoden zur Einschätzung der Performanz

#### 3.1.4.3.3 Beispiel: Überprüfen des Systemdesigns

Da als Ausgangsbasis für die Implementierung des Systems lediglich eine Architekturbeschreibung und eine relativ grobe Anforderungsdefinition vorlag, waren für die Entwickler die konkreten Anforderungen an den Webshop relativ unklar. Aus diesem Grund wurde ein Fragebogen zu funktionalen und technischen Details erstellt, der dem Kunden übermittelt wurde. Einzelne Fragen und Antworten wurden auf einem anschließenden Workshop mit dem Kunden diskutiert. Ergebnis des Workshops war eine priorisierte Liste der im Detail umzusetzenden Funktionalitäten.

Aus einem vorangegangenen Projekt existierten Erfahrungen und vor allem Bibliotheken für die Anbindung des Webshops an eine Tamino-Datenbank. Die durch Verwendung dieser Bibliotheken möglichen Zeiteinsparungen wurden bei der Erstellung der Prioritätenliste berücksichtigt.

### 3.1.4.4 Verfeinern der Systementwürfe

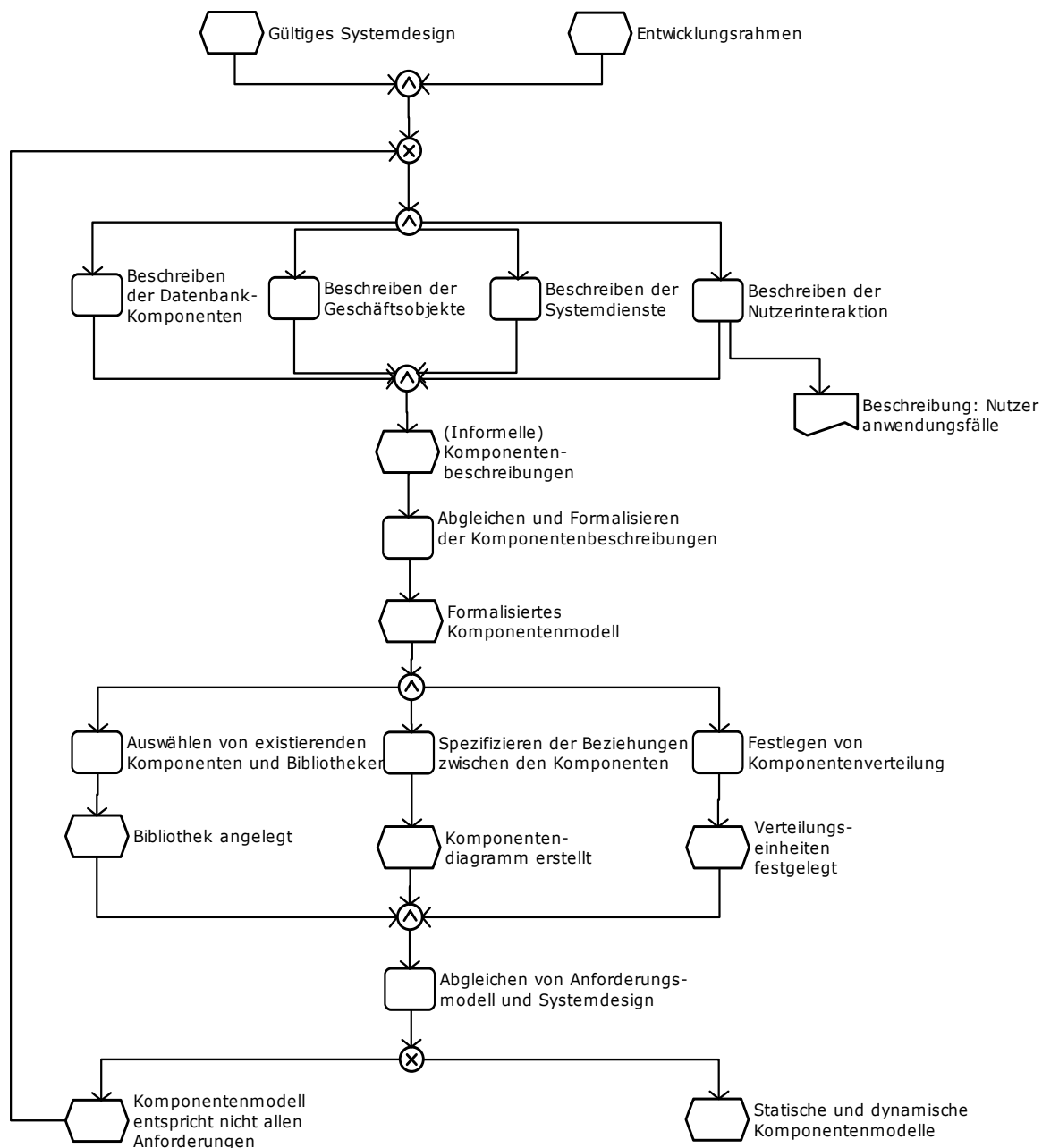


Abbildung 11: Verfeinern der Systementwürfe.

Anmerkung: Der SWD ist hier, im Gegensatz zum Teilprozess "Unterstützen von Systemanalyse und -design", aktiv und kreativ tätig.

#### 3.1.4.4.1 Tätigkeiten: Verfeinern der Systementwürfe

- Beschreiben der Datenbankkomponenten
- Beschreiben der Geschäftsobjekte
- Beschreiben der Systemdienste
- Beschreiben der Nutzerinteraktion
- Abgleichen und Formalisieren der Komponentenbeschreibungen

- Auswählen von existierenden Komponenten und Bibliotheken
- Spezifizieren der Beziehungen zwischen den Komponenten
- Festlegen der Komponentenverteilung
- Abgleichen von Anforderungsmodell und Systemdesign

#### **3.1.4.4.2 Kompetenzfelder: Verfeinern der Systementwürfe**

##### *Fähigkeiten/Fertigkeiten*

- alle Teile der zu entwickelnden Software in implementierungsnahe Komponentenmodelle übersetzen können
- Schnittstellen und Parameter der Komponenten festlegen können
- bereits existierende einsetzbare Komponenten und Bibliotheken kennen, finden und einbeziehen können
- vertragsrechtliche Risiken und Verpflichtungen aus dem Einsatz von bereits existierenden Komponenten und Bibliotheken untersuchen und zur Bewertung durch einen Entscheider aufbereiten können
- den entstandenen Softwareentwurf mit Anforderungsmodell und Systemdesign vergleichen können

##### *Wissen*

- Bibliotheken
- Vertragsrecht
- Middleware
- Client-Server-Architekturen

##### *Werkzeuge/Methoden*

- Modellierungstechniken und -methoden

#### **3.1.4.4.3 Beispiel: Verfeinern der Systementwürfe**

Jeder der Entwickler war für eine Teilkomponente des Systems verantwortlich und hat diese zunächst als Klassendiagramm dargestellt. Anforderungen an externe Systeme und Teilkomponenten wurden formal spezifiziert (d. h. welche Daten werden in welchem Grad der Aufbereitung benötigt, welche Eingabeparameter können geliefert werden?).



### 3.1.4.5 Abstimmen der internen Schnittstellen und Datenformate

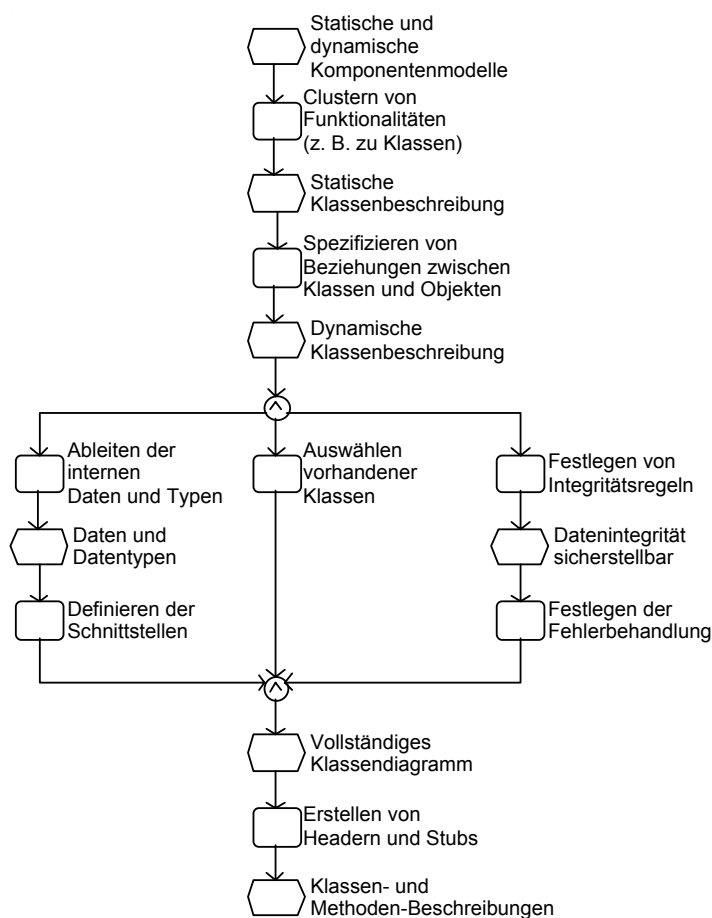


Abbildung 12: Abstimmen der internen Schnittstellen und Datenformate.

#### 3.1.4.5.1 Tätigkeiten: Abstimmen der internen Schnittstellen und Datenformate

- Clustern von Funktionalitäten (z. B. zu Klassen)
- Spezifizieren von Beziehungen zwischen Klassen und Objekten
- Ableiten der internen Daten und Typen
- Auswählen vorhandener Klassen
- Festlegen von Integritätsregeln
- Definieren der Schnittstellen
- Erstellen von Headern und Stubs

#### 3.1.4.5.2 Kompetenzfelder: Abstimmen der internen Schnittstellen und Datenformate

##### Fähigkeiten/Fertigkeiten

- Komponentenmodelle in Klassendiagramme übersetzen können
- Schnittstellen und Parameter der späteren Einheiten (z. B. Klassen, Objekte) festlegen können
- bereits existierende, einsetzbare Klassen und deren Methoden kennen, finden und einbeziehen können

- vertragsrechtliche Risiken und Verpflichtungen aus dem Einsatz von bereits existierenden Klassen untersuchen und zur Bewertung durch einen Entscheider aufbereiten können
- den entstandenen Softwareentwurf mit Anforderungsmodell und Systemdesign vergleichen können
- Datenkonsistenz sicherstellen können

#### *Wissen*

- Datentypen

#### *Werkzeuge/Methoden*

- Modellierungstechniken und -methoden

#### **3.1.4.5.3 Beispiel: Abstimmen der internen Schnittstellen und Datenformate**

Zur Abstimmung der Schnittstellen und Datenformate wurde ein eintägiger Workshop durchgeführt, auf dem jeder SWD seine Module kurz vorstellte und Anforderungen an andere Module konkret benannte. In der Diskussion untereinander wurden anschließend konkrete Schnittstellen zwischen den Modulen festgelegt und dokumentiert.

### 3.1.4.6 Ableiten von Testdaten und -szenarien

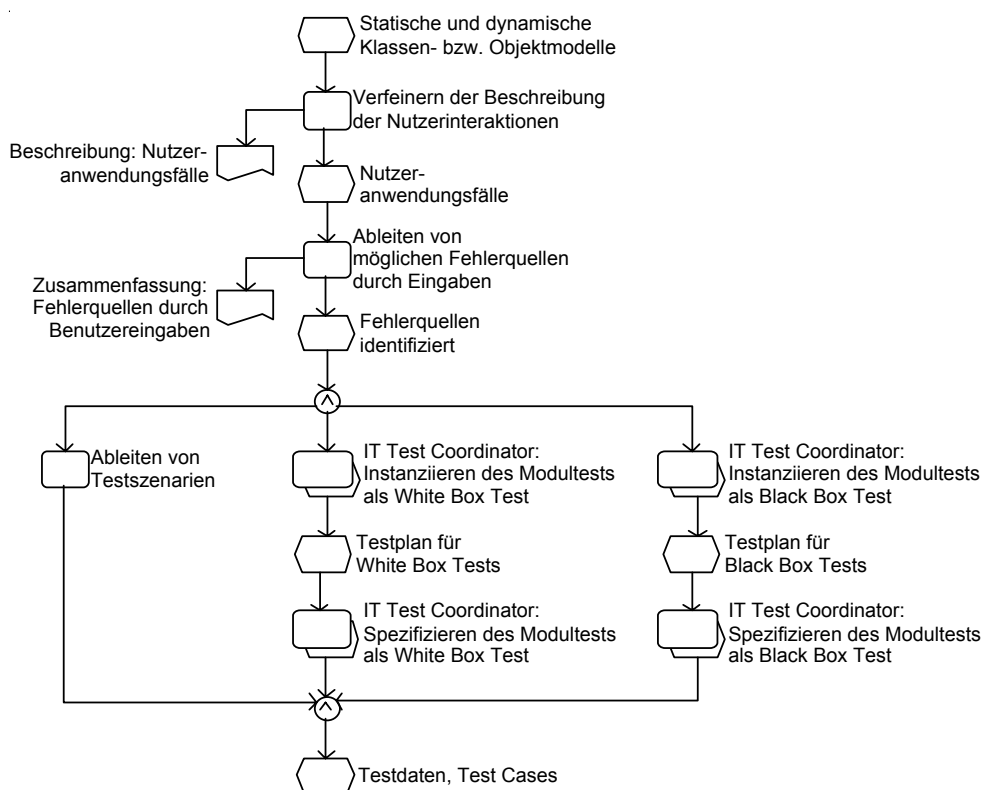


Abbildung 13: Ableiten von Testdaten und -szenarien.

#### 3.1.4.6.1 Tätigkeiten: Ableiten von Testdaten und -szenarien

- Verfeinern der Beschreibung der Nutzerinteraktionen
- Ableiten von möglichen Fehlerquellen durch Eingaben – Anmerkung: um die Fehlerquellen zu finden und einschätzen zu können, arbeitet der SWD mit Grenzdaten und möglichen (nicht gültigen) Wertebereichen
- Ableiten von Testszenarien

#### 3.1.4.6.2 Kompetenzfelder: Ableiten von Testdaten und -szenarien

##### Fähigkeiten/Fertigkeiten

- Anwendungsfälle detailliert beschreiben können
- Anwendungsfälle und Vorgänge zu Szenarien zusammenfassen können
- Testdaten ableiten können
- Softwareentwurf hinsichtlich potenzieller Fehlerquellen untersuchen können
- Risiken identifizieren und beschreiben können

##### Wissen

- Theorie der Fehlerbehandlung

*Werkzeuge/Methoden*

- statische und dynamische Testmethoden

**3.1.4.6.3 Beispiel: Ableiten von Testdaten und -szenarien**

Die vom Kunden gelieferten Daten wurden in das benötigte Eingangsformat transformiert. Zusätzlich wurden in eine Kopie dieses Datensatzes Grenzwerte und mögliche, aber nicht sinnvolle Eingangsdaten eingefügt.

### 3.1.4.7 Implementieren von Testprogrammen

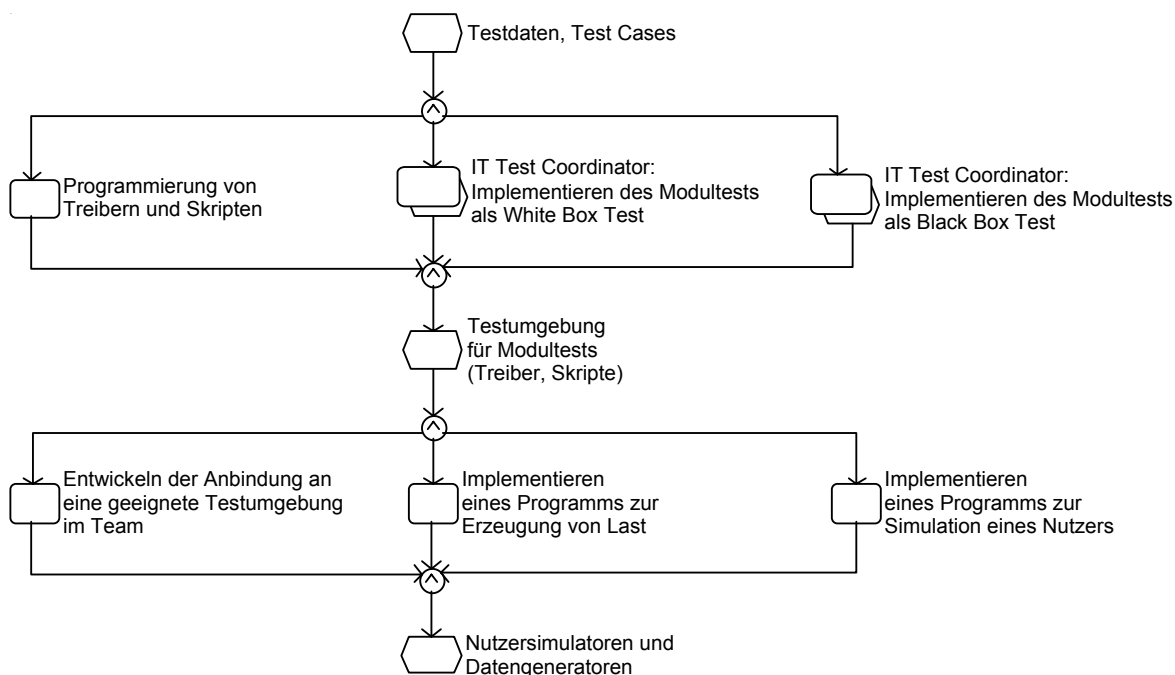


Abbildung 14: Implementieren von Testprogrammen.

#### 3.1.4.7.1 Tätigkeiten: Implementieren von Testprogrammen

- Programmieren von Treibern und Skripten
- Entwickeln der Anbindung an eine geeignete Testumgebung im Team
- Implementieren eines Programms zur Erzeugung von Last
- Implementieren eines Programms zur Simulation eines Nutzers

#### 3.1.4.7.2 Kompetenzfelder: Implementieren von Testprogrammen

##### Fähigkeiten/Fertigkeiten

- Programme und Skripte zur Steuerung von Eingaben und Datenlast implementieren und anbinden können

##### Wissen

- Kenntnis der Testmethoden und Testarten

##### Werkzeuge/Methoden

- Testumgebungen (z. B. JUnit)

#### 3.1.4.7.3 Beispiel: Implementieren von Testprogrammen

Gemeinsam wurde eine Testumgebung aufgesetzt, die die festgelegten Testdaten in die Anwendung einspeist. Für jede Komponente wurden zusätzlich Module implementiert, die ausschließlich der Sichtbarmachung von Zwischen- und Endergebnissen dienen.

### 3.1.4.8 Implementieren der Nutzerschnittstelle

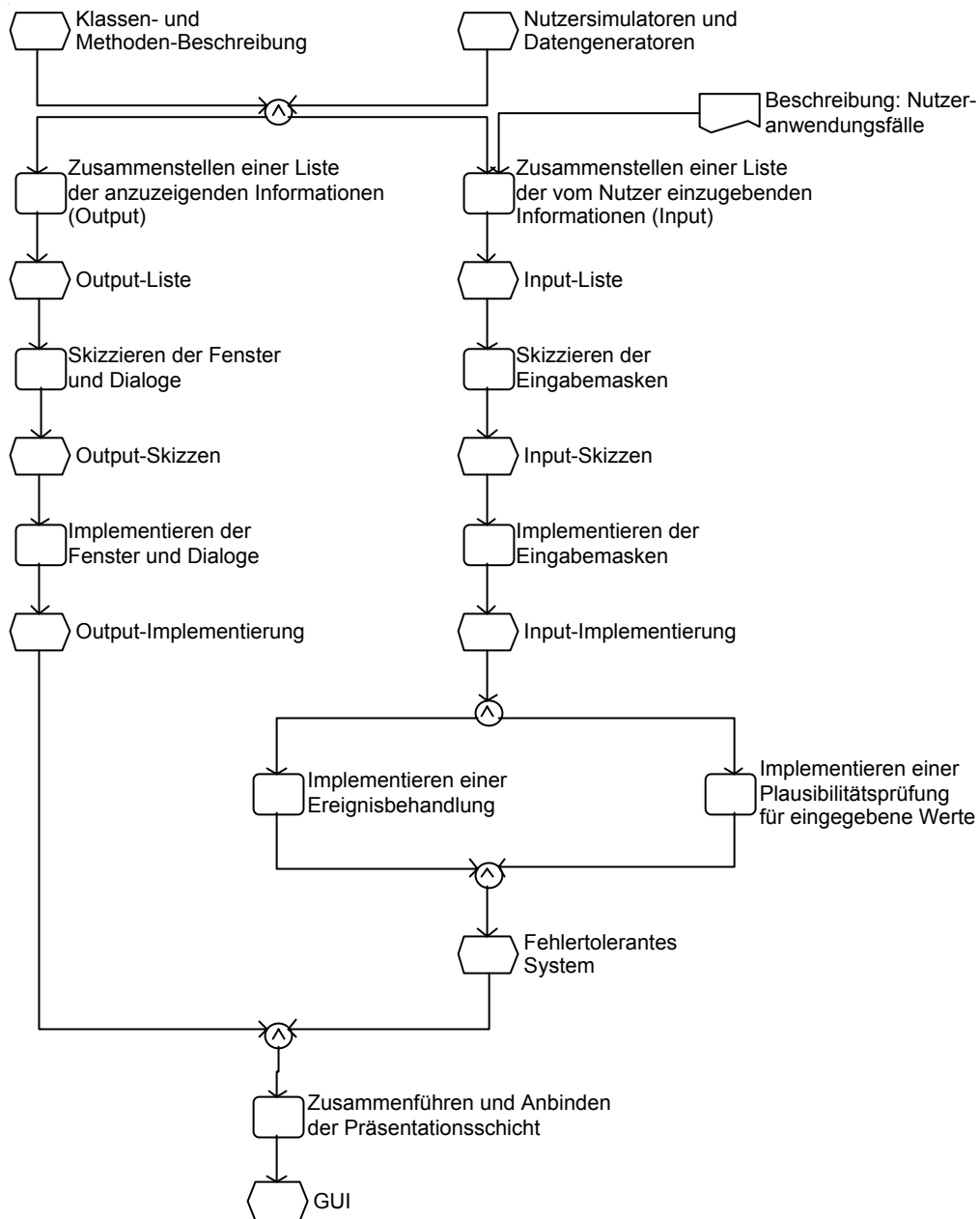


Abbildung 15: Implementieren der Nutzerschnittstelle.

Anmerkung: Je nach Größe des Projekts kann der Prozess der Nutzerschnittstellen-Entwicklung auch durch den User Interface Developer ausgeführt werden. In solchen Fällen ist der SWD lediglich für das Anbinden der Präsentationsschicht an die von ihm entwickelte Komponente verantwortlich.

#### 3.1.4.8.1 Tätigkeiten: Implementieren der Nutzerschnittstelle

- Zusammenstellen einer Liste der anzuzeigenden Informationen (Output)
- Zusammenstellen einer Liste der vom Nutzer einzugebenden Informationen (Input)
- Skizzieren der Fenster und Dialoge
- Skizzieren der Eingabemasken

- Implementieren der Fenster und Dialoge
- Implementieren der Eingabemasken
- Implementieren einer Ereignisbehandlung
- Implementieren einer Plausibilitätsprüfung für eingegebene Werte
- Zusammenführen und Anbinden der Präsentationsschicht

#### **3.1.4.8.2 Kompetenzfelder: Implementieren der Nutzerschnittstelle**

##### *Fähigkeiten/Fertigkeiten*

- Fenster, Dialoge und Eingabemasken ergonomisch und den Designrichtlinien entsprechend gestalten können
- Aktionen und Reaktionen des späteren Nutzers einschätzen und einbeziehen können
- GUI implementieren können

##### *Wissen*

- Softwareergonomie
- Usability
- firmen- oder projektinterne Designrichtlinien und -standards

##### *Werkzeuge/Methoden*

- Bibliotheken für GUI-Entwicklung (z. B. Java Swing)
- Modelle der Ereignisverarbeitung (z. B. Model View Control Pattern)
- GUI-Builder

#### **3.1.4.8.3 Beispiel: Implementieren der Nutzerschnittstelle**

Zur Interaktion des Nutzers mit der Anwendung wurden mithilfe eines GUI Design Tool geeignete Dialoge und Eingabemasken entworfen. Für alle möglichen, aber falschen Eingaben wurde ein Error Handling implementiert. Abzufangende Events wurden identifiziert und entsprechende Event Handler implementiert. Für alle interaktiven Elemente wurde sichergestellt, dass zumindest Dateneingabe, MouseOver und MouseClick mit sinnvollen Handlern belegt sind.

### 3.1.4.9 Spezifizieren und Kapseln der Datenbankzugriffe

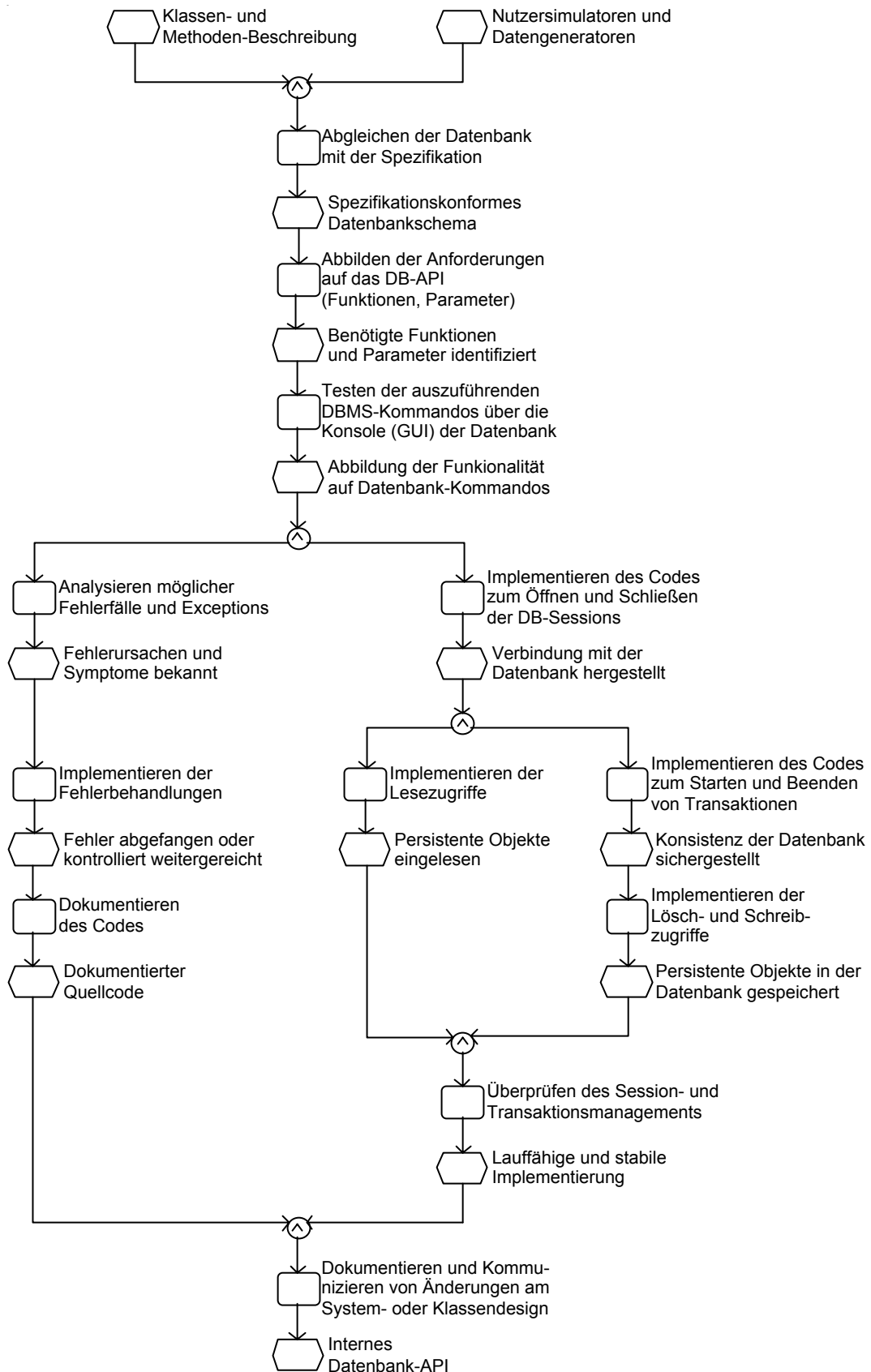


Abbildung 16: Spezifizieren und Kapseln der Datenbankzugriffe.



#### **3.1.4.9.1 Tätigkeiten: Spezifizieren und Kapseln der Datenbankzugriffe**

- Abgleichen der Datenbank mit der Spezifikation
- Abbilden der Anforderungen auf das DB-API (Funktionen, Parameter)
- Testen der auszuführenden DBMS-Kommandos über die Konsole (GUI) der Datenbank
- Analysieren möglicher Fehlerfälle und Exceptions
- Implementieren des Codes zum Öffnen und Schließen der DB-Sessions
- Implementieren der Fehlerbehandlungen
- Implementieren der Lesezugriffe
- Implementieren des Codes zum Starten und Beenden von Transaktionen
- Dokumentieren des Codes
- Implementieren der Lösch- und Schreibzugriffe
- Überprüfen des Session- und Transaktionsmanagements
- Dokumentieren und Kommunizieren von Änderungen am System- oder Klassendesign

#### **3.1.4.9.2 Kompetenzfelder: Spezifizieren und Kapseln der Datenbankzugriffe**

##### *Fähigkeiten/Fertigkeiten*

- Klassendiagramm unter Berücksichtigung der Anforderungen in ein Datenmodell übersetzen können
- Datentypen der Datenbank in die Typen der verwendeten Programmiersprache transformieren können und umgekehrt
- Modul oder Schicht für Datenbankzugriffe entwerfen und implementieren können
- entstandenen Code den Richtlinien entsprechend ausführlich dokumentieren können
- technische Risiken und mögliche Fehlerquellen identifizieren und ausräumen können

##### *Wissen*

- Dokumentationsrichtlinien
- Datenbanksysteme und -technologien
- Transaktionskonzepte
- Nutzerrechtekonzept
- Datentypen von Datenbanken

##### *Werkzeuge/Methoden*

- Datenbanksprachen (z. B. SQL)
- Modellierungsmethoden und -techniken

#### **3.1.4.9.3 Beispiel: Spezifizieren und Kapseln der Datenbankzugriffe**

Dieses Beispiel stammt nicht aus dem Praxisprojekt, sondern aus einem Projekt des Fraunhofer ISST für eine große deutsche Bank, ist aber so passend, dass es hier angeführt wird:

Ziel des Projekts war die Erstellung einer E-Learning-Anwendung mit einem kleinen in PHP programmierten Redaktionssystem. Für das Redaktionssystem mussten u. a. Angaben über die Zugriffsrechte von Autoren und die Seitenstruktur des Lernsystems in einer Datenbank verwaltet werden. Der Datenbankadministrator des Kunden hatte dazu auf dem DBS anhand der gelieferten Spezifikation die benötigten Tabellen aufgesetzt. Bei der Überprüfung der Tabellen fiel auf, dass der Datenbereich des Datentyps Integer in der Spezifikation +- 65536

und bei der Datenbank +- 32768 betrug. Darüber hinaus wurde der Datentyp „Boolean“ von der Datenbank nicht unterstützt. In beiden Fällen musste das vorher spezifizierte Datenbankschema angepasst werden. Im API der Datenbank konnten beim Aufbau der Verbindung zur Datenbank mehr als zehn Konfigurationsparameter angegeben werden. Für jeden dieser Parameter musste überprüft werden, welche Option im Sinne der Kundenanforderungen am sinnvollsten ist und gleichzeitig auch von der aktuellen Konfiguration des DBS überhaupt unterstützt wird. Beim ersten Test des Systems gab es nach ca. zehn Durchläufen einen Systemabsturz, da nur die typischen Fehler abgefangen wurden, aber offensichtlich ein laut Interpretation des Handbuchs ziemlich exotischer Fehler aufgetreten war. Nach diesem ersten Absturz konnte anschließend keine Session mehr aufgebaut werden. Da der Code sowohl mit MS Access als auch mit der MySQL- und der Oracle-Datenbank des Fraunhofer ISST fehlerfrei lief, lag eine unpassende Konfigurationseinstellung im Verbindungsaufbau oder in der Datenbank des Kunden nahe. Es dauerte mehr als zwei Tage, bis herausgefunden wurde, dass der zur PHP-Distribution gehörende Wrapper für die Datenbank des Kunden bei jedem Zugriff auf die Datenbank fehlerhafterweise implizit eine Session öffnete, die explizit geschlossen werden musste. Da in der Konfiguration des DBS eine Maximalzahl von Sessions pro Nutzer und Datenbank festgelegt war, konnte nach einer gewissen Anzahl von Zugriffen kein weiterer Zugriff erfolgen. Leider stand die ausgegebene Fehlermeldung in keinerlei Bezug zu dieser Fehlerursache. Da dieser Fehler dazu führte, dass keine geschachtelten Sessions möglich waren, mussten sowohl das Design des Datenbankwrappers als auch das Schema der Datenbank komplett umgestellt werden.

### 3.1.4.10 Spezifizieren und Kapseln von entfernten Aufrufen

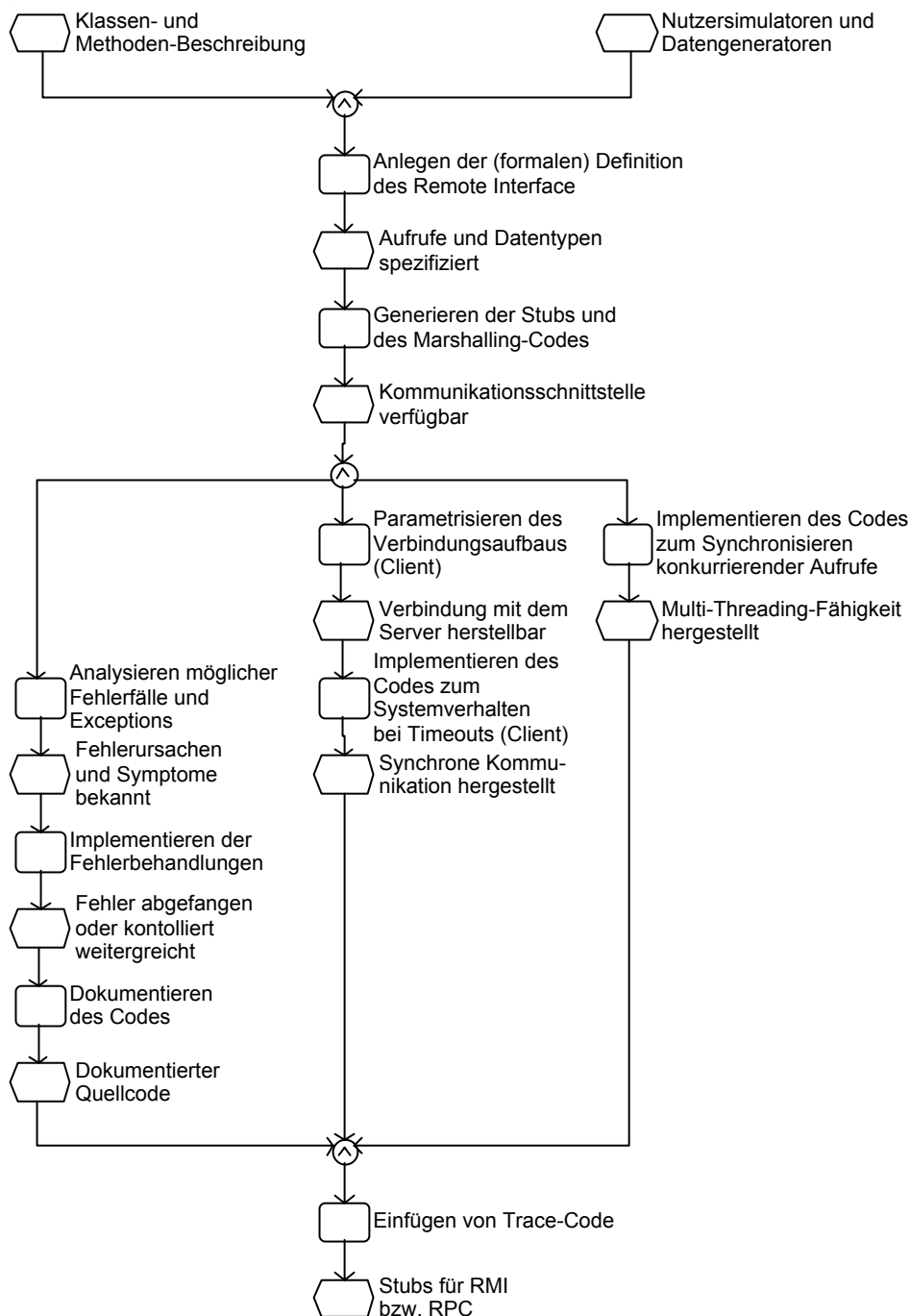


Abbildung 17: Spezifizieren und Kapseln von entfernten Aufrufen.

Anmerkung: Zum Anforderungsprofil eines SWD gehört die Fähigkeit, die Funktionalität einer Anwendungskomponente auf mehrere Rechner verteilt zu implementieren. Hierbei sollten sowohl Pull-Paradigmen (ein Client stellt eine Anforderung, die vom Server bearbeitet wird) als auch Push-Paradigmen (ein Ereignis auf dem Server löst eine Aktion auf allen Clients aus, für die dieses Ereignis relevant sein kann) umsetzbar sein.

Die besonderen programmiertechnischen Herausforderungen von verteilten Anwendungen:

- das Nichtvorhandensein eines gemeinsamen Adressraums, wodurch z. B. globale Variablen des Clients auf dem Server nicht verfügbar sind
- bei einer objektorientierten Implementierung ist eine dynamische Bindung der Parameter nicht möglich bzw. sie führt zu einem erhöhten Aufwand bei der Übermittlung von Aufrufparametern
- Anfragen an einen Server und Ereignisbenachrichtigungen für einen Client sind potenziell konkurrierend und in der Regel asynchron, d. h. hier muss der Synchronisation – insbesondere beim Zugriff auf nicht implizit multi-threading fähige Ressourcen – ein besonderes Augenmerk gewidmet werden

Zuweilen findet die Kommunikation zwischen verschiedenen Hardware-Architekturen und Betriebssystemen statt, oder es wird eine Middleware wie z. B. CORBA eingesetzt, die auch die Verwendung verschiedener Programmiersprachen für Client und Server erlaubt. In diesem Fall muss der Anwendungsentwickler eine formale Beschreibung in einer Interface Definition Language anfertigen, aus der die Stubs für die entfernten Aufrufe sowie der Code für das Marshalling der zu übergebenden Parameter und der Rückgabewerte generiert werden können.

#### **3.1.4.10.1 Tätigkeiten: Spezifizieren und Kapseln von entfernten Aufrufen**

- Anlegen der (formalen) Definition des Remote Interface
- Generieren der Stubs und des Marshalling Codes
- Parametrisieren des Verbindungsaufbaus (Client)
- Implementieren des Codes zum Synchronisieren konkurrierender Aufrufe – Anmerkung: es muss gewährleistet sein, dass durch parallele Aufrufe gestartete Prozesse bzw. Threads ohne Gefahr von Deadlocks, Race Conditions und anderen Verletzungen von impliziten Transaktionslogiken auf Ressourcen des Servers zugreifen können (z. B. schreibende Zugriffe auf globale Variablen oder Dateien)
- Analysieren möglicher Fehlerfälle und Exceptions
- Implementieren des Codes zum Systemverhalten bei Timeouts (Client)
- Implementieren der Fehlerbehandlungen
- Dokumentieren des Codes
- Einfügen von Trace Code – Anmerkung: der Trace-Code erleichtert das Debugging

#### **3.1.4.10.2 Kompetenzfelder: Spezifizieren und Kapseln von entfernten Aufrufen**

##### *Fähigkeiten/Fertigkeiten*

- Spezifikationssprache der Middleware (z. B. IDL bei Einsatz von CORBA) anwenden können
- Datentypen der Middleware in die Typen der verwendeten Programmiersprache transformieren können und umgekehrt
- Modul für entfernte Aufrufe implementieren können
- Risiken der Parallelverarbeitung kennen, einschätzen und auf sie reagieren können
- entstandenen Code den Richtlinien entsprechend ausführlich dokumentieren können

*Wissen*

- Middleware (z. B. CORBA)
- Dokumentationsrichtlinien und -standards
- Transaktionskonzepte

*Werkzeuge/Methoden*

- Methoden der Parallelverarbeitung und Synchronisation
- RMI

**3.1.4.10.3 Beispiel: Spezifizieren und Kapseln von entfernten Aufrufen**

Im Beispielprojekt konnte auf das einzubindende Warenwirtschaftssystem nur lokal zugegriffen werden. Es wurde daher geeignet gekapselt, sodass entfernte Aufrufe über die das Warenwirtschaftssystem kapselnden Klassen realisiert werden konnten. Hierzu wurden die sichtbaren Methoden auf Funktionen (mit dem betreffenden Objekt als erstem Parameter) abgebildet und diese in Corba IDL beschrieben. Die Parametrisierung der generierten Stubs war problemlos, da die Adresse des Servers und die Parameter für die Authentifizierung bereits an anderer Stelle aus einer XML-Datei gelesen wurden und somit zur Verfügung standen. Da das Handbuch des Warenwirtschaftssystems keine Hinweise auf das Verhalten bei parallelen Schreibzugriffen gab und auch der Hersteller nicht abschließend sagen konnte, ob das System intern eine Transaktionsverwaltung besitzt, wurden alle Schreibzugriffe sicherheitshalber über eine Lock-Variable synchronisiert.

### 3.1.4.11 Implementieren der Systemfunktionalität

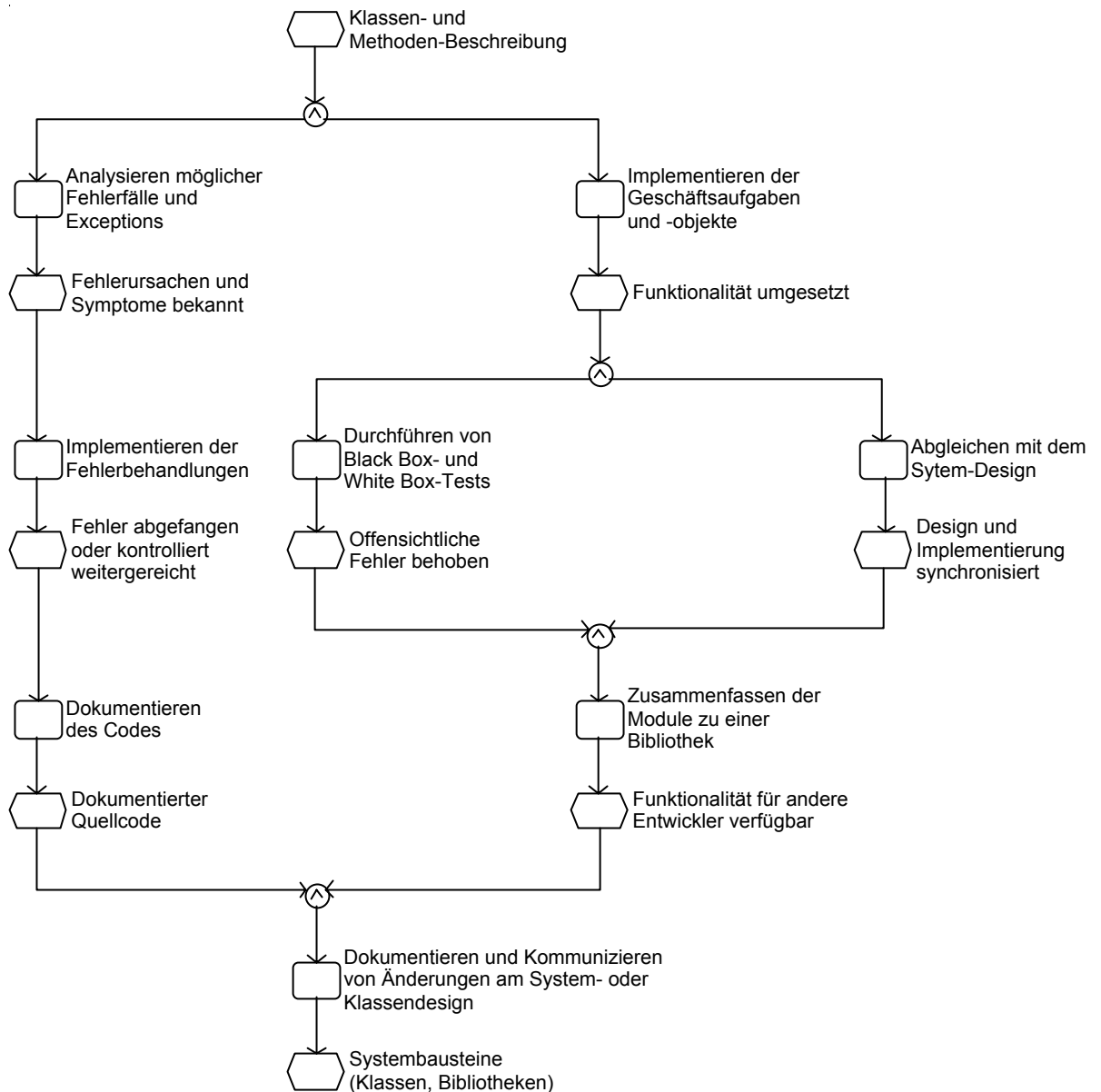


Abbildung 18: Implementieren der Systemfunktionalität.

#### 3.1.4.11.1 Tätigkeiten: Implementieren der Systemfunktionalität

- Analysieren möglicher Fehlerfälle und Exceptions
- Implementieren der Geschäftsaufgaben und -objekte
- Implementieren der Fehlerbehandlungen
- Durchführen von Black-Box- und White-Box-Tests
- Abgleichen mit dem Systemdesign
- Dokumentieren des Codes
- Zusammenfassen der Module zu einer Bibliothek
- Dokumentieren und Kommunizieren von Änderungen am System- oder Klassendesign

### **3.1.4.11.2 Kompetenzfelder: Implementieren der Systemfunktionalität**

#### *Fähigkeiten/Fertigkeiten*

- Klassendiagramm unter Berücksichtigung der Anforderungen in eine Implementierung übersetzen können
- Modul oder Schicht für Systemfunktionalität entwerfen und implementieren können
- entstandenen Code den Richtlinien entsprechend ausführlich dokumentieren können
- technische Risiken und mögliche Fehlerquellen identifizieren und ausräumen können

#### *Wissen*

- Dokumentationsrichtlinien und -standards

#### *Werkzeuge/Methoden*

- Modelle der Ereignisverarbeitung

### **3.1.4.11.3 Beispiel: Implementieren der Systemfunktionalität**

Die Implementierung erfolgte bottom-up vom Mainframe zum GUI. Programmiert wurde in Bolero und HTML. Dafür standen die Bolero-Entwicklungsumgebung, Allaire Homepage und ein XML Notepad zur Verfügung. Als Bibliotheken wurden neben diversen Standardbibliotheken ein Mail API, eine Implementierung des HTTP und die Apache Servlet Engine verwendet. Insbesondere aufwändig war es, die bestehende Logik des Webshops zu verändern. Diese Stellen waren die größte Herausforderung für die Implementierung. Insbesondere Veränderungen an der Navigation führten zu unerwünschten Seiteneffekten. Insgesamt wurde im Piloten zu 60 % Code erweitert und geändert und nur zu 40 % neu erstellt.

### 3.1.4.12 Kapseln von Fremdsystemen

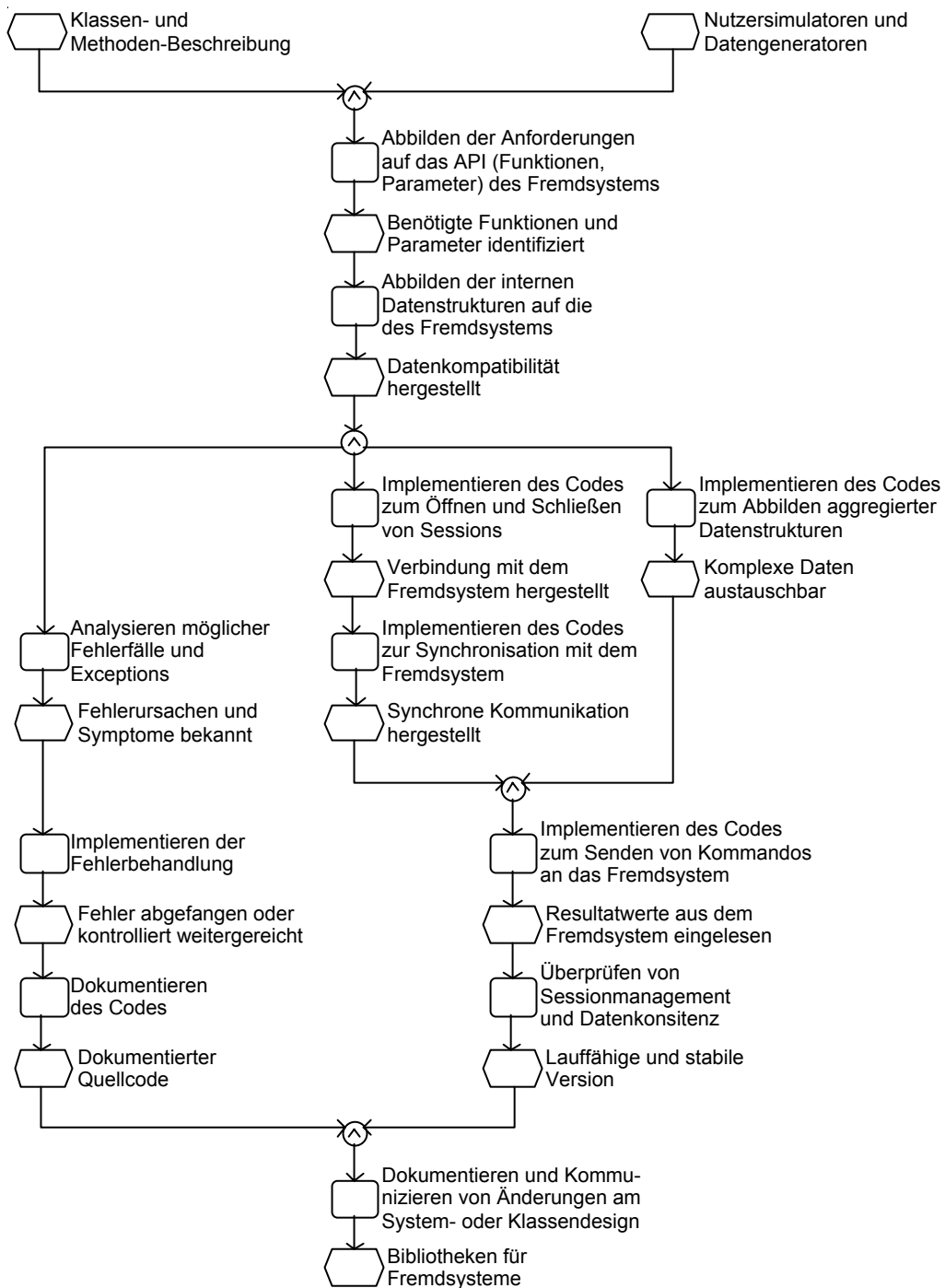


Abbildung 19: Kapseln von Fremdsystemen.

#### 3.1.4.12.1 Tätigkeiten: Kapseln von Fremdsystemen

- Abbilden der Anforderungen auf das API (Funktionen, Parameter) des Fremdsystems
- Abbilden der internen Datenstrukturen auf die des Fremdsystems
- Implementieren des Codes zum Öffnen und Schließen von Sessions
- Implementieren des Codes zum Abbilden aggregierter Datenstrukturen
- Analysieren möglicher Fehlerfälle und Exceptions



- Implementieren des Codes zur Synchronisation mit dem Fremdsystem
- Implementieren der Fehlerbehandlung
- Implementieren des Codes zum Senden von Kommandos an das Fremdsystem
- Dokumentieren des Codes
- Überprüfen von Sessionmanagement und Datenkonsistenz
- Dokumentieren und Kommunizieren von Änderungen am System- oder Klassendesign

#### **3.1.4.12.2 Kompetenzfelder: Kapseln von Fremdsystemen**

##### *Fähigkeiten/Fertigkeiten*

- Datentypen der Fremdsysteme in die Typen der verwendeten Programmiersprache transformieren können und umgekehrt
- Modul für das Anbinden von Fremdsystemen implementieren können
- Risiken der Parallelverarbeitung kennen, einschätzen und auf sie reagieren können
- entstandenen Code den Richtlinien entsprechend ausführlich dokumentieren können

##### *Wissen*

- Dokumentationsrichtlinien und -standards
- Transaktionskonzepte

##### *Werkzeuge/Methoden*

- Methoden der Parallelverarbeitung und Synchronisation

#### **3.1.4.12.3 Beispiel: Kapseln von Fremdsystemen**

Beim Kunden wurden alle Produkt- und Kundendaten über ein ERP-System verwaltet. Die Schnittstelle zu diesem System wurde in einer Klasse gekapselt, die lediglich eine Hand voll Methoden zum Lesen und Schreiben der benötigten Daten enthielt. Die Implementierung bestand aus einem Abbilden der benötigten Funktionalität auf die Schnittstelle (API) des ERP-Systems. Da der Datenexport über ein ASCII-Format erfolgte, mussten die eingehenden Daten über einen einfachen Scanner und LR (recursive descent) Parser geparkt werden. Zusätzlich wurde ein umfangreiches Error Handling implementiert, das alle möglichen Fehlermeldungen des ERP-Systems berücksichtigte.

### 3.1.4.13 Durchführen von Unit-Tests

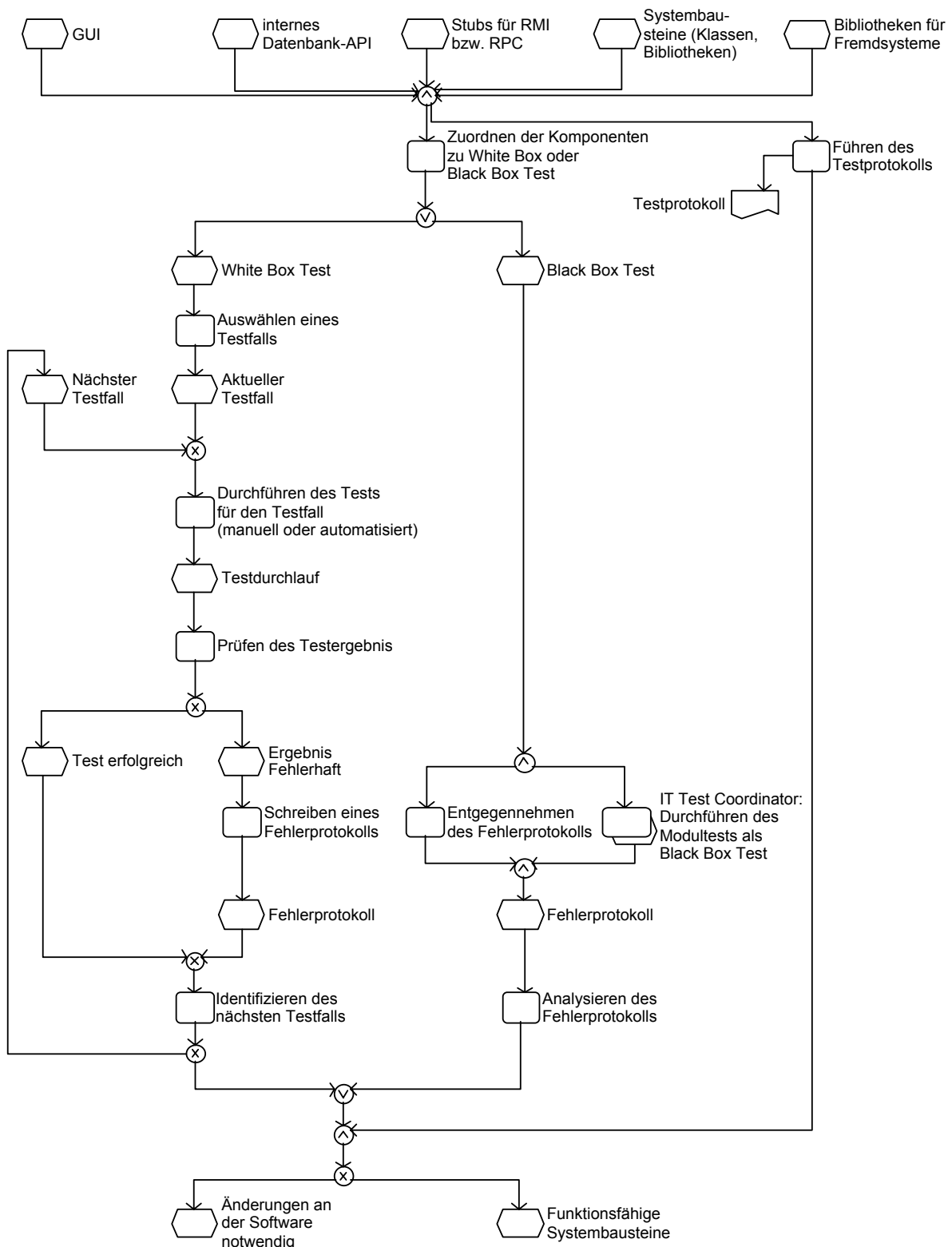


Abbildung 20: Durchführen von Unit-Tests.

#### 3.1.4.13.1 Tätigkeiten: Durchführen von Unit-Tests

- Zuordnen der Komponenten zu White-Box- oder Black-Box-Test
- Führen des Fehlerprotokolls

- Auswählen eines Testfalls
- Durchführen des Tests für den Testfall (manuell oder automatisiert)
- Prüfen des Testergebnisses
- Schreiben eines Fehlerprotokolls
- Entgegennehmen des Fehlerprotokolls
- Identifizieren des nächsten Testfalls
- Analysieren des Fehlerprotokolls

#### **3.1.4.13.2 Kompetenzfelder: Durchführen von Unit-Tests**

##### *Fähigkeiten/Fertigkeiten*

- je Komponente und Testfall eine oder mehrere passende Testarten (Black-Box-Test, White-Box-Test) und Testmethoden (automatisiert oder manuell) auswählen können
- Fehlerursachen identifizieren können
- Abweichungen im Output bewerten können
- aus den Ergebnissen Empfehlungen für das weitere Vorgehen ableiten können
- Fehler- und Testprotokolle entsprechend den Richtlinien führen können

##### *Wissen*

- Testarten
- Dokumentationsrichtlinien und -standards

##### *Werkzeuge/Methoden*

- Testumgebungen und Methoden zum automatisierten Testen (z. B. JUnit)

#### **3.1.4.13.3 Beispiel: Durchführen von Unit-Tests**

Alle Tests wurden auf einer Kopie der Kundendatenbank durchgeführt. Für alle Testdaten war bekannt, wie der Datensatz nach dem Durchlauf aussehen müsste. Über ein spezielles Programm wurde überprüft, ob die Datenbank nach dem Test den vorhergesagten Zustand besaß. Für Massentest stand ein entsprechendes Testsystem zur Verfügung, das mehrere parallele Zugriffe von Clients auf den Server simulieren konnte.



#### **3.1.4.14.2 Kompetenzfelder: Erstellen und Anbinden der Online-Hilfe**

##### *Fähigkeiten/Fertigkeiten*

- Texte strukturieren können
- Indizes erstellen können
- zielgruppenorientiert denken und formulieren können
- technische Sachverhalte für Laien verständlich darstellen können
- Dokumentation auf Vollständigkeit und Konsistenz prüfen können

##### *Wissen*

- gängige Formen und Standards der Online-Hilfe
- Dokumentationsrichtlinien und -standards

##### *Werkzeuge/Methoden*

- Help Compiler (z. B. WinHelp)
- Textverarbeitung (z. B. Microsoft Word)

#### **3.1.4.14.3 Beispiel: Erstellen und Anbinden der Online-Hilfe**

Eine Zielgruppe der Anwendung waren die Administratoren im Rechenzentrum des Kunden. Aus diesem Grund wurden im Handbuch vor allem die Datenformate und die Schnittstellen zu den Fremdsystemen beschrieben. Alle möglichen Fehlerfälle und deren mögliche Ursachen wurden ausführlich dokumentiert.

Die zweite Zielgruppe waren die Bauleiter, die über das System von der Baustelle aus Material bestellen bzw. abrufen können sollten. Deren Kommunikation mit dem System geschah über ein GUI. Zu diesem GUI wurden für jeden Dialog und jedes Eingabefeld Hilfetexte erstellt, die anschließend über eine kontextsensitive Hilfe angebunden wurden. Zusätzlich wurde ein prozessorientiertes Handbuch mit typischen Nutzungsszenarien und Beispielen erstellt und über das Menu eingebunden. Hierzu waren auch ein Index und eine Schlagwortsuche verfügbar.

### 3.1.4.15 Implementieren von Werkzeugen zur Installation und Konfiguration

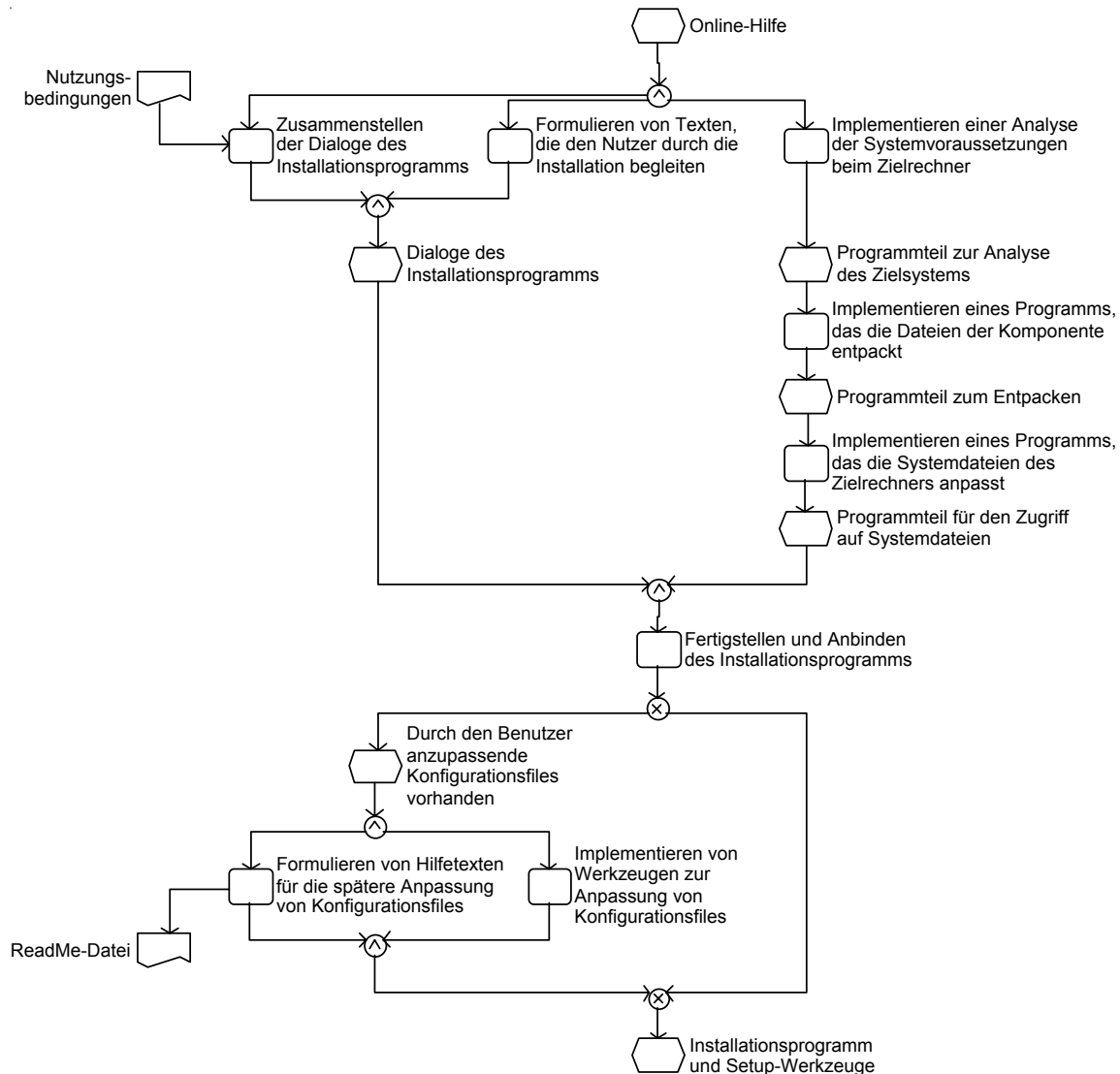


Abbildung 22: Implementieren von Werkzeugen zur Installation und Konfiguration.

#### 3.1.4.15.1 Tätigkeiten: Implementieren von Werkzeugen zur Installation und Konfiguration

- Zusammenstellen der Dialoge des Installationsprogramms
- Formulieren von Texten, die den Nutzer durch die Installation begleiten
- Implementieren einer Analyse der Systemvoraussetzungen beim Zielrechner – Anmerkung: hier kann alternativ zur Neuentwicklung auch ein bestehendes Programm zum Analysieren der Hardware/Software des Zielrechners eingebunden werden
- Implementieren eines Programms, das die Dateien der Komponente entpackt – Anmerkung: hier kann alternativ zur Neuentwicklung auch ein bestehendes Programm zum Entpacken der Dateien auf dem Zielrechner eingebunden werden
- Implementieren eines Programms, das die Systemdateien des Zielrechners anpasst – Anmerkung: hier kann alternativ zur Neuentwicklung auch ein bestehendes Programm zur Anpassung der Systemdateien auf dem Zielrechner eingebunden werden
- Fertigstellen und Anbinden des Installationsprogramms
- Formulieren von Hilfetexten für die spätere Anpassung von Konfigurationsfiles

- Implementieren von Werkzeugen zur Anpassung von Konfigurationsfiles – Anmerkung: hier kann alternativ zur Neuentwicklung auch ein bestehendes Programm zur Anpassung von Konfigurationsfiles eingebunden werden

#### **3.1.4.15.2 Kompetenzfelder: Implementieren von Werkzeugen zur Installation und Konfiguration**

##### *Fähigkeiten/Fertigkeiten*

- Installationswerkzeug entwerfen und implementieren können
- Zielumgebung kennen und konfigurieren können
- Texte formulieren können, die den Nutzer bei der späteren Installation unterstützen
- Installationsanleitung erstellen können

##### *Wissen*

- Nutzungsbedingungen
- Dokumentationsrichtlinien und -standards
- gängige Installationswerkzeuge und -standards

##### *Werkzeuge/Methoden*

- Werkzeuge zum Erstellen von Installationsassistenten (z. B. InstallShield)
- Werkzeuge zum Entpacken von Daten (z. B. WinZIP)
- Werkzeuge für den Zugriff auf Systemdateien eines Rechners

#### **3.1.4.15.3 Beispiel: Implementieren von Werkzeugen zur Installation und Konfiguration**

Ein Installationsprogramm wurde lediglich als Batch-Datei zur Verfügung gestellt, da die Entwicklungsumgebung eine exakte Kopie der Laufzeitumgebung beim Kunden war.

Wesentliche Parameter des GUI (z. B. Labeltexte und Schriftgrößen) und der Kommunikation zwischen Client und Server (Adressen, Pfade etc.) wurden vom System dynamisch aus verschiedenen in XML kodierten Konfigurationsdateien eingelesen. Für die wichtigsten Konfigurationsdateien wurden graphische Oberflächen erstellt, um die Konfiguration lesen und schreiben zu können.

### 3.1.4.16 Mitarbeiten bei Nutzerschulungen

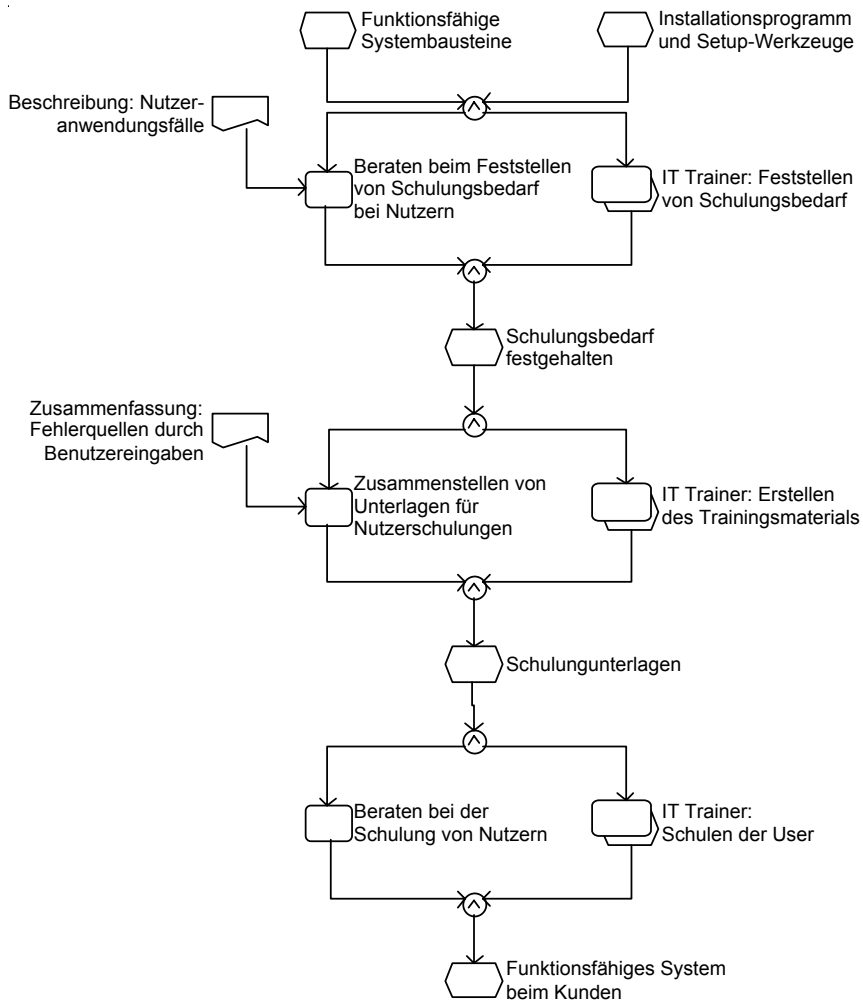


Abbildung 23: Mitarbeiten bei Nutzerschulungen.

#### 3.1.4.16.1 Tätigkeiten: Mitarbeiten bei Nutzerschulungen

- Beraten beim Feststellen von Schulungsbedarf bei Nutzern
- Zusammenfassen von Unterlagen für Nutzerschulungen
- Beraten bei der Schulung von Nutzern

#### 3.1.4.16.2 Kompetenzfelder: Mitarbeiten bei Nutzerschulungen

##### Fähigkeiten/Fertigkeiten

- Kompetenzen der Nutzer einschätzen können
- zielgruppenorientiert denken und formulieren können
- technische Sachverhalte für Außenstehende verständlich darstellen können
- fachliche Hinweise zu Schulungsbedarf und Schulungsplan geben können
- Kenntnisse aussuchen und vermitteln können, die für die unterschiedlichen Nutzergruppen notwendig und hinreichend sind



*Wissen*

- Metawissen (Abhängigkeiten zwischen Wissen und Vorwissen)

*Werkzeuge/Methoden*

- Visualisierungs- und Präsentationstechniken

**3.1.4.16.3 Beispiel: Mitarbeiten bei Nutzerschulungen**

Die Integration in die vorhandene Infrastruktur wurde gemeinsam mit dem Kunden durchgeführt. Hierbei wurden den späteren Administratoren von den Softwareentwicklern die Funktionalität des Systems sowie die Verwendung und Kodierung der verschiedenen Konfigurationsdateien erklärt. Alle während der Testphase vom Kunden den Entwickler und den Projektleitern gestellten Fragen (sowie deren Antworten) wurden zu einer FAQ-Liste zusammengefasst. Gemeinsam mit dem Projektleiter wurden auf Basis dieser Frageliste sowie der während der Systemintegration mit den Entwicklern geführten Gespräche die Inhalte für einen eintägigen Workshop mit den Administratoren festgelegt und priorisiert.



### **3.1.4.17.2 Kompetenzfelder: Unterstützen von Systemintegration und -test**

#### *Fähigkeiten/Fertigkeiten*

- die eigenen Komponenten und Module für die Integration vorbereiten können
- Fehlerursachen identifizieren können
- Abweichungen im Output bewerten können
- aus den Ergebnissen Empfehlungen für das weitere Vorgehen ableiten können
- Fehler- und Testprotokolle entsprechend den Richtlinien führen können

#### *Wissen*

- Testarten
- Dokumentationsrichtlinien und -standards

#### *Werkzeuge/Methoden*

- Testumgebungen und Methoden zum automatisierten Testen (z. B. JUnit)

### **3.1.4.17.3 Beispiel: Unterstützen von Systemintegration und -test**

Bereits zu Beginn der Implementierung wurde die Umgebung des Kunden beim Software AG Systemhaus nachgebildet. Alle Datenbanken und das ERP-System wurden mit realen Kundendaten bestückt. Aus diesem Grund war zunächst keine explizite Systemintegration notwendig, da die Software quasi schrittweise in die Umgebung "hineinwuchs".

Auch alle vorher festgelegten Tests wurden innerhalb der nachgebauten Umgebung durchgeführt. Insbesondere Tests mit großen Datenmengen und vielen parallelen Zugriffen konnten so erheblich vereinfacht werden, da aufgetretene Fehler ohne eine nennenswerte Verzögerung der Testläufe unmittelbar behoben werden konnten.

Die Installation beim Kunden war anschließend problemlos und konnte innerhalb eines Tages durchgeführt werden.