

Referenzprofil

User Interface Developer

Alexander Karosseit

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



Deutsche Angestellten Aka-
demie

Fraunhofer ISST

Unternehmenspartner

Bildungspartner

Danksagung

Diese Profilbeschreibung entstand auf Basis von Praxisprojekten der *Materna GmbH*. Wir danken dem Projektleiter Herrn Georg Lodde sowie dem Produktmanager Herrn Dirk Struck herzlich für ihre fachkundige und umfassende Hilfe. Fachlich beratend mitgewirkt haben Herr Oliver Thies, *Deutsche Angestellten Akademie DAA*, und Herr Thomas Kafurke, *Intershop AG*, sowie Frau Marleen Kiral vom *Fraunhofer ISST* Berlin. Ohne ihre Hilfe hätte dieses Dokument nicht entstehen können.

Inhalt

1	EINFÜHRUNG: REFERENZPROZESSE ALS CURRICULA	4
1.1	EREIGNIS-PROZESS-KETTEN: SYMBOLIK	4
1.2	REFERENZPROZESS UND TEILPROZESSE	6
2	DAS PROFIL: USER INTERFACE DEVELOPER (NUTZERSCHNITT STELLENENTWICKLER/IN)	8
2.1	TÄTIGKEITSBESCHREIBUNG	8
2.2	PROFILTYPISCHE ARBEITSPROZESSE	9
2.3	PROFILPRÄGENDE KOMPETENZFELDER.....	10
2.4	QUALIFIKATIONSERFORDERNISSE.....	11
2.5	EINORDNUNG INS SYSTEM UND KARRIEREPFADE	11
3	REFERENZPROZESS	12
3.1	NUTZERSCHNITTSTELLEN-ENTWICKLUNG	12
3.1.1	Referenzprozess: Nutzerschnittstellen-Entwicklung	13
3.1.2	Das Beispielprojekt: DX-Union	14
3.1.3	Prozesskompass: Nutzerschnittstellen-Entwicklung	15
3.1.4	Teilprozesse: Nutzerschnittstellen-Entwicklung	15
3.1.4.1	Unterstützen von Systemanalyse und -design	17
3.1.4.2	Mitwirken bei der Festlegung des Entwicklungsrahmens	20
3.1.4.3	Überprüfen der Anforderungsmodelle und Systemdesign-Dokumente.....	22
3.1.4.4	Einbeziehen und Abgleichen invarianter Rahmenbedingungen	24
3.1.4.5	Verfeinern der Entwürfe der Interaktionslogik	26
3.1.4.6	Verfeinern der Entwürfe der Nutzersicht	28
3.1.4.7	Reverse Engineering bestehender Komponenten	30
3.1.4.8	Festlegen von Szenarien für den Akzeptanztest.....	32
3.1.4.9	Abstimmen der Spezifikation mit dem Nutzer	34
3.1.4.10	Festlegen der Schnittstellen zur Anwendungs- bzw. Datenbankschicht.....	36
3.1.4.11	Festlegen von Datenformaten und Constraints	39
3.1.4.12	Entwerfen der Softwaremodule	41
3.1.4.13	Spezifizieren von Modultests	43
3.1.4.14	Abstimmen des Entwurfs mit dem Team.....	46
3.1.4.15	Konfigurieren und Anpassen von COTS-Komponenten	48
3.1.4.16	Realisieren der Softwaremodule	50
3.1.4.17	Integrieren der Module und Komponenten in die Gesamtarchitektur.....	53
3.1.4.18	Unterstützen des Deployment-Prozesses	55

1 Einführung: Referenzprozesse als Curricula

Das Referenzprojekt des User Interface 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 User Interface 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 User Interface 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-Ketten¹ 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.

¹ 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:

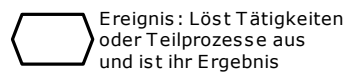
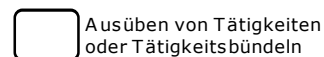
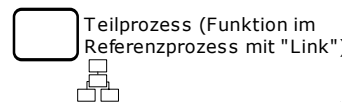


Abbildung 1: Grundlegende Symbole der Referenz- und Teilprozessmodelle.

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.

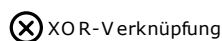
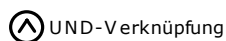


Abbildung 2: Konnektoren.

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.

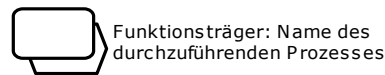


Abbildung 3: Schnittstelle.

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 User Interface 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 als Ergebnis hat. Ereignisse und Funktionen können mit AND, OR oder XOR, den Konnektoren, verbunden sein.

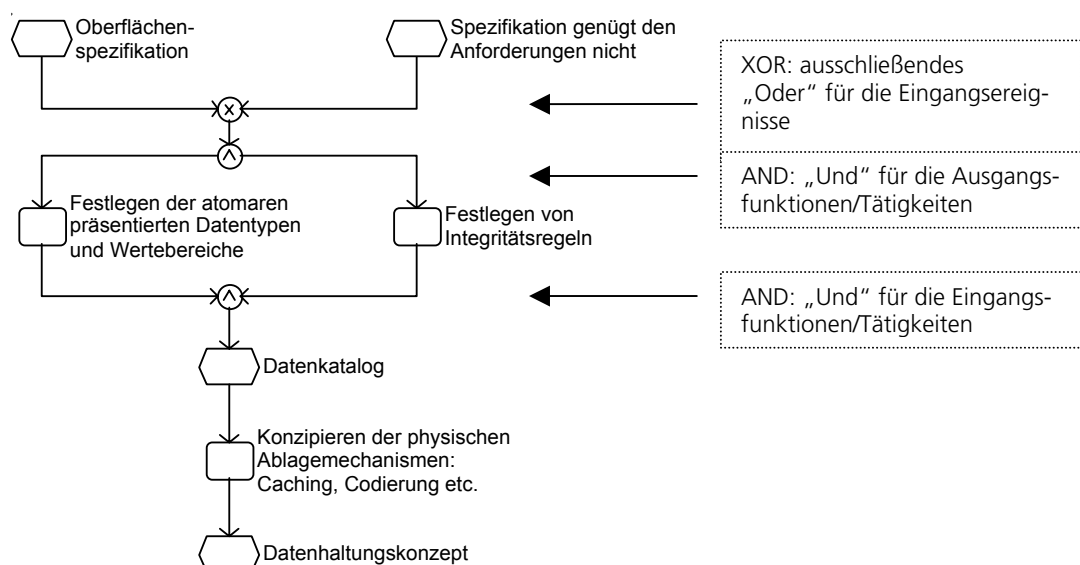


Abbildung 4: Beispielprozess (Teilprozess "Festlegen von Datenformaten und Constraints") mit unterschiedlicher Verwendung von Konnektoren.

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.

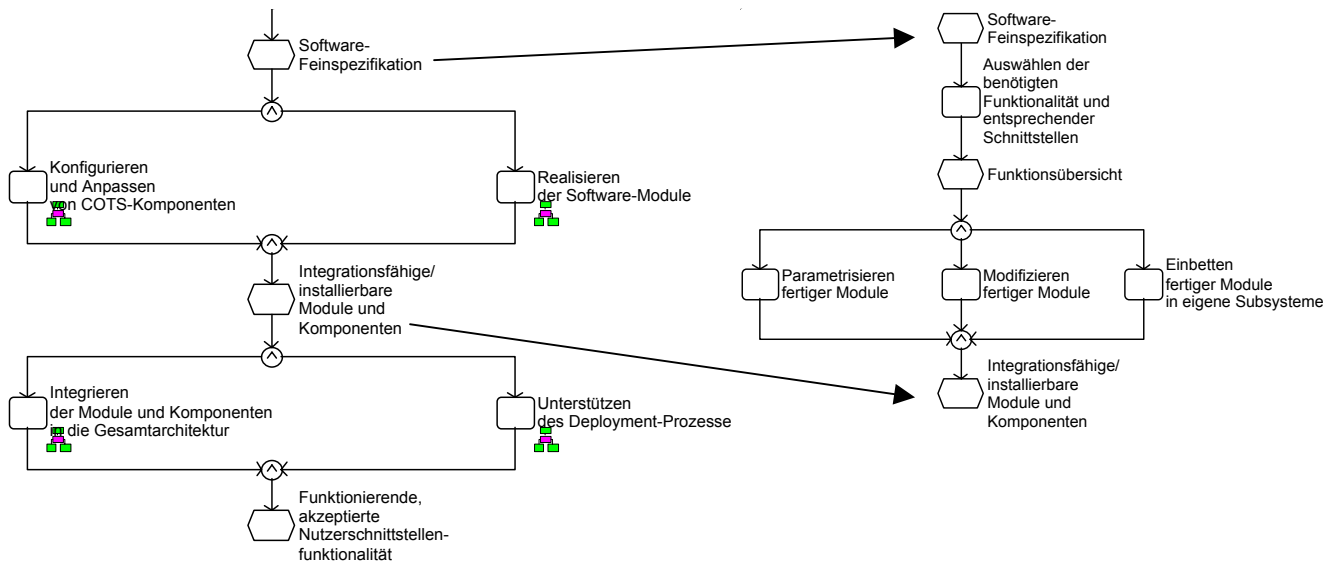


Abbildung 5: Ausschnitt aus dem Referenzprozess des User Interface Developer (links) und Teilprozess des User Interface Developer "Konfigurieren und Anpassen von COTS-Komponenten" (rechts).

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: User Interface Developer (Nutzerschnittstellenentwickler/in)

Der User Interface Developer² plant, konzipiert und implementiert die Schnittstellen für die Interaktion zwischen Softwaresystemen und deren menschlichen Bedienern. Hauptaufgabe dabei ist die Analyse und ggf. das Reengineering der Arbeitsprozesse, die der Benutzer durch das System unterstützen lassen möchte.

2.1 Tätigkeitsbeschreibung

Den Ausgangspunkt für die Arbeiten des User Interface Developer bildet die intensive Kommunikation mit den Fachexperten der Anwendungsdomäne sowie den Endnutzern des Systems bzw. generell die Entwicklung eines tiefen Verständnisses für die Belange des Problemfeldes. Allgemeine kulturelle, psychologische Prinzipien, ergonomische Richtlinien und Standards aber auch unternehmenspolitische bzw. -strategische Erwägungen geben einen entsprechenden Rahmen vor. Iterativ inkrementelle Vorgehensweisen zur Softwareentwicklung, unterstützt durch die Erstellung von Prototypen, ermöglichen die Berücksichtigung der Kundenwünsche dabei am besten.

Die eingesetzte Technologie orientiert sich grundsätzlich an den Anforderungen des Kunden, folgt in der Regel jedoch auch den Gesetzmäßigkeiten gewisser Industriestandards. So wird die Entwicklung zweidimensionaler graphischer Benutzeroberflächen, die dialogorientiert und mithilfe von metaphorischen Steuerelementen (Menüs, Checkboxes, Radiobuttons etc.) arbeiten und sich in die Arbeitsplatzmetaphern verschiedener Betriebssystemhersteller einbetten lassen, die zentrale Rolle spielen. Aber auch einfachste textbasierte Terminal-Anwendungen sowie weniger komplexe Bedienpanels so genannter „Embedded Systems“ benötigen unter Umständen den Sachverstand eines Experten für Nutzerschnittstellen. Um elaboriertere mediale Unterstützung der Benutzung (Audio, Video etc.) anbieten zu können, ist es sinnvoll auf die Fähigkeiten eines Multimedia Developer zurückzugreifen.

Der User Interface Developer muss in der Lage sein, seine Arbeit in die Architektur und den Entwicklungszyklus des Gesamtsystems zu integrieren. Dazu benötigt er Erfahrung mit gängigen Entwurfsmustern (etwa dem Modell-View-Controller-Pattern) bzw. Kommunikationsprotokollen zum Informationsaustausch mit tiefer liegenden, eventuell räumlich verteilten Architekturkomponenten sowie verbreiteten Vorgehensmodellen (z. B. dem V-Modell). Um die Konsistenz der mit anderen Schichten ausgetauschten, sich üblicherweise ändernden Datenformate zu sichern, bedarf es der ständigen Kommunikation unter den Entwicklerteams sowie eines grundsätzlichen Verständnisses für die technischen Probleme der einzelnen Subsysteme. So sollte etwa vermieden werden, Daten in einer Art zu organisieren bzw. abzurufen, mit der die Infrastruktur nicht oder nur mit großem Aufwand zurechtkommt. Der User Interface Developer sollte dazu die vom jeweiligen Unternehmen eingesetzte CASE-Tool-Kette in den Teilen beherrschen, die er benötigt. Außerdem muss er in der Lage sein, die geforderte Dokumentation und Spezifikationen, abgefasst mittels gängiger Notationen (UML, ER etc.), zu verfassen bzw. zu verstehen. Die User-Interface-Schicht trägt nicht zuletzt die Verantwortung für die Qualität der Daten, die in das System gelangen, und damit die Robustheit des Systems.

² Das 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.

2.2 Profiltypische Arbeitsprozesse

Die im Folgenden beschriebenen Teilprozesse dokumentieren den gesamten profiltypischen Arbeitsprozess des User Interface Developer. Die Beherrschung dieses Arbeitsprozesses in Verbindung mit den Kompetenzen in den jeweiligen Kompetenzfeldern und der Berufserfahrung bildet 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 beim Festlegen des Entwicklungsrahmens und der Entwicklungsumgebung, Abschätzen der Aufwände, Festlegen von Meilensteinen und Identifizieren von Implementierungsrisiken
3. Überprüfen von Anforderungsmodellen und Systemdesign-Dokumenten auf Korrektheit, Eindeutigkeit und Vollständigkeit sowie auf Realisierbarkeit der Systemanforderungen bezüglich Ergonomie der Nutzung und der angestrebten Akzeptanz beim Nutzer; Abstimmen von Änderungen und Erweiterungen mit IT-Systemanalytikern, IT-Systemplanern und weiteren Spezialisten aus dem Bereich Entwicklung
4. Einbeziehen und Abgleichen von Standards, Richtlinien sowie kultureller bzw. politischer Aspekte, um die Bedienbarkeit, Einheitlichkeit, Wartbarkeit und Beschreibbarkeit der Systeme und ihrer Handbücher festzulegen
5. Verfeinern von Systementwürfen durch Abbilden der spezifizierten Systemkomponenten beispielsweise auf Softwarearchitektur-Modelle und Oberflächendesign-Studien. Spezifizieren des dynamischen Verhaltens der Systemkomponenten in Form geeigneter Diagramme bzw. Storyboards
6. Erstellen von Migrationsmodellen und -szenarien zur (ggf. schrittweisen) Integration der Unterstützung des neuen Systems in bestehende Arbeitsabläufe; ggf. Ablösen bzw. Einbeziehen von vorhandenen Altsystemen
7. Ableiten von Testfällen und -szenarien aus den Spezifikationen und Bereitstellen von Testdaten und Prüfprozeduren für Unit- und Systemtests
8. Erstellen von Software-Feinentwürfen entsprechend der eingesetzten Entwicklungsmethodik (Abbildung auf entsprechende Modelle) in Abhängigkeit vom konkret eingesetzten Oberflächenframework
9. Abstimmen mit den Software Developern, um Schnittstellen für die Anbindung der Anwendungslogik festzulegen; Abgleichen von Interaktions- sowie Datenmodelle; Absprachen bzgl. der Verwendung der ggf. eingesetzten Middleware treffen
10. Realisieren der Funktionalität der Nutzerschnittstelle: Umsetzung der Screenentwürfe sowie der Nutzer-Interaktionslogik auf der Basis des ausgewählten Entwicklungsrahmens; Implementieren der Schnittstellenfunktionalität zur Anbindung der Anwendungsschicht; Durchführen der Unit-Tests, Festhalten der Testergebnisse
11. Unterstützen der Systemintegration und der Systemtests bzw. bei kleineren Projekten Durchführen der Systemintegration mit Unterstützung der am Projekt beteiligten Entwickler
12. Unterstützen des Deployment-Prozesses bzw. der Installation bei Kunden inklusive der Migration von Arbeitsprozessen und der Anbindung von Fremdsystemen; Begleiten der Nutzer
13. Mitarbeiten bei der Erstellung von Handbüchern, Installationsanleitungen und Trainingsmaterialien sowie Durchführung von Schulungen für Endnutzer

2.3 Profilprägende Kompetenzfelder

Die Beherrschung der profiltypischen Arbeitsprozesse setzt Kompetenzen unterschiedlicher Reichweite in den nachstehend aufgeführten beruflichen Kompetenzfeldern³ voraus. Den Kompetenzfeldern sind Wissen und Fähigkeiten sowie typische Methoden und Werkzeuge unterschiedlicher Breite und Tiefe zugeordnet.

Grundlegend zu beherrschende, gemeinsame Kompetenzfelder⁴:

- 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
- Software-Ergonomie

Routiniert zu beherrschende, profilspezifische Kompetenzfelder:

- Moduldesign, Designmuster
- ästhetisch bildnerisches Arbeiten
- Softwaremodellierungsmethoden, -regeln, -verfahren
- Prozessmodellierung
- Beherrschung der Werkzeuge und Frameworks zur Erstellung von Nutzeroberflächen

³ 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.

⁴ Jeder Spezialist muss in den in diesem Abschnitt genannten „weichen“ Kompetenzfeldern wie „Kommunikation, Präsentation“, „Konflikterkennung, -lösung“ usw. ein Niveau erreichen, dass 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 gegebenenfalls über den Rahmen bekannter Verfahren und Lösungen hinausgehen können.

2.4 Qualifikationserfordernisse

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

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 User Interface Developer angesiedelt ist
2. die Ebene der operativen Professionals
3. die Ebene der strategischen Professionals

Selbstverständlich kann sich der User Interface Developer sukzessive zu einem Professional weiterqualifizieren.

Verwandte Profile

Die Tätigkeitsprofile des IT-Weiterbildungssystems können Schnittstellen zu anderen Profilen aufweisen. Der User Interface Developer interagiert mit dem IT Configuration Coordinator, der die Entwicklungsumgebung für die technische Realisierung des Projekts bereitstellt, und dem IT Test Coordinator, der den User Interface Developer beim Testen der entstandenen Software unterstützt. Die Berührungspunkte der Arbeitsprozesse dieser unterschiedlichen Profile werden in der Prozessdarstellung ausdrücklich aufgezeigt.

Zu den dem User Interface Developer verwandten Profilen gehören die anderen Profile in der Gruppe der Software Developer, die einem ähnlichen Ablauf folgen. Insbesondere zu nennen sind dabei:

- der IT Systems Developer, der sich ebenfalls mit der Architektur eines Systems beschäftigt
- der Software Developer, dessen Prozesse und Tätigkeiten denen des User Interface Developer ähnlich sind, sowie
- der Multimedia Developer, der sich insbesondere mit Datenformaten und anderen Anforderungen an die multimediale, interaktive Gestaltung von Benutzeroberflächen beschäftigt

Aufstiegsqualifizierung

Das Tätigkeitsfeld des User Interface Developer ist eine ideale Grundlage für Aufstiegsqualifizierungen insbesondere zum IT Business Consultant mit den Schwerpunkten Erarbeiten und Umsetzen von (auch sehr spezifischen und hochkarätigen) IT-Systemlösungen und zum IT Systems Manager, der den gesamten Entwicklungsprozess von Systemen und Softwarelösungen verantwortet.

3 Referenzprozess

Der Referenzprozess des User Interface Developer beschreibt die Entwicklung aller Komponenten eines Softwaresystems, die dem Nutzer die angebotenen Funktionalitäten zugänglich machen, vom Systementwurf bis zum endgültigen Systemtest.

3.1 Nutzerschnittstellen-Entwicklung

Der Beitrag, den der User Interface Developer innerhalb des Wertschöpfungsprozesses der Software-Entwicklung leistet, lässt sich in folgende Phasen einteilen:

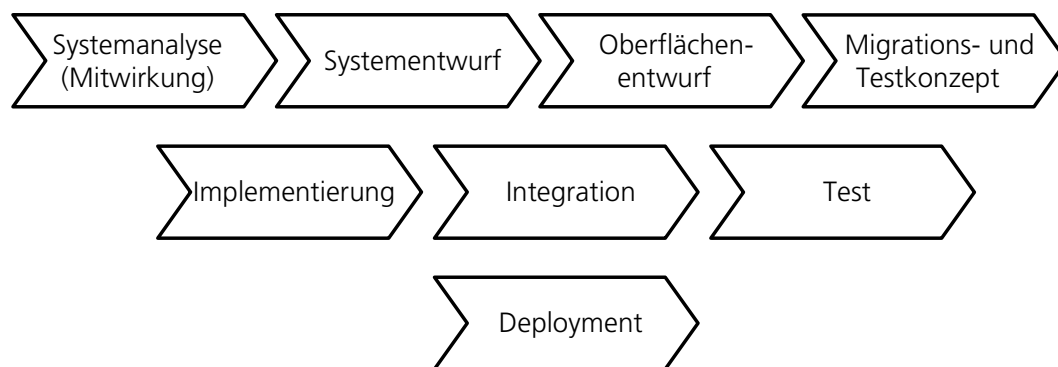


Abbildung 6: Die Phasen des Referenzprozesses User Interface Development.

Die dabei ablaufenden Prozesse werden im Folgenden ausführlich dargestellt:

Der Referenzprozess gibt den gesamten Entwicklungsprozess von Nutzerschnittstellenfunktionalität 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: Nutzerschnittstellen-Entwicklung

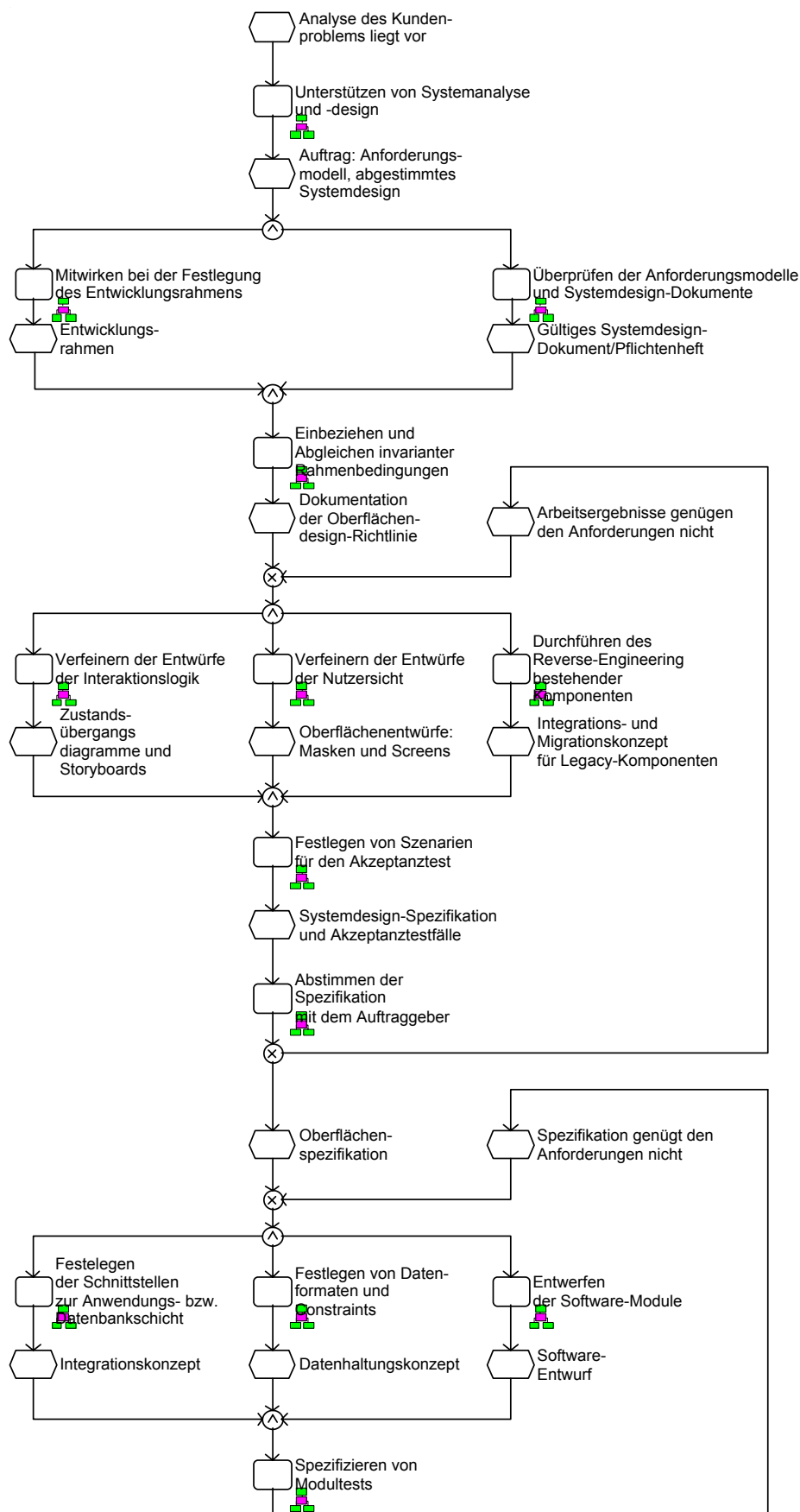


Abbildung 7: Referenzprozess des User Interface Developer, Teil 1.

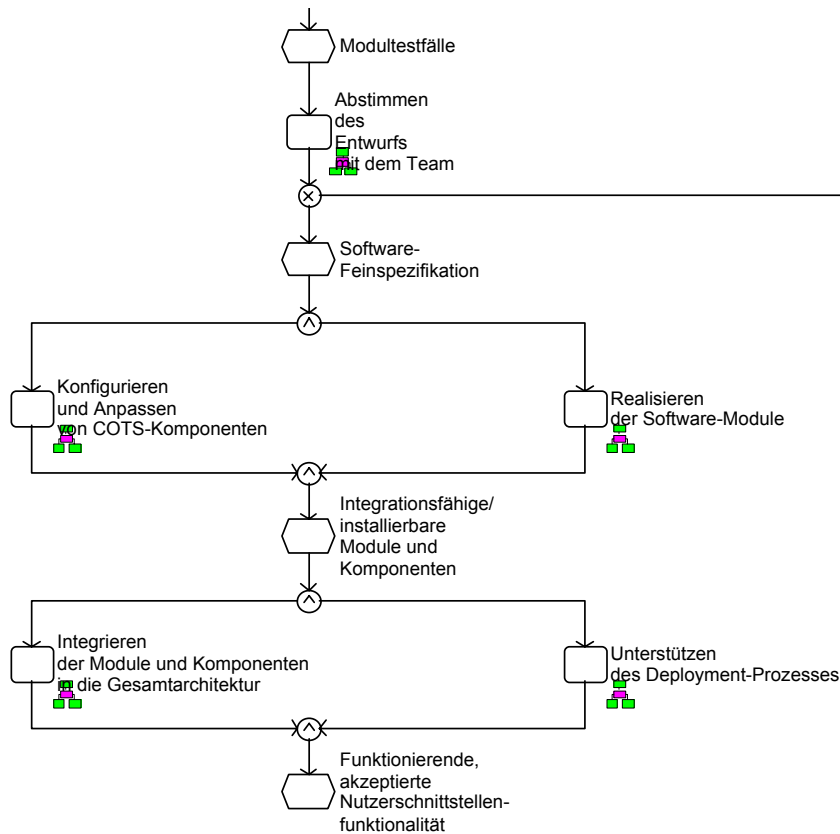


Abbildung 8: Referenzprozess des User Interface Developer, Teil 2.

Der Entwicklungsprozess für Nutzeroberflächen ist typisch für ein Developer-Profil: Nach der fachlichen und formalen Festlegung des Projektrahmens geht es mit der Festlegung von Standards, dem Migrationsmodell und der Analyse der Nutzergewohnheiten sowie der kulturellen und unternehmensstrategischen Charakteristika um die Spezifika der Anwendungsdomäne und deren partielle Abbildung in einem Anwendungssystem. Sind auch die Testfälle und der Software-Feinentwurf festgelegt sowie die Entwicklungsumgebung eingerichtet, erfolgt die Programmierung im engeren Sinne, selbstverständlich in Abstimmung mit dem Team. Nach der Systemintegration, der Installation und Migration sowie der Unterstützung der Systemeinführung endet das Projekt damit, dass der Kunde ein funktionierendes System hat.

Dieser Prozess läuft in kleinen wie in großen Projekten ab, in denen die Erstellung von Nutzer-Computer-Schnittstellen eine Rolle spielt. Allerdings wird nicht in jedem Projekt jeder Teilprozess den gleichen Umfang und die gleiche Komplexität haben. So werden in großen Projekten sicherlich mehrere User Interface Developer zusammenarbeiten, und es wird Migrations- und Integrationsspezialisten geben. Kleine Projekte hingegen kann der User Interface Developer allein durchführen, einschließlich der Systemanalyse der Etablierung einfacher Lösungen zur persistenten Datenhaltung und der Nutzerschulung.

3.1.2 Das Beispielprojekt: DX-Union

Ursprünglich unterstützte DX-Union als "In-House"-Lösung die Softwareverteilung und die zentrale Erstinstallation von Arbeitsplatzrechnern ausschließlich innerhalb der Materna GmbH. Aus dem steigenden Interesse auch vonseiten der Kunden resultierte der Bedarf nach einer kompletten Neuentwicklung der Software zu einem ausgewachsenen Systemmanagement-Werkzeug. Um ein wirklich marktaugliches Produkt entwickeln zu können, musste zunächst das Bewusstsein für die speziellen Belange des Systemmanagements geschaffen werden, sehr komplexe Zusammenhänge, eingebunden in Routine-Arbeitsprozesse, gegenüber einem – unter Umständen bzgl. des Systemmanagements unbedarften – Nutzer zu präsentieren.

Auf diese Weise entstand ein System, welches sich bewusst an allen gängigen Standards orientierte:

- an Industriestandards, die sich aus den Style Guides der Betriebssystemhersteller ableiten
- Quasistandards der Software-Ergonomie
- Richtlinien, die die Wahrnehmungspsychologie vorgibt

Das Vorgehen während der Entwicklung sowie die verwendeten Werkzeuge zu deren Unterstützung waren folglich ebenfalls charakterisiert durch die angestrebte Orientierung an der Nutzer-Interaktion. Viele Feedback-Zyklen auf der Basis von zum Teil horizontalen Prototypen und der Einsatz von marktgängigen Frameworks wie der Microsoft Foundation Classes (MFC) trugen entscheidend zum Erfolg des Projekts bei.

Aufgrund seiner Eigenschaften eignet sich dieses Projekt sehr gut, um einige der Aspekte der Nutzerschnittstellen-Entwicklung zu illustrieren – vor allem in Abgrenzung zur reinen Multimedia-Entwicklung und zur traditionellen Software-Entwicklung. Da selbstverständlich nicht alle Bereiche der sehr vielschichtigen Thematik eine Rolle im Rahmen des Projekts spielten, werden die Beispiele jeweils um allgemeinere Erfahrungen von Entwicklern ergänzt, die User-Interface-getriebene Software erstellen. Gerade bzgl. nichtgraphischer Interaktionsmöglichkeiten, also der multimodalen Schnittstellengestaltung, fehlen zum Teil Erfahrungswerte. Diese werden hier lediglich aus konzeptioneller Sicht erwähnt werden können.

3.1.3 Prozesskompass: Nutzerschnittstellen-Entwicklung

1. Unterstützen von Systemanalyse und -design
2. Mitwirken bei der Festlegung des Entwicklungsrahmens
3. Überprüfen der Anforderungsmodelle und Systemdesign-Dokumente
4. Einbeziehen und Abgleichen invarianter Rahmenbedingungen
5. Verfeinern der Entwürfe der Interaktionslogik
6. Verfeinern der Entwürfe aus Nutzersicht
7. Durchführen des Reverse Engineering bestehender Komponenten
8. Festlegen von Szenarien für den Akzeptanztest
9. Abstimmen der Spezifikation mit dem Nutzer
10. Festlegen der Schnittstellen zur Anwendungs- bzw. Datenbankschicht
11. Festlegen von Datenformaten und Constraints
12. Entwerfen der Softwaremodule
13. Spezifizieren von Modultests
14. Abstimmen des Entwurfs mit dem Team
15. Konfigurieren und Anpassen von COTS-Komponenten
16. Realisieren der Software
17. Integrieren der Module und Komponenten in die Gesamtarchitektur
18. Unterstützen des Deployment-Prozesses

3.1.4 Teilprozesse: Nutzerschnittstellen-Entwicklung

Die Teilprozesse geben den Entwicklungsprozess von Funktionalitäten zur System-Nutzer-Interaktion ausführlich und detailliert wieder. Sie entsprechen so der Essenz aus Erfahrun-

gen mehrerer realer Kundenprojekte, welche als Grundlage für den Referenz- und die Teilprozesse gedient haben. Sie liefern Beispiele zur Veranschaulichung.

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 von der konkreten Einbettung des Projekts abhängig. Ein User Interface Developer auf der Spezialistenebene sollte allerdings alle diese Prozesse beherrschen. Die Betonung liegt dennoch auf den Prozessen, die für die Entwicklung von User Interfaces zentral sind, angefangen vom Überprüfen der Anforderungsmodelle bis hin zum Installieren und zur Einführung der Nutzer.

3.1.4.1 Unterstützen von Systemanalyse und -design

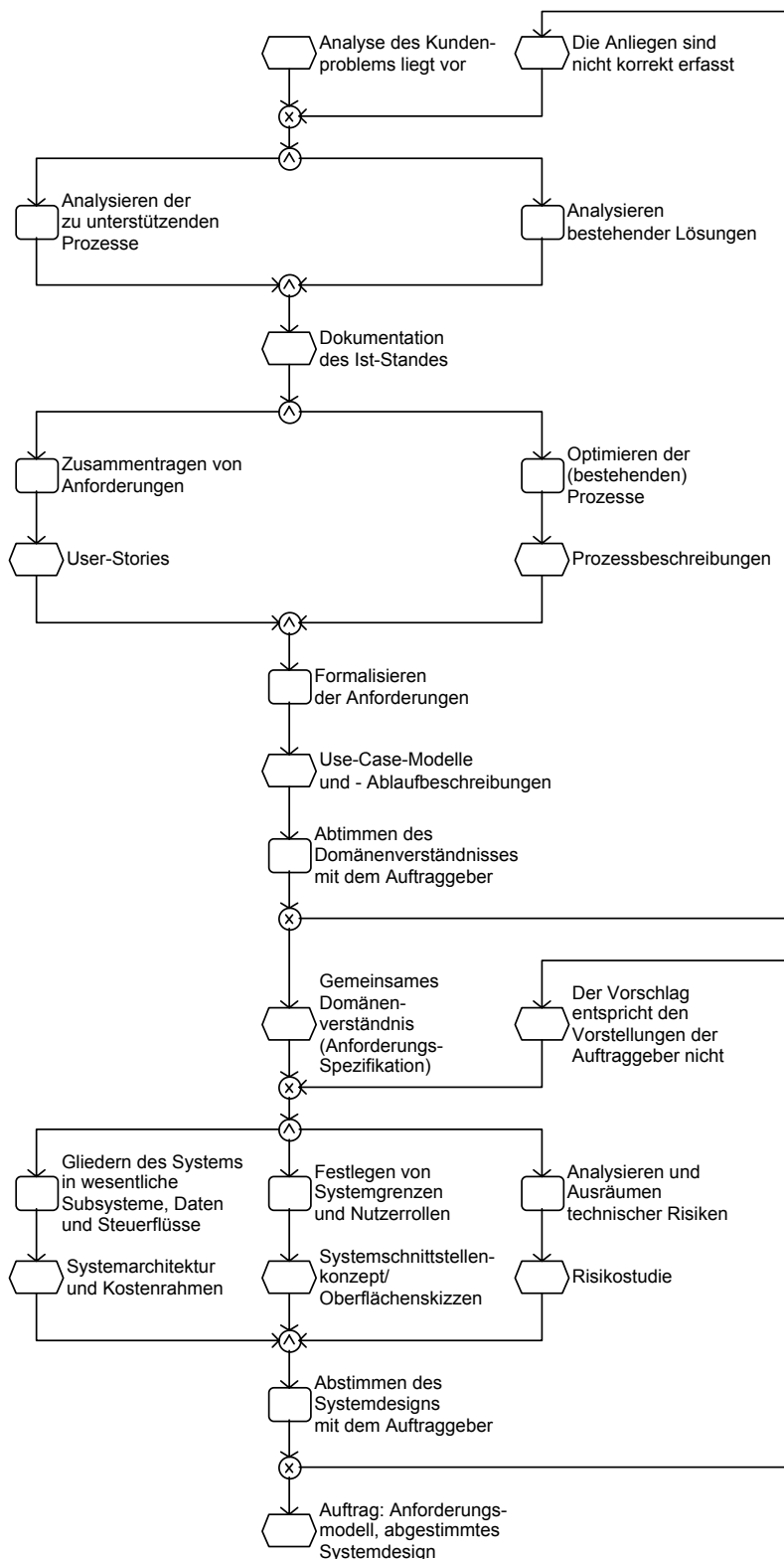


Abbildung 9: Unterstützen von Systemanalyse und -design.

Anmerkung: Während dieses Prozesses kann ein horizontaler Prototyp entstehen, bei dessen Entstehung der User Interface Developer (UID) kreativ mitwirkt.

3.1.4.1.1 Tätigkeiten: Unterstützen von Systemanalyse und -design

- Analysieren der zu unterstützenden Prozesse – Anmerkung: Analysieren der Nutzeranforderungen an das zu erstellende System aus Nutzersicht, ggf. in Zusammenarbeit mit Systemanalytiker und -designer sowie dem Vertriebsbeauftragten als Kundenbetreuer
- Analysieren bestehender Lösungen
- Zusammentragen von Anforderungen
- Optimieren der (bestehenden) Prozesse
- Formalisieren der Anforderungen
- Abstimmen des Domänenverständnisses mit dem Auftraggeber
- Gliedern des Systems in wesentliche Subsysteme, Daten und Steuerflüsse
- Festlegen von Systemgrenzen und Nutzerrollen – Anmerkung: Dokumentieren der Arbeitsablaufbeschreibungen und der Rahmen für die Präsentation der Daten (z. B. Skizzen zu Screens und Masken)
- Analysieren und Ausräumen technischer Risiken
- Abstimmen des Systemdesigns mit dem Auftraggeber – Anmerkung: Durchführen von Beratungsgesprächen beim Kunden, Vor- und Nachbereiten dieser Gespräche im Team bzw. fachliche Unterstützung dieser Gespräche

3.1.4.1.2 Kompetenzfelder: Unterstützen von Systemanalyse und -design

Fähigkeiten/Fertigkeiten

- Projektanalyse überprüfen können
- Nutzeranforderungen analysieren können
- Daten und Datensätze und deren Präsentationsformen analysieren können
- für Wünsche, Sorgen und Gewohnheiten von Nutzern sensibel sein
- vorhandenes System analysieren können
- zugrunde liegende Arbeitsabläufe aus vorhandenen Anwendungen (auch ohne Dokumentation) erschließen können
- Soll-Ist-Vergleiche erstellen können
- Entwürfe und Ideen unter Anwendung von Standards fixieren können
- Geschäftsprozesse, Funktionen, komplexe Strukturen und Unternehmensregeln modellieren können
- Prototypen aufgrund der Verhandlungsergebnisse und/oder der Projektanalyse erstellen können oder bei der Erstellung mitwirken können
- Prototypen testen können (sinnvolle Use Cases, insbesondere für Demonstrationszwecke, erstellen bzw. aussuchen können)
- kurzfristig Änderungen und Kundenwünsche in den Prototypen einpflegen können
- bei der Überprüfung der Kundenanforderungen, der Schätzung der Aufwände und Ressourcen mitwirken können
- sinnvoll im Team arbeiten können
- präsentieren können bzw. bei Präsentationen sinnvoll mitwirken können
- Kostenverhandlungen unterstützen können

Wissen

- Systemdesign, Vorgehensweise der Systementwicklung kennen
- Designstrategien
- Systemarchitektur
- Datentypen
- Datenformate
- Produktkenntnisse zu Grafikwerkzeugen und User-Interface-Entwicklungsframeworks
- Geschäftsprozesse
- Modellierung, Modellierungsregeln, Modellierungsverfahren
- Fähigkeit zur Umsetzung des ersten Anwendungsdesigns
- Dokumentationsstandards für die Anforderungsanalyse
- Software-Entwicklung und Software Engineering (gängige Vorgehensmodelle wie V-Modell oder Rational Unified Process)
- Grundkenntnisse Projektmanagement

Werkzeuge/Methoden

- Case-Tool für das Design des Prototyps und zur Unterstützung der Analyseprozesse
- Graphiksoftware zur Erstellung der Oberflächenentwürfe
- Werkzeuge zur Unterstützung von Kreativtechniken
- ggf. vorhandene Software zur Erstellung der Analyse vorhandener Datenstrukturen, Abläufe und Nutzergewohnheiten
- Planungs- und Schätzsysteme für Softwareprojekte
- kaufmännische Software
- Kreativtechniken (z. B. Brainstorming und Mindmapping)

3.1.4.1.3 Beispiel: Unterstützen von Systemanalyse und -design

Die Materna GmbH benennt in der Regel Produktmanager, die sich mit den Belangen der Anwendungsdomäne auskennen. Große projektgetriebene Firmen bündeln solche Kräfte auch in so genannten Programmabteilungen. Diese sind dann primär mit der Erfassung der Kundenwünsche und der Beratung der Kunden bzgl. der Möglichkeiten befasst. Häufig ist die Verflechtung technischer Belange mit den inhaltlichen Erfordernissen an das zu erstellende System so groß, dass die jeweiligen Spezialisten ihren Anteil an der Anforderungsspezifikation selbst verfassen. So ist gewährleistet, dass möglichst direkt Wünsche mit technischem Hintergrund interpretiert werden und das Wissen um die Machbarkeit in die Diskussion einfließt.

3.1.4.2 Mitwirken bei der Festlegung des Entwicklungsrahmens

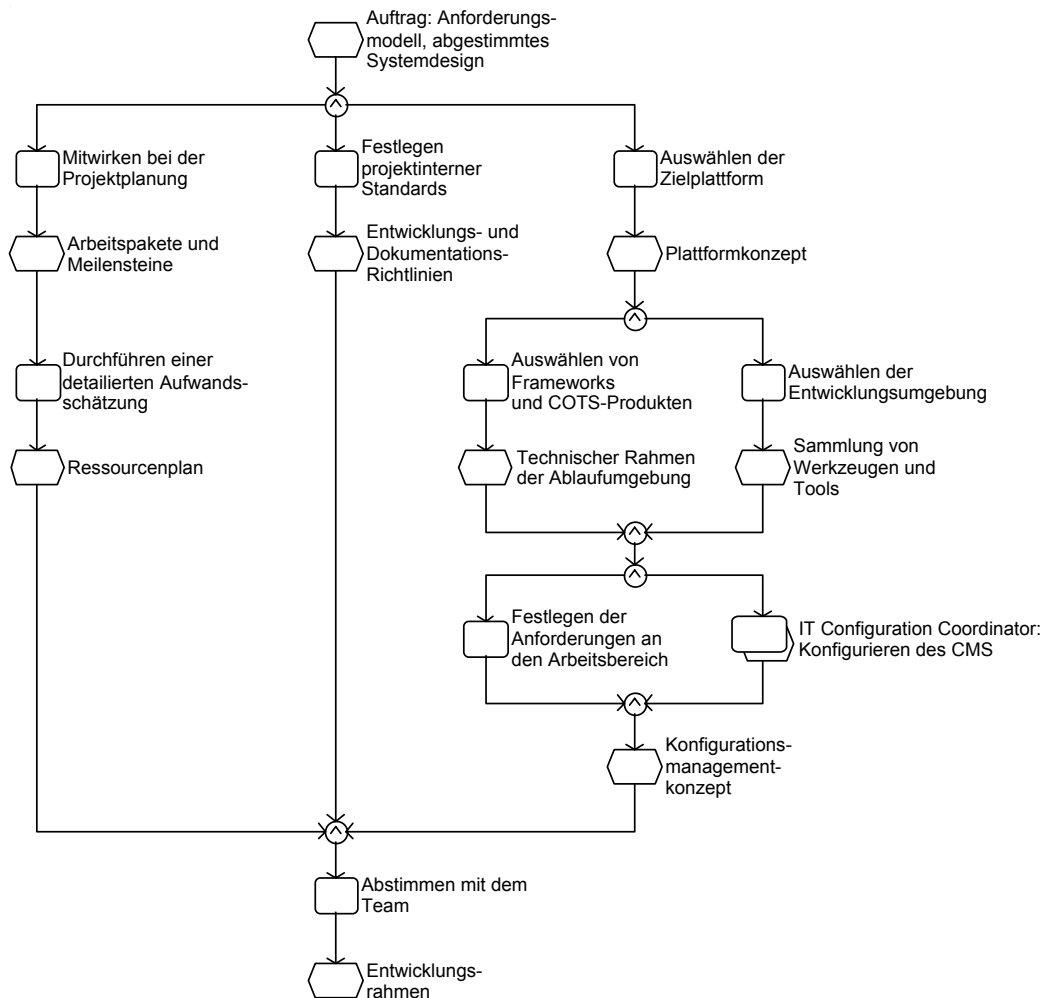


Abbildung 10: Mitwirken bei der Festlegung des Entwicklungsrahmens.

3.1.4.2.1 Tätigkeiten: Mitwirken bei der Festlegung des Entwicklungsrahmens

- Mitwirken bei der Projektplanung
- Festlegen projektinterner Standards
- Auswählen der Zielplattform – Anmerkung: Ableiten der adäquaten Zielplattform(en) bzw. Zielsystemumgebung aus den Dokumenten der Anforderungsspezifikation und unter Berücksichtigung unternehmenspolitischer Maßgaben
- Durchführen einer detaillierten Aufwandsschätzung
- Auswählen von Frameworks und COTS-Produkten – Anmerkung: Recherche nach adäquaten bereits vorhandenen, marktverfügbaren Common-off-the-shelf-Produkten ("von der Stange") wie Komponenten, Frameworks oder vollständigen Teilsystemen
- Auswählen der Entwicklungsumgebung – Anmerkung: Wahl einer Entwicklungsumgebung (Programmiersprache, Editoren, Build-Werkzeuge, CASE-Tools im weitesten Sinne)
- Festlegen der Anforderungen an den Arbeitsbereich
- Abstimmen mit dem Team

3.1.4.2.2 Kompetenzfelder: Mitwirken bei der Festlegung des Entwicklungsrahmens

Fähigkeiten/Fertigkeiten

- mit einer oder mehreren Entwicklungsplattformen (einschließlich integrierter Entwicklungsumgebung: CASE-Tools, Build-Unterstützung und Quelltextverwaltung) umgehen können
- für verschiedene Zielplattformen entwickeln können
- kollegial im Team arbeiten können (das umfasst administrative/organisatorische und strategische Aspekte eines Software-Entwicklungsprojekts)

Wissen

- unternehmenspolitische Rahmenbedingungen, die den Software-Einsatz (für die Entwicklung sowie für die einzubindende COTS-Software) beeinflussen
- Grundverständnis für (formale) Verfahren zur Unterstützung von Entscheidungen
- Technical Due Diligence von Fremdsoftware und angebotenen Software-Entwicklungsleistungen
- Überblick über den Markt der gängigsten GUI-Frameworks und Toolkits

Werkzeuge/Methoden

- Methoden des objektorientierten Designs
- Entwurfsmuster
- entsprechende Programmiersprache sowie die passende integrierte Entwicklungsumgebung
- GUI-Builder zum graphischen Arrangieren von Komponenten
- Methoden und Werkzeuge für Projektplanung (z. B. Gantt-Diagramme oder andere Balkendiagramme)

3.1.4.2.3 Beispiel: Mitwirken bei der Festlegung des Entwicklungsrahmens

Die Philosophie der Materna GmbH liegt im Produktgeschäft begründet und strebt nach Unabhängigkeit von Lizenzkosten für eingebundene Fremdprodukte. Es wird in der Regel versucht, möglichst viel eigene Entwicklungsleistung mit dem Produkt zu verkaufen. Selbstverständlich werden dennoch so viele Systemteile wie möglich wieder verwendet sowie kostenfreie Komponenten eingebunden. So kommen etwa bei allen in Java geschriebenen Anwendungen auch Swing-Klassen zum Einsatz.

Die Anforderung, dass DX-Union als Snap In für die Microsoft-Management-Konsole arbeiten sollte, ergab die Notwendigkeit, diese auf der Basis der Microsoft Foundation Classes (dem kommerziellen windowsbasierten Pendant zu Swing) in C++ und damit mithilfe der Entwicklungsumgebung von Visual C++ zu implementieren. Diese Entscheidung ist aber auch in Teilen historisch motiviert dadurch, dass die am Projekt beteiligten Entwickler erfahrene C++-Programmierer sind.

Diese beiden Beispiele machen deutlich, dass die Entscheidungen über die Etablierung des richtigen Projektumfelds nicht ausschließlich von den Zielen des Projekts abhängig zu machen sind. Die Rahmenbedingungen sollten aber dennoch bewusst etabliert und auch fixiert werden. Ferner führt mangelndes Mitspracherecht der Entwickler häufig zu ausschließlich politisch motivierten Festlegungen, die dann in der Regel die Machbarkeit gefährden.

3.1.4.3 Überprüfen der Anforderungsmodelle und Systemdesign-Dokumente

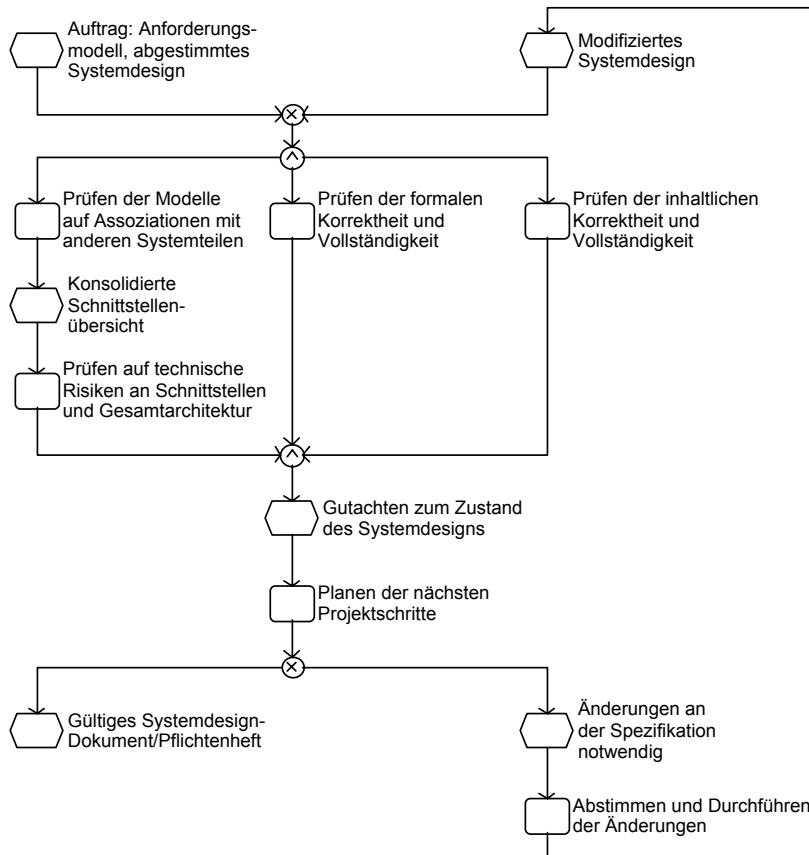


Abbildung 11: Überprüfen der Anforderungsmodelle und Systemdesign-Dokumente.

Anmerkung: In diesem Teilprozess werden diejenigen Teile der Anforderungsdokumente (Anwendungsfälle) identifiziert, die für die Arbeit des Nutzerinterfaces und seine Schnittstellen zu anderen Systemteilen relevant sind. Inhaltliche und formale Fehler, Widersprüche oder Ungenauigkeiten werden identifiziert und korrigiert.

3.1.4.3.1 Tätigkeiten: Überprüfen der Anforderungsmodelle und Systemdesign-Dokumente

- Prüfen der Modelle auf Assoziationen mit anderen Systemteilen
- Prüfen der formalen Korrektheit und Vollständigkeit
- Prüfen der inhaltlichen Korrektheit und Vollständigkeit
- Prüfen auf technische Risiken an Schnittstellen und Gesamtarchitektur
- Planen der nächsten Projektschritte – Anmerkung: Vertreten von Änderungsvorschlägen gegenüber dem Kunden bzw. dem Verantwortlichen der Analysephase (IT Systems Developer/IT Systems Analyst)
- Abstimmen und Durchführen der Änderungen

3.1.4.3.2 Kompetenzfelder: Überprüfen der Anforderungsmodelle und Systemdesign-Dokumente

Fähigkeiten/Fertigkeiten

- Anforderungsmodelle und Spezifikationen erschließen können
- Anforderungsmodelle (um)formulieren können (z. B. durch das Erstellen von Use-Case-Diagrammen)
- Verbindungen zwischen Anwendungsdomäne und technischen Umsetzungsmöglichkeiten ad hoc herstellen können
- Chancen neuer Technologien im Kontext einer Anwendungsdomäne vermitteln können

Wissen

- Struktur von Anforderungsspezifikationen (z. B. entsprechend V-Modell)

Werkzeuge/Methoden

- CASE-Tools zur Unterstützung des Anforderungs-Aufnahme- und Verfolgungsprozesses (z. B. Telelogic DOORS)
- CASE-Tools zur Unterstützung der Anforderungsmodellierung (z. B. UML-Werkzeuge wie Rational Rose)

3.1.4.3.3 Beispiel: Überprüfen der Anforderungsmodelle und Systemdesign-Dokumente

Häufig werden Anforderungsanalysen von Systemingenieuren bzw. Produktmanagern erstellt, die ihren Blick eher für das Gesamtsystem schärfen und deren Verständnisschwerpunkt eher in der Anwendungsdomäne liegt. Bezüglich neuester technischer Entwicklungen und damit neuer Möglichkeiten werden daher immer Zuarbeiten vonseiten der Spezialisten benötigt. Dabei wird der User Interface Developer häufig auch Überzeugungsarbeit leisten müssen, um Paradigmenwechsel wie etwa am Übergang von textbasierten zu multimedialen Schnittstellen plausibel zu machen.

3.1.4.4 Einbeziehen und Abgleichen invarianter Rahmenbedingungen

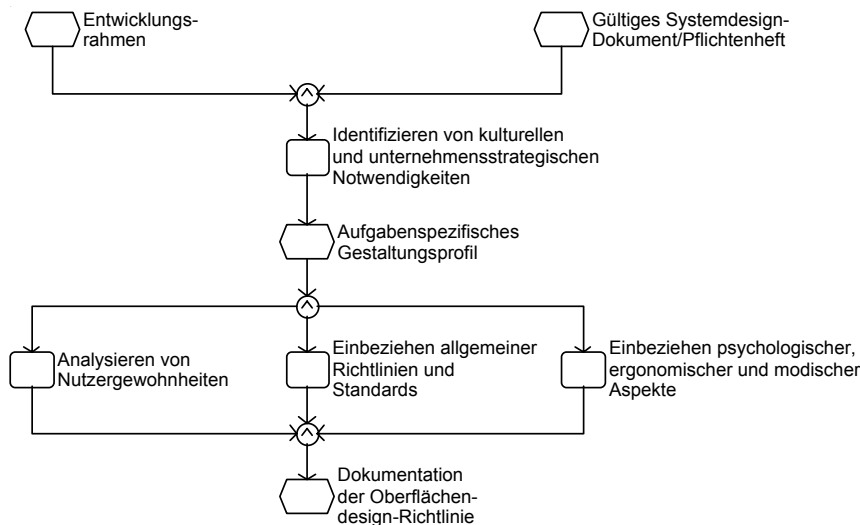


Abbildung 12: Einbeziehen und Abgleichen invarianter Rahmenbedingungen.

3.1.4.4.1 Tätigkeiten: Einbeziehen und Abgleichen invarianter Rahmenbedingungen

- Identifizieren von kulturellen und unternehmensstrategischen Notwendigkeiten
- Analysieren von Nutzergewohnheiten – Anmerkung: Analysieren von Nutzergewohnheiten aus gegebenen Arbeitsprozessen und ggf. vorhandenen unterstützenden Systemen
- Einbeziehen allgemeiner Richtlinien und Standards
- Einbeziehen psychologischer, ergonomischer und modischer Aspekte – Anmerkung: Berücksichtigen aller für die Anwendung relevanten allgemein gültigen Entwurfsmaßgaben, die bestimmt sind durch die Psychologie, Physiologie sowie durch ästhetische Ansprüche der Nutzer; Ableiten und Zusammenstellen von Entwurfsmaßgaben, deren Einhaltung für die Akzeptanz bzw. die Anwendbarkeit im gegebenen kulturellen bzw. unternehmenspolitischen Nutzerumfeld entscheidend ist (es sollte beispielsweise klar sein, wie viele Informationsblöcke ein Mensch im Allgemeinen auf einen Blick erfassen und behalten kann; oder auch, dass die Farbe Rot immer warnende Signalwirkung hat und bei häufigem, unbedachtem Gebrauch leicht zu Irritationen führen kann); Erarbeiten der in Anwendung zu bringenden Standards (Style Guides) auf Grundlage der psychologischen, ergonomischen und modischen Aspekte

3.1.4.4.2 Kompetenzfelder: Einbeziehen und Abgleichen invarianter Rahmenbedingungen

Fähigkeiten/Fertigkeiten

- andere Kulturen verstehen und deren Handlungsweisen und Traditionen einschätzen können
- Anwendungen ausgewogen, ansprechend und funktional gestalten können
- sich durch intensive Interviews in die Problemdomäne des Nutzers einarbeiten können
- sich in den Nutzer einfühlen können
- grundlegende Kenntnisse zu Unternehmens- bzw. Marketingstrategien (ermöglichen das Verstehen von in dieser Hinsicht übergeordneten Einflüssen auf die Gestaltung; etwa durch Corporate Designs)

Wissen

- Style Guides (zu Quasistandards wie z. B. Microsoft Windows oder Java)
- Zeitgeist
- grundlegendes Wissen zu Weltkulturen bzw. den großen Weltreligionen
- grundlegende Kenntnisse zu Unternehmens- bzw. Marketingstrategien (ermöglichen das Verstehen von in dieser Hinsicht übergeordneten Einflüssen auf die Gestaltung; etwa durch Corporate Designs)
- für die ergonomische Gestaltung von Software nötige wahrnehmungspsychologische Grundkenntnisse
- ästhetisch-bildnerisches Grundverständnis (z. B. Farbenlehre)

Werkzeuge/Methoden

- Interviewtechniken
- Moderationstechniken

3.1.4.4.3 Beispiel: Einbeziehen und Abgleichen invarianter Rahmenbedingungen

Die Entwickler des betrachteten Beispielprojekts (DX-Union) waren zuerst dem Style Guide von Microsoft-Windows-Applikationen verpflichtet. Das heißt, die grundlegende Anordnung von Menüs, die Gestaltung von Steuerelementen und das Interaktionsverhalten entsprechen denen aller Windows-Anwendungen.

Außerdem gibt es eine Art untergeordneten, fachbezogenen „DX-Union-Style-Guide“, der DX-Union zu einem unverwechselbaren Produkt der Materna GmbH macht, das Einarbeiten von Nutzern, die mit Produkten der Firma vertraut sind, erleichtert und nicht zuletzt den Wiedererkennungswert steigert. Dieser Style Guide ist allerdings nicht schriftlich fixiert, lediglich als Konsens in dem überschaubaren Team latent.

Ein ganz essenzielles Beispiel für die Gegenstände, die in diesem Teilprozess zu fixieren sind, ist die Lokalisierung der Software: Die Erfahrung der Beispielprojekte lehrt, dass es ganz entscheidend ist, ob ein System etwa nach Großbritannien oder in die USA geliefert wird, denn in beiden Fällen unterscheiden sich die akzeptierten Formatierungen der Ausgabe des Datums erheblich.

Bewusste interkulturelle Sensibilität im eigentlichen Sinne ist besonders gefragt, wenn der Endanwender aus einem anderen Kulturkreis kommt und unter Umständen auch noch sehr viel stärker seinen Traditionen verpflichtet ist. So sollte etwa beachtet werden, dass heilige oder reservierte Farben nicht „aus Versehen“ benutzt werden (Grün im muslimischen Raum, Gold in Japan etc.). Gleiches gilt für Symbolik und Ornamentik.

3.1.4.5 Verfeinern der Entwürfe der Interaktionslogik

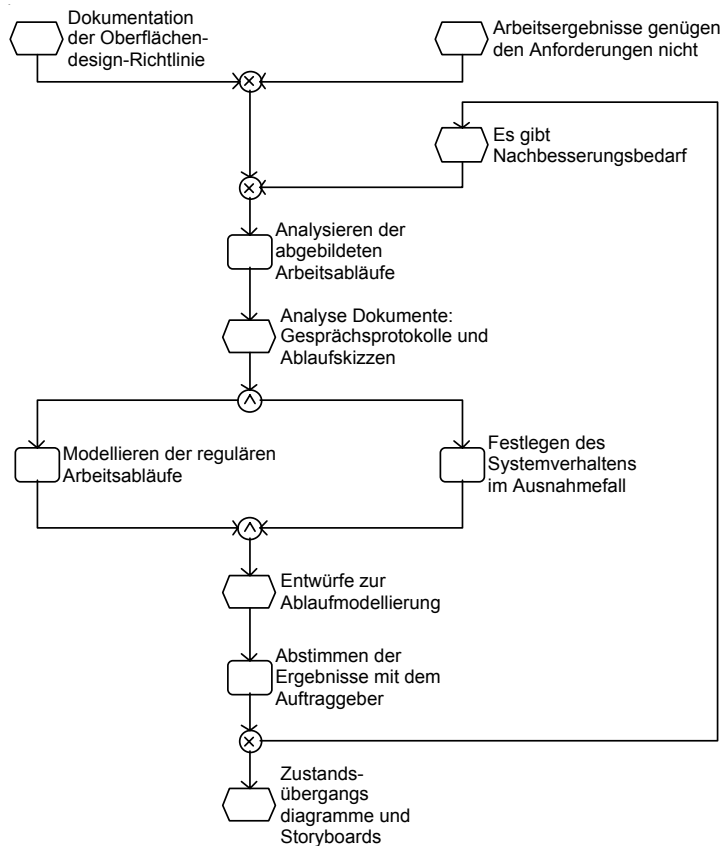


Abbildung 13: Verfeinern der Entwürfe der Interaktionslogik.

Anmerkung: Innerhalb dieses Teilprozesses werden die Anforderungen der Nutzer dahin gehend präzisiert, dass resultierende technische Anforderungen abgeleitet werden. Diese bilden die Basis für die Konzeption der Lösung. Der Schwerpunkt der Betrachtungen liegt dabei auf der Nutzerinteraktion – der Optimierung von Arbeitsabläufen in der Mensch-Maschine-Kommunikation – in klarer Abgrenzung zu den rein softwaretechnologischen Überlegungen. Diese Phase stellt in der Regel die letzte Phase dar, die schwerpunktmäßig fachlich orientiert ist und es damit dem Nutzer erlaubt, Einfluss auf das Produkt zu nehmen. Je intensiver davon profitiert werden kann, umso geringer ist das Risiko, in den folgenden rein technisch geprägten Abschnitten der Entwicklung an den Bedürfnissen des Nutzers vorbeizuarbeiten.

3.1.4.5.1 Tätigkeiten: Verfeinern der Entwürfe der Interaktionslogik

- Analysieren der abgebildeten Arbeitsabläufe – Anmerkung: das Analysieren der Arbeitsabläufe beinhaltet das Führen von Interviews mit potenziellen Nutzern, die Systematisierung der Lösungsansätze in Modellen und „Drehbüchern“ bzw. horizontalen Prototypen – als Basis für die Abstimmung mit dem Nutzer und das Durchführen von „Feldversuchen“ mit horizontalen Prototypen
- Modellieren der regulären Arbeitsabläufe – Anmerkung: organisatorische Schnittstellen mit dem abgebildeten Geschäftsprozess präzisieren
- Festlegen des Systemverhaltens im Ausnahmefall
- Abstimmen der Ergebnisse mit dem Auftraggeber – Anmerkung: Vorschlagen von Optimierungsmöglichkeiten bzgl. des unterstützten Prozesses (durch den Einsatz des Systems)

3.1.4.5.2 Kompetenzfelder: Verfeinern der Entwürfe der Interaktionslogik

Fähigkeiten/Fertigkeiten

- beliebige Domänen und deren Erfordernisse bzgl. des IT-Einsatzes verstehen können
- Wünsche von Nutzern wahrnehmen und respektieren können
- Ängsten von Nutzern begegnen können
- von realen Arbeitsabläufen auf maschinenunterstützte abstrahieren können
- Prozesse formalisieren und visualisieren können
- Prozesse optimieren können
- Prototypen implementieren können

Wissen

- Kognition und Kommunikation (Prozess-Sicht)
- Ergonomie von Mensch-Maschine-Kommunikationsprozessen

Werkzeuge/Methoden

- Prozess-/Zustandsübergangsmodelle
- Interviewtechniken

3.1.4.5.3 Beispiel: Verfeinern der Entwürfe der Interaktionslogik

In der Regel wird versucht, sehr etablierte Geschäftsprozesse mithilfe von Informationstechnologie effektiver zu gestalten – Zyklen zu verkürzen, zeitaufwändige Schritte, geprägt von Routinearbeit, einzusparen. Es stellt sich in der Praxis häufig als sehr schwierig heraus, von menschlichen Nutzern getriebene etablierte Prozesse zu verändern. Somit ist es wichtig, die Effektivität der veränderten Abläufe offensichtlich zu machen – sich über jeden Verdacht der Umständlichkeit zu erheben. Löst man etwa papiergetriebene Vorgangsbearbeitung durch eine elektronische Lösung ab, muss dem Nutzer ins Auge springen, dass ihm nun eine platz- und organisationsintensive Ablage mit schlechten Recherchemöglichkeiten erspart bleibt, indem er etwa klare Übersichten über Zustände und Lokalisierungen von Dokumenten erhält, genau zu dem Zeitpunkt im Prozess, wo diese Sicherheit nötig ist. Der Nutzer darf dabei (in Ablösung des Papiers) nicht mit einer neuen Flut von Meldungen und Dialogangeboten überfordert werden – sodass augenscheinlich ein noch komplexerer, noch mehr beschleunigter Ablauf auf ihn einwirken würde.

3.1.4.6 Verfeinern der Entwürfe der Nutzersicht

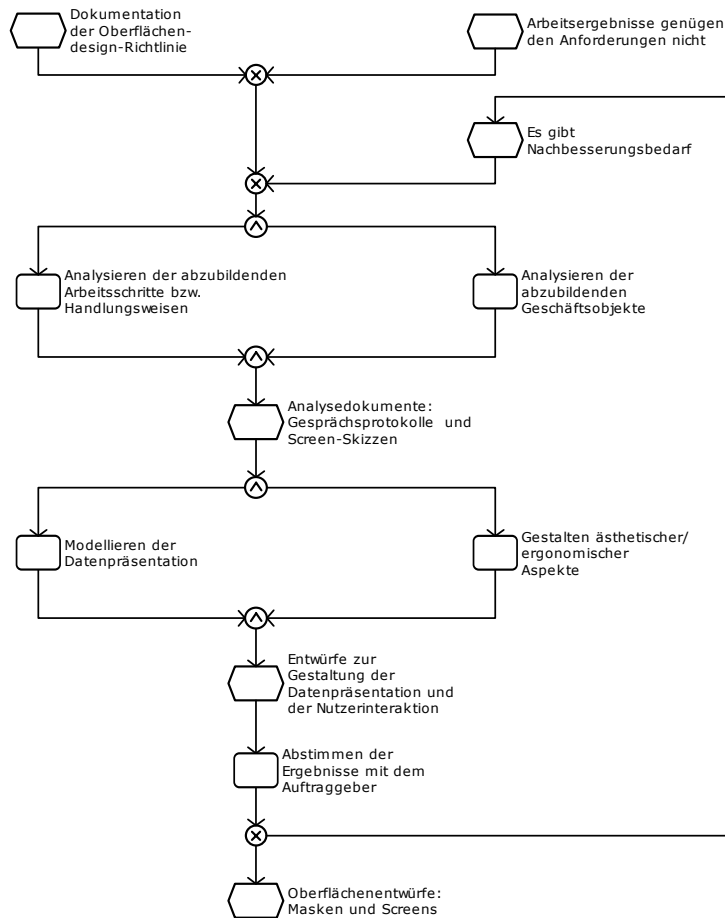


Abbildung 14: Verfeinern der Entwürfe der Nutzersicht.

Anmerkung: Neben der Konzeption der dynamischen Aspekte der Nutzerschnittstelle, die den Arbeitsablauf prägen, ist das statische Erscheinungsbild der Nutzerschnittstelle zu entwerfen. Diese Arbeiten werden im Zeitalter graphisch orientierter Schnittstellen geprägt sein vom Entwerfen visueller Metaphern für Gegenstände der abgebildeten Anwendungsdomäne. Es ist dennoch die Aufgabe des UID, das Interface unter Ausschöpfung aller sensorischen und motorischen Mittel der Interaktion mit Menschen zu optimieren.

3.1.4.6.1 Tätigkeiten: Verfeinern der Entwürfe der Nutzersicht

- Analysieren der abzubildenden Arbeitsschritte bzw. Handlungsweisen – Anmerkung: Führen von Interviews mit potenziellen Nutzern; Durchführen von „Feldversuchen“ mit horizontalen Prototypen; Beschreiben von Geräten zur Kommunikation mithilfe anderer sensorischer bzw. motorischer Modi (z. B. Blindendisplay, Audiosynthesizer, Datenhandschuhe etc.)
- Analysieren der abzubildenden Geschäftsobjekte
- Modellieren der Datenpräsentation
- Gestalten ästhetischer/ergonomischer Aspekte – Anmerkung: Strukturieren der User Interfaces, d. h. Anordnen und Gruppieren von Bedienelementen unter wahrnehmungspsychologischen Gesichtspunkten; Skizzieren graphischer Entwürfe (ggf. freihändig)
- Abstimmen der Ergebnisse mit dem Auftraggeber

3.1.4.6.2 Kompetenzfelder: Verfeinern der Entwürfe der Nutzersicht

Fähigkeiten/Fertigkeiten

- beliebige Domänen und deren Erfordernisse bzgl. des IT-Einsatzes verstehen können
- Wünsche von Nutzern wahrnehmen und respektieren können
- Ängsten von Nutzern begegnen können
- von realen Objekten auf adäquate Maschinenrepräsentationen (etwa Metaphern) abstrahieren können
- Entitäten (Objekte der Anwendungsdomäne) formalisieren können
- Entwürfe präsentieren können
- gestalterisch arbeiten können

Wissen

- Kognition und Kommunikation (Struktursicht)
- Ergonomie der Mensch-Maschine-Kommunikation
- Wahrnehmungspsychologie
- Schnittstellendesign (z. B. im graphischen Sinne)

Werkzeuge/Methoden

- Handskizzieren
- Grafikwerkzeuge (z. B. Adobe Photoshop)
- integrierte Entwicklungsumgebungen mit Werkzeugen zur visuellen Erstellung von User Interfaces
- ggf. Modellbau – zur Illustration alternativer (multimodaler) Interaktionsperipherie
- Interviewtechniken

3.1.4.6.3 Beispiel: Verfeinern der Entwürfe der Nutzersicht

Dieser Teilprozess ist in der Regel die größte Herausforderung für die Kreativität des Entwicklers, da er die Möglichkeiten rasanter technischer Entwicklung mit den Bedürfnissen der Nutzer vereinbaren muss. Er muss in der Lage sein, die Fähigkeiten und die Wahrnehmung der Nutzer einzuschätzen, um dann Mittel zu finden, Gegenstände der Problemwelt zu virtualisieren. Ein extremes Beispiel ist die Schreibtischmetapher, die hauptsächlich über die Mouse zugänglich ist und heute jedem Windows-System zugrunde liegt. Dieses Prinzip ist zum Teil das Ergebnis eines längeren evolutionären Prozesses, es erforderte aber auch die Arbeit kreativer User Interface Developer, konkret in den Labors von Xerox Parc.

3.1.4.7 Reverse Engineering bestehender Komponenten

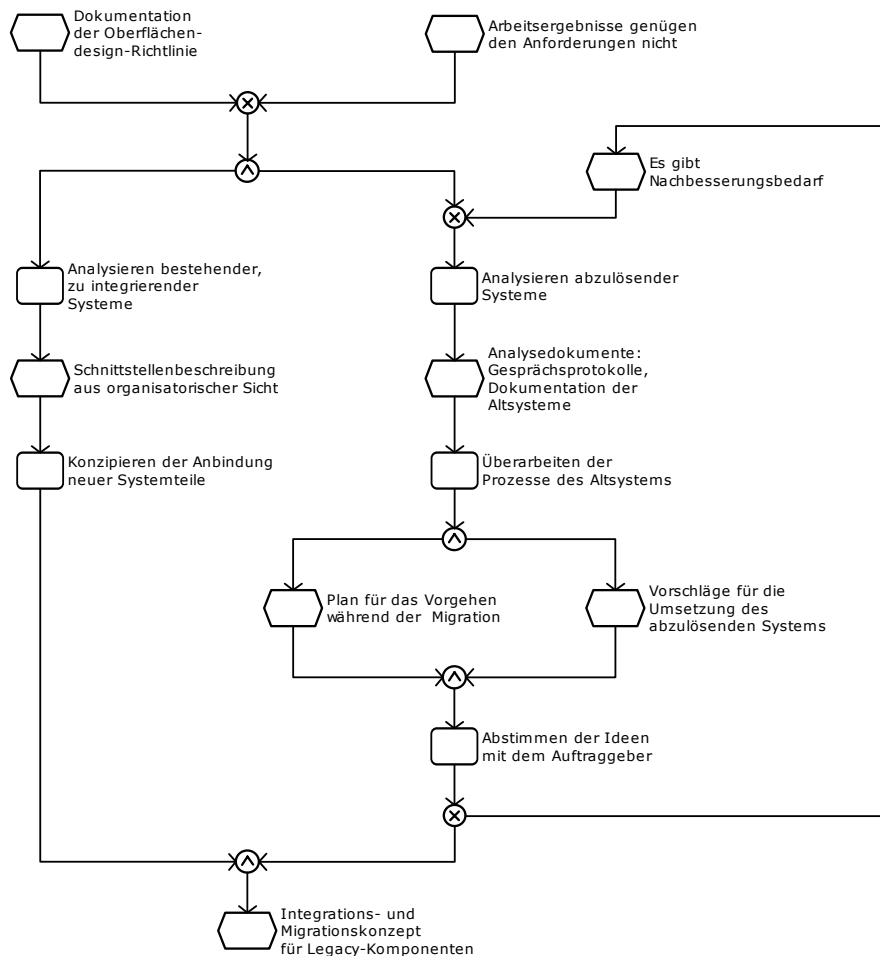


Abbildung 15: Reverse Engineering bestehender Komponenten.

Anmerkung: Komplexe Anwendungssysteme, gerade auch die durch Nutzerinteraktion getriebenen, werden häufig nicht komplett neu entwickelt. In vielen Fällen werden bestehende Systeme (auch Legacy-Systeme genannt) erweitert bzw. in die Neuentwicklung integriert.

3.1.4.7.1 Tätigkeiten: Reverse Engineering bestehender Komponenten

- Analysieren bestehender zu integrierender Systeme – Anmerkung: dieser Teilschritt beinhaltet das Analysieren der Schnittstellen zu integrierender bzw. zu erweiternder Systeme, deren Quelltext nicht verfügbar ist; das umfasst in der Regel die Recherche nach Informationen und das Studium der verfügbaren Dokumentation; Analyse der bestehenden Nutzerschnittstellen (unterstützte Arbeitsabläufe, verwendete Präsentation) abzulösender System(teile) – hieraus lassen sich Schlüsse ziehen in Bezug auf Nutzergewohnheiten, die Entwurfsentscheidungen beeinflussen müssen (in Rücksprache mit dem Nutzer des Systems)
- Analysieren abzulösender Systeme
- Konzipieren der Anbindung neuer Systemteile – Anmerkung: Erarbeiten technischer Konzepte für die Anbindung von Legacy-Systemen, etwa auf der Basis von Middleware; Planen des Vorgehens während der Migration (z. B. Konzeption und Umsetzung technischer Hilfsmittel etwa zur Übertragung von Legacy-Daten, organisatorische Einbettung von Übergangsphasen, politische und psychologische „Lobby“-Arbeit)
- Überarbeiten der Prozesse des Altsystems
- Abstimmen der Ideen mit dem Auftraggeber

3.1.4.7.2 *Kompetenzfelder: Reverse Engineering bestehender Komponenten*

Fähigkeiten/Fertigkeiten

- fremde Software und deren Quelltexte analysieren können
- Nutzerverhalten analysieren können
- Nutzergewohnheiten aufnehmen und respektieren können
- Analyseergebnisse und Konzeption systematisieren und verwalten können
- (unpopuläre) Entscheidungen kommunizieren und durchsetzen können
- technisch bzw. organisatorisch flexibel und situationsangepasst reagieren (improvisieren) können

Wissen

- Interoperabilität in heterogenen Systemumgebungen
- Legacy-Architekturen und -Schnittstellentechniken (z. B. Host-Systeme und Terminal-Emulatoren)
- Prinzipien von Legacy-Programmiertechniken (z. B. Maschinensprache oder prozedurale/jobbasierte Datenverarbeitung)
- Vorgehensweisen für die Übertragung von Daten und Anwendung auf neue Systeme bzw. Plattformen (z. B. inkrementell, „Big Bang“ etc.)

Werkzeuge/Methoden

- Middleware (z. B. CORBA, RMI, SOAP)
- Gateways, Mediatoren
- CASE-Tools zur Unterstützung des Reverse-Engineering-Prozesses (z. B. Extraktion von UML-Modellen aus bestehendem Code)

3.1.4.7.3 *Beispiel: Reverse Engineering bestehender Komponenten*

Ein typischer Grund für eine nötige Migration ist der Wechsel der Plattform. Dem Trend folgend stellt sich beispielsweise die Materna GmbH der Herausforderung, Linux als Betriebssystemplattform zu unterstützen, d. h. ihre Systemmanagementtools auch dafür bereitzustellen. Konkret die Routine zum einheitlichen initialen Installieren und Konfigurieren eines PC muss dabei komplett neu erstellt werden. Im Zuge dessen wurde die vorhandene Dokumentation ausgewertet; außerdem wurden in regelmäßigen Sitzungen die Entwickler des Altsystems befragt.

Eine wichtige Entscheidung, die im Zuge der Ablösung einer Betriebssystem- bzw. Hardwareplattform oder grundsätzlich einer Technologie immer getroffen werden muss, ist die zwischen der Portierung des Programmcodes oder der gesamten Anwendung. Die Portierung des Codes setzt voraus, dass zumindest die gleiche Programmiersprache eingesetzt werden kann. So ist es idealerweise möglich, unter Einbindung plattformabhängig passender Bibliotheken den „alten“ Programmcodes für die neue Plattform zu übersetzen und damit ablauffähig zu machen. Oft will man jedoch von den neuen Paradigmen, die eine weiterentwickelte Plattform mit sich bringt, profitieren – etwa am Übergang von einem Host-System zu einer graphisch gesteuerten Lösung. Das bringt es in der Regel mit sich, dass man die Anwendung komplett neu erstellen muss. Dabei werden die grundlegenden Funktionen übertragen und meist auch überarbeitet.

Solche tief greifenden Einschnitte, die sich direkt auf die Nutzerinteraktion auswirken, führen nicht selten zu Akzeptanzschwierigkeiten, die eine Mischung aus Einfühlungsvermögen und Überzeugungskraft erfordern. Der Übergang vom ISA-Dialog-Manager (altes, bekanntes „Look and Feel“) zur Realisierung von DX-Union innerhalb der Microsoft-Management-Konsole war begleitet von „Protesten“ seitens der Nutzer. Erforderliche umständliche Screen-Wechsel und die Nutzung der Mouse machten das Arbeiten vermeintlich umständlicher. Sowohl Korrekturen am Design als auch die sukzessive Eingewöhnung der Nutzer haben letztlich die Probleme gelöst.

3.1.4.8 Festlegen von Szenarien für den Akzeptanztest

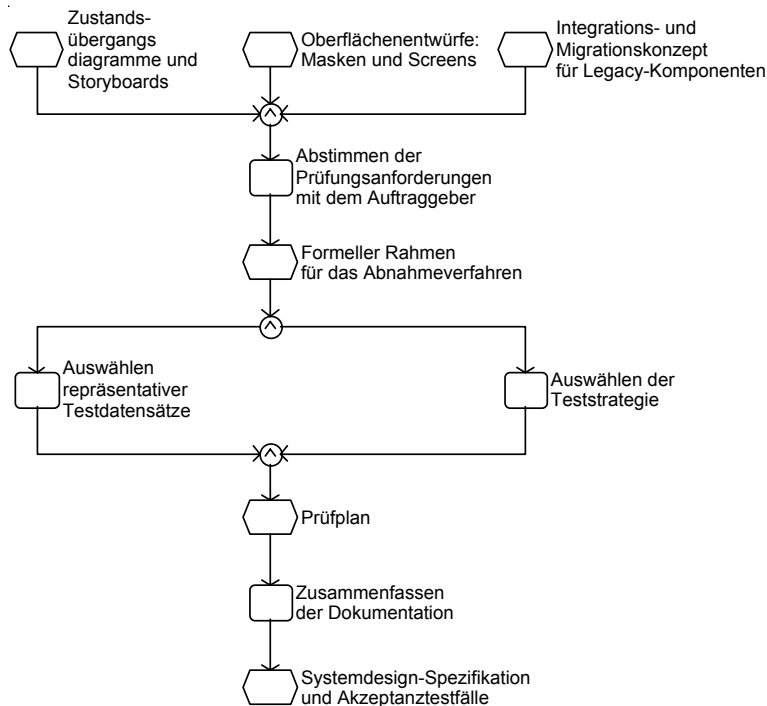


Abbildung 16: Festlegen von Szenarien für den Akzeptanztest.

Anmerkung: Die Erarbeitung der Prüfprozeduren wird in der Regel durch den Quality Management Coordinator geleitet und durch die Festlegungen des Vertragswerks mit dem Kunden geregelt. Die dort abgesteckten Rahmen für die Prüfung werden hier entsprechend den Designergebnissen konkretisiert.

3.1.4.8.1 Tätigkeiten: Festlegen von Szenarien für den Akzeptanztest

- Abstimmen der Prüfungsanforderungen mit dem Auftraggeber – Anmerkung: Erarbeiten von Vorschlägen für Prüfabläufe zur Demonstration bzw. zum Nachweis der Funktionsfähigkeit, Bedienbarkeit sowie der Leistungsfähigkeit des Anwendungssystems in den Teilen, für die der User Interface Developer verantwortlich ist; dies geschieht idealerweise in Zusammenarbeit mit dem Auftraggeber
- Auswählen repräsentativer Testdatensätze – Anmerkung: Auswählen geeigneter Datensätze, die die Prüfabläufe unterstützen; diese sollten möglichst repräsentativ sein, um das gesamte Spektrum der Leistungen des Systems prüfen zu können; ferner sollten es Daten sein, die ein möglichst authentisches Arbeiten mit dem System ermöglichen – idealerweise sind es Daten, die aus operativen Prozessen des Kunden zu extrahieren sind
- Auswählen der Teststrategie – Anmerkung: das Festlegen der Teststrategie umfasst die Entscheidung für die angewendeten Prüfverfahren, in Abhängigkeit vom Grad der Kritikalität bzw. der Qualitätsschwerpunkte, die die Anwendung setzt, wählt man etwa Kombinationen aus automatisierten Tests, Stresstests, Pfadtests etc.; im Bereich des User Interfaces wird auch der „Feldversuch“, bei dem der Endnutzer zu so genannten Beta-Tests herangezogen wird, eine wichtige Rolle spielen
- Zusammenfassen der Dokumentation

3.1.4.8.2 Kompetenzfelder: Festlegen von Szenarien für den Akzeptanztest

Fähigkeiten/Fertigkeiten

- Kritik am entwickelten Produkt konstruktiv in eine Verbesserung umsetzen können
- eigene Leistung kritisch einschätzen können
- potenzielle Risikoquellen einschätzen können
- auf die Bedürfnisse des Kunden eingehen können
- eigene Positionen vertreten können
- systematisch dokumentieren bzw. protokollieren können

Wissen

- Kenntnis der vertraglich vereinbarten und i. d. R. unternehmensabhängigen Geschäftsprozesse um Abnahmeverfahren
- Kenntnis üblicher Prüfmethoden
- Kenntnis der gängigen Dokumentations- und Protokolliertechniken und der üblicherweise dafür verwendeten Vorlagen

Werkzeuge/Methoden

- Systemtestverfahren (Initiieren repräsentativer Beta-Tests, Stresstests, formale Test- bzw. Beweisverfahren etc.)
- Werkzeuge zur Durchführung automatisierter Tests (z. B. JUnit)

3.1.4.8.3 Beispiel: Festlegen von Szenarien für den Akzeptanztest

Die Beispielprojekte der Materna GmbH machen deutlich, dass der Schwerpunkt der Prüfungsarbeit an User-Interface-getriebenen Systemen beim Feldversuch liegt. Die integrierten Systeme werden nach standardmäßigen internen Trockenläufen (Durchspielen der grundlegenden Funktionen entlang den Anforderungen des Lastenhefts) an den Endnutzer ausgeliefert unter der Maßgabe, alle gefundenen Fehler zu protokollieren und bezüglich der Schwere bzw. der Dringlichkeit zu priorisieren.

Im Lastenheft explizit ausgewiesene Qualitätsmerkmale wie maximale Antwortzeiten sowie zu bewältigende Mengengerüste werden allerdings in systematischen Prüfungen wie gezielten Lastprüfungen vor der Auslieferung durchgeführt. Die Durchführung wird federführend durch einen Mitarbeiter einer dezidierten Abteilung für Qualitätssicherung durchgeführt – der Entwickler hinterlegt aber Hinweise auf Risikopunkte und entsprechende Methoden zur Prüfung und damit zur Ausräumung von Unsicherheiten.

Formale Beweise der Korrektheit von Systemteilen spielen nur bei wirklich sicherheitskritischen Systemen, die potenzielle Gefahrenquellen für Leib und Leben von Menschen bergen – etwa in der Luft- und Raufahrtindustrie –, eine wichtige Rolle. Sie stellen besonders in nutzerinteraktiver Software eine große Herausforderung dar, da größtmögliche Flexibilität auch einen sehr großen Raum an zu prüfenden Systemzuständen nach sich zieht. Das entsprechende Spezialwissen ist also mit Sicherheit beim User Interface Developer anzusiedeln.

3.1.4.9 Abstimmen der Spezifikation mit dem Nutzer

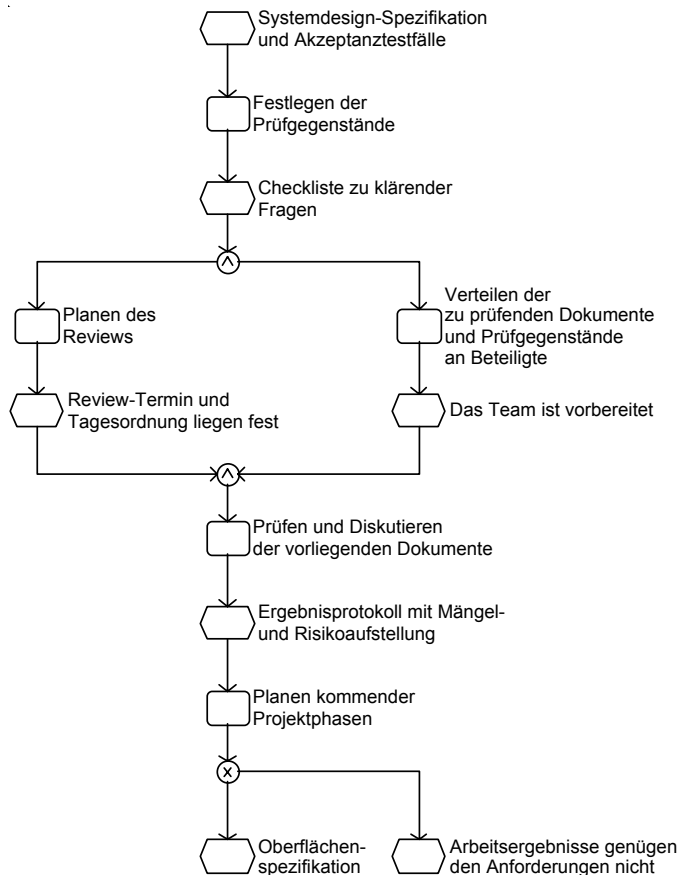


Abbildung 17: Abstimmen der Spezifikation mit dem Nutzer.

Anmerkung: In der Regel werden in der mit dem Auftraggeber abgestimmten Projektplanung Meilensteine fixiert, die Termine und auch die inhaltliche Gestaltung von Reviews regeln.

3.1.4.9.1 Tätigkeiten: Abstimmen der Spezifikation mit dem Nutzer

- Festlegen der Prüfgegenstände – Anmerkung: Prüfgegenstände (ggf. entsprechend den vertraglichen Vereinbarungen) zusammenstellen
- Planen des Reviews – Anmerkung: organisatorische Abstimmung mit dem Auftraggeber; ggf. Ausarbeiten einer Agenda mit festen Zielen und klarem Vorgehen für das Review – u. U. abzuleiten aus den Liefergegenständen, wie sie mit dem Auftraggeber abgestimmt sind
- Verteilen der zu prüfenden Dokumente und Prüfgegenstände an Beteiligte – Anmerkung: Instruktion der Beteiligten bzgl. der Vorbereitung; Bereitstellung aller nötigen Materialien
- Prüfen und Diskutieren der vorliegenden Daten – Anmerkung: Ableiten des Projektstatus' und damit des Fortgangs aus der Beurteilung der Kommission
- Planen kommender Projektphasen

3.1.4.9.2 Kompetenzfelder: Abstimmen der Spezifikation mit dem Nutzer

Fähigkeiten/Fertigkeiten

- externe Sitzungen planen und einberufen können
- Sitzungen zielführend moderieren können
- Sensibilität für die Fokussierung des Kunden entwickeln können
- in einer Prüfungssituation mit Kundenautorität umgehen können
- Intentionen des eigenen Unternehmens gegenüber dem Kunden vertreten können
- Diskussionsgegenstände fixieren/protokollieren können
- aus dem Gespräch Schlüsse und Konsequenzen ziehen können
- mit von anderen gezogenen (und möglicherweise nicht rational erscheinenden) Konsequenzen umgehen können

Wissen

- Protokollierung
- Gesprächsführung

Werkzeuge/Methoden

- Präsentationswerkzeuge zur Unterstützung der Moderation
- Moderationstechniken
- Methoden der Projektplanung

3.1.4.9.3 Beispiel: Abstimmen der Spezifikation mit dem Nutzer

Im Rahmen einer Präsentation hatte der Kunde Gelegenheit seine Wünsche zu äußern. Die Schnittstelle stellt der Produktmanager dar. Die Ergebnisse des Gesprächs wurden in das Design aufgenommen, wobei Differenzen zum Lastenheft markiert wurden.

3.1.4.10 Festlegen der Schnittstellen zur Anwendungs- bzw. Datenbankschicht

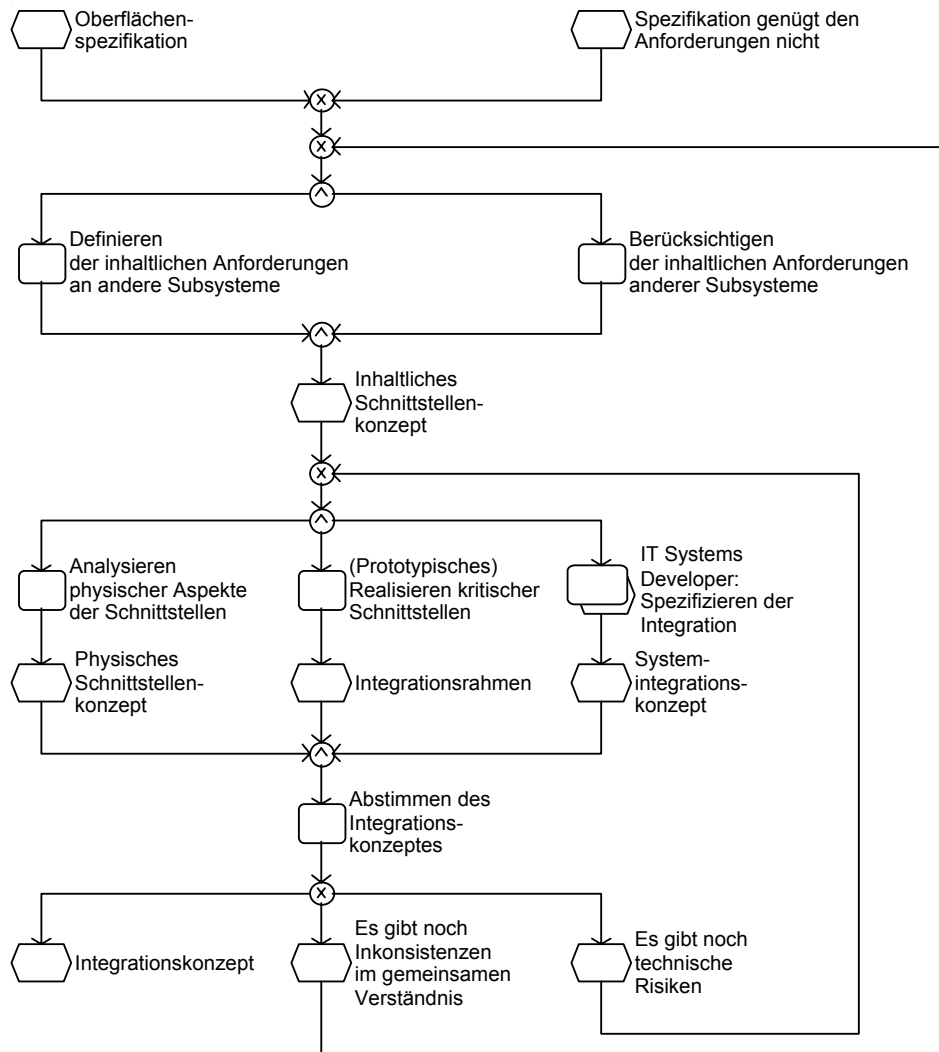


Abbildung 18: Festlegen der Schnittstellen zur Anwendungs- bzw. Datenbankschicht.

Anmerkung: Die wesentlichen Entscheidungen und Festlegungen zur Systemintegration trifft im Allgemeinen der IT Systems Developer. Er koordiniert die Arbeiten der auf die Erstellung der Teilsysteme spezialisierten Entwickler. Das heißt, der UID richtet sich in erster Linie nach den Vorgaben des Systementwicklers, er arbeitet ihm zu bezüglich der für ihn relevanten Bereiche.

3.1.4.10.1 Tätigkeiten: Festlegen der Schnittstellen zur Anwendungs- bzw. Datenbankschicht

- Definieren der inhaltlichen Anforderungen an andere Subsysteme – Anmerkung: Zusammenstellen der vom Backend (z. B. Datenbanken, Echtzeitdatenquellen, Netzdienste) benötigten Daten
- Berücksichtigen der inhaltlichen Anforderungen anderer Subsysteme
- Analysieren physischer Aspekte der Schnittstellen – Anmerkung: Detaillierung des Datenkatalogs: Definition von Datentypen, Aggregaten und ggf. Integritätsregeln; Analysieren physischer und technischer Spezifika der Schnittstellen, etwa beim Einsatz von Netzwerkanbindungen (Bandbreitenrestriktionen) oder der Nutzung von Middleware und der damit verbundenen Typkonvertierungen

- (prototypisches) Realisieren kritischer Schnittstellen – Anmerkung: Umsetzen (Ausprobieren) kritischer Schnittstellenfunktionen, um Risiken auszuräumen
- Abstimmen des Integrationskonzepts

3.1.4.10.2 Kompetenzfelder: Festlegen der Schnittstellen zur Anwendungs- bzw. Datenbankschicht

Fähigkeiten/Fertigkeiten

- Fremdsysteme analysieren können
- kommunizieren können
- (ggf. verteilte) Softwarearchitekturen und damit verwandte Problemstellungen (z. B. Transaktionssteuerung oder Knotenresidenz von Daten) verstehen und einschätzen können
- Entwurfs- und Architekturmuster anwenden/wieder verwenden können
- kleinere Teams koordinieren können

Wissen

- Software-Entwurfstechniken
- gängige Entwurfsmuster und -ideen
- Konzepte verteilter Softwarearchitektur
- Prinzipien des Einsatzes von Middleware
- Grundkenntnisse Datenbanken (Begrifflichkeiten: Transaktion, Authentifizierung etc.)
- Grundkenntnisse Netzwerktechnik (Begrifflichkeiten: Bandbreite, Kommunikationsprotokolle etc.)
- Software-Entwurfsmodelle (objektorientiert, strukturiert) und entsprechende Notationen (z. B. UML)

Werkzeuge/Methoden

- Modellierungswerkzeuge/CASE-Tools (z. B. Rational Rose oder Aonix STP – Software-Through-Pictures)
- Werkzeuge zur Erstellung einfacher technischer Zeichnungen
- Entwicklungsumgebung zur Implementierung der Prototypen
- gängige Middleware-Lösungen (z. B. CORBA oder ORBs, Sun J2EE, Microsoft .NET)
- gängige Protokolle zur Netzwerk- bzw. Interprozesskommunikation auf Anwendungsebene (z. B. TCP/IP Socket, RMI, http, JDBC)
- Standards zur Dateninteroperabilität (z. B. XML, LDAP)
- Software-Entwurfsmethodiken (z. B. objektorientierter Entwurf bzw. strukturierter Entwurf)

3.1.4.10.3 Beispiel: Festlegen der Schnittstellen zur Anwendungs- bzw. Datenbankschicht

Die Entwickler der Oberflächen von DX-Union beispielsweise extrahieren die darzustellenden Daten aus einem LDAP-Server. Ferner muss die Software als Snap In für die Microsoft-Management-Konsole fungieren und damit den Anforderungen der entsprechenden Schnittstellen genügen.

Die Produkte aus dem Hause Materna, die auf Java basieren und verteilte Systeme im Sinne der klassischen 3-Tier-Architektur realisieren, nutzen RMI (Remote Method Invocation –

entfernte Methodenaufrufe) zur Kommunikation zwischen einzelnen Komponenten. Zugriffe auf Datenbanken nutzen in der Regel JDBC (Java DataBase Connectivity).

Das Bewusstsein für die Verantwortung des User Interface Developer bei der Gestaltung der Schnittstellen zum Systemumfeld wird in der Regel erst durch negative Erfahrungen geschärft. Ist ihm beispielsweise nicht klar, dass er für das Anfordern eines Datums – durch einen einfachen Funktionsaufruf gekapselt – eine Anfrage über ein schmalbandiges Netzwerk anstößt (das schließt in der Regel einen enormen Overhead für den Verbindungsauf- bzw. abbau mit ein), wird er zwar sehr handliche Schnittstellen mit sehr hoher Kohäsion definieren, aber damit auch die Zahl der Zugriffe und damit den Overhead vervielfachen. Lange Antwortzeiten und die Notwendigkeit der Überarbeitung der Schnittstellen sind die Folge. So genannte Bulkloads – das Laden größerer Datensätze auf einen Schlag – waren so auch in einem Projekt der Materna GmbH die adäquate Lösung von Antwortzeitproblemen beim Einsatz von analogen Wählmodemverbindungen.

Auf der anderen Seite kann es gerade unabdingbar sein, mit feingranularen Schnittstellen zu arbeiten, etwa wenn das Sperrgranulat für die Transaktionsverarbeitung in Mehrbenutzersystemen sehr fein sein muss und die Zugriffseffizienz nicht durch physische Rahmenbedingungen (wie geringe Bandbreite) beeinflusst wird.

3.1.4.11 Festlegen von Datenformaten und Constraints

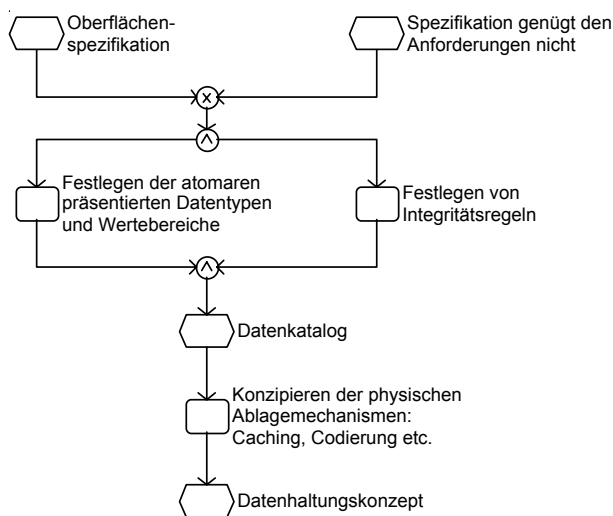


Abbildung 19: Festlegen von Datenformaten und Constraints.

3.1.4.11.1 Tätigkeiten: Festlegen von Datenformaten und Constraints

- Festlegen der atomaren präsentierten Datentypen und Wertebereiche – Anmerkung: Abbilden der zu visualisierenden Informationen auf Datentypen der User-Interface-internen Datenhaltung
- Festlegen von Integritätsregeln
- Konzipieren der physischen Ablagemechanismen: Caching, Codierung etc.

3.1.4.11.2 Kompetenzfelder: Festlegen von Datenformaten und Constraints

Fähigkeiten/Fertigkeiten

- Constraints in Bezug auf atomare Datentypen formalisieren können

Wissen

- Grundlagen der Kodierungstheorie
- Anwendung von Verschlüsselungsmechanismen
- Grundlagen der Datenmodellierung (ER-Modelle)
- Umgang mit formalen Sprachen zur Formulierung von Integritätsregeln
- Ablagemechanismen der verwendeten Plattform

Werkzeuge/Methoden

- Methoden der Datenmodellierung
- Sprachen zur Implementierung (deren Spezifika der Kodierung von Daten)
- CASE-Tools zur Datenmodellierung (z. B. Rational Rose)
- Methoden der Abstraktion/Modellbildung

3.1.4.11.3 Beispiel: Festlegen von Datenformaten und Constraints

Ein negatives Beispiel aus dem Erfahrungsschatz der Materna-Entwickler illustriert die Auswirkungen nachlässig gestalteter Datenhaltungskonzepte: Viele „altgediente“ C-Entwickler neigen dazu, mit untypisierten so genannten Void-Pointern zu arbeiten. Das führt zwar zu größtmöglicher Flexibilität (dynamischer Speichernutzung), leider aber auch zu unlesbarem, kaum zu wartendem Code. Ferner ist das Risiko von Laufzeitfehlern zur Compile-Zeit schwer einzugrenzen, da der Raum der möglichen Systemzustände (Testfälle) ungleich höher ist.

Analog zur Problematik der Erarbeitung der Schnittstellen müssen auch hier der Aufbau des Gesamtsystems, also Aspekte der Verteilung und der Transaktionsverarbeitung berücksichtigt werden. Entscheidungen bezüglich des Einsatzes von Caches und der Granularität der lokal vorgehaltenen Daten hängen auch hier davon ab, ob Zugriffe auf Ressourcen über Netzwerke erfolgen oder die Nutzung durch mehrere Clients synchronisiert werden muss. DX-Union verhindert z. B. pauschal das konkurrierende Schreiben von Konfigurationsdaten. Der Client, der als Erster das Schreibrecht anfordert, erhält dies exklusiv bis zu dem Zeitpunkt, an dem er die Freigabe explizit signalisiert.

3.1.4.12 Entwerfen der Softwaremodule

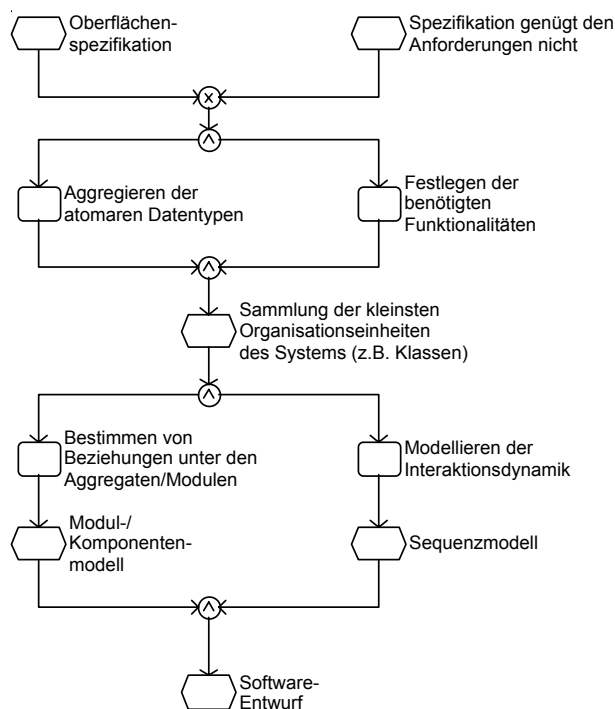


Abbildung 20: Entwerfen der Softwaremodule.

3.1.4.12.1 Tätigkeiten: Entwerfen der Softwaremodule

- Aggregieren der atomaren Daten
- Festlegen der benötigten Funktionalitäten – Anmerkung: Überführen der abstrakten, anwendungsnah formulierten Spezifikationen (z. B. Use Cases) in Modelle, die Komponenten und Module (je nach Komplexität bzw. Bedarf der Detaillierung) der Software (im Sinne eines technischen Systems) repräsentieren
- Bestimmen von Beziehungen unter den Aggregaten/Modulen – Anmerkung: Identifizieren und Modellieren von Beziehungen zwischen einzelnen Bausteinen, um Kommunikationswege, Abhängigkeiten, Verantwortlichkeiten und Integritätsbedingungen zu beschreiben
- Modellieren der Interaktionsdynamik – Anmerkung: die Modellierung erfolgt in der Regel unterstützt durch CASE-Tools, entsprechend einer festen Methodik – diese auszuwählen ist Teil der Aufgaben, die im Rahmen der Festlegung des Entwicklungsrahmens zu erledigen sind

3.1.4.12.2 Kompetenzfelder: Entwerfen der Softwaremodule

Fähigkeiten/Fertigkeiten

- Software-Entwürfe im Rahmen ingenieurmäßig betriebener Software-Entwicklungsprozesse erstellen bzw. aus den Anforderungsmodellen ableiten können
- Sachverhalte beliebiger Anwendungsdomänen auf informationstechnologische Konstrukte abstrahieren können
- Entwurfsmuster anwenden/wieder verwenden können

Wissen

- Software-Entwurfstechniken.
- gängige Entwurfsmuster und -ideen
- Software-Entwurfsmodelle (objektorientiert, strukturiert) und entsprechende Notationen (z. B. UML)
- Software-Entwurfsmethodiken (z. B. objektorientierter Entwurf bzw. strukturierter Entwurf)
- Software-Entwurfsmuster

Werkzeuge/Methoden

- Modellierungswerkzeuge/CASE-Tools (z. B. Rational Rose oder Aonix STP)
- Werkzeuge zur Erstellung einfacher technischer Zeichnungen
- Methoden der Selbstorganisation

3.1.4.12.3 Beispiel: Entwerfen der Softwaremodule

Im Rahmen des Beispielprojekts DX-Union wurde die Software objektorientiert mithilfe des CASE-Tools Rational Rose modelliert. Dabei wurden sowohl Klassendiagramme erstellt als auch die Interaktionen zwischen den Instanzen in Form von Sequenzdiagrammen und Prozessmodellen festgehalten. Die Modellierungssprache UML ist im Hause Materna Standard für die Modellierung objektorientiert zu entwickelnder Systeme. Allerdings ist die Werkzeuglandschaft insofern heterogen, dass auch STP zum Einsatz kommt – mit dem gleichen Zweck.

Mehr oder minder bewusst bzw. implizit durch die Nutzung von Frameworks wie der MFC- bzw. der Java-Swing-Klassen kommt auch das (für die User-Interface-Entwicklung wichtigste) Architekturmuster Model View Control zum Einsatz.

3.1.4.13 Spezifizieren von Modultests

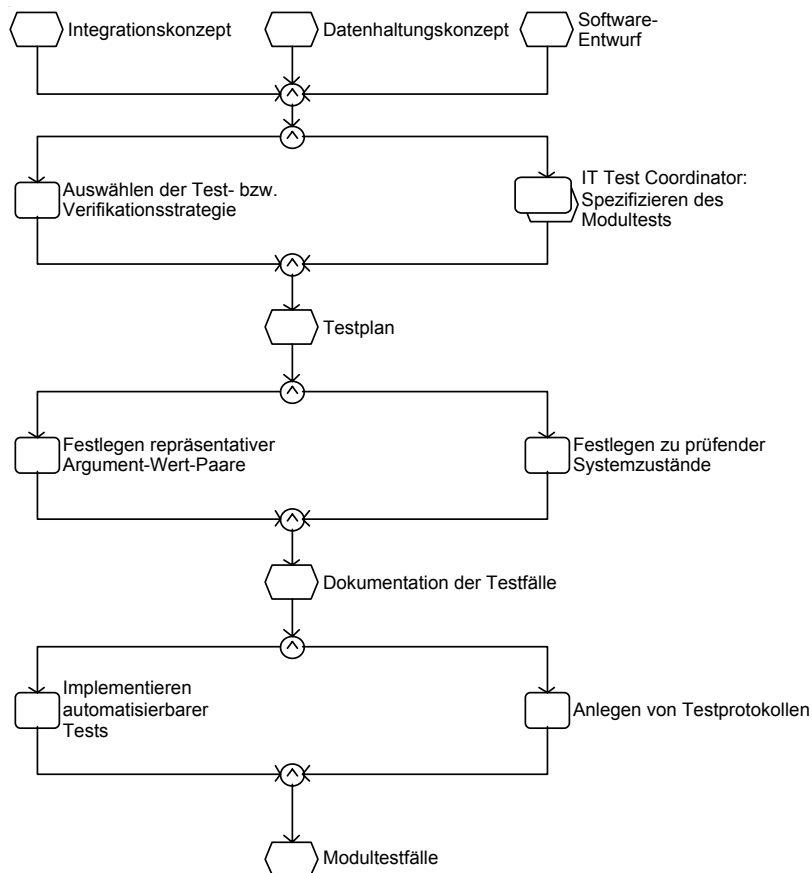


Abbildung 21: Spezifizieren von Modultests.

Anmerkung: Module sind in der Regel das kleinste Granulat, welches in der Konfigurationsverwaltung berücksichtigt wird, d. h. lediglich lauffähige Module werden durch den Entwickler freigegeben für die Nutzung durch das Gesamtsystem (z. B. eingechekkt in ein Versionsverwaltungssystem) – sie verlassen den Verantwortungsbereich des Erstellers. Damit ist es nahe liegend, dem Entwickler auch die Qualitätssicherung auf dieser Stufe zu überlassen. Auch wenn es wünschenswert wäre, so „nahe“ am Quelltext durch unabhängige Instanzen zu testen, ist es gängige Praxis, dass die verantwortlichen Software- und Systemdeveloper davon ausgehen, zuverlässige Module zu erhalten oder zumindest klare Vorschriften und Daten für eine Prüfung. Dass die Testfälle vor der Implementierung spezifiziert werden, ist auch idealtypisch und nicht die Regel – sollte aber angestrebt werden. Die Modultestfälle können auch als eine Präzisierung der Entwurfsmodelle verstanden werden.

3.1.4.13.1 Tätigkeiten: Spezifizieren von Modultests

- Auswählen der Test- bzw. Verifikationsstrategie – Anmerkung: in Abstimmung mit dem IT Test Coordinator und unter Berücksichtigung der Rahmenbedingungen, die das Projekt vorgibt (z. B. Absprachen mit dem Kunden, Festlegungen im Projekthandbuch) werden die Ziele und der Umfang des Testes geplant; ausgehend vom Charakter des Systems (Komplexität, angestrebte Umsetzungsmethodik, Kritikalität) wird die Methode der Prüfung ausgewählt (es werden sich z. B. so genannte White-Box-Tests oder gar formale Korrektheitsbeweise anbieten, wenn umfangreiche Funktionalitäten durch den Entwickler selbst zu implementieren sind und die Kritikalität hoch ist – sind hingegen viele vorgefertigte Komponenten zu integrieren und ist die Kritikalität nicht hoch zu bewerten,

werden Black-Box-Tests unter Beobachtung des Ein-/Ausgabe-Verhaltens der Module ausreichend sein)

- Festlegen repräsentativer Argument-Wert-Paare – Anmerkung: es sind Eingabedaten festzulegen und die korrespondierenden, erwarteten Ausgaben; dabei ist es wichtig, sich nicht auf typische Fälle zu beschränken, sondern die Grenzen der Definitions- bzw. Wertebereiche zu betrachten und auch ungültige Kombinationen einzubeziehen
- Festlegen zu prüfender Systemzustände – Anmerkung: Gleiches gilt für die Definition der zu prüfenden Systemzustände und der entsprechenden Übergänge zwischen Zuständen als Reaktionen auf Ereignisse; viele der Ereignisse werden sich lediglich durch Simulation herbeiführen lassen – dies ist zu spezifizieren (besonders die Reaktionen in Ausnahmefällen sind hier zu betrachten)
- Implementieren automatisierbarer Tests – Anmerkung: die Werkzeuge für automatisierbare Tests sind zu implementieren bzw. im Falle der Verwendung von Testframeworks zu konfigurieren; Versuchsaufbauten zur Messung physikalischer Größen am System sind zu installieren
- Anlegen von Testprotokollen – Anmerkung: mit den Angaben der Spezifikation der Modultests sind Testprotokolle anzulegen, die im Zuge der Durchführung der Tests mit den Ergebnissen befüllt werden können, um dann als Dokumentation für die ggf. nötige Fehlerbeseitigung bzw. als Teil der Liefergegenstände dienen zu können

3.1.4.13.2 Kompetenzfelder: Spezifizieren von Modultests

Fähigkeiten/Fertigkeiten

- Stellenwert von Tests projektabhängig einschätzen können
- potenzielle technische Risikopunkte (selbstkritisch) voraussehen können
- Prüfungsszenarien planen können
- Verhalten komplexer dynamischer Systeme einschätzen können
- relevante und signifikante Prüfzuszenarien erkennen können
- Ergebnisse von Prüfungen systematisieren können

Wissen

- Algebren für das Verständnis der Zuordnung Funktion–Argumentmenge–Wertemenge
- inhaltliches Verständnis für die Argument- und Wertemengen (Domänenwissen)
- Zustandsmodelle
- Vorlagen für die Anlage der Dokumentation

Werkzeuge/Methoden

- Frameworks und CASE-Tools für den Softwaretest – speziell für die Simulation von Nutzerinteraktion
- Vorgehensweisen bei der Softwareverifikation (Black-Box-/White-Box-/Stresstests, formale Beweisverfahren etc.)
- Werkzeuge zur Unterstützung der Dokumentation (Trouble-Ticket-Systeme etc.)
- Messgeräte zum Prüfen des Verhaltens von Sensoren und Aktuatoren zur Nutzerinteraktion (verschieden von den üblichen Displays und Eingabegeräten)

3.1.4.13.3 Beispiel: Spezifizieren von Modultests

Bei der Materna GmbH gibt es derzeit keinen zentral definierten Prozess, der Modultests unter Einbeziehung einer eigenen Prüfinstanz systematisiert. Das bedeutet, man geht davon

aus, dass die Entwickler selbst ausschließlich qualitätsgesicherte Module veröffentlichen. Gerade bei der Entwicklung von Projekten auf der Basis von Java ist allerdings der Trend erkennbar, dass viele Entwickler sich des JRun-Frameworks bedienen, um Tests zu automatisieren und damit auch zu systematisieren.

Die durchschnittliche Komplexität der Projekte bzgl. des HCI bei Materna macht auch eine Simulation der Nutzerinteraktion nicht notwendig. Sollten eine hohe Interaktivität und unüberschaubar viele Systemzustände (eine große Anzahl von Masken mit vielen voneinander abhängigen Bedienelementen) den Charakter eines Systems prägen, ist zu empfehlen entsprechende CASE-Tools einzusetzen. Das setzt allerdings voraus, dass man die eigentliche Nutzerinteraktion mithilfe eines verbreiteten Tool Kits erstellen möchte, also entsprechende Werkzeuge verfügbar sind.

3.1.4.14 Abstimmen des Entwurfs mit dem Team

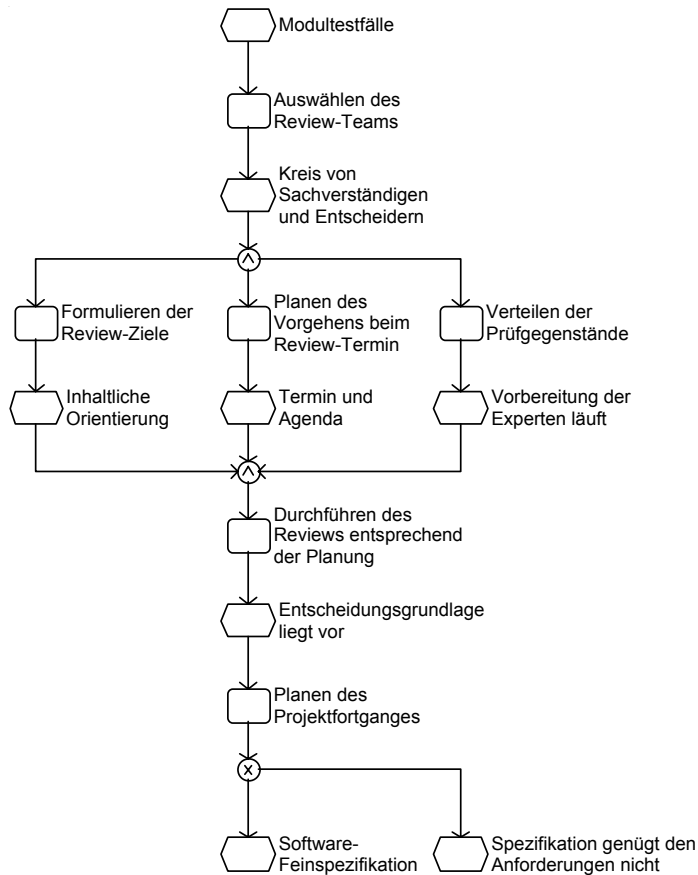


Abbildung 22: Abstimmen des Entwurfs mit dem Team.

3.1.4.14.1 Tätigkeiten: Abstimmen des Entwurfs mit dem Team

- Auswählen des Review-Teams – Anmerkung: Auswahl kompetenter, idealerweise unabhängiger Sachverständiger
- Formulieren der Review-Ziele
- Planen des Vorgehens beim Review-Termin – Anmerkung: Ausarbeiten einer Agenda mit festen Zielen und klarem Vorgehen für das Review
- Verteilen der Prüfgegenstände – Anmerkung: Instruktion der Beteiligten bzgl. der Vorbereitung; Bereitstellung aller nötigen Materialien
- Durchführen des Reviews entsprechend der Planung – Anmerkung: Moderation der Sitzung
- Planen des Projektfortganges – Anmerkung: Ableiten des Projektstatus' und damit des Fortgangs aus der Beurteilung der Kommission

3.1.4.14.2 Kompetenzfelder: Abstimmen des Entwurfs mit dem Team

Fähigkeiten/Fertigkeiten

- interne Sitzungen einberufen und planen können
- Sitzungen zielführend moderieren können
- Diskussionsgegenstände fixieren/protokollieren können
- aus dem Gespräch Schlüsse und Konsequenzen ziehen können

Wissen

- Überblick über potenzielle Gutachter (Know-how und Zuständigkeiten im eigenen Hause)
- Protokollierung
- Gesprächsführung

Werkzeuge/Methoden

- Groupware zur Besprechungsorganisation
- Präsentationswerkzeuge zur Unterstützung der Moderation
- Moderationstechniken
- Methoden der Projektplanung

3.1.4.14.3 Beispiel: Abstimmen des Entwurfs mit dem Team

Die anfallenden Dokumente wurden zwischen den Teammitgliedern hin- und hergereicht. Dabei wurden besonders formale Aspekte/Wording abgestimmt.

Es wurden auch QM-Checklisten zurate gezogen.

3.1.4.15 Konfigurieren und Anpassen von COTS-Komponenten

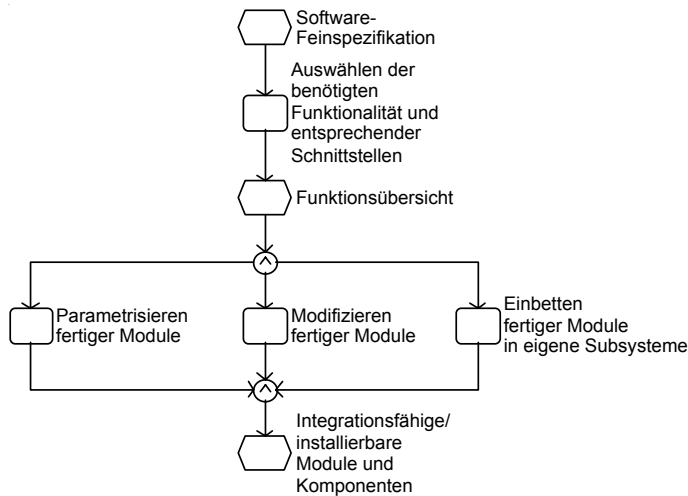


Abbildung 23: Konfigurieren und Anpassen von COTS-Komponenten.

3.1.4.15.1 Tätigkeiten: Konfigurieren und Anpassen von COTS-Komponenten

- Auswählen der benötigten Funktionalität und entsprechender Schnittstellen
- Parametrisieren fertiger Module
- Modifizieren fertiger Module
- Einbetten fertiger Module in eigene Subsysteme

3.1.4.15.2 Kompetenzfelder: Konfigurieren und Anpassen von COTS-Komponenten

Fähigkeiten/Fertigkeiten

- branchenübliche Standardprodukte kennen und bewerten können
- die „Community“ kennen und auf ihrer Basis Recherchearbeiten durchführen und Problemlösungen erarbeiten können
- Aufwände und Risiken des eigenen Produkts abschätzen und es mit ähnlichen Systemen kritisch vergleichen können
- die Leistung anderer bewerten und respektieren können
- Fremdsysteme und deren Schnittstellen analysieren können
- für Modifikationen vorgesehene Fremdsysteme (mehr oder minder gut dokumentiert) entsprechend den eigenen Anforderungen anpassen können

Wissen

- Kenntnis der wichtigsten Frameworks zur GUI-Entwicklung
- Kenntnis der gängigen Framework-Architekturen und von deren Wirkprinzipien sowie der entsprechenden Entwicklungswerkzeuge
- entsprechende Muster zur Nutzung bzw. Anknüpfung wieder verwendbarer Softwarebausteine
- Erfahrung im Umgang mit Infrastrukturen/Protokollen (z. B. Middleware) zur Etablierung von Interoperabilität zwischen Systemteilen
- unternehmenspolitische Rahmenbedingungen und Maßgaben

Werkzeuge/Methoden

- Assessment und Vergleich von Software sowie Softwarekomponenten
- Methoden der Aufwandsschätzung
- GUI-Entwicklungsumgebungen

3.1.4.15.3 Beispiel: Konfigurieren und Anpassen von COTS-Komponenten

Ein eher untypisches, aber interessantes Beispiel für den Einsatz spezieller Werkzeuge zur Erstellung von Nutzerschnittstellenfunktionalität bietet eine ältere Version eines Systems zur Fernwartung heterogener, verteilter Systeme, die mithilfe des Produkts Dialogmanager (der Firma ISA in Stuttgart) entwickelt wurde. Damit konnte relativ einfach und auf graphischem Wege die Oberfläche gestaltet werden. Das Ergebnis ist kein Programmcode im eigentlichen Sinne, sondern lediglich eine konfigurierte Instanz des Dialogmanagers. Durch seine Verfügbarkeit für die verschiedensten Plattformen erreichte man die gewünschte Interoperabilität in diesem heterogenen Netzwerk.

Mit HP-Openview wurde ein fertiges Systemmanagement-Produkt eingesetzt und nur geringfügig an die Wünsche der Kunden angepasst: die Farbe von Symbolen geändert etc. In ähnlicher Weise wird die Microsoft-Management-Konsole, die den Rahmen für DX-Union bildet, verändert: über entsprechende Schnittstellen konfiguriert.

Weit verbreitet in den Entwicklungsabteilungen der Firma Materna ist ansonsten das gängige JAVA-Swing-Framework. Dieses bietet einen umfangreichen „Baukasten“ mit den verschiedensten Steuerelementen – ganz grundlegenden wie Texteingabefeldern und Check-boxen, aber auch komplexeren wie Tabellendarstellungen und Standarddialogboxen. Diese Steuerelemente besitzen alle grundsätzlichen (querschnittlich gebrauchten) Eigenschaften wie das Verfolgen von Nutzerinteraktionen (Mausklicks und Tastenanschlägen). Grundlegende Reaktionen wie etwa das Ausklappen von Menüs sind vorgegeben. Diese Elemente werden von einem kontrollierenden (dispatchenden) Rahmen (der Applikation) zusammengefasst. Die Anwendung mit all ihren Bedienelementen folgt einem zentral definierbaren, so genannten portablen „Look and Feel“ (PLAF). Das heißt, die graphische Erscheinung der Anwendung und damit der befriedigte Style Guide kann an die entsprechende Zielplattform (Microsoft Windows oder auch Unix Motif) angepasst werden. Der Entwickler (der Anwender des Frameworks) erstellt eine Instanz einer Swing-Anwendung üblicherweise, indem er innerhalb einer graphischen Entwicklungsumgebung die benötigten Steuerelemente zu Dialogmasken arrangiert und initiale Inhalte wie Bezeichnungen einfügt und Schalterzustände vorgibt. Die Reaktionen, also auch Interaktionen zwischen den Steuerelementen, werden dann innerhalb des Ereignisgerüsts (Geflecht von Funktionen, die das Framework bei Ereignissen standardmäßig aufruft) in Java implementiert.

3.1.4.16 Realisieren der Softwaremodule

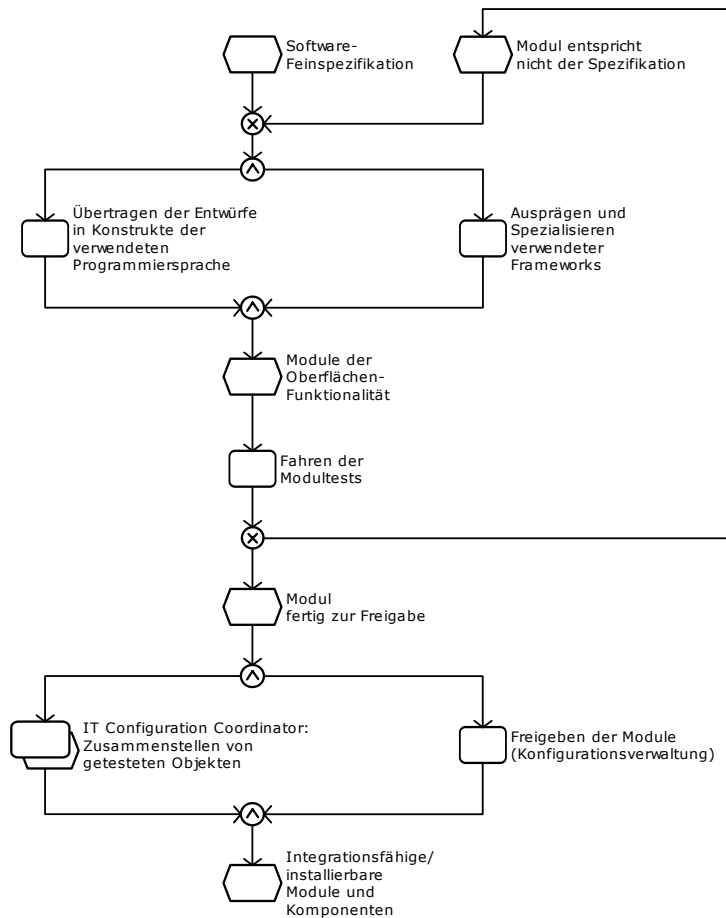


Abbildung 24: Realisieren der Softwaremodule.

Anmerkung: Dieser Teilprozess dominiert in der Praxis häufig den Gesamtentwicklungsprozess, da hier das eigentliche Produkt der Software-Entwicklung erzeugt wird. Viele der anderen Aktivitäten wie der Software-Entwurf laufen implizit – besonders bei kleineren Projekten. Genau davon sollte man sich jedoch lösen, um die tatsächlich oft schwer kalkulierbare Komplexität überschaubar zu machen. Hier sollte bewusst lediglich ein Schritt der Transformation (Abstraktion) bereits formalisierter Konzepte (Feinentwurfmodelle) in Konstrukte der Zielplattform vollzogen werden. Es werden Modellelemente des Entwurfs (z. B. Klassen, aber auch ganze Patterns etc.) überführt in die Sprache (unter Verwendung entsprechender Idio-me), auf deren Basis lauffähige Programme generiert werden können. Diese Abbildung ist aufgrund der Ausdrucksmächtigkeit von gängigen Programmiersprachen selbst bei qualitativ hochwertigen detaillierten Entwürfen nicht eindeutig und nur in Ansätzen automatisierbar. Nötige Optimierungen an Algorithmen und Datenstrukturen ergeben sich oft aus Spezifika der verwendeten Compiler bzw. der Ablaufumgebung für das Softwareprodukt und erfordern viel Erfahrung mit eben dieser Konstellation.

3.1.4.16.1 Tätigkeiten: Realisieren der Softwaremodule

- Übertragen der Entwürfe in Konstrukte der verwendeten Programmiersprache – Anmerkung: fortgesetztes Abstrahieren der Entwurfsmodelle hin zu „maschinenverarbeitbaren“ Konstrukten (Programmcode); selbstständiges oder auch paarweises Formulieren und Strukturieren von verständlichem, ggf. wieder verwertbarem Quelltext/Programmcode im weitesten Sinne (rein visuelle Entwicklungsumgebungen sind gerade auf diesem Gebiet der Trend der Entwicklung); Anpassen und Prüfen vorhandener Quelltextteile; Erzeugen (Übersetzen) lauffähiger Artefakte, ggf. mit simulierten angrenzenden Schnittstellenfunktionalitäten (z. B. mit Compilern); Einhalten (u. U. auch Etablieren) sog. Coding Rules –

dabei handelt es sich um Regeln, die die Gestaltung des Quelltexts vereinheitlichen (etwa wie Bezeichner zu benennen sind oder wann und wie Zeilen umgebrochen sein sollten)

- Ausprägen und Spezialisieren verwendeter Frameworks
- Fahren der Modultests
- Freigeben der Module (Konfigurationsverwaltung)

3.1.4.16.2 Kompetenzfelder: Realisieren der Softwaremodule

Fähigkeiten/Fertigkeiten

- „reale“ Prozesse und deren Modelle abstrahieren und formalisieren können
- mit Code-Generatoren zur (semi-)automatisierten Überführung von Entwurfsmodellen in übersetzbaren Quelltext umgehen können
- Code verfassen können
- mit Quelltexten, die durch Dritte verfasst sind, umgehen können
- Übersetzungs- bzw. Build-Prozesse verstehen und durchführen können
- die eigene Leistung kritisch bewerten können
- getroffene Entscheidungen revidieren können

Wissen

- gängige Programmiersprachen und deren Laufzeitumgebungen
- Algebren und Programmiermodelle zur verwendeten Entwicklungssprache
- Berechnungskomplexität
- Erfahrung mit dem kooperativen Verfassen von Quelltexten (unterstützt durch entsprechende Werkzeuge)
- Standards zur Vermeidung undurchsichtigen (obfuscated) Programmcodes

Werkzeuge/Methoden

- Methoden und Werkzeuge zur (ggf. automatisierten) Überführung der Entwurfsmodelle in Programmcode
- integrierte Entwicklungsumgebungen
- Compiler und ggf. Linker
- Werkzeuge zur Verwaltung von kooperativ erstellten Quelltexten (Versionsverwaltung etc.)
- Methoden der Selbstorganisation

3.1.4.16.3 Beispiel: Realisieren der Software-Module

Das in 3.1.4.17.3 beschriebene System zur Verwaltung beliebiger Konfigurationen (Continuus) wird neben allen Dokumenten, die im Zuge des Projekts erstellt werden, hauptsächlich auch den entwickelten Programmcode verwalten (versionieren, konkurrierende Zugriffe steuern etc.). Das System bildet damit auch das Rückgrat der Implementierung wobei jeder Entwickler lokal den Quelltext verfasst und testet, um ihn nach Fertigstellung im Repository von Continuus zu veröffentlichen.

Die Implementierung selbst nimmt die im Zuge des Entwurfs entstandenen UML-formulierten Klassen- bzw. Ablaufdiagramme als Grundlage in der Art technischer Zeichnungen. Die dort identifizierten Klassen von so genannten Geschäftsobjekten werden in Klassen im Sinne der Sprache Java bzw. C++ übertragen. Attribute werden angelegt und Funktionalitäten, die sich

aus der Interaktionsmodellierung ergeben, implementiert. In der Regel werden während des Entwurfs nur die wesentlichen Geschäftsklassen bzw. tragende Infrastrukturklassen erfasst, sodass der Entwurf meist noch um Hilfsklassen bzw. Funktionalitäten ergänzt werden muss. Auf diesem Wege werden konzeptionelle Brüche zwischen Modellierungs- und Programmiersprache geschlossen.

Die schwerwiegendsten Fehler während der Implementierung ergeben sich meist aus Unkenntnis der Ablaufumgebung und deren Zusammenspiel mit dem erzeugten Programm. So sollte man sich beispielsweise dessen bewusst sein, dass mit dem Aufruf „new“ erzeugte Speicherobjekte in C++ so lange Ressourcen belegen, bis sie explizit zerstört werden, während in Java etwa bereits konzeptionell ein Mechanismus verankert ist, der nicht mehr benötigte Objekte erkennt und den durch sie belegten Speicher freigibt.

Die Kenntnis der wichtigsten Fakten der Theorie zur Berechnungskomplexität sollte immer zu den effizientesten Umsetzungen der Algorithmik führen bzw. zur realistischen Einschätzung der Machbarkeit. So bergen auch einfache Such- und Sortieraufgaben bzw. Probleme, die damit verwandt sind (darauf reduzierbar), das Risiko von Implementierungen, die besonders bei der Verarbeitung großer operativer Datenmengen nach der Auslieferung versagen, d. h. inakzeptabel langsam reagieren.

3.1.4.17 Integrieren der Module und Komponenten in die Gesamtarchitektur

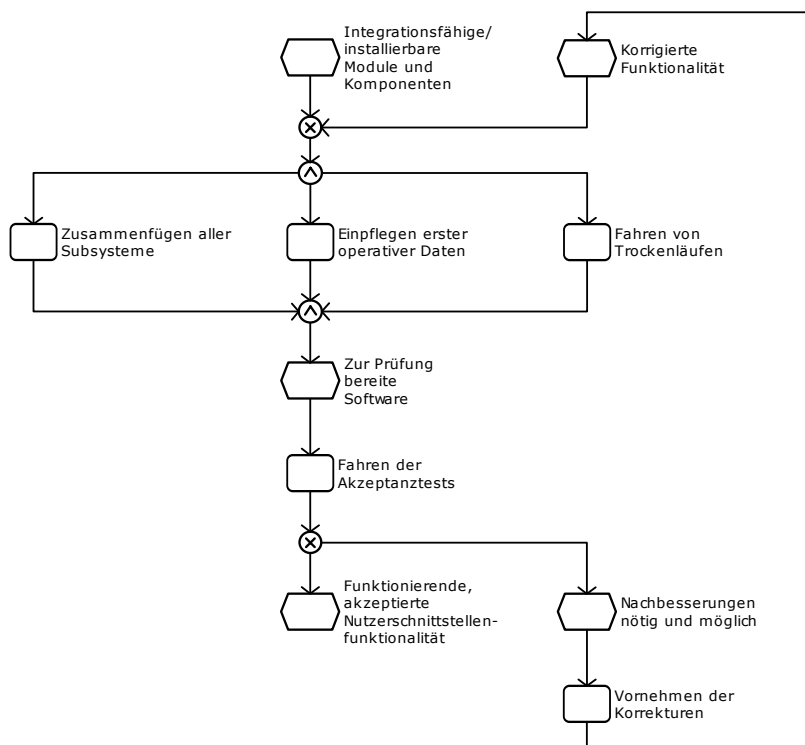


Abbildung 25: Integrieren der Module und Komponenten in die Gesamtarchitektur.

Anmerkung: Separat (inkrementell) entwickelte und getestete Module und Teilsysteme werden zu einem System zusammengefügt. Die Schnittstellen und das System als Ganzes werden geprüft, eventuell korrigiert und letztlich abgenommen.

3.1.4.17.1 Tätigkeiten: Integrieren der Module und Komponenten in die Gesamtarchitektur

- Zusammenfügen aller Subsysteme – Anmerkung: Anbinden der benachbarten Subsysteme bzw. Module; ggf. Einbinden der erzeugten Code-Teile in den Gesamt-Build-Prozess – das bedeutet, dass parallel entwickelte Code-Stränge zusammengeführt werden müssen (was grundsätzlich in der Verantwortung des IT Configuration Coordinator liegt)
- Einpflegen erster operativer Daten – Anmerkung: Einpflegen erster Daten bedeutet für den User Interface Developer häufig das Eingeben von Daten über das erstellte Interface (also mehr oder minder „von Hand“) oder aber über zu erstellende Ladeskripte; bei großen, mehrschichtigen Systemen mit Datenbankbindung fällt diese Aufgabe allerdings eher dem User Interface Developer zu; es müssen Konstanten (etwa für die Lokalisierung) in die Konfiguration eingetragen werden
- Fahren von Trockenläufen – Anmerkung: dabei werden die Funktionen des Systems durchgespielt und auf ihr grundsätzliches Funktionieren untersucht; der User Interface Developer spielt auch hier eine herausragende Rolle, da er das tiefste Verständnis von der Handhabung des Systems und des gewünschten Verhaltens haben sollte
- Fahren der Akzeptanztests – Anmerkung: das Durchführen der Abnahmetests systematisiert die Trockenläufe entsprechend dem aufgestellten Prüfplan und in der Regel im Beisein des Kunden bzw. des Endnutzers

3.1.4.17.2 Kompetenzfelder: Integrieren der Module und Komponenten in die Gesamtarchitektur

Fähigkeiten/Fertigkeiten

- sich und das entwickelte Produkt in Projektteams, die sich aus den verschiedensten Rollen zusammensetzen bzw. von Ihnen beeinflusst werden, einordnen können
- Methoden der Koordination von Projektteams verstehen und mit ihnen umgehen können
- unter zeitlichem oder auch politischem Druck arbeiten können
- mit großen Pools von Quelltext umgehen können (wobei große Teile davon von anderen Entwicklern erstellt und strukturiert worden sein können)
- Integrationstests und Abnahmeprüfungen entsprechend den vorliegenden Prüfplänen vornehmen können

Wissen

- Konfigurationsmanagement (Verwaltung und Strukturierung von Softwareartefakten)
- Projektmanagement
- Vorgehensweise bei der Software-Integration

Werkzeuge/Methoden

- Middleware-Plattformen (Funktionsweise und Methoden der Fehlerdiagnose)
- Werkzeuge zur Unterstützung des Konfigurationsmanagements.
- integrierte (meist skriptgesteuerte) Build-Werkzeuge (z. B. Make, Java ANT)

3.1.4.17.3 Beispiel: Integrieren der Module und Komponenten in die Gesamtarchitektur

Die Materna GmbH strebt an, alle größeren Softwareprojekte bzgl. des Konfigurationsmanagements auf der Basis des Produkts Continuum CMM zu entwickeln. Dort wird ein Projekt mit all den entstehenden Artefakten (Handbuchseiten, Code, Dokumentation) – in Continuum CMM als Tasks bezeichnet – in einem so genannten Problem zusammengefasst. So bildet etwa ein Subsystem wie die vom User Interface Developer entwickelte Funktionalität einen einzelnen Task, der losgelöst vom gesamten Problem lokal am Arbeitsplatz des Entwicklers realisiert und getestet wird, um dann in einem Freigabeprozess für das restliche Team und die Integration in das Gesamtsystem verfügbar gemacht zu werden.

Die Heterogenität der Bezeichnungen für inhaltlich dieselben Verfahren wird am Beispiel der Trockenläufe deutlich. Diese werden im Hause Materna als Implementer-Test bezeichnet. So werden mit der Systemmanagementsoftware etwa probeweise Nutzer angelegt und entsprechende Ressourcen zugeordnet. Dafür werden sporadisch Spezialisten der Qualitätssicherung herangezogen. Häufig werden diese „Generalproben“ aber auch von den Entwicklern selbst durchgeführt.

3.1.4.18 Unterstützen des Deployment-Prozesses

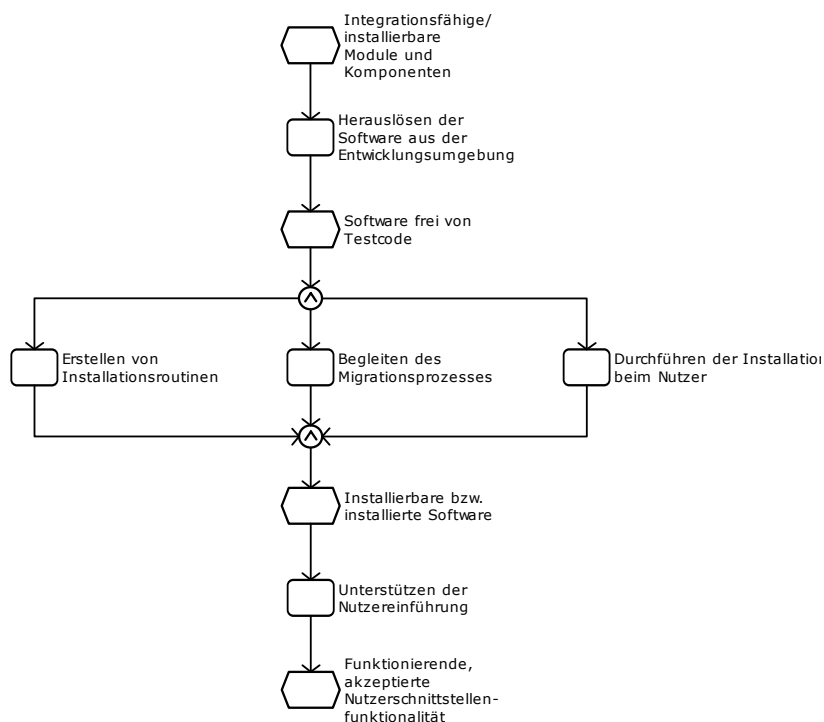


Abbildung 26: Unterstützen des Deployment-Prozesses.

Anmerkung: Dieser Teilprozess wird stark davon geprägt sein, in welcher Form das System auszuliefern ist – ob es sich dabei etwa um eine einmalige individuelle Installation oder ein vielfach vertriebenes Produkt für den Massenmarkt handelt. Die individuelle Installation (ggf. unter Migration von einem Altsystem) und die damit in der Regel verbundene direkte Betreuung der Nutzer kann diese Phase im Ganzen sehr aufwändig werden lassen. Ein Massenprodukt hingegen wird einen singulären Schwerpunkt bei der nutzerfreundlichen Gestaltung der Installationsroutinen setzen.

3.1.4.18.1 Tätigkeiten: Unterstützen des Deployment-Prozesses

- Herauslösen der Software aus der Entwicklungsumgebung – Anmerkung: Befreien der Software von Testcode; finales Übersetzen aller Quellen; Zusammenstellen aller benötigten Dateien (Konfigurationsdateien, Ressourcen, Programmbibliotheken, Executables) – das erfordert im Idealfall lediglich die Nutzung der entsprechenden Funktion der integrierten Entwicklungsumgebung bzw. des angeschlossenen Konfigurationsmanagement-Werkzeugs
- Erstellen von Installationsroutinen – Anmerkung: Dokumentieren der für die Installation wichtigen Einstellungen, Maßgaben (Dateiorganisation, Parametereinstellungen, Systemvoraussetzungen); Erstellen der Dokumentation zum Vorgehen während der Installation; Erstellen so genannter Skripte bzw. integrierter Pakete zur Steuerung der Extraktion sowie der eigentlichen Installation der Software – häufig erfordert die Nutzung dieser Funktionen die Interaktion mit dem Endnutzer, weshalb es wünschenswert ist, wenn der User Interface Developer seinen Sachverstand hier schwerpunktmäßig einbringt
- Begleiten des Migrationsprozesses
- Durchführen der Installation beim Nutzer – Anmerkung: ggf. Durchführen manueller Installationsschritte vor Ort beim Nutzer

- Unterstützen der Nutzereinführung – Anmerkung: Begleiten bzw. Durchführen von Nutzerschulungen; Unterstützen der Handbucherstellung

3.1.4.18.2 Kompetenzfelder: Unterstützen des Deployment-Prozesses

Fähigkeiten/Fertigkeiten

- Software-Gesamtkonfiguration überblicken können
- plattformabhängige Standards und Konventionen für Installationen anwenden können (z. B. die Ablage eigener Java-Bibliotheken im /lib/ext-Verzeichnis der virtuellen Maschine oder von Standard-Installationsdialogen etc.)
- Verständnis für die Schwierigkeiten des Nutzers beim „Erstkontakt“ mit neuer Software aufbringen können

Wissen

- Spezifika der Zielplattform (Betriebssystem/Hardware) und resultierende Abhängigkeiten

Werkzeuge/Methoden

- Werkzeuge zur Erstellung von Installationspaketen (z. B. Installshield)
- Skriptsprachen zur Erstellung einfacher Installationsroutinen (z. B. Shell-Script oder Windows-Batch-Dateien)

3.1.4.18.3 Beispiel: Unterstützen des Deployment-Prozesses

Die Firma Materna profitiert auch hier vom Einsatz der Software Continuum CMM, mit deren Hilfe alle Artefakte des Entwicklungsprozesses zentral verwaltet werden. Damit ist auch unterstützt, dass die Build-Plattform der Zielplattform weitgehend entspricht. Dem Quasistandard folgend werden alle Produkte, die dem Nutzer zur Installation überlassen werden, mit der Installshield-Software „verpackt“. Die Multiplattform-Edition dieser Software ermöglicht es etwa, Java-Programm-Installationsdialoge auf allen Plattformen mit immer dem gleichen „Look-and-Feel“ zu präsentieren. Die Dialoge entsprechen dabei immer dem vom Nutzer erwarteten Standard.

Abweichend von diesem Idealfall werden viele, vor allem große, evolutionäre Softwareprojekte in relativ heterogenen Umgebungen entwickelt, bei denen beispielsweise die Patch-Stände der Betriebssysteme bzw. Compiler der Entwicklerrechner variieren. Wenn dann zusätzlich mit der Auslieferung ein Generationswechsel folgt – nach langer Entwicklungszeit werden Kunden beispielsweise sicher die dann aktuellen Maschinen anschaffen –, wird der Installationsprozess mit Sicherheit aufwändig. So müssen beispielsweise veränderte Strukturkonventionen berücksichtigt und die benötigten Bibliotheken zusammengetragen werden.