MASTERARBEIT

AUF DEM WEG ZU EINEM STRATEGIEBASIERTEN VORGEHENSMODELL ZUR SOFTWARE-ENTWICKLUNG

Eine Kombination von Software- und Service-Engineering

ausgeführt am

Studiengang
Informationstechnologien und Wirtschaftsinformatik

Von: Nadalina Pscheidt
Personenkennzeichen: 1910320036

Graz, am 05. Juli 2022 .....

Unterschrift

EHRENWÖRTLICHE ERKLÄRUNG

Ich erkläre ehrenwörtlich, dass ich die vorliegende Arbeit selbstständig und ohne fremde Hilfe verfasst, andere als die angegebenen Quellen nicht benützt und die benutzten Quellen wörtlich zitiert sowie inhaltlich entnommene Stellen als solche kenntlich gemacht habe.

.....

Unterschrift

DANKSAGUNG

Für die Geduld und Unterstützung bedanke ich mich bei meiner Betreuerin FH-Prof. Dr. Elisabeth Pergler.

Ohne die enorme Unterstützung und Ermutigung meines Mannes, Dipl.-Ing. Dr. Markus Pscheidt, wäre diese Masterarbeit nicht zustande gekommen, ich danke dir sehr.

Ich bedanke mich auch sehr bei meiner Familie, die für mich immer da ist.

Nicht zuletzt ein großer Dank an alle Arbeitskollegen von der Softwareabteilung der Firma Bartelt, insbesondere an Dipl.-Ing. Peter Schöttel und Ing. Robert Löscher.

KURZFASSUNG

Hauptziel der vorliegenden Masterarbeit ist es, eine Kombination von Service-Engineering und Software-Engineering zu finden, die geeignet ist, um Softwarelösungen mit Mehrwert für Kunden und Softwareunternehmen zu generieren.

Die meisten Softwareunternehmen haben bereits eine hohe Expertise bei der Entwicklung von Softwarelösungen entwickelt und hierfür klare Methoden und Prozesse definiert.

Die Anzahl neuer Leistungen, die sich als Flop erweisen und aufgrund zu geringen Umsatzes vom Markt genommen werden, ist extrem hoch. Viele Autoren sind sich einig, dass das fehlende Glied, um Innovationen erfolgreich zu kreieren, der Einsatz einer Entwicklungsstrategie ist.

Bedauerlicherweise gibt es noch eine große Zahl von IT-Unternehmen, die in der Praxis eine Ad-hoc-Entwicklung von Services vornehmen, oft unvollständig und unstrukturiert. Aber um Innovation richtig zu fördern, benötigen Unternehmen ein Vorgehensmodell, um neue Chancen zu identifizieren und bestehende Geschäftsfelder weiter zu entwickeln und auszubauen.

Aus den Erkenntnissen des Software- und Service-Engineering, insbesondere des Engineering IT-basierter Dienstleistungen, kommt diese Masterarbeit zu dem Schluss, dass die Entwicklung von Softwarelösungen nicht ohne strategischen Kontext passieren soll. Was benötigt wird, ist ein Vorgehensmodell, das die integrierte Entwicklung von Software und Dienstleitungen explizit behandelt.

Das vorgeschlagene strategiebasierte Vorgehensmodell zur Softwareentwicklung ist eine Kombination aus Software- und Dienstleistungsentwicklungsmodellen. Es behandelt die integrierte Entwicklung von Software und Dienstleistungen.

Die Ergebnisse dieser Arbeit bestätigen, dass das strategiebasierte Vorgehensmodell in der Praxis mit eigenen Mitteln anwendbar ist und wichtige Informationen liefert, um strategische Entscheidungen für bestehende oder neue Software-Lösungen zu treffen.

ABSTRACT

The main goal of this master thesis is to find a combination of service engineering and software engineering that is suitable for generating software solutions with added value for customers and software companies.

Most software companies have already developed a high level of expertise in the development of software solutions and have defined clear methods and processes for this.

The number of new services that have proven to be flops and are withdrawn from the market due to insufficient sales is extremely high. Many authors agree that the missing link to successfully develop innovations is the use of a development strategy.

Unfortunately, a large number of IT companies still practices ad hoc services development, often incomplete and/or unstructured. However, in order to properly promote innovation, companies need a process model to identify new opportunities and to further develop and expand existing business areas.

From the knowledge of software and service engineering, in particular the engineering of E-services, this master thesis comes to the conclusion that the development of software solutions should not happen without a strategic context. What is needed is a process model that explicitly deals with the integrated development of software and services.

The proposed strategy-based process model for software development is a combination of software and service development models. It deals with the integrated development of software and services.

The results of this work confirm that the strategy-based process model is applicable in practice and provides important information for making strategic decisions for existing or new software solutions.

INHALTSVERZEICHNIS

1 EINLEITUNG

„Die zunehmend sich verschärfende Konkurrenzsituation, verändertes Kundenverhalten wie auch technologischer Fortschritt machen für viele Anbieter die Beschreitung neuer Wege zur Erringung von Wettbewerbsvorteilen notwendig“ (Reichwald und Schaller 2003).

Ohne Innovation kann keine Firma langfristig überleben, und das gilt auch für Software Firmen. Wie Pikkarainen et al. (2011) formuliert haben, war die pünktliche, budgetgerechte und akzeptable Lieferung von Softwareprodukten für die meisten Softwareunternehmen schon immer eine große Herausforderung. In diesem historischen Kontext haben sich Softwareunternehmen auf die sogenannte „Kunst des Software-Engineerings“ spezialisiert. Sie beherrschen den Prozess der Erstellung von Software-Assets wie Code und Anforderungsdokumenten. Heute stehen sie vor einer neuen Generation von Herausforderungen, die als „Kunst der Software-Innovation“ eingestuft werden können. Wie kann ich (als Unternehmen) mit Software innovieren und Werte schaffen?

Laut Reichwald und Schaller (2006), haben sich zwischen 30% und 50% aller neu Leistungen am Markt als Flops erwiesen und mussten Aufgrund geringen Umsatzes vom Markt genommen werden.

Studien aus dem Service Engineering haben sich mit den entscheidenden Erfolgsfaktoren für neue Dienstleistungen beschäftigt. Dazu zählen integrierte Teams, Mitgestaltung der Kunden (Kunden-Co-Kreation), eine klar definierte und kommunizierte Serviceentwicklungsstrategie und ein formalisierter Serviceentwicklungsprozess. Eine Service-Entwicklungsstrategie ist das „fehlende Glied“ bei der Verbesserung der Leistung (Edvardsson et al. 2013; Edvardsson und Olsson 1996).

Das strategiebasierte Vorgehensmodell zur Dienstleistungsentwicklung der Studienrichtung IT und Wirtschaftsinformatik an der FH CAMPUS 02 kann für Softwarefirmen besonders praktikabel sein, weil diese verlässlichen Werkzeuge benötigen, um Richtungsentscheidungen bezüglich ihren Software-Lösungen zu treffen.

Die Grund-Idee für die Masterarbeit ist eine Kombination zwischen dem strategiebasierten Dienstleistungsentwicklungsprozess und einem Softwareentwicklungsprozess. Ziel dieser Arbeit ist es, die ideale Kombination zwischen Service und Software Engineering zu finden, die es Softwareunternehmen ermöglicht, strategische Aspekte in die Planung und Implementierung von Softwarelösungen zu integrieren.

Anhand dieses kombinierten Vorgehensmodells sollen Softwarelösungen analysiert werden, und die Ergebnisse sollen der Geschäftsführung als Grundlage für den Entscheidungsprozess zur Verfügung gestellt werden können.

Um die Anwendbarkeit eines derart kombinierten Vorgehensmodells zu untersuchen, wird der strategiebasierte Prozess in der Softwareentwicklungsabteilung der Firma Bartelt durchgeführt.

Diese Masterarbeit will einen Beitrag dazu leisten, die Nützlichkeit dieses Prozesses zu untersuchen.

1.1 Zielsetzung

Das allgemeine Ziel ist es, eine passende Kombination von Service-Engineering und Software-Engineering für die Entwicklung von Softwarelösungen zu finden, um ein strategiebasiertes Vorgehensmodell zu erarbeiten.

Es soll festgestellt werden, ob das Vorgehensmodell mit eigenen Mitteln in einem Unternehmen sinnvoll anwendbar ist.

Es soll für einzelne Software-Lösungen erkennbar sein, ob die strategische Richtung stimmt oder wie sich diese ändern sollte.

1.2 Forschungsfrage

Forschungsfrage:

> **Welche Kombination von Service Engineering und Software Engineering ist geeignet, um Softwarelösungen mit Mehrwert für Kunden und Softwareunternehmen zu generieren?**

1.3 Hypothesen

Hypothese 1: Eine Kombination von Service- und Software-Engineering ist möglich, um ein strategiebasiertes Softwareentwicklungsmodell zu formulieren.

Hypothese 2: Es ist möglich, diesen Prozess eigenständig durchzuführen, d.h. selbst zu steuern.

Hypothese 3: Die Ergebnisse des strategiebasierten Vorgehensmodells ermöglicht den Firmen, Grundlagen für eine fundierte Entscheidung zu erhalten, um die Strategie einer oder mehrerer Software-Lösungen zu gestalten.

1.4 Methodologie und Aufbau der Arbeit

Um das Ziel dieser Arbeit zu erreichen und die Forschungsfrage zu beantworten, taucht die vorliegende Arbeit zunächst durch eine Literaturrecherche tief in die Bereiche des Software- und Service-Engineering ein.

Basierend auf den Erkenntnissen aus der Literaturrecherche wird ein Modell zur strategiebasierten Entwicklung von Software entworfen.

Als nächstes wird dieses theoretisch erarbeitete Modell in der Praxis getestet und analysiert um die Forschungsfrage zu beantworten, die Hypothesen zu bewerten, und schließlich um Schlussfolgerungen anzuziehen.

Diese Arbeit wurde strukturiert, um zunächst alle relevanten theoretischen Konzepte zu erhalten und dann ein Modell aufzubauen, das in der Praxis in einem Unternehmen getestet und evaluiert werden kann.

Kapitel 2 analysiert, was Software ist und wie Software Engineering funktioniert. In diesem Kapitel wird der Softwareentwicklungsprozess dargestellt, der Aspekte wie Agilität, Iterationen und Inkremente integriert. Außerdem wird der von Pressman und Maxim (2020) empfohlene Prozess beschrieben.

In Kapitel 3 konzentriert sich die vorliegende Arbeit auf die Entwicklung von Services, mit besonderem Augenmerk auf E-Services und wie deren Engineering funktioniert, inklusive einer Beschreibung des strategiebasierten Vorgehensmodells zur Dienstleistungsentwicklung der Studienrichtung IT und Wirtschaftsinformatik der FH CAMPUS 02.

Kapitel 4 bildet den Kern dieser Arbeit und versucht, aufbauend auf den Erkenntnissen aus den Kapiteln 2 und 3, ein strategiebasiertes Vorgehensmodell zur Softwareentwicklung zu entwerfen, das als Grundlage für den praktischen Teil in Kapitel 5 dienen soll.

Kapitel 6 zeigt die in der Praxis erzielten Ergebnisse und versucht, sie in Bezug auf die in dieser Arbeit aufgestellten Hypothesen zu analysieren.

Die Kapitel 7 und 8 bieten die reflektierten Schlussfolgerungen und Einschränkungen sowie Vorschläge für zukünftige Forschung.

2 SOFTWARE ENGINEERING

Wir leben in einer Zeit, in der sich fast alles um die Digitalisierung herum entwickelt, sowohl in der Wirtschaft als auch im privaten Bereich. Diese Zeit zeichnet sich dadurch aus, dass die digitale Technologie immer stärker in die Geschäftsmodelle integriert wird (Broy 2018) (Marquardt 2017).

Wenn Unternehmen erfolgreich *Customer Experiences* und *Customer Journeys* generieren wollen, müssen sie die Potenziale der digitalen Technologie mit ihren Geschäftsmodellen und ihren Möglichkeiten am Markt kombinieren. Dabei spielt Software eine entscheidende Rolle, da sie die Funktionalität digitaler Systeme bestimmt und in vielen Branchen der Schlüssel für effektive Geschäftsprozesse ist.

Im Software Engineering geht es darum, die Charakteristik von Software-Prozessen zu verstehen, geeignete Architekturen von Software-Systemen zu finden und die wesentlichen und wertschöpfenden Aktivitäten in der Software-Entwicklung zu identifizieren (Gruhn und Striemer 2018).

Laut Pressman und Maxim (2020) ist Software das Schlüsselelement in der Entwicklung computergestützter Systeme und Produkte und eine der wichtigsten Technologien auf der Weltbühne. In den letzten 60 Jahren hat sich Software von einem spezialisierten Problemlösungs- und Informationsanalysewerkzeug zu einer eigenständigen Industrie entwickelt. Dennoch haben wir immer noch Schwierigkeiten, qualitativ hochwertige Software rechtzeitig und innerhalb des Budgets zu entwickeln.

Zunächst ist es wichtig, Software zu charakterisieren.

2.1 Was ist Software?

Software liefert das wichtigste Produkt unserer Zeit - Informationen. Sie wandelt persönliche Daten (z. B. die Finanztransaktionen einer Person) um, so dass die Daten in einem lokalen Kontext nützlich sind; sie verwaltet Geschäftsinformationen, um die Wettbewerbsfähigkeit zu verbessern; sie bietet ein Tor zu weltweiten Informationsnetzwerken (z. B. das Internet) und stellt die Mittel zur Beschaffung von Informationen in all ihren Formen bereit.

Heutzutage nimmt Software eine Doppelrolle ein (Pressman und Maxim 2020):

- Produkt – stellt das Datenverarbeitungspotenzial bereit, das in der Computerhardware oder, weiter gefasst, in einem Netzwerk von Computern enthalten ist, die über lokale Hardware zugänglich sind. Unabhängig davon, ob sie sich in einem mobilen Gerät, auf dem Desktop, in der Cloud oder in einem Großrechner oder einer autonomen Maschine befindet, ist Software ein Informationsumwandler - sie produziert, verwaltet, erfasst, verändert, zeigt oder überträgt Informationen, die so einfach wie ein einzelnes Bit oder so komplex wie eine Augmented-Reality-Darstellung sein können, die aus Daten Dutzender unabhängiger Quellen abgeleitet und dann der realen Welt überlagert wird.

- Mittel zur Bereitstellung eines Produkts - Als Vehikel zur Bereitstellung eines Produkts dient Software als Grundlage für die Steuerung des Computers (Betriebssysteme), die Kommunikation von Informationen (Netzwerke) und die Erstellung und Steuerung anderer Programme (Softwaretools und -umgebungen).

Pressman und Maxim (2020) beschreiben Software als:

(1) Anweisungen (Computerprogramme), die bei ihrer Ausführung die gewünschten Merkmale, Funktionen und Leistungen bereitstellen;

(2) Datenstrukturen, die es den Programmen ermöglichen, Informationen angemessen zu bearbeiten; und

(3) beschreibende Informationen in gedruckter und virtueller Form, die den Betrieb und die Verwendung der Programme beschreiben.

Software kann man nicht anfassen, es ist eine Kodifizierung einer riesigen Menge von Verhaltensweisen (Wenn dies geschieht, dann sollte jenes geschehen und so weiter). Wenn man sich Software ansieht, während sie geschrieben wird, erscheint sie als eine Abfolge von Anweisungen. Die Komplexität eines Programms hängt jedoch nicht nur von den Anweisungen ab, sondern auch von den Wechselwirkungen zwischen den Anweisungen (Stepanek 2005).

Software scheint dadurch gekennzeichnet zu sein, dass sie nicht physisch und nicht greifbar ist und dennoch durch potenziell große und komplexe Gruppen von Beschränkungen strukturiert ist, die das, was anscheinend häufig geändert und weiterentwickelt werden muss, erschweren. Während Software selbst nicht physisch und nicht greifbar ist, besteht ein Hauptziel für Instanzen des Typs Software darin, dass sie eine oder mehrere Komponenten enthalten, deren Ausführung die Verwaltung und Kontrolle von greifbaren Entitäten bewirkt. Computersoftware ist dadurch gekennzeichnet, dass sie für die Ausführung auf einem Computer bestimmt ist (Osterweil 2018).

Veränderung ist für Software etwas ganz Natürliches: Kein anderes menschliches Artefakt hat diese Eigenschaft. Ihre immaterielle Natur und das Fehlen physikalischer Zwänge machen sie vollkommen formbar: jede Änderung ist prinzipiell möglich. Durch einfache textuelle Operationen können Software-Ingenieure Funktionalitäten hinzufügen, löschen oder verändern und ihre Eigenschaften (z. B. ihre Leistung) verbessern. Technisch gesehen können sowohl die funktionalen als auch die nicht-funktionalen Eigenschaften geändert werden. Die Fähigkeit, extrem leistungsfähige Funktionalitäten auf äußerst flexible und veränderbare Weise bereitzustellen, war der Schlüsselfaktor, der zur heutigen, von Software dominierten Welt geführt hat (Ghezzi 2018).

Pressman und Maxim (2020) unterscheiden sieben große Kategorien von Computersoftware:

(1) Systemsoftware. Eine Sammlung von Programmen, die geschrieben wurden, um andere Programme zu bedienen. Manche Systemsoftware (z. B. Compiler, Editoren und Dateiverwaltungsprogramme) verarbeitet komplexe, aber bestimmte Informationsstrukturen. Andere Systemanwendungen (z. B. Betriebssystemkomponenten, Treiber, Netzwerksoftware, Telekommunikationsprozessoren) verarbeiten weitgehend unbestimmte Daten.

(2) Anwendungssoftware. Eigenständige Programme, die eine bestimmte Geschäftsanforderung erfüllen. Anwendungen in diesem Bereich verarbeiten geschäftliche oder technische Daten in einer Weise, die den Geschäftsbetrieb oder die Entscheidungsfindung von Management und Technik erleichtert.

(3) Technische/wissenschaftliche Software. Eine breite Palette von Programmen zur "Zahlenverarbeitung" oder Datenwissenschaft, die von der Astronomie bis zur Vulkanologie, von der Belastungsanalyse von Kraftfahrzeugen bis zur Orbitaldynamik, von computergestütztem Design bis zu Konsumgewohnheiten und von der Genanalyse bis zur Meteorologie reichen.

(4) Eingebettete Software. Sie befindet sich in einem Produkt oder System und wird zur Implementierung und Steuerung von Merkmalen und Funktionen für den Endbenutzer und das System selbst verwendet. Eingebettete Software kann eng eingegrenzte Funktionen ausführen (z. B. Tastatursteuerung für einen Mikrowellenherd) oder bedeutende Funktionen und Steuerungsmöglichkeiten bieten (z. B. digitale Funktionen in einem Auto wie Kraftstoffsteuerung, Anzeigen auf dem Armaturenbrett und Bremssysteme).

(5) Produktliniensoftware. Besteht aus wiederverwendbaren Komponenten und ist so konzipiert, dass sie spezifische Funktionen für die Nutzung durch viele verschiedene Kunden bietet. Sie kann sich auf einen abgegrenzten Markt konzentrieren (z. B. Produkte zur Lagerverwaltung) oder versuchen, den Massenmarkt anzusprechen.

(6) Web-/Mobilanwendungen. Diese netzzentrierte Softwarekategorie umfasst ein breites Spektrum von Anwendungen und beinhaltet browserbasierte Anwendungen, Cloud-Computing, servicebasiertes Computing und Software, die auf mobilen Geräten installiert ist.

(7) Software mit künstlicher Intelligenz. Verwendet Heuristiken, um komplexe Probleme zu lösen, die sich nicht mit normalen Berechnungen oder einfachen Analysen lösen lassen. Zu den Anwendungen in diesem Bereich gehören Robotik, Entscheidungsfindungssysteme, Mustererkennung (Bild und Stimme), maschinelles Lernen, Theorem Beweise und Spiele.

Im Großen und Ganzen ist Software die Gesamtheit der Programme, Inhalte (Daten) und anderen Arbeitsprodukte, die sie unterstützen. Für den Benutzer ist Software ein Werkzeug das die Welt irgendwie besser macht und seine Arbeit erleichtert.

Wenn Computersoftware erfolgreich ist – wenn sie die Bedürfnisse der Benutzer erfüllt, wenn sie über einen langen Zeitraum fehlerfrei funktioniert, wenn sie leicht zu modifizieren und noch einfacher zu verwenden ist – kann sie die Dinge zum Besseren verändern und tut es auch. Aber wenn Software versagt – wenn ihre Benutzer unzufrieden sind, wenn sie fehleranfällig ist, wenn sie schwer zu ändern und noch schwieriger zu bedienen ist – können schlimme Dinge passieren. Wir alle wollen Software entwickeln, die die Dinge besser macht und schlechte Effekte vermeidet, die im Schatten gescheiterter Bemühungen lauern. Um erfolgreich zu sein, brauchen wir Disziplin beim Entwerfen und Bauen von Software. Wir brauchen einen Engineering-Ansatz (Pressman und Maxim 2020).

2.2 Das Engineering von Software

Pressman und Maxim (2020) betonnen dass einige einfache Realitäten anerkannt werden müssen, um Software so zu entwickeln, dass sie den Herausforderungen des 21. Jahrhunderts gewachsen ist:

- Software ist in praktisch jedem Aspekt unseres Lebens tief verankert. Die Zahl der Menschen, die ein Interesse an den Merkmalen und Funktionen einer bestimmten Anwendung haben, ist dramatisch gestiegen. Bevor eine Softwarelösung entwickelt wird, sollte man sich bemühen, das Problem zu verstehen.
- Die Anforderungen an die Informationstechnologie, die von Privatpersonen, Unternehmen und Behörden gestellt werden, werden von Jahr zu Jahr anspruchsvoller. Große Teams von Mitarbeitern erstellen heute Computerprogramme. Komplexe Software, die früher in einer vorhersehbaren, in sich geschlossenen Computerumgebung implementiert wurde, ist heute oftmals eingebettet, von Unterhaltungselektronik über medizinische Geräte bis hin zu autonomen Fahrzeugen. Design ist zu einer zentralen Tätigkeit geworden.
- Privatpersonen, Unternehmen und Regierungen verlassen sich bei strategischen und taktischen Entscheidungen sowie bei der täglichen Arbeit und Kontrolle zunehmend auf Software. Wenn die Software versagt, kann das Auswirkungen haben, die für Menschen und Unternehmen von kleinen Unannehmlichkeiten bis hin zu katastrophalen Folgen reichen. Software sollte eine hohe Qualität aufweisen.
- Je größer der wahrgenommene Wert einer bestimmten Anwendung ist, desto wahrscheinlicher ist es, dass auch ihre Nutzerbasis und ihre Langlebigkeit zunehmen. Mit zunehmender Benutzerzahl und Nutzungsdauer steigt auch der Bedarf an Anpassungen und Verbesserungen. Software sollte wartbar sein.

Diese einfachen Tatsachen lassen nur einen Schluss zu: Ein Engineering-Ansatz sollte zur Erstellung von Software in all ihren Formen und Anwendungsbereichen gewählt werden.

Software-Engineering umfasst einen Prozess, eine Sammlung von Methoden (Praxis) und eine Reihe von Werkzeugen, die es Fachleuten ermöglichen, qualitativ hochwertige Computersoftware zu erstellen.

Die IEEE hat die folgende Definition für Software Engineering entwickelt (IEEE Computer Society):

> „Software Engineering: Die Anwendung eines systematischen, disziplinierten und quantifizierbaren Ansatzes für die Entwicklung, den Betrieb und die Wartung von Software; das heißt, die Anwendung von Engineering auf Software.“

Aber wie soll nun dieser systematische, disziplinierte und quantifizierbare Ansatz für die Entwicklung ausschauen? Ein Ansatz kann für ein Softwareteam gut funktionieren, für ein anderes aber eine Belastung sein.

Laut Pressman und Maxim (2020) brauchen wir nicht nur Disziplin, sondern auch Anpassungsfähigkeit und Agilität. Software Engineering ist eine vielschichtige Technologie:

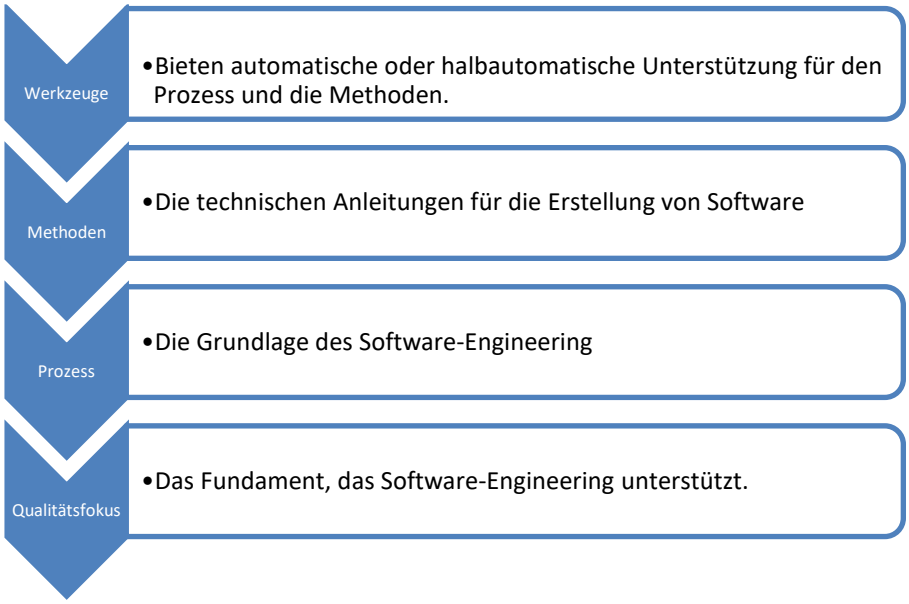


Abbildung 1: Software-Engineering-Schichten nach (Pressman und Maxim 2020)

Qualitätsfokus: Das Fundament, welches das Software-Engineering unterstützt, ist der Fokus auf Qualität. Jeder Engineering-Ansatz muss auf einer organisatorischen Verpflichtung zur Qualität beruhen. Philosophien wie Total Quality Management oder Six Sigma fördern eine Kultur der kontinuierlichen Prozessverbesserung. Es ist diese Kultur, die letztlich zu effektiveren Ansätzen für die Softwareentwicklung führt.

Prozess: Die Prozessebene ist die Grundlage für die Softwareentwicklung. Der Software-Engineering-Prozess ist der Klebstoff, der die Technologieschichten zusammenhält und eine rationelle und zeitgerechte Entwicklung von Computersoftware ermöglicht. Der Prozess definiert einen Rahmen, der für die effektive Bereitstellung von Software-Engineering-Technologie geschaffen werden muss. Der Softwareprozess bildet die Grundlage für die Steuerung von Softwareprojekten und legt den Kontext fest, in dem technische Methoden angewendet, Arbeitsprodukte (Modelle, Dokumente, Daten, Berichte, Formulare usw.) erstellt, Meilensteine festgelegt, die Qualität sichergestellt und Änderungen ordnungsgemäß verwaltet werden.

Methoden: Software-Engineering-Methoden liefern die technischen Anleitungen für die Erstellung von Software. Die Methoden umfassen ein breites Spektrum von Aufgaben, darunter Kommunikation, Anforderungsanalyse, Entwurfsmodellierung, Programmerstellung, Testen und Support. Software-Engineering-Methoden beruhen auf einer Reihe von Grundprinzipien, die jeden Bereich der Technologie regeln und Modellierungsaktivitäten und andere beschreibende Techniken umfassen.

Werkzeuge: Software-Engineering-Tools bieten automatische oder halbautomatische Unterstützung für den Prozess und die Methoden. Wenn die Werkzeuge so integriert werden, dass die von einem Werkzeug erzeugten Informationen von einem anderen verwendet werden können, entsteht ein System zur Unterstützung der Softwareentwicklung, das als computergestütztes Software-Engineering bezeichnet wird.

Zusammenfassend lässt sich sagen, dass Software-Engineering Prozesse, Methoden und Werkzeuge umfasst, die es ermöglichen, komplexe computergestützte Systeme zeitnah und in hoher Qualität zu erstellen.

Relevant für unser Thema ist insbesondere der Software-Engineering-Prozess.

2.3 Software-Engineering-Prozess

Ein allgemeines Prozessmodell für Software-Engineering umfasst fünf Rahmenaktivitäten (Kommunikation, Planung, Modellierung, Konstruktion und Bereitstellung), die auf alle Softwareprojekte anwendbar sind. Die Rahmenaktivitäten des Softwareentwicklungsprozesses werden durch eine Reihe von übergeordneten Aktivitäten ergänzt. Im Allgemeinen werden die übergeordneten Aktivitäten während des gesamten Softwareprojekts angewendet und helfen dem Softwareteam, Fortschritt, Qualität, Änderungen und Risiken zu verwalten und zu kontrollieren (Pressman und Maxim 2020).

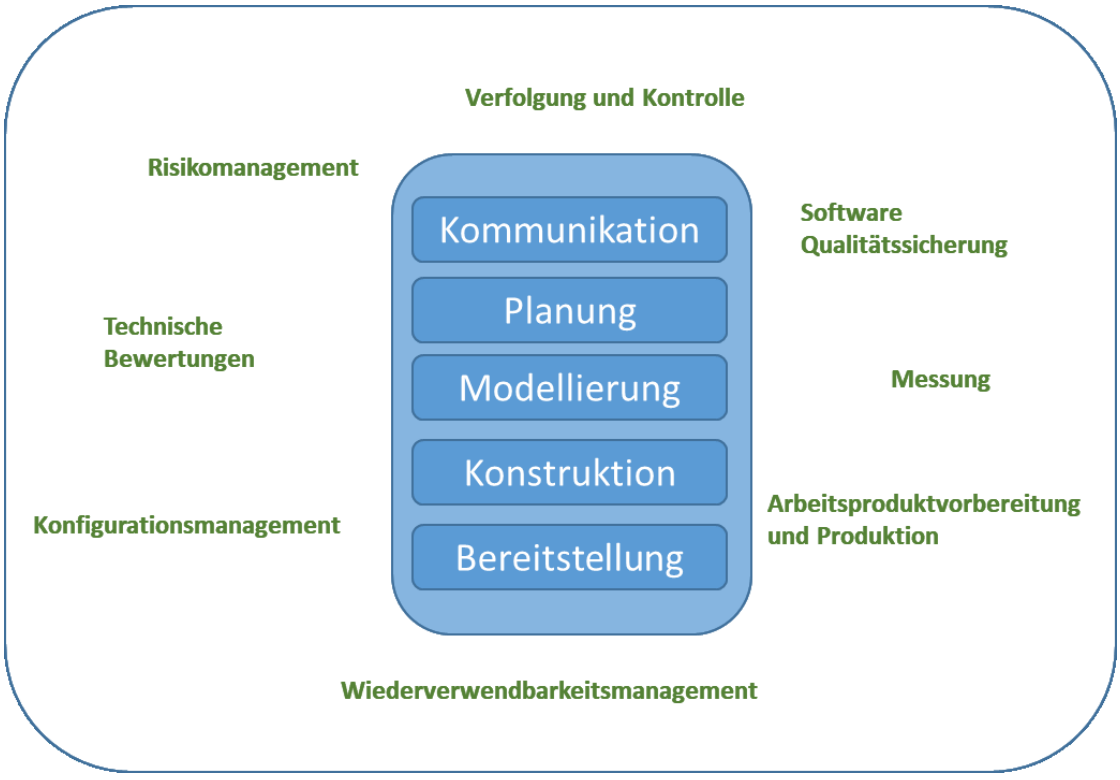


Abbildung 2: Allgemeiner Prozess-Framework für Software-Engineering

Kommunikation: bevor mit der technischen Arbeit begonnen werden kann, ist es von entscheidender Bedeutung, mit dem Kunden (und anderen Interessengruppen) zu kommunizieren und zusammenzuarbeiten. Dabei geht es darum, die Ziele der Interessengruppen für das Projekt zu verstehen und Anforderungen zu sammeln, die bei der Definition der Softwaremerkmale und -funktionen helfen.

Planung: jede komplizierte Reise kann vereinfacht werden, wenn es eine Karte gibt. Ein Softwareprojekt ist eine komplizierte Reise, und die Planungsaktivität erstellt eine „Landkarte“, die das Team auf seinem Weg begleitet. Die Karte, der so genannte Softwareprojektplan, definiert

die Softwareentwicklungsarbeit, indem er die durchzuführenden technischen Aufgaben, die wahrscheinlichen Risiken, die erforderlichen Ressourcen, die zu erstellenden Arbeitsprodukte und einen Arbeitsplan beschreibt.

Modellierung: ob Sie nun Landschaftsgärtner, Brückenbauer, Luftfahrtingenieur, Schreiner oder Architekt sind, Sie arbeiten jeden Tag mit Modellen. Sie erstellen eine "Skizze" der Sache, damit Sie das Gesamtbild verstehen - wie es architektonisch aussehen wird, wie die einzelnen Teile zusammenpassen und viele andere Merkmale. Falls erforderlich, verfeinern Sie die Skizze immer weiter, um das Problem und seine Lösung besser zu verstehen. Ein Software-Ingenieur tut dasselbe, indem er Modelle erstellt, um die Software-Anforderungen und den Entwurf, der diese Anforderungen erfüllen soll, besser zu verstehen.

Konstruktion: was entworfen wurde, muss erst gebaut werden. Diese Tätigkeit umfasst die (manuelle oder automatisierte) Codegenerierung sowie die Tests, die erforderlich sind, um Fehler im Code aufzudecken.

Deployment: die Software (als vollständige Einheit oder als teilweise fertiges Inkrement) wird an den Kunden geliefert, der das gelieferte Produkt bewertet und auf der Grundlage dieser Bewertung Feedback gibt.

Diese fünf generischen Rahmenaktivitäten können bei der Entwicklung kleiner, einfacher Programme, bei der Erstellung von Webanwendungen und bei der Entwicklung großer, komplexer computergestützter Systeme eingesetzt werden. Die Details des Softwareprozesses sind in jedem Fall sehr unterschiedlich, aber die Rahmenaktivitäten bleiben die gleichen, d.h., dass der Softwareentwicklungsprozess keine starre Vorschrift ist, vielmehr sollte er agil und anpassungsfähig sein (an das Problem, das Projekt, das Team und die Organisationskultur).

Software-Prozesse werden immer zuverlässiger und effektiver, aber sie unterscheiden sich in ihrer Art von Produktionsprozessen (Gruhn und Striemer 2018).

Die verschiedenen Prozessmodelle können durch einen so genannten Prozessfluss beschrieben werden. Dieser beschreibt, wie die Rahmenaktivitäten und die Aktionen und Aufgaben, die innerhalb jeder Rahmenaktivität auftreten, in Bezug auf Reihenfolge und Zeit organisiert sind.

Die verschiedenen Prozessflüsse sind:

- **Linear** – jede der fünf Rahmenaktivitäten werden nacheinander ausgeführt, beginnend mit der Kommunikation und endend mit der Deployment.
- **Iterativ** – eine oder mehrere der Aktivitäten werden wiederholt, bevor mit der nächsten Aktivität fortzufahren.
- **Evolutionär** – die Aktivitäten werden in einer „zirkulären“ Weise ausgeführt. Jeder Kreislauf durch die fünf Aktivitäten führt zu einer vollständigeren Version der Software
- **Parallel** – eine oder mehrere Aktivitäten werden parallel zu anderen Aktivitäten ausgeführt.

Seit vielen Jahren werden präskriptive Prozessmodelle eingesetzt, um Ordnung und Struktur in die Softwareentwicklung zu bringen (Pressman und Maxim 2020). Jedes dieser Modelle schlägt

einen etwas anderen Prozessablauf vor, aber alle führen dieselbe Reihe von allgemeinen Rahmenaktivitäten aus: Kommunikation, Planung, Modellierung, Konstruktion und Deployment.

Sequentielle Prozessmodelle, wie das Wasserfallmodell, gehören zu den ältesten Paradigmen der Softwareentwicklung. Sie suggerieren einen linearen Prozessablauf, der mit den modernen Gegebenheiten (z. B. ständige Veränderungen, sich entwickelnde Systeme, enge Zeitvorgaben) in der Softwarewelt oft nicht vereinbar ist. Sie sind jedoch in Situationen anwendbar, in denen die Anforderungen gut definiert und stabil sind.

Inkrementelle Prozessmodelle sind von Natur aus iterativ und erzeugen recht schnell funktionierende Softwareversionen.

Evolutionäre Prozessmodelle erkennen den iterativen, inkrementellen Charakter der meisten Softwareentwicklungsprojekte an und sind so konzipiert, dass sie Änderungen berücksichtigen.

Evolutionsmodelle, wie z. B. das Prototyping und das Spiralmodell, erzeugen schnell inkrementelle Arbeitsprodukte (oder Arbeitsversionen der Software). Diese Modelle können auf alle Softwareentwicklungsaktivitäten angewandt werden – von der Konzeptentwicklung bis zur langfristigen Systemwartung.

Der „Unified Process“ ist ein anwendungsfallorientierter, architekturzentrierter, iterativer und inkrementeller Softwareprozess, der als Rahmen für UML-Methoden und -Tools konzipiert wurde.

Die traditionellen Softwareentwicklungsprozesse waren hauptsächlich phasenweise und starr geplant, was im Idealfall zu robusten Prozessen führen sollte, die ein Überdenken und Ändern früherer Designentscheidungen überflüssig machen würde (Ghezzi 2018).

Es hat sich jedoch gezeigt, dass Änderungen in den meisten praktischen Fällen unvermeidlich sind. Die Anforderungen für eine bestimmte Anwendung sind zu Beginn einer neuen Entwicklung oft nur vage bekannt. Mit fortschreitender Entwicklung werden sie immer besser bekannt, und es fließen Rückmeldungen von Kunden und aus dem Betrieb ein. Das Wissen über den betrieblichen Kontext, in den die Software eingebettet sein wird, ist zum Zeitpunkt der Entwicklung oft ungewiss. Und selbst wenn die Anforderungen und der Kontext in der Anfangsphase bekannt sind, können sie sich ändern, z. B. aufgrund von Änderungen im breiteren Unternehmenskontext, in den die Anwendung eingebettet ist.

Turbulenzen bei den Anforderungen führen dazu, dass alternative Software-Prozessmodelle entwickelt werden müssen, die Veränderungen auf natürliche Weise in den Prozess einbeziehen können, um eine iterative und inkrementelle Entwicklung zu unterstützen und die Softwareprodukte besser an die sich verändernden Anforderungen anzupassen. Der Begriff "agile Softwareentwicklung" hat sich eingebürgert, um eine Reihe von Bemühungen der Industrie zu charakterisieren, die darauf abzielen, die Kosten von Anforderungsänderungen durch mehrere kurze Entwicklungszyklen anstelle langer monolithischer Zyklen zu senken.

2.4 Agilität in Software Engineering Prozess

Im Jahr 2001 unterzeichnete eine Gruppe bekannter Softwareentwickler, Autoren und Berater das "Manifest für agile Softwareentwicklung", in dem sie sich dafür aussprachen, dass Individuen und Interaktionen wichtiger sind als Prozesse und Werkzeuge, funktionierende Software wichtiger als umfassende Dokumentation, Zusammenarbeit mit dem Kunden wichtiger als Vertragsverhandlungen, und das Reagieren auf Veränderungen wichtiger als das Befolgen eines Plans.

Agile Softwaretechnik ist eine Kombination aus einer Philosophie und einer Reihe von Entwicklungsrichtlinien. Die Philosophie fördert die Kundenzufriedenheit, die frühzeitige inkrementelle Bereitstellung von Software, kleine, hoch motivierte Projektteams, informelle Methoden, minimale Software-Engineering-Arbeitsprodukte und die allgemeine Einfachheit der Entwicklung. In den Entwicklungsrichtlinien wird der Schwerpunkt auf die Auslieferung und nicht auf die Analyse und den Entwurf gelegt (Pressman und Maxim 2020).

In der modernen Wirtschaft ist es oft schwierig oder unmöglich vorherzusagen, wie sich ein computergestütztes System im Laufe der Zeit entwickeln wird. Die Marktbedingungen ändern sich schnell, die Bedürfnisse der Nutzer entwickeln sich, und neue Wettbewerbsbedrohungen tauchen ohne Vorwarnung auf. In vielen Situationen ist man nicht in der Lage, die Anforderungen vor Beginn des Projekts vollständig zu definieren. Es ist also nötig, agil genug zu sein, um auf ein fluides Geschäftsumfeld zu reagieren.

Pressman und Maxim (2020) bezeichnen Agile Entwicklung als „Software-Engineering Lite“. Die grundlegenden Rahmenaktivitäten – Kommunikation, Planung, Modellierung, Konstruktion und Bereitstellung – bleiben bestehen. Sie werden jedoch zu einem minimalen Aufgabensatz umgewandelt, der das Projektteam zur Konstruktion und Bereitstellung antreibt.

Auch nach SWEBOK -Software Engineering Body of Knowledge (IEEE Computer Society), gelten agile Methoden als leichtgewichtige Methode, da sie sich durch kurze, iterative Entwicklungszyklen, selbstorganisierende Teams, einfachere Entwürfe, Code-Refactoring, testgetriebene Entwicklung, häufige Einbindung des Kunden und die Betonung der Schaffung eines nachweislich funktionierenden Produkts in jedem Entwicklungszyklus auszeichnen.

Agilität kann auf jeden Softwareprozess angewendet werden. Um dies zu erreichen, muss der Prozess jedoch so gestaltet sein, dass das Projektteam Aufgaben anpassen und straffen kann, die Planung so durchgeführt wird, dass die Fluidität eines agilen Entwicklungsansatzes verstanden wird, alle Arbeitsprodukte mit Ausnahme der wichtigsten eliminiert und schlank gehalten werden und eine inkrementelle Lieferstrategie betont wird, die dem Kunden so schnell wie möglich funktionierende Software für den Produkttyp und die Betriebsumgebung liefert.

Agile Methoden unterstützen die Softwareentwicklung in einer iterativen und inkrementellen Weise, um kontinuierliche Änderungen zu ermöglichen. In den Anfangsphasen eines neuen Projekts müssen die Anforderungen schrittweise kalibriert und die Erkundung verschiedener Designalternativen unterstützt werden. Sobald eine Softwareversion einsatzbereit ist und läuft, werden bei gleichzeitigen Entwicklungsaktivitäten neue Versionen erstellt, die z. B. die Erfüllung nichtfunktionaler Anforderungen verbessern, zusätzliche Anforderungen erfüllen oder sich mit

Umgebungsänderungen befassen, die nicht vorhergesehen wurden und von den selbstanpassenden Strategien der aktuellen Betriebsversionen nicht unterstützt werden (Ghezzi 2018).

Bei einem agilen Ansatz wird die Softwareentwicklung durch häufige Iterationen strukturiert: Eine unvollständige Spezifikation wird zu einer vollständigen Spezifikation, sobald die unbekannten oder unsicheren Aspekte bekannt werden. Darüber hinaus können Teile des Systems in einer bestimmten Phase bewusst unvollständig bleiben und ihre Fertigstellung wird auf eine spätere Phase verschoben.

2.4.1 Agile Softwareentwicklungsmethoden

In der Literatur finden sich viele agile Methoden (Pressman und Maxim 2020) (IEEE Computer Society); Im Folgenden sind einige der populäreren Ansätze kurz besprochen:

RAD: Rapid Software Development-Methoden werden hauptsächlich bei der Entwicklung datenintensiver Geschäftssystemanwendungen eingesetzt. Die RAD-Methode wird durch spezielle Datenbankentwicklungswerkzeuge ermöglicht, die von Softwareingenieuren zur schnellen Entwicklung, Prüfung und Bereitstellung neuer oder geänderter Geschäftsanwendungen eingesetzt werden.

XP: Extreme Programming; Dieser Ansatz verwendet Stories oder Szenarien für die Anforderungen, entwickelt zuerst Tests, hat eine direkte Kundenbeteiligung im Team (in der Regel die Definition von Akzeptanztests), verwendet Pair Programming und sieht kontinuierliches Code Refactoring vor. Die Stories werden in Aufgaben zerlegt, nach Prioritäten geordnet, geschätzt, entwickelt und getestet. Jedes Inkrement der Software wird mit automatisierten und manuellen Tests getestet; ein Inkrement kann häufig, z. B. alle paar Wochen, veröffentlicht werden.

Scrum: der Name leitet sich von einer Aktivität ab, die während eines Rugbyspiels stattfindet, ist eine sehr populäre agile Softwareentwicklungsmethode, die von Jeff Sutherland und seinem Entwicklungsteam in den frühen 1990er Jahren konzipiert wurde. Die Weiterentwicklung der Scrum-Methode wurde von Schwaber und Beedle durchgeführt.

Dieser agile Ansatz ist projektmanagementfreundlicher als die anderen. Der Scrum Master verwaltet die Aktivitäten innerhalb des Projektinkrements; jedes Inkrement wird als Sprint bezeichnet und dauert nicht länger als 30 Tage. Es wird eine PBI-Liste (Product Backlog Item) erstellt, anhand derer die Aufgaben identifiziert, definiert, nach Prioritäten geordnet und geschätzt werden. In jedem Inkrement wird eine Arbeitsversion der Software getestet und freigegeben. Tägliche Scrum-Meetings stellen sicher, dass die Arbeit nach Plan verwaltet wird.

Kanban: Der Schwerpunkt von Kanban liegt auf dem Änderungsmanagement und der Erbringung von Dienstleistungen. Das Änderungsmanagement definiert den Prozess, durch den eine gewünschte Änderung in ein softwarebasiertes System integriert wird. Die Erbringung von Dienstleistungen wird gefördert, indem man sich auf das Verständnis der Kundenbedürfnisse und -erwartungen konzentriert. Die Teammitglieder verwalten die Arbeit und haben die Freiheit, sich

selbst zu organisieren, um sie zu erledigen. Die Richtlinien werden nach Bedarf weiterentwickelt, um die Ergebnisse zu verbessern.

Kanban hat seinen Ursprung bei Toyota und wurde von David Anderson an die Softwareentwicklung angepasst.

DevOps: wurde von Patrick DeBois entwickelt, um Entwicklung und Betrieb miteinander zu verbinden. DevOps versucht, die Grundsätze der agilen und schlanken Entwicklung auf die gesamte Software-Lieferkette anzuwenden. Der DevOps-Ansatz umfasst mehrere Phasen, die sich kontinuierlich wiederholen, bis das gewünschte Produkt vorliegt:

- *Kontinuierliche Entwicklung*. Die Software wird in mehrere Sprints aufgeteilt und entwickelt, wobei die Inkremente an die Qualitätssicherungsmitglieder des Entwicklungsteams zum Testen übergeben werden.
- *Kontinuierliche Tests*. Automatisierte Testwerkzeuge helfen den Teammitgliedern, mehrere Code-Inkremente gleichzeitig zu testen, um sicherzustellen, dass sie vor der Integration frei von Fehlern sind.
- *Kontinuierliche Integration*. Die Codestücke mit den neuen Funktionen werden dem vorhandenen Code und der Laufzeitumgebung hinzugefügt und nach der Bereitstellung auf Fehlerfreiheit geprüft.
- *Kontinuierliches Deployment*. In dieser Phase wird der integrierte Code in der Produktionsumgebung bereitgestellt (installiert), die mehrere Standorte weltweit umfassen kann, die für den Empfang der neuen Funktionen vorbereitet werden müssen.
- *Kontinuierliche Überwachung*. Betriebsmitarbeiter, die Mitglieder des Entwicklungsteams sind, helfen bei der Verbesserung der Softwarequalität, indem sie die Leistung in der Produktionsumgebung überwachen und proaktiv nach möglichen Problemen suchen, bevor die Benutzer sie finden.

DevOps verbessert das Kundenerlebnis, indem es schnell auf Änderungen der Bedürfnisse oder Wünsche der Kunden reagiert. Dies kann die Markentreue erhöhen und den Marktanteil steigern.

In der Literatur und in der Praxis gibt es viele weitere Varianten agiler Methoden. Es wird immer einen Platz für schwergewichtige, planbasierte Software-Engineering-Methoden geben, aber häufig sind agile Methoden passend. Es gibt neue Methoden, die aus Kombinationen von agilen und planbasierten Methoden hervorgegangen sind, bei denen Praktiker neue Methoden definieren, die ein Gleichgewicht zwischen den erforderlichen Merkmalen von schwergewichtigen und leichtgewichtigen Methoden herstellen und in erster Linie auf den vorherrschenden Geschäftsanforderungen des Unternehmens basieren. Diese geschäftlichen Anforderungen, wie sie typischerweise von einigen der Projektbeteiligten vertreten werden, sollten die Entscheidung für eine Software-Engineering-Methode gegenüber einer anderen oder für die Konstruktion einer neuen Methode aus den besten Merkmalen einer Kombination von Software-Engineering-Methoden bestimmen und tun dies auch (IEEE Computer Society).

2.4.2 Pressman und Maxim: Empfohlenes Softwareentwicklungsprozessmodell

Pressman und Maxim (2020) erläutern, dass jeder agile Softwareprozess auf eine Art und Weise charakterisiert ist, die eine Reihe von Schlüsselannahmen über die Mehrheit der Softwareprojekte berücksichtigt:

1. Es ist schwierig, im Voraus zu sagen, welche Softwareanforderungen bestehen bleiben und welche sich ändern werden. Ebenso schwierig ist es, vorherzusagen, wie sich die Prioritäten der Kunden im Laufe des Projekts ändern werden.
2. Bei vielen Softwaretypen sind Design und Konstruktion ineinander verschachtelt. Das heißt, beide Tätigkeiten sollten parallel durchgeführt werden, damit sich die Entwurfsmodelle während ihrer Erstellung bewähren. Es lässt sich nur schwer vorhersagen, wie viel Entwurf notwendig ist, bevor die Konstruktion zum Nachweis des Entwurfs herangezogen wird.
3. Analyse, Entwurf, Konstruktion und Prüfung sind (aus planerischer Sicht) nicht so vorhersehbar, wie wir es uns wünschen würden.

Angesichts dieser drei Annahmen können wir feststellen, dass ein agiler Prozess anpassungsfähig sein, aber auch Fortschritte bringen muss. Der Prozess soll auch inkrementell angepasst werden. Dafür ist Kundenfeedback wichtig, und ein wirksamer Katalysator für Kundenfeedback ist ein funktionsfähiger Prototyp oder ein Teil eines funktionsfähigen Systems. Daher sollte eine inkrementelle Entwicklungsstrategie eingeführt werden. Software-Inkremente (lauffähige Prototypen oder Teile eines operativen Systems) müssen in kurzen Zeiträumen geliefert werden, damit die Anpassung mit dem Wandel (Unvorhersehbarkeit) Schritt halten kann. Dieser iterative Ansatz ermöglicht es dem Kunden, das Software-Inkrement regelmäßig zu bewerten, dem Software-Team das notwendige Feedback zu geben und die Prozessanpassungen zu beeinflussen, die zur Berücksichtigung des Feedbacks vorgenommen werden.

Letztendlich schlagen Pressman und Maxim (2020) die Verwendung eines hochgradig interaktiven, inkrementellen Prototyping-Prozesses vor. Sie sind der Meinung, dass dies besser ist als die Erstellung starrer Produktpläne und umfangreicher Dokumente, bevor man mit der Programmierung beginnt. Anforderungen ändern sich. Der Input und das Feedback der Interessengruppen sollten früh und häufig im Entwicklungsprozess erfolgen, um die Lieferung eines nützlichen Produkts zu gewährleisten.

Vorgeschlagen ist die Verwendung eines evolutionären Prozessmodells, das die häufige Einbindung der Interessengruppen in die Erstellung und Bewertung von inkrementellen Softwareprototypen betont.

Die Beschränkung der Anforderungs-Artefakte auf die minimal nützlichen Dokumente und Modelle ermöglicht die frühzeitige Erstellung von Prototypen und Testfällen. Die Erstellung von evolutionären Prototypen reduziert die Zeit, die für die Erstellung von Wegwerfprototypen benötigt wird. Die Verwendung von Papierprototypen in einem frühen Stadium des Entwurfsprozesses kann dazu beitragen, die Programmierung von Produkten zu vermeiden, die die Erwartungen der

Kunden nicht erfüllen. Der richtige Architekturentwurf vor Beginn der eigentlichen Entwicklung ist ebenfalls wichtig, um Zeit- und Kostenüberschreitungen zu vermeiden.

Die Planung ist wichtig, sollte aber zügig durchgeführt werden, um den Beginn der Entwicklung nicht zu verzögern. Die Entwickler sollten eine ungefähre Vorstellung davon haben, wie lange ein Projekt dauern wird, aber sie müssen sich darüber im Klaren sein, dass sie wahrscheinlich nicht alle Projektanforderungen kennen werden, bevor die Softwareprodukte geliefert werden. Die Entwickler tun gut daran, eine detaillierte Planung zu vermeiden, die über die Planung des aktuellen Prototyps hinausgeht.

Die Entwickler und die Beteiligten sollten ein Verfahren für das Hinzufügen von Funktionen einführen, die in zukünftigen Prototypen implementiert werden sollen, und die Auswirkungen dieser Änderungen auf den Projektzeitplan und das Budget bewerten.

Risikobewertung und Akzeptanztests sind ein wichtiger Teil des Prototypenbewertungsprozesses. Eine agile Philosophie für die Verwaltung von Anforderungen und das Hinzufügen neuer Funktionen zum Endprodukt ist ebenfalls wichtig. Die größte Herausforderung für Entwickler bei evolutionären Prozessmodellen besteht darin, den Umfang des Projekts zu begrenzen und gleichzeitig ein Produkt zu liefern, das den Kundenerwartungen entspricht, und dies alles unter Einhaltung des Zeitplans und des Budgets. Das ist es, was die Softwareentwicklung so herausfordernd und lohnend macht.

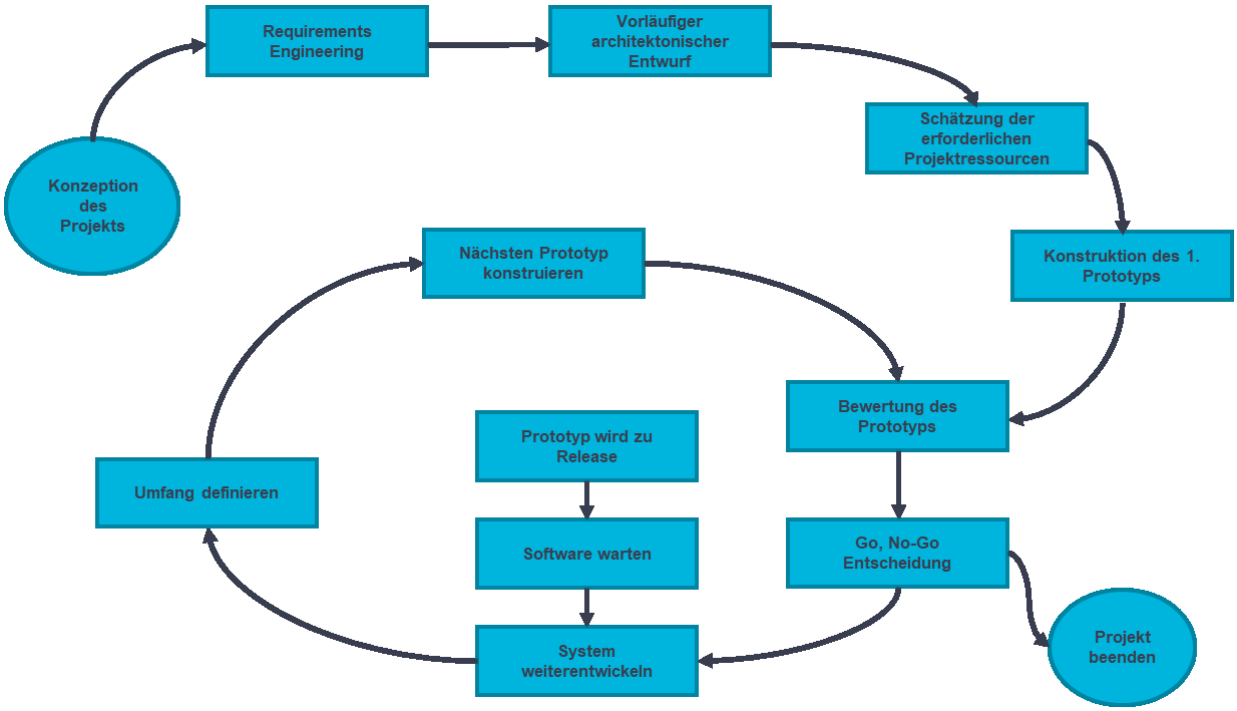


Abbildung 3: Pressman und Maxim Empfohlenes Softwareentwicklungsprozessmodell

Die empfohlenen Software-Prozess-Schritte sind wie folgt:

1. Requirements Engineering
    - User Stories von allen Beteiligten sammeln
    - Akzeptanzkriterien für die User Stories von Stakeholdern beschreiben lassen
2. Vorläufiger Architekturentwurf

- Verwendung von Papierprototypen und Modellen
- Bewertung von Alternativen anhand nichtfunktionaler Anforderungen
- Dokumentation der Entscheidungen zum Architekturentwurf

3. Schätzung der erforderlichen Projektressourcen
   - Historische Daten verwenden, um die Zeit für die Fertigstellung jeder User Story zu schätzen
   - User Stories in Sprints organisieren
   - Anzahl der Sprints bestimmen, die für die Fertigstellung des Produkts erforderlich sind
   - Zeitschätzungen überprüfen, wenn User Stories hinzugefügt oder gelöscht werden
4. Erstellung des ersten Prototyps
   - Teilmenge der für die Stakeholder wichtigsten User Stories auswählen
   - Papierprototyp als Teil des Designprozesses erstellen
   - Prototyp der Benutzeroberfläche mit Eingaben und Ausgaben entwerfen
   - Algorithmen entwickeln, die für die ersten Prototypen benötigt werden
   - Prototyp unter Berücksichtigung des Deployments entwickeln
5. Prototyp evaluieren
   - Testfälle erstellen, während der Prototyp entwickelt wird
   - Prototyp mit geeigneten Benutzern testen
   - Feedback der Beteiligten für den Überarbeitungsprozess erfassen
6. Go / No-go Entscheidung
   - Qualität des aktuellen Prototyps bestimmen
   - Zeit- und Kostenschätzungen für den Abschluss der Entwicklung überprüfen
   - Risiko bestimmen, dass die Erwartungen der Interessengruppen nicht erfüllt werden
   - Zusage zur Fortsetzung der Entwicklung einholen
7. System weiterentwickeln
   - Umfang des neuen Prototyps definieren
   - Konstruktion des neuen Prototyps
   - Bewertung des neuen Prototyps und Durchführung von Regressionstests
   - Bewertung der mit der weiteren Entwicklung verbundenen Risiken
8. Prototyp Release
   - Akzeptanztests durchführen
   - Festgestellte Mängel dokumentieren
   - Qualitätsrisiken mit dem Management teilen
9. Wartung der Software
   - Code verstehen, bevor Änderungen vorgenommen werden
   - Software testen, nachdem Änderungen vorgenommen wurden
   - Änderungen dokumentieren
   - Bekannte Fehler und Risiken an alle Beteiligten kommunizieren

2.5 Probleme beim Management von Softwareprojekte

The Standish Group (2015) hat in ihrem Chaos Report 2015 veröffentlicht, dass von mehr als 25.000 Softwareprojekten im Jahr 2015 nur 29% erfolgreich waren, 52% in Frage gestellt wurden und 19% scheiterten.

Erfolgreich bedeutet, dass das Projekt termingerecht, im Rahmen des Budgets und mit einem zufriedenstellenden Ergebnis abgeschlossen wurde. In Frage gestellte Projekte lagen entweder

über dem Budget, waren verspätet und/oder nicht zufriedenstellend. Und gescheiterte Projekte sind Projekte, die abgebrochen oder nicht genutzt werden.

Zwischen 30% und 50% aller neuen Leistungen am Markt haben sich in einer Untersuchung als Flops erwiesen und mussten aufgrund zu geringen Umsatzes vom Markt genommen werden (Reichwald und Schaller 2003).

Stepanek (2005) stellte in „Why Software Projects fail“ fest, dass der Grund für das Scheitern von Software-Projekten in der Einzigartigkeit von Software liegt:

- ***Software ist komplex***: Selbst kleine Software kann eine erschreckende Komplexität anhäufen. Ein kleines Programm mit einem oder zwei Autoren kann leicht Zehntausende von Codezeilen umfassen.

  Software ist eine Folge von Anweisungen. Die Komplexität eines Programms hängt jedoch nicht nur von den Anweisungen ab, sondern auch von den Wechselwirkungen zwischen den Anweisungen.

- ***Software ist abstrakt:*** Man kann Software nicht physisch anfassen. Software ist eine Kodifizierung einer riesigen Menge von Verhaltensweisen, was es schwierig macht, Blueprints (Baupläne, Vorlagen) zu erstellen.

  Software kann immer nur auf der Ebene der einzelnen Anweisungen genau beschrieben werden. Wenn das System einmal vollständig beschrieben ist, dann ist die Software geschaffen. Es muss nichts weiter getan werden.

  Jede Architektur, jedes Design und jedes Diagramm, das wir für Software erstellen, sind im Grunde unzureichend, denn, wenn wir jedes Detail darstellen, duplizieren wir die Software lediglich in einer anderen Form, was einer Verschwendung von Zeit und Mühe gleichkommt.

- ***Es ist schwierig, vollständige Anforderungen für Software zu definieren***: Benutzer gewinnen neue Erkenntnisse über ihre Bedürfnisse, wenn die Software Gestalt annimmt.

  Um erfolgreich zu sein, müssen Benutzer und Entwickler zusammenarbeiten, um die Anforderungen an die Software zu verfeinern. Wenn der Funktionsumfang der Software wächst, können die Benutzer die verbleibenden Funktionen auf der Grundlage ihrer Tests des im Aufbau befindlichen Systems überarbeiten.

- ***Softwareentwicklungstechnologien ändern sich schnell***: Die Geschwindigkeit von Computern verdoppelt sich etwa alle zwei Jahre, und das eröffnet den Softwareentwicklern immer mehr Möglichkeiten.

- ***Best Practices sind nicht ausgereift***: Die meisten Softwareentwicklungstechnologien sind noch nicht ausgereift genug, um über eine Reihe von Best Practices zu verfügen.

- ***Technologie ist eine riesige Domäne***: Die Softwareentwicklung umfasst eine kaum überschaubare Anzahl an Technologien, und ihre Komplexität übersteigt die Möglichkeiten einer einzelnen Person, Fachwissen in allen Technologien zu erwerben.

- ***Technologieerfahrung ist unvollständig***: Das Fachwissen über bestimmte Softwareentwicklungs-Technologien ist schnell veraltet, weshalb die meisten spezifischen Fertigkeiten erst am Arbeitsplatz erlernt werden.
- ***Softwareentwicklung ist Forschung***: Bei der Softwareentwicklung geht es nicht nur um die Erstellung von Software, sondern auch darum, zu lernen, wie man die für den jeweiligen Zweck am besten geeignete Software erstellt.
- ***Sich wiederholende Arbeiten sind automatisiert***: Die Softwareentwicklung ist in höherem Maße automatisiert als andere projektbasierte Tätigkeiten.
- ***Konstruktion ist eigentlich Design***: Im Gegensatz zu anderen Produkten wird Software nicht konstruiert, sondern ins Leben gerufen.
- ***Änderungen werden als einfach angesehen***: Es stimmt, dass wesentliche Änderungen oft schnell und einfach durchgeführt werden können, aber um sie richtig zu implementieren, muss man die Architektur der Software so überarbeiten, dass sie die neuen Funktionen zuverlässig unterstützt; andernfalls verursacht man nur ein Chaos und macht die Software fehleranfälliger. Software kann schnell geändert werden, und dieses Tempo wird erwartet, aber es ist notwendig, die Änderungen korrekt umzusetzen.
- ***Veränderung ist unvermeidlich***: Keine Software ist so perfekt, wie man sie sich vorgestellt hat; sie wird immer Änderungen erfordern, damit sie ihrer Aufgabe gerecht wird.

  Wenn eine Software nicht so gebaut ist, dass sie Veränderungen unterstützt, dann wird dies schon während der Entwicklung Schwierigkeiten verursachen. Die Qualität von Software zeigt sich, wenn die Software zum ersten Mal erweitert oder geändert wird.

Stepanek (2005) geht noch weiter und erklärt, dass Software-Projekte aufgrund ihrer Eigenschaften nicht wie andere Projekte gemanagt werden sollten. Die Analyse des Projektmanagements hat zehn Annahmen aufgedeckt, die für Softwareentwicklungsprojekte nicht zu gelten scheinen:

1. Der Umfang kann vollständig definiert werden.
2. Die Definition des Umfangs kann vor Beginn des Projekts erfolgen.
3. Die Softwareentwicklung besteht aus ganz unterschiedlichen Aktivitäten.
4. Die Softwareentwicklungsaktivitäten können in eine Reihenfolge gebracht werden.
5. Die Teammitglieder können individuell den Aktivitäten zugeordnet werden.
6. Die Größe des Projektteams hat keinen Einfluss auf den Entwicklungsprozess.
7. Es gibt immer eine Möglichkeit, aussagekräftige Schätzungen zu erstellen.
8. Annehmbar genaue Schätzungen können erstellt werden.
9. Ein Entwickler ist einem anderen gleichwertig.
10. Metriken sind ausreichend, um die Qualität von Software zu bewerten.

Die von Stepanek vorgeschlagene Lösung besteht darin, agile Methoden anzuwenden, um nicht Opfer der oben genannten ungültigen Annahmen zu werden, denn:

- Agile Methoden erfordern nicht, dass der gesamte Umfang im Voraus definiert wird: Die Definition des Umfangs erfolgt während des gesamten Projektverlaufs.
- Agile Methoden behandeln die Entwicklung als eine einzige Aktivität, d. h. Design, Kodierung und Tests werden gleichzeitig durchgeführt. Die Entwickler machen alle dieselbe Arbeit.
- Schätzungen können auf den Erfahrungen früherer Iterationen beruhen, so dass eine Historie echter Metriken aufgebaut werden kann, die eine genauere Schätzung ermöglichen. Es ist nicht nötig, zu schätzen, wie viel Arbeit ein beliebiger Entwickler erledigen kann, weil bekannt ist, wie produktiv das Team gewesen ist.
- Die Entwickler überprüfen ständig den Code der anderen, um dessen Qualität zu bewerten und Probleme zu finden.
- Der Vorteil der iterativen Entwicklung besteht darin, dass jede Iteration eine weitere Chance bietet, herauszufinden, wenn etwas schiefläuft, und eine weitere Chance, es zu beheben.

Die Feststellung, dass agile Methoden eine Lösung für die Probleme des Softwareprojekt-Managements sind, wird durch die Ergebnisse des Chaos Report 2015 bestätigt (The Standish Group 2015). Insgesamt mehr als 10.000 Software-Projekte zeigen, dass agile Projekte fast viermal so erfolgreich sind wie Wasserfall-Projekte, und dass Wasserfall-Projekte dreimal so häufig scheitern wie agile Projekte.

Agile Methoden können Teil einer Lösung für die Probleme bei Softwareentwicklungsprojekten sein, aber sie sind nicht die ganze Lösung. Sie können helfen, erfolgreich Software zu entwickeln, aber sie können nicht garantieren, dass unsere Projekte rentabel sind.

Viele der agilen Praktiken verbessern die Softwarequalität, ohne den Projektverlauf zu beeinträchtigen, und verringern das Gesamtrisiko in Projekten, aber sie machen es auch schwer, weitere Praktiken aus dem Projektmanagement parallel zu Praktiken aus den agilen Methoden einzusetzen.

Die pünktliche, budgetgerechte und akzeptable Lieferung von Softwareprodukten war für die meisten Softwareunternehmen schon immer eine große Herausforderung. In diesem historischen Kontext haben sich Softwareunternehmen auf die sogenannte „Kunst des Software-Engineerings“ spezialisiert. Sie beherrschen den Prozess der Erstellung von Software-Assets wie Code und Anforderungsdokumenten. Heute stehen sie vor einer neuen Generation von Herausforderungen, die als „Kunst der Software-Innovation“ eingestuft werden können. Wie kann ich (als Unternehmen) mit Software Innovationen vornehmen und Werte schaffen? (Pikkarainen et al. 2011)

Kreuzer und Aschbacher (2011) erkannten in einer im November 2009 gestarteten empirischen Studie der FH CAMPUS 02 (mit Unterstützung der Wirtschaftskammer Steiermark), dass fast

50% der befragten IT-Unternehmen nur eine Ad-hoc-Entwicklung von Services vornehmen. Das bedeutet, dass keine wirklichen Prozesse beschrieben werden.

Software ist komplex, abstrakt und fließend. Bei der Planung eines Softwareentwicklungsprojekts ist es hilfreich, ein tiefes Verständnis für die Besonderheiten von Software zu haben. Nicht-technische Personen – Kunden und Projektmanager – haben oft nicht die Art von praktischer Erfahrung, die nötig ist, um dieses Verständnis zu erlangen. Den technischen Teammitgliedern – den Entwicklern, Architekten und Analysten – fehlen jedoch häufig die betriebswirtschaftlichen und Managementkenntnisse, die für die erfolgreiche Organisation eines Projekts erforderlich sind.

Die konzeptionelle Kluft zwischen den technischen und nichttechnischen Mitgliedern eines Softwareentwicklungsteams ist der offensichtlichste Grund für das Scheitern von Softwareprojekten (Stepanek 2005).

Im nächsten Kapitel befassen wir uns mit dem Service-Engineering, um zu sehen, ob es dazu beitragen kann, die Kluft zwischen den technischen und nichttechnischen Aspekten von Softwareprojekten zu verringern.

3 SERVICE ENGINEERING

Services haben in den letzten Jahren vermehrt an Bedeutung gewonnen. Dies kann nicht zuletzt am stetig steigenden Anteil des Dienstleistungssektors abgelesen werden. In Österreich sind bereits etwa vier Fünftel aller Unternehmen sowie 68% aller Beschäftigten im Dienstleistungssektor tätig (Johannes Kepler Universität Linz 2017). Auch in der Wirtschafts-Literatur ist das Service-Konzept stark in den Vordergrund getreten. Das gilt im speziellen für die Disziplinen Service-Marketing und Service-Management, aber auch allgemein für ein fortgeschrittenes, Service-orientiertes Wirtschaftssystem (Ng und Vargo 2018). Unternehmen verlagern den eigenen Fokus weg von traditioneller Produktion hin zu Services bzw. einer integrierten Sicht, die sowohl Produkte als auch Services einschließt.

Die Informationstechnologie (IT) hat diesen Trend hin zu mehr Service-Angeboten maßgeblich beeinflusst und tut dies weiterhin. Zunächst war die IT in der Rolle eines Unterstützers für bestehende Services. Später entstanden aber auch neue Formen der Dienstleistung, etwa Software-As-A-Service (SaaS). Mittlerweile hat sich das Wirtschaftssystem so verändert, dass nicht mehr das physische Produkt im Mittelpunkt steht, sondern Software. Dabei spielt das Sammeln und die Nutzung von Daten als Basis für Smart Services eine wichtige ökonomische Grundlage für Unternehmen (Marquardt 2017).

Aber was ist unter einem Service zu verstehen?

3.1 Was ist ein Service?

Für den Begriff „Dienstleistung“ (Service) gibt es keine eindeutige, präzise Definition. In Abgrenzung zum primären Sektor (Urproduktion) und sekundären Sektor (Industrie) werden aus volkswirtschaftlicher Sicht alle Aktivitäten zum tertiären Sektor (Dienstleistungen) gezählt, die nicht den anderen beiden Sektoren zugerechnet werden können. Es handelt sich also um eine Negativ-Definition, die sich auch folgendermaßen ausdrücken lässt: Dienstleistungen sind alles das, was nicht fest oder flüssig ist. Die prägnanteste Eigenschaft von Dienstleistungen ist also ihre Unstofflichkeit (Fähnrich und Opitz 2006).

Eine andere Herangehensweise ist die Einteilung der konstitutiven Merkmale von Dienstleistungen (Reichwald und Schaller 2003), wie in Abbildung 4 dargestellt:

- Die potenzialorientierte Dimension beschreibt ein angebotenes Leistungspotenzial. Voraussetzungen für die Realisierung des Potenzials sind Dienstleistungsfähigkeit und -bereitschaft des Anbieters.
- Die prozessorientierte Dimension ist eine Darstellung der Dienstleistung als andauernden, noch nicht abgeschlossenen Prozess. Der Leistungserbringer bindet einen externen Faktor (z.B. Kunde selbst, Maschinen des Kunden) ein, der vom Kunden bereitgestellt wird. Es herrscht das Uno-actu-Prinzip: die Dienstleistungserbringung und die Konsumation fallen zeitlich zusammen, sind also ein nicht trennbarer Akt (lateinisch Uno

actu: Ein Akt). Die prozessorientierte Dimension ist relevant, denn „Dienstleistungen sind von Natur aus weniger Sache als Prozess“ (Fähnrich und Opitz 2006).

- Die ergebnisorientierte Dimension konzentriert sich auf das Resultat der Leistungserbringung. Ein zentrales Charakteristikum ist das der Immaterialität: Dienstleistungen gelten als überwiegend immaterielle Leistungen. Die Wirkungen einer Dienstleistung können jedoch sowohl immateriell also auch materiell sein.

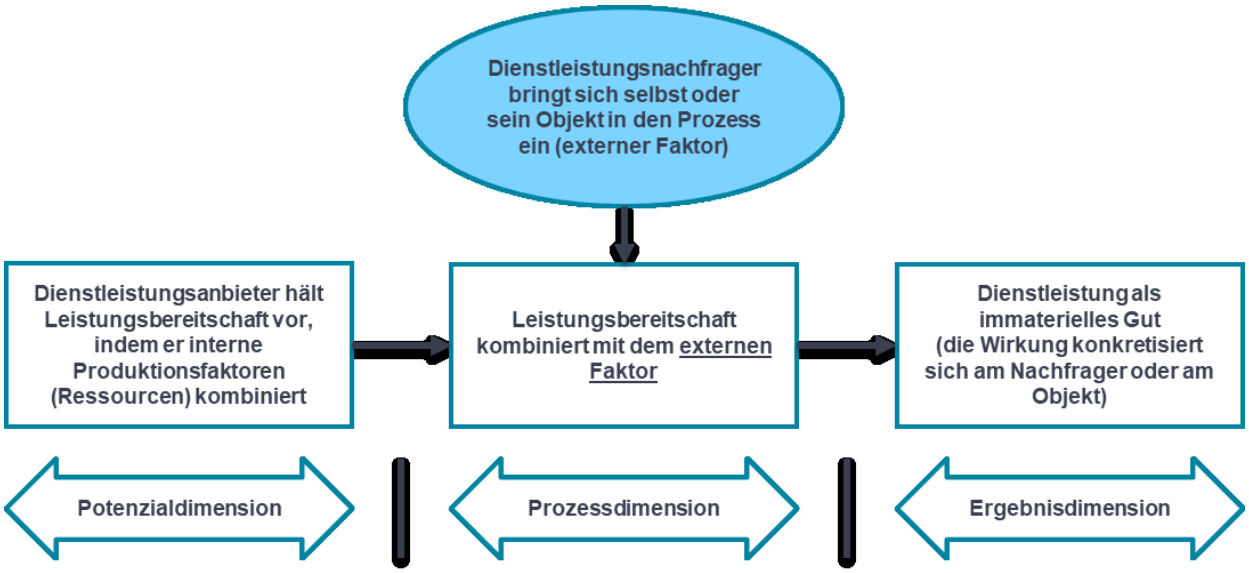


Abbildung 4: Dimensionen einer Dienstleistung (Scheer et al. 2006)

3.1 Service-dominante Logik

Die Grundannahme der “Service-dominanten Logik“ (Vargo et al. 2019) besteht in der Überzeugung, dass Organisationen, Märkte und die Gesellschaft fundamental service-orientiert sind. Kompetenzen, also Wissen und Fähigkeiten, werden demnach zum Vorteil eines Beteiligten eingesetzt. Die service-dominante Logik stellt sich explizit gegen eine Produkt-dominante Logik (G-D: „goods-dominant logic“), welche Produkte und dessen Herstellung in den Fokus stellt. Produkte sind ohne dazu passende Service-Leistungen oft nicht verkaufbar. Nach der SD-Logik ist jedes Unternehmen ein Service-Dienstleister. Daher betrachtet die SD-Logik Produkte als Teil eines umfassenden Service-Angebots. Die Überlegenheit von Services wird in folgendem Zitat ausgedrückt: „Die Bedeutung von physischen Produkten liegt weniger in dessen Besitz, sondern in den Services, die sie bieten“ (Zitat Philip Kotler).

Vargo und Lusch (2016) haben fünf Grundsätze für ein service-zentriertes System ermittelt:

- Das Service ist die fundamentale Basis für den Austausch: Services werden gegen andere Services getauscht.
- Wert entsteht durch Co-Kreation, immer unter Einbeziehung des Nutzers.
- Alle sozialen und ökonomischen Akteure sind Ressourcen-Integratoren.
- Der Wert wird durch den Nutzer bestimmt; Wert ergibt sich aus der Nutzung von Services.

- Die Co-Kreation des Werts wird mit Hilfe von Institutionen und Vereinbarungen koordiniert, die von den Akteuren gesteuert werden.

Die service-dominante Logik hat klare Konsequenzen für das Marketing: der Fokus liegt auf dem Nutzwert sowie auf gemeinsamer Wertschöpfung eines Produzenten mit den Kunden und Partnern (Vargo et al. 2019).

3.2 Service-Engineering-Definition

Die systematische Entwicklung und Gestaltung innovativer Dienstleistungen werden als „Service-Engineering“ bezeichnet. Es betrifft sowohl die strategische als auch die operative Ebene. Es richtet sich nicht nur an klassische Dienstleistungsunternehmen, sondern darüber hinaus auch an Hersteller von Sachgütern, die ihr produktbegleitendes Dienstleistungsangebot professionalisieren wollen (Bullinger et al. 2006).

Service Engineering ist nicht als Selbstzweck oder als Aufgabe wissenschaftlicher Grundlagenforschung zu verstehen, sondern als konkrete Unterstützung von Unternehmen bei der Gestaltung von Dienstleistungen mit der gewünschten Qualität und Effizienz gemäß den Erwartungen des Marktes (Bullinger und Scheer 2006).

Erkenntnisse aus der klassischen Produktentwicklung, nämlich Aspekte aus dem Customer-Relationship-Management, Qualitätsmanagement und Marketing fließen in die Methodik des Service Engineering ein. Durch die rasche Entwicklung des Internets ergab sich eine neue Anforderung, die bei der Planung und Gestaltung von Services berücksichtigt werden muss, nämlich die Unterstützung von Dienstleistungen mit Hilfe moderner Informations- und Kommunikationstechnologie und die dazu notwendige Integration der zu verarbeitenden Daten und Informationen in die bestehenden IT-Strukturen.

Service Engineering ist eine Disziplin, die sich mit der Entwicklung und Gestaltung von Dienstleistungsprodukten unter Verwendung geeigneter Vorgehensmodelle, Methoden und Werkzeuge beschäftigt (Bullinger und Schreiner 2006).

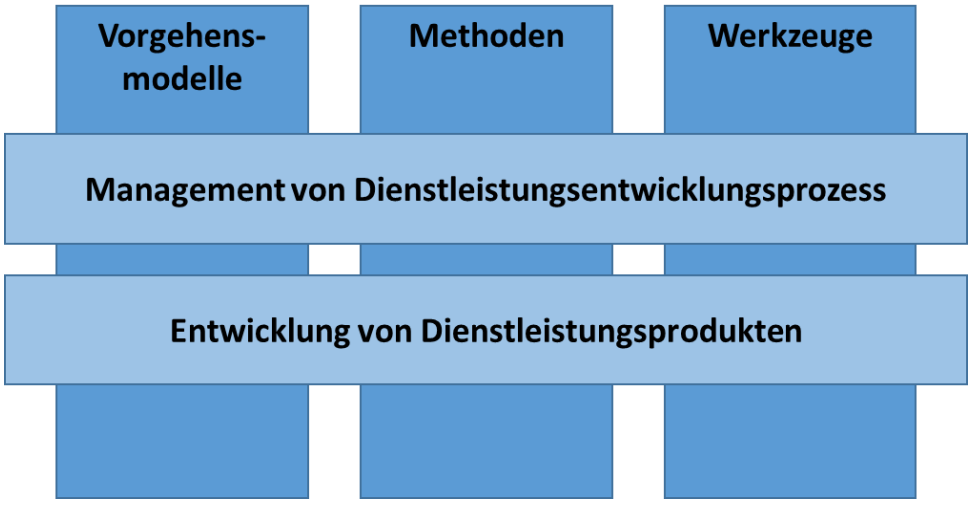


Abbildung 5: Service Engineering Strukturierung nach Fähnrich und Opitz (2006)

Abbildung 5 zeigt zwei überlagernde Dimensionen: die inhaltliche Dimension die sich auf die Entwicklung und den Einsatz von Vorgehensmodellen, Methoden und Werkzeugen bezieht, und die Betrachtungsdimension, die sich entweder auf die Entwicklung einer singulären Dienstleistung oder auf das Management von Entwicklungsprozessen bezieht (Fähnrich und Opitz 2006).

Vorgehensmodelle stellen definierte Vorgehensweisen dar, die in Dienstleistungsentwicklungsprozessen befolgt werden. Sie enthalten eine ausführliche Dokumentation von Abläufen, Strukturen und Verantwortlichkeiten und unterstützen die Planung, Steuerung und Überwachung von Projekten mit dem Ziel, Entwicklungsprozesse zu strukturieren und Komplexität zu reduzieren.

Je nach Vorgehensmodell können die Prozessschritte sequenziell, parallel oder in Zyklen erfolgen.

Methoden sind detaillierte und systematische Handlungsvorschriften dafür, wie ein vorgegebenes Ziel nach bestimmten Prinzipien erreicht werden kann. Ihre Anwendung ist abhängig von der Situation des Unternehmens, der Komplexität der zu entwickelnden Dienstleistung und der Erfahrung der Mitarbeiter mit dem Methodeneinsatz.

In der Betrachtungsdimension kann die Entwicklung von Dienstleistungsprodukten neue Implementierungen oder die Verbesserung von bestehenden Services bedeuten. Die Lenkung der Projekte, die Gestaltung von Rahmenbedingungen und die Entwicklung des Service Engineering-Systems bilden das Management von Dienstleistungsentwicklungsprojekten.

3.3 Vorgehensmodelle für die Dienstleistungsentwicklung

Laut (Bullinger und Schreiner 2006) definieren Vorgehensmodelle des Service Engineering den Dienstleistungsentwicklungsprozess, indem sie die einzelnen Schritte festlegen, die von der Generierung der Service-Idee bis zur Einführung der marktreifen Dienstleistung durchlaufen werden.

Der Dienstleistungsentwicklungsprozess kann in sechs Phasen eingeteilt werden:

- Startphase: Dienstleistungsideen-Generierung.
- Analysephase: Nachfrage- und anbieterseitige Anforderungen aufnehmen. Ideen-Bewertung auf Basis dieser Informationen. Falls keine geeigneten Ideen generiert wurden, erfolgt ein Rücksprung in die Startphase.
- Konzeptionsphase: Einzelspezifikationen auf der Potenzial-, Prozess-, Ergebnis- und Marktdimension erstellen und am Ende der Konzeptionsphase zu einer Gesamtspezifikation integrieren.
- Vorbereitungsphase: Vorbereitung des Potenzials, das im Rahmen der Dienstleistungserbringung benötigt wird.

- Testphase: Gesamtspezifikation testen, um eventuelle Schwachstellen zu identifizieren, und betreffende Einzelspezifikationen überarbeiten.
- Implementierungsphase

Nach der Implementierung wird die Dienstleistung am Markt angeboten, bis diese das Ende ihres Lebenszykus erreicht hat.

Die Entwicklung von Dienstleistungen ist eine fortlaufende Aufgabe; obsolete Angebote werden substiuirt und neue Dienstleistugen werden für Wachstumspotenziale angeboten.

In jeder Phase eines Vorgehensmodells werden verschiedene Methoden verwendet, d.h. definierte Handlungsanweisungen, die vorgeben, welche Aktivitäten durchzuführen sind, um ein bestimmtes Ziel zu erreichen.

In der Literatur sind mehrere verschiedene Vorgehensmodelle zu finden, wobei drei Ausprägungsformen, die den „Weg zum Ziel“ beschreiben, unterscheidbar sind:

1. **Lineare Vorgehens- oder Phasenmodelle**:

   Die Entwicklungsschritte laufen in einer sequentiellen Abfolge. Die nächste Phase startet erst, wenn die vorherige erfolgreich abgeschlossen ist. Das Produkt wird sukzessive konkretisiert und der Prozess hat eine hohe Transparenz. Anderseits ist ein Rückschritt in eine vorangehende Phase nicht vorgesehen, um Änderungen zu tätigen, was einen Mangel an Flexibilität verursacht.

2. **Iterative Vorgehensmodelle**:

   Im Vergleich zum linearen Modell ist das iterative Modell flexibler. Das Modell ermöglicht es, wiederholt in die vorangehende Phase zurück zu springen, um einen Fehler zu beseitigen und die anschließende Phase erneut zu durchlaufen.

3. **Prototyping-Modelle**:

   Eine Vorabversion der Dienstleistung wird entwickelt und anhand der erwünschten Merkmale und Funktionalitäten getestet. Das Besondere an diesem Modell ist, dass die Phasen überlappend ablaufen können.

Die Auswahl einer Vorgehensweise hängt davon ab, wie groß die zu entwickelnde Dienstleistung ist, sowie von damit verbundenem Aufwand und Kosten. Da die Phasenmodelle einfach und leicht verständlich sind, finden sie in der Praxis bisher die größte Verbreitung.

3.4 IT-basierte Dienstleistung / E-Service

Im Dienstleistungssektor sind zwei dominante Entwicklungen zu beobachten (Laqua 2007a):

- Industrialisierung: Etablierte Dienstleistungen werden mithilfe von Informationstechnologie automatisiert und damit industrialisiert.
- Innovationssprünge: Die konsequente Anwendung von IT führt zu sprunghafter Innovation.

Im Gegensatz zu traditionellen Industriezweigen wie etwa dem Maschinenbau existiert im Dienstleistungssektor kein systematisches Engineering. Die beiden Entwicklungen Industrialisierung und Verlangen nach Innovation erfordern aber gerade eine systematische, modellbasierte Entwicklung von IT-basierten Dienstleistungen.

Ein solches systematisches Vorgehen erfordert die Integration von Software Engineering und Management, da Informations- und Kommunikationstechnologie sowohl für Produktion als auch Distribution zentral sind (Laqua 2007a).

Der tertiäre Sektor hat sich nicht nur als dominierender Sektor vor allem in den stärker entwickelten Ländern etabliert, sondern ist auch selbst Veränderungen unterworfen (Meyer und van Husen 2007). Dieser Wandel ist insbesondere durch die Verbindung zwischen dem Dienstleistungssektor und der Informations- und Kommunikationstechnologie gekennzeichnet. Kunden wünschen sich moderne und integrierte Lösungen aus Dienstleistung und Informationstechnologie, was wiederum zur Entstehung des Forschungs- und Praxisgebietes IT-basierter Dienstleistungen oder E-Services führte.

IT-basierte Dienstleistungen sind Dienstleistungen, die hauptsächlich durch Informations- und Kommunikationstechnologien realisiert werden. Dazu gehören beispielsweise die Bereiche Finanzdienstleistungen, Versicherungen, Mediendienste, Telekommunikationsdienstleistungen, E-Business, E-Logistik oder E-Learning (Laqua 2007a).

Die Entwicklung von IT-basierten Dienstleistungen erfordert ein Co-Design von Informationstechnologie und Dienstleistungen (Meyer und van Husen 2007). Damit ergeben sich neue Herausforderungen, um Vollständigkeit, Konsistenz und Korrektheit der bereitgestellten Gesamtleistung zu erreichen. Entwicklungsverfahren, die in der IT eingesetzt werden, sind alleine nicht ausreichend, um auch den Dienstleistungsaspekt zu entwickeln. Es müssen also IT-Entwicklungsverfahren mit Dienstleistungsentwicklungsverfahren in Einklang gebracht werden (Laqua 2007b).

Ein spezifisches Problem für die Entwicklung IT-basierter Dienstleistungen ist die Interdependenz, also die gegenseitige Abhängigkeit der beiden Komponenten Dienstleistung und Software. Die Spezifikation der Dienstleistung hat einen großen Einfluss auf die Spezifikation der Software. Umgekehrt ist die Spezifikation der Dienstleistung abhängig von der Machbarkeit der IT-Lösung: die Dienstleistung kann nur so gut sein wie die Qualität der unterstützenden Software. Werden die Entwickler erst spät eingebunden, erschwert das die Berücksichtigung von Interdependenzen. Das führt in weiterer Folge zur Ineffizienz des Gesamtsystems (Meyer et al. 2007).

Je nachdem, ob Dienstleistung oder Informationstechnologie das primäre Produkt in der Dienstleistungs-IT-Kombination darstellt, können laut Meyer und van Husen (2007) IT-basierte Dienstleistungen in drei Gruppen untergliedert werden:

- Durch IT unterstützte Dienstleistungen: Primärprodukt ist die Dienstleistung (z. B. Terminals für den Fahrkartenverkauf).
- IT-begleitende Dienstleistungen: Primärprodukt ist die Informationstechnologie (z. B. „Full Lifecycle Support“ für verkaufte Informationstechnologie).

- Integriertes Hybridprodukt: beide Bereiche sind bezüglich der Priorität gleichberechtigt (z. B. Video on Demand über das Internet, Beratung via Nutzung mobiler Geräte).

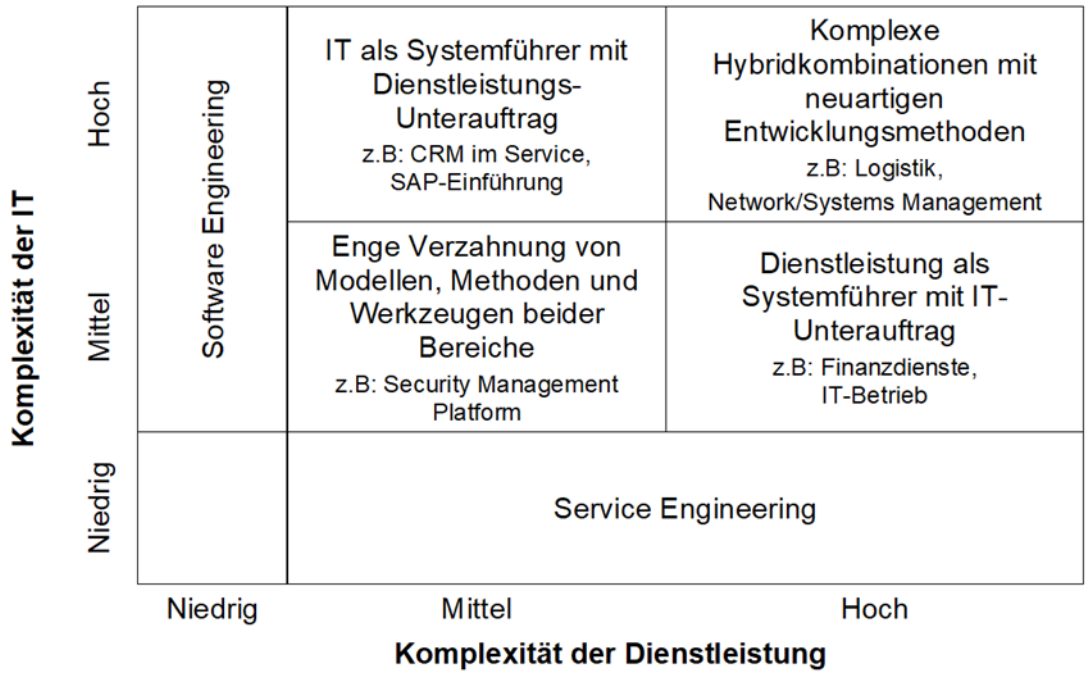


Abbildung 6: Entwicklungsbereiche IT-basierter Dienstleistungen - Meyer und van Husen (2007), mit Beispielen von (van Husen et al. 2007).

Laut Meyer und van Husen (2007) beschäftigt sich der Forschungsgegenstand IT-basierter Dienstleistungen mit Konstellationen, in denen sowohl IT als auch Dienstleistung mittlere bis hohe Komplexität aufweisen. Wenn nur einer der beiden Bereiche eine entsprechende Komplexität aufweist, dann ist es ausreichend, sich auf entweder klassisches Software-Engineering oder Service Engineering zu beschränken. Sobald jedoch eine mittlere oder hohe Komplexität von sowohl IT als auch Dienstleistung auftritt, ist es zielführend, Ansätze von Software-Engineering und Service Engineering zu kombinieren. Im Zusammenspiel von Software Engineering und Service Engineering können je nach Komplexitätskonstellation einige Bereiche unterschieden werden (siehe Abbildung 6).

Im Fall von mittlerer Komplexität von sowohl Software als auch Service Engineering ist es noch möglich, das Produkt durch Abstimmung der beiden Prozesse zu entwickeln.

Bei hoher Service-Komplexität und mittlerer Software-Komplexität kann die IT-Komponente als Unterauftrag ausgeführt werden. In diesem Fall ist die Fachabteilung die treibende Kraft und gibt die Produktspezifikation maßgeblich vor.

Analog gilt für den Fall von hoch komplexen IT-Lösungen, die um Services erweitert werden: in diesem Fall kann die Dienstleistung als Unterauftrag ausgeführt werden. Die IT-Abteilung gibt die Produktspezifikation maßgeblich vor.

Sind sowohl Services als auch IT hochkomplex, reichen traditionelle Ansätze nicht mehr aus, um Produkte systematisch zu entwickeln. Die Entwicklung hochgradig verzahnter Dienstleistungs- und IT-Produkte wurde z.B. im Projekt „ServCASE“ untersucht (siehe Abschnitt 3.4.2).

Unternehmen, die bereits IT-basierte Dienstleistungen entwickeln, sehen diese Aufgabe bisher nicht als integriertes Thema. Stattdessen arbeiten Software- und Dienstleistungsentwickler oft getrennt voneinander, und die Abstimmung zwischen ihnen ist schwierig (Meyer et al. 2007).

Verschiedene Umfragen zeigen die Notwendigkeit des Zusammenwachsens der beiden Dimensionen Dienstleistung und Informationstechnologie.

In einer Expertenbefragung kamen (van Husen et al. 2007), zum Ergebnis, dass zwar der größere Teil der untersuchten Unternehmen nicht über einen formalisierten Prozess zur Entwicklung von Dienstleistungen verfügt, und dass die Entwicklung der Dienstleistungen dort eher ad hoc und auf Basis individueller Erfahrungen erfolgt. Allerdings wurde ein klarer Bedarf für ein systematischeres Vorgehen gesehen. Darüber hinaus war der Wunsch erkennbar nach einer intensiveren Unterstützung des Entwicklungsprozesses durch geeignete Methoden, Tools und sogar eine entsprechende IT-Plattform. Die Interviews ergaben insbesondere vier Erfolgsfaktoren für die Entwicklung IT-basierter Dienstleistungen:

- Formalisiertes Vorgehen nach einem definierten Prozess
- Kundennähe und Kundenintegration in den Entwicklungsprozess
- detaillierte Anforderungsanalyse
- Schnelligkeit und Flexibilität

Eine Breitenerhebung mittels Fragebogen in deutschen Unternehmen aus der Dienstleistungs- und IT-Branche von (Meyer et al. 2007) ergab, dass alle drei Typen praktische Relevanz haben: durch IT unterstützte Dienstleistungen, IT-begleitende Dienstleistungen oder Hybridprodukte. In vielen Fällen waren der Aufwand für die Dienstleistungskomponente und jener für die IT-Komponenten ähnlich hoch. Auch diese Umfrage bestätigte den Wunsch nach formalisierten Vorgehensmodellen und Methoden bei der Entwicklung von Dienstleistungen und Software.

Die befragten Unternehmen nannten drei wichtige Erfolgsfaktoren für die Entwicklung IT-basierter Dienstleistungen:

- Kundenorientierung: Nähe zu den Kunden, Kenntnis der Anforderungen, gutes Verständnis der Kundenprozesse
- Projektmanagement: Kooperation in kleinen, handlungsfähigen Teams, gute Strukturierung des Projekts, Information aller Beteiligten, geeignete Dokumentation, Projektcontrolling
- Qualität: fehlerfreies und effizientes Service, definierte Service Level Agreements (SLAs), Verfügbarkeit, Datenqualität, Unterstützung, Bedienungskomfort, richtiges Preis-Leistungsverhältnis

3.4.1 Vorgehensmodelle

Ein Vorgehensmodell im Bereich E-Service dient der integrierten Entwicklung von Software und Dienstleistung. Ein solches Co-Design ist von Bedeutung, da Softwareentwickler meist eine andere Sprache sprechen als Dienstleistungsentwickler, die häufig einen betriebswirtschaftlichen Hintergrund haben. Die unterschiedliche Sprache erschwert die Zusammenarbeit. Ein Vorgehensmodell hat die Aufgabe, einen geeigneten konzeptionellen Rahmen für die Zusammenarbeit zu schaffen (Meyer und van Husen 2007).

Produktmodelle stellen dar, was eine Dienstleistung leistet. Diese sind für das Co-Design von Service und IT nicht ausreichend, da keine Produktmodelle existieren, die die Sicht von Entwicklung, IT, Marketing und Kunde gleichermaßen repräsentieren. Ergänzend zu Produktmodellen beschreiben Prozessmodelle, wie die Ergebnisse eine Dienstleistung zustande kommen. Dabei werden Prozesse mit dem Ziel dokumentiert, bereits in der Konzeptionsphase Transparenz zu schaffen und unrentable Aktivitäten zu eliminieren (Laqua 2007a).

Ohne ein explizites Vorgehensmodell für E-Services muss man sich entscheiden, worauf der Fokus gesetzt werden soll. Steht die Softwareentwicklung im Vordergrund, wird ein Softwareentwicklungs-Modell gewählt und die Dienstleistung entsteht nebenbei. Handelt es sich hauptsächlich um eine Dienstleistung, die durch Software unterstützt wird, wird ein Service Engineering-Modell als Basis gewählt und die Software darum herum gebaut. In beiden Fällen entsteht eine der beiden Komponenten unstrukturiert. Bei Hybrid-Produkten mit komplexen IT- und Service-Komponenten wird das Fehlen eines übergeordneten Vorgehensmodells zur Co-Entwicklung besonders deutlich (Meyer und van Husen 2007).

Eine Dienstleistung ist wesentlich mehr als die dafür zu erstellenden Anwendungssysteme. Die Methoden des Software Engineering reichen alleine nicht aus, um IT-basierte Dienstleistungen systematisch zu erstellen. Ein Software-Service-Co-Design muss Software Engineering-Methoden als einhüllende Methodik integrieren. Service Engineering ist also eine über die Software Engineering-Methoden übergeordnete Methodik (Laqua 2007a).

Die Anforderungen an Vorgehensmodelle zur Entwicklung IT-basierter Dienstleistungen sind wie folgt (Meyer und van Husen 2007):

- Allgemeingültigkeit: umfassende Beschreibung des Erstellungsprozesses IT-basierter Dienstleistungen
- Anwendbarkeit: konstruktive Begleitung des Entwicklungsprozesses mit Beachtung von Elementen des Software- und Service Engineerings.
- Anpassbarkeit: Anwendung für eine Vielzahl möglicher Entwicklungsprojekte mit unterschiedlich starker Ausprägung von Dienstleistungs- und IT-Komponente.
- Zielorientierung: das Ergebnis des Entwicklungsprozesses ist eine IT-basierte Dienstleistung.

- Wiederverwertbarkeit: sowohl das Vorgehensmodell selbst als auch die während des Erstellungsprozesses einer Dienstleistung erstellten Informationen müssen gespeichert und bei Bedarf verfügbar sein

Gesucht ist also ein Vorgehensmodell, das diese Anforderungen erfüllt und explizit Rücksicht auf die Interdependenzen beim Co-Design von Dienstleistung und Software nimmt. Mit einem guten Vorgehensmodell können Projekte mit sowohl hoher Dienstleistungskomplexität als auch hoher IT-Komplexität effizienter höherwertiger erstellt werden (Meyer und van Husen 2007).

(Memminger und Wäsch 2007) schlagen als Leitfaden für einen strukturierten Entwicklungsprozess fünf aufeinander aufbauende Phasen vor:

- Ideenfindung
- Anforderungsdefinition
- Konzeption
- Realisierung und Qualitätssicherung
- Markteinführung

Orthogonal zu den Phasen werden in der integrierten Dienstleistungsentwicklung folgende Entwicklungsobjekte betrachtet (Memminger und Wäsch 2007):

- Produktmodell: Beschreibung, was die Dienstleistung leistet und aus welchen Teilen sie besteht, sowie das Nutzenmodell für den Kunden und das Ertragsmodell für das Unternehmen
- Prozessmodell: Beschreibung, wie die Leistung erbracht wird
- Ressourcenkonzept: Die zur Leistungserbringung nötigen Humanressource, IT-Ressourcen, etc.
- Kundeninteraktionsmodell: Beschreibung, welche Kundeninterkationen in welchen Prozessschritten stattfinden

3.4.2 Das Projekt ServCASE

Im Rahmen des Projekts ServCASE wurde ein Vorgehensmodell für die Entwicklung von e-Services entwickelt, das die genannten Anforderungen – Allgemeingültigkeit, Anwendbarkeit, Anpassbarkeit, Zielorientierung und Wiederverwertbarkeit – erfüllt.

Über die gesamte Lebensdauer eines Projekts, von Projektinitiierung bis Projektabschluss, werden die Aktivitäten einem von drei Layer zugeordnet (siehe Abbildung 7):

- Service Engineering Layer: Dienstleistungsentwicklung
- Software Engineering Layer: Softwareentwicklung

- Integrated Engineering Layer: Komplex verzahnte Arbeitsschritte sowie Abstimmungs- und Review-Elemente

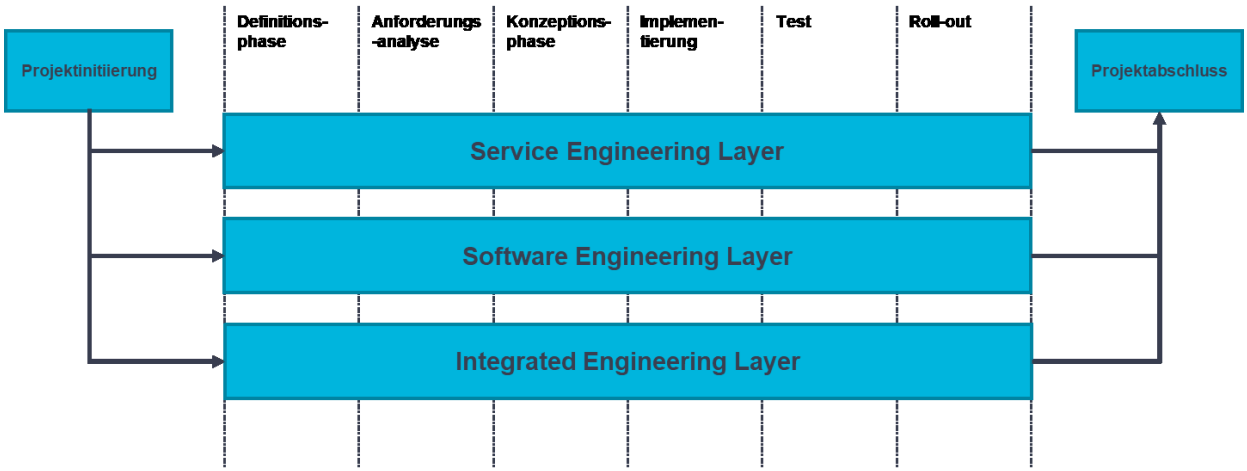


Abbildung 7: Grundstruktur der Vorgehensmodells nach Laqua (2007a)

Je nachdem, ob die Service- oder die IT-Komponente im Vordergrund steht oder ob es sich um ein komplexes Hybridprodukt handelt, wird entweder der Service Engineering Layer, der Software Engineering Layer oder der Integrated Engineering Layer als sogenannter „Master" definiert, dem die anderen Entwicklungsbausteine untergeordnet werden (Meyer und van Husen 2007).

Die Parallelisierung und Ausrichtung der Prozesse am Standardfall und nicht am kompliziertesten Fall geschieht mit dem Ziel, die Effizienz zu erhöhen (Laqua 2007a).

3.4.3 Strategiebasiertes Vorgehensmodell zur Dienstleistungsentwicklung der Studienrichtung IT & Wirtschaftsinformatik der FH CAMPUS 02

Das strategiebasierte Vorgehensmodell zur Dienstleistungsentwicklung der FH CAMPUS 02 ist aus einer Reihe von Studien mit lokalen Unternehmen hervorgegangen (Kreuzer und Aschbacher 2011). Das Modell folgt dem State-Gate-System Cooper (1983). Bei jedem Schritt gibt es eine Art Ampelsystem, das bei der Entscheidung hilft, ob die Qualitätsanforderungen des sogenannten „Quality Gate" für die jeweilige Stufe erreicht wurden. Erst wenn eine Stufe zufriedenstellende Ergebnisse liefert, wird die nächste Stufe in Angriff genommen.

Das Modell schreibt nicht vor, welche Methoden und Management-Tools während des Prozesses in den jeweiligen Stufen eingesetzt werden müssen, sondern erlaubt es, je nach Kontext und Unternehmen spezifische Methoden und Tools zu wählen und anzuwenden. Diese Flexibilität soll es den eingebundenen Personen erleichtern, den Prozess anzuwenden. Es können Tools gewählt werden, die für den jeweiligen Kontext als passend empfunden werden und die gegebenenfalls bereits bekannt sind (Kreuzer et al. 2011).

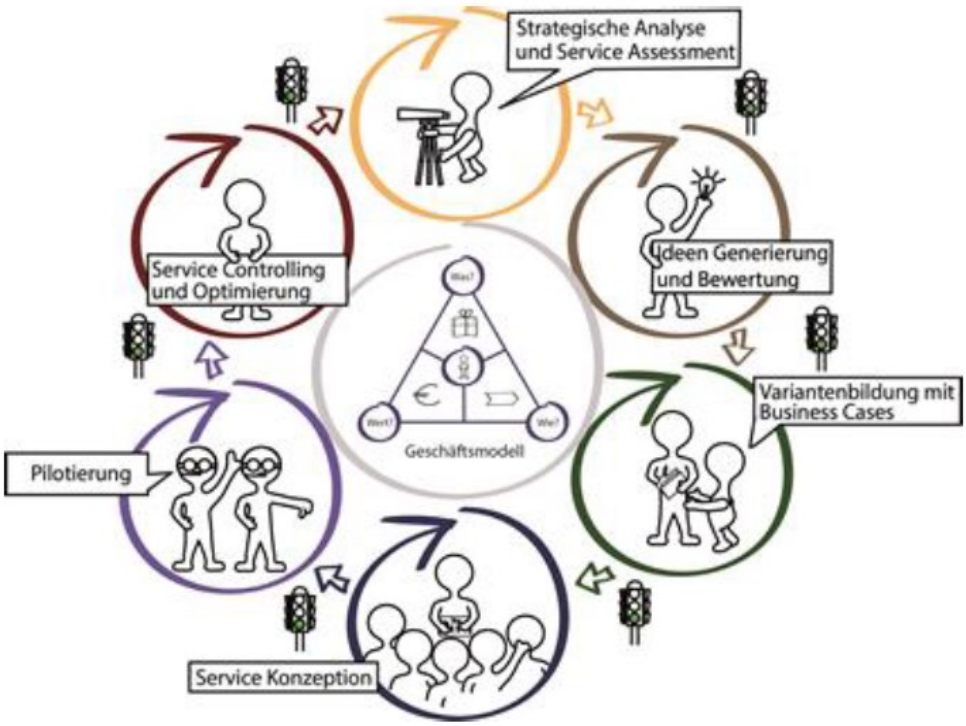


Abbildung 8: Strategiebasiertes Vorgehensmodell zur Dienstleistungsentwicklung der Studienrichtung IT & Wirtschaftsinformatik der FH CAMPUS 02

Eine schematische Darstellung des Modells ist in Abbildung 8 zu sehen. Das Geschäftsmodell steht im Zentrum und dient als Ankerpunkt während sämtlicher Prozessphasen. Am Geschäftsmodell richtet sich die Service-Entwicklung aus, und das Geschäftsmodell wird im Prozessverlauf weiterentwickelt (Ehrenhofer und Kreuzer 2012).

Der gesamte Prozess besteht aus sechs iterative Stufen (Kreuzer und Aschbacher 2011), die in Tabelle 1 zusammengefasst sind.

1. **Strategische Analyse und Service Assessment**: Die strategische Analyse ist ein essentieller erster Schritt. Es handelt sich um einen strategischen Blick durch die Service-Innovations-Brille auf die Ist-Situation im Unternehmen, insbesondere auf die aktuell angebotenen Produkte und Services.

   Diese Phase endet mit einer SWOT-Analyse. Die Methoden und Tools, die zuvor angewendet werden, um den Input für die SWOT-Analyse zu liefern, können jedoch im Unternehmen selbst gewählt werden. Dazu gehören etwa Dienstleistungs-Portfolio-Analyse, Kundenkontaktkreis und Personas.

2. **Ideengenerierung und Bewertung**: Nach dem Abstecken eines Suchfeldes für neue Services geht es in der zweiten Stufe darum, Ideen zu identifizieren und zu bewerten. Zur Ideenfindung können etwa kreative Techniken wie Brainstorming zum Einsatz kommen. Ideen sollen nicht kritisiert werden, auch wenn sie sehr weit hergeholt erscheinen.

Aus den entstandenen Ideen können durch systematische Bewertung jene eingegrenzt werden, die das größte Potenzial besitzen. Die wichtigste Frage in dieser Stufe ist jene, ob der derzeitige Produkt-Service-Mix den Zielen des Unternehmens entspricht und ob das geplante Service in das Portfolio passt.

3. **Variantenbildung mit Business Cases:** Aus den Ideen werden Geschäftsfälle ausgearbeitet, die die Eigenschaften des Marktes, der Wettbewerber und der Kunden ebenso berücksichtigen wie die zu erwartenden Auswirkungen auf Verkauf, Kosten und Profit. Für IT-basierte Services ist insbesondere zu beachten, ob die geplanten Lösungen technisch realisierbar sind.
4. **Service-Konzeption**: Hierbei handelt es sich um die Ausarbeitung von Service-, Ressourcen- und Prozess-Modellen, die es ermöglichen, das Service und den damit verbundenen gewünschten Kundennutzen bereitzustellen.
5. **Pilotierung**: Ein Prototyp des geplanten Service wird erstellt und getestet. Die Produktidee wird eine Gruppe potenzieller Kunden präsentiert, um das Potenzial zu verifizieren und das Service zu adaptieren. Die abschließenden Schritte dieser Stufe bilden das Ausrollen des Service am Markt, Trainingsmaßnahmen und weitere Aktivitäten, die dem kommerziellen Erfolg förderlich sind.
6. **Service-Controlling und Optimierung**: Die erfolgreiche Implementation des innovativen Service am Markt führt direkt zum Service Management, welches das Service-Angebot abrundet. Zusätzlich ist diese Stufe der Beginn einer fortwährenden Weiterentwicklung des Service.

Phase	Ziel	Hauptaufgaben	Werkzeuge	Ergebnisse
Strategische Analyse und Service Assessment	*Suchfelder abstecken* • *Identifikation strategischer Lücken* • *Standortbestimmung*	*Strategische Ausrichtung entwickeln*	*Systemanalyse, Kundenkontaktkreis, DL-Portfolio, SWOT*	*Überblick über alle Services im Unternehmen - klar abgegrenztes Suchfeld für neue Services*
Ideen-Generierung und Bewertung	*Serviceideen kreiert*	*Entscheidung neue Dienstleistungen entwickeln oder bestehende innovieren*	*Kreativitätstechniken (z.B. Brainstorming, 6-3-5 etc.) und Bewertungsmodelle (sticking dots, Paarvergleich etc.)*	*Neue Ideen für neue Services - Neue Ideen für bestehende Services*
Variantenbildung mit Business Cases	*Varianten-Entscheidung getroffen. Grobdesign erstellt.*	*Erstellung eines Mini-Businessplanes für die Geschäftsführung, um Go/No Go Entscheidung zu erwirken*	*Business Opportunity Description (BOD) = Mini Business Plan, Business Model Canvas*	*Grob formuliertes Geschäftsmodell für das neue Service mit Risikoabschätzung*
Service Konzeption	*Feinkonzept erstellt*	*Das Service mit allen notwendigen Ressourcen und Anforderungen ausdesignen*	*Prozessmodell, Produktmodell, Ressourcenmodell*	*Ausformuliertes Service, bereit für Erprobung*

Phase	Ziel	Hauptaufgaben	Werkzeuge	Ergebnisse
Pilotierung	*Pilotversuch durchgeführt*	*Pilotkunden identifizieren, Service erproben, Service verbessern*	*Beurteilungsbögen für Prozessablauf, Ressourcen- Einsatz und Service-attraktivität*	*Service ist bei Testkunden getestet und verbessert worden*
Service-Controlling und Optimierung	***Ziel:** Service am Markt einführen, testen und optimieren*			

Tabelle 1: Überblick über die Phasen des Strategiebasierten Vorgehensmodell zur Dienstleistungsentwicklung (basierend auf den Vorlesungsunterlagen „E-Service Engineering“ an der FH CAMPUS 02)

4 VORGESCHLAGENES STRATEGIEBASIERTES VORGEHENSMODELL ZUR SOFTWAREENTWICKLUNG

"The world has changed over the last decade from a mainly physical to a software controlled economy and the information technology has become an integral part of our industry and society in its entirety" (Marquardt 2017)

In Kapitel 2 wurde festgestellt, dass das Softwarevorgehensmodell sehr gut entwickelt ist und die Firmen die Kunst, Software zu entwickeln, meistern. Heutzutage benutzen viele Firmen eine Variation des Vorgehensmodells von Pressman und Maxim (2020). Das Modell soll agil, interaktiv und inkrementell sein. Aber die Forschungsfrage besteht weiterhin: Welche Kombination von Service Engineering und Software Engineering ist geeignet, um Softwarelösungen mit Mehrwert für Kunden und Softwareunternehmen zu generieren?

4.1 Relevanz von Innovation

Die globale Wirtschaft unterliegt vielen Herausforderungen wie z.B. demografische Veränderungen, verringerte Verfügbarkeit von Ressourcen, Globalisierung von Märkten und verschärftem internationalem Wettbewerb. Weiters verlangen stetig steigende Kunden- und Qualitätsanforderungen, technologischer Fortschritt und der damit zusammenhängende Innovationsdruck eine Neuorientierung und eine höhere Flexibilität von Firmen, um deren eigene Zukunft abzusichern. Kein Anbieter von Produkten und Dienstleistungen kann sich dem Thema Innovationsmanagement dauerhaft entziehen (Marquardt 2017) (Reichwald und Schaller 2003).

Innovation gilt seit jeher als Schlüssel zu Wachstum und Unternehmenserfolg. Innovation ist oft ein nicht klar definierter Begriff. Es geht stets um etwas „neues": ein neues Produkt, eine neue Dienstleistung, ein neues Verfahren, ein neuer Vertriebsweg usw. Innovation kann in zwei Dimensionen unterschieden werden (Reichwald und Schaller 2003):

- Die prozessorientierte Dimension hat eine zeitliche Dimension und umfasst alle Aktivitäten von der Ideenfindung bis zur Erreichung der Marktreife.
- Die ergebnisorientierte Dimension bezieht sich auf das Resultat des Innovationsprozesses und fragt „Was ist neu?", „Für wen ist es neu?" und „Wie sehr ist es neu?"

Marquardt (2017) führt als große Herausforderungen für Unternehmen an, dass diese agil, fortschrittlich und visionär sein sollen. Um dies zu erreichen, benötigen Unternehmen eine Möglichkeit, um neue Chancen zu identifizieren, um die bestehenden Geschäftsfelder weiter zu entwickeln und auszubauen. „Smart Services" werden zusammen mit Begriffen wie „Digitalisierung", „Internet der Dinge", „Industrie 4.0" und „Big Data" als mögliche Antwort auf diese Herausforderungen genannt, damit Unternehmen Innovationen realisieren können.

Wenn Innovationen entwickelt werden, gibt es neben Chancen allerdings auch das Risiko des Misserfolgs. Viele Studien zeigen, dass sich je nach Branche 30-50 Prozent aller am Markt eingeführten Innovationen als „Flops" erweisen und wieder vom Markt genommen werden. Um

Flops zu vermeiden, müssen Innovationen geplant werden. Innovationsmanagement ist substanziell anders als das Management von Routineentscheidungen. Dazu kommt, dass Dienstleistungsinnovationen im Vergleich zu Produktinnovationen mit zahlreichen weiteren Herausforderungen konfrontiert sind. Innovationsprozesse für Dienstleistungen werden in der Praxis oft unvollständig und unstrukturiert durchgeführt. Gerade die marktorientierten Aktivitäten, die eine explizite Kundenintegration erfordern und einen großen Beitrag zum Erfolg der Innovation liefern, werden oft weggelassen. Die Gestaltung der Marktorientierung sollte die Verknüpfung von Marktbedürfnis („Market pull“) und technologischen Chancen („Technology push“) sein. Kunden und Hersteller sollen dabei bereits möglichst früh interagieren, schon bevor die eigentliche Entwicklung beginnt. In Abbildung 6 ist ein strukturierter Prozess dargestellt, der über alle Prozessschritte hinweg Kundeninformationen einbezieht und gleichzeitig Potenziale im Unternehmen und beim Kunden aufbaut (Reichwald und Schaller 2003).

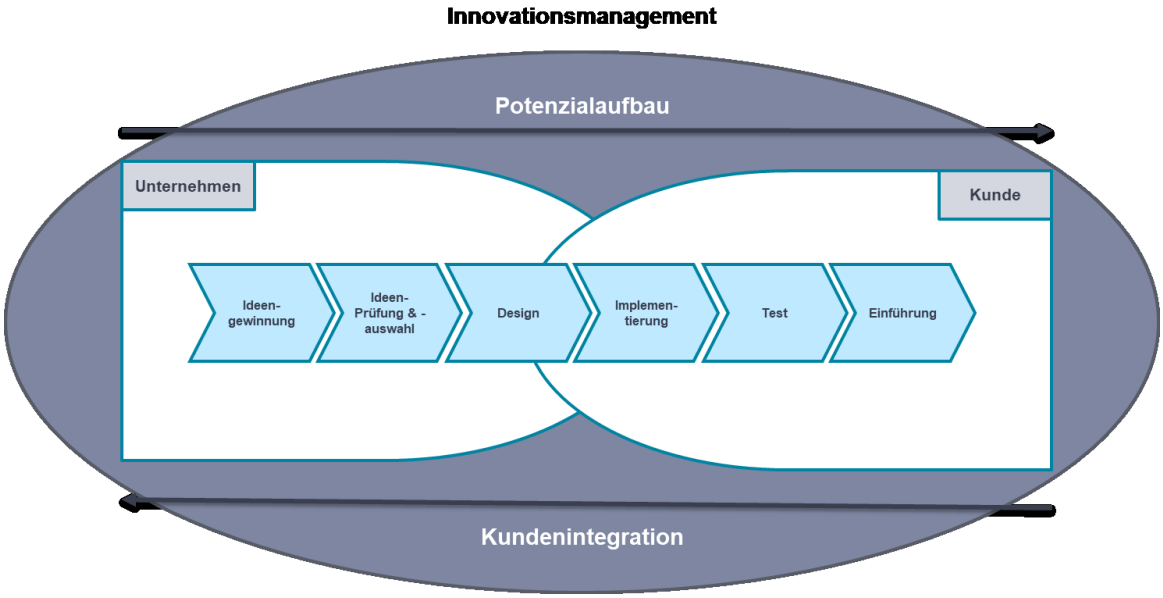


Abbildung 9: Gestaltung von Innovationsprozessen und Kundenorientierung nach Reichwald und Schaller (2006)

Laut Edvardsson et al. (2013) gibt es vier Faktoren, die nachgewiesenermaßen großen Einfluss auf NSD-Projekte (NSD: „New Service Development“) haben:

- Eine Service-Entwicklungsstrategie
- Ein formalisierter Service-Entwicklungsprozess
- Integrierte Entwicklungsteams: Team-Mitglieder kommen aus verschiedenen Abteilungen und bringen eine Vielzahl verschiedener Kompetenzen ein
- Kunden-Co-Kreation: Sammlung von Wissen über Kunden sowie Interaktion mit Kunden während der Service-Entwicklung

Für das Design von Services betonen Prestes Joly et al. (2019) die Relevanz von Multidisziplinarität. Dabei wird das Wissen verschiedener Disziplinen nebeneinandergestellt, und deren unterschiedliche Konzepte, Methoden und Herangehensweisen werden kombiniert. Dieses Potenzial wird leider oft nicht genutzt. Sechs Bereiche sollen dabei möglichst zusammenwirken, um nützliche Services zu gestalten: (a) Service-Forschung, (b) Design, (c) Marketing, (d)

Operations-Management, (e) Informationssysteme und (f) Interaktionsdesign. Diese sechs Bereiche sind folgendermaßen zu verstehen: Die Forschung steuert theoretische Grundlagen bei, um zu verstehen, wie ein Wert durch Co-Kreation in einem Service-System erzeugt werden kann. Designer haben die Aufgabe, Service Interfaces und Interaktionsmöglichkeiten zwischen den Akteuren eines Service-Systems zu gestalten. Das Marketing erlaubt ein detailliertes Verständnis des Kunden, sodass dessen Perspektive ins Zentrum des Service Design gestellt werden kann. Operations-Management ermöglicht es, das Service bereitzustellen bzw. auszurollen. Die Perspektive der Informationssysteme bringt den technologischen Aspekt ein, um das Service zu entwickeln. Das Interaktionsdesign betrifft Service-Interaktionen, wie etwa das Nutzererlebnis, aber auch Co-Design-Aktivitäten und die Visualisierung von User-Journeys in Service-Systemen.

4.2 Relevanz von Strategie

Ein Geschäftsmodell beschreibt den Plan einer Firma, wie Profite erzielt werden sollen. Geschäftsmodelle dienen dazu, fundamentale Fragen wie etwa die nach der „Value proposition“ zu beantworten: Wer sind die Kunden und wie werden Kundenwünsche erfüllt? Im Gegensatz zum Geschäftsmodell, das erklärt, wie ein Unternehmen funktioniert, dient die Geschäftsstrategie dazu, sich gegen die Konkurrenz zu behaupten, z.B. durch ein besseres Geschäftsmodell oder durch Positionierung in einem anderen Markt (Andrea Ovans 2015).

Edvardsson et al. (2013) kamen in einer Studie mit über 500 Service-Entwicklungsprojekten zu dem Schluss, dass die Service-Entwicklungsstrategie ein entscheidender Faktor für den Markterfolg neu entwickelter Services ist. Von den 500 NSD-Projekten in dieser Studie wurden 43% der Services aufgrund schlechter Verkaufszahlen vom Markt wieder zurückgezogen. Eine potenzielle Hürde zu besserer NSD-Performance liegt in einem fehlenden Verständnis der tatsächlichen Komplexität von NSD. Manager unterliegen oft der Einschätzung, NSD ausreichend verstanden zu haben und Kundenwissen bereits gut nutzen. Sie verzichten daher auf eine NSD-Strategie, was sich als fatal erweisen kann. Für Edvardsson et al. (2013) ist eine Dienstleistungsentwicklungsstrategie deshalb das fehlende Glied („Missing link“), um die Entwicklung von neuen Services zu verbessern.

Auch Kreuzer und Aschbacher (2011) kommen in einer empirischen Studie mit über 500 Unternehmen in Deutschland, Schweden und der Schweiz zum Ergebnis, dass der Einsatz einer Service-Entwicklungsstrategie den größten Einfluss auf die Service-Entwicklungs-Performance hat.

In einer Befragung steirischer IT-Unternehmen wurde festgestellt, dass in beinahe 50% aller Fälle überhaupt kein formalisierter Prozess und in insgesamt etwa 80% ein unzureichender Prozess eingesetzt wurde, um neue IT-basierte Services zu entwickeln (Kreuzer und Aschbacher 2011).

Fehlende formalisierte Prozesse für die Innovation in IT-basierte Dienstleistungen sind besonders bemerkbar in KMUs. Höber et al. (2015) sind in ihrem Artikel zu zwei relevante Ergebnisse gekommen:

- Wenige KMUs befolgen standardisierte (d. h. formalisierte und/oder systematische) Verfahren zur Entwicklung, zum Betrieb und zur Messung der Leistung bestehender oder neuartiger Dienste; und
- Selbst KMU, die über definierte Methoden und Techniken verfügen, setzen diese aus KMU-bedingten Gründen (Time-to-Market-Beschränkungen, Ressourcen- und/oder Kapazitätsengpässe usw.) nicht im täglichen Geschäft ein.

Der Einsatz einer strategiebasierten Service-Entwicklungs-Vorgehensweise bringt folgende allgemeinen Eigenschaften mit sich (Kreuzer und Aschbacher 2011):

- Geringere Zeit vom Design-Konzept zur Markteinführung
- Geringeres Fehlerrisiko
- Positive Auswirkung auf die Gesamtqualität von Services
- Frühere Möglichkeiten zu laufenden Verbesserungen von Services
- Vereinfachtes After-Sales-Service
- Höhere Life-Cycle-Rentabilität
- Behebt das Design-Dilemma und hilft, sämtliche strategische Ziele zu erreichen

4.3 Das strategiebasierte Vorgehensmodell zur Softwareentwicklung

New Service Development (NSD) wird häufig als Ad-hoc-Prozess kritisiert, der mit Improvisation und versteckten Kosten verbunden ist (Edvardsson et al. 2013).

Kreuzer und Aschbacher (2011) beobachteten in einer empirischen Studie, dass fast 50% der befragten IT-Unternehmen lediglich eine Ad-hoc-Entwicklung von Services haben. Das bedeutet, dass eigentlich keine wirklichen Prozesse beschrieben werden. Es lässt sich feststellen, dass ca. 80% der befragten IT-Unternehmen über unzureichende Entwicklungsprozesse zur effizienten Entwicklung ihrer (IT-basierten) Services verfügen.

Softwareunternehmen haben sich auf die eine oder andere Weise bereits in Bezug auf die Erstellung von Software organisiert, viele verwenden bereits agile, iterative und inkrementelle Prozesse (z. B. Scrum).

Jedes Softwareprojekt wird durch eine geschäftliche Notwendigkeit ausgelöst – die Notwendigkeit, einen Fehler in einer bestehenden Anwendung zu beheben; die Notwendigkeit, ein „Legacy-System“ an ein sich änderndes Geschäftsumfeld anzupassen; die Notwendigkeit, die Funktionen und Merkmale einer bestehenden Anwendung zu erweitern; oder die Notwendigkeit, ein neues Produkt, eine neue Dienstleistung oder ein neues System zu entwickeln (Pressman und Maxim 2020).

Das Softwareentwicklungs-Vorgehensmodell startet mit der Projekt Konzeption, aber wie soll diese ausschauen?

In der Praxis kommen die Unternehmen zu Innovation durch Wunsch des Kunden, eine super Idee von Mitarbeitern, erkannte Trends in Markt, Produkte von Konkurrenten, etc.

Sowohl Pressman und Maxim (2020) als auch IEEE Computer Society erklären das Projekt Konzeption nicht ausführlich.

Was genau passiert hier, gibt es ein Vorgehen dafür, eine strukturierte Denkweise? Wie kommt man zu einer Software Idee und wie weiß man, ob dadurch Wert für die Firma und die Kunden schaffen wird?

Laut Bullinger und Scheer (2006) wird die Notwendigkeit eines systematischen Vorgehens im Rahmen der Vorbereitung angesichts einer Vielzahl gescheiterter Dienstleistungen deutlich, bei denen unternehmensseitig einzelne oder mehrere Entwicklungsschritte nicht oder nur unzureichend bearbeitet wurden.

Die Entwicklung von Leistungen (im vorliegenden Fall IT-basierten Dienstleistungen) eines Unternehmens soll nicht isoliert ohne strategischen Kontext passieren (Aschbacher 2014). Aus Software-Engineering-Erkenntnissen scheint es, dass der Konzeption-Phase mehr Aufmerksamkeit geschenkt werden sollte. Diese Phase soll strategische Aspekte der beabsichtigten Innovationen einbeziehen.

In Kapitel 3 wurde das Thema Service Engineering beleuchtet und festgestellt, dass die Service-Konzeption ein umfangreicher und strukturierter Vorgang ist.

Das wird auch am Vorgehensmodell des FH CAMPUS 02 deutlich, in dem die ersten 3 Phasen sich intensiv und strukturiert mit der Projekt-Konzeption beschäftigten. Basierend auf dem Business Modell wird die Strategie der Firma analysiert und es werden strategische Lücken entdeckt, woraus verschiedene Ideen für neue Services entstehen, bewertet und selektiert werden, um darauf aufbauend einen Mini Business Plan zu generieren. Dieser wiederum dient als Basis für eine Go/No-Go-Entscheidung – eine strukturierte und strategische Vorgehensweise, die für die Software Entwicklung benutzt werden kann.

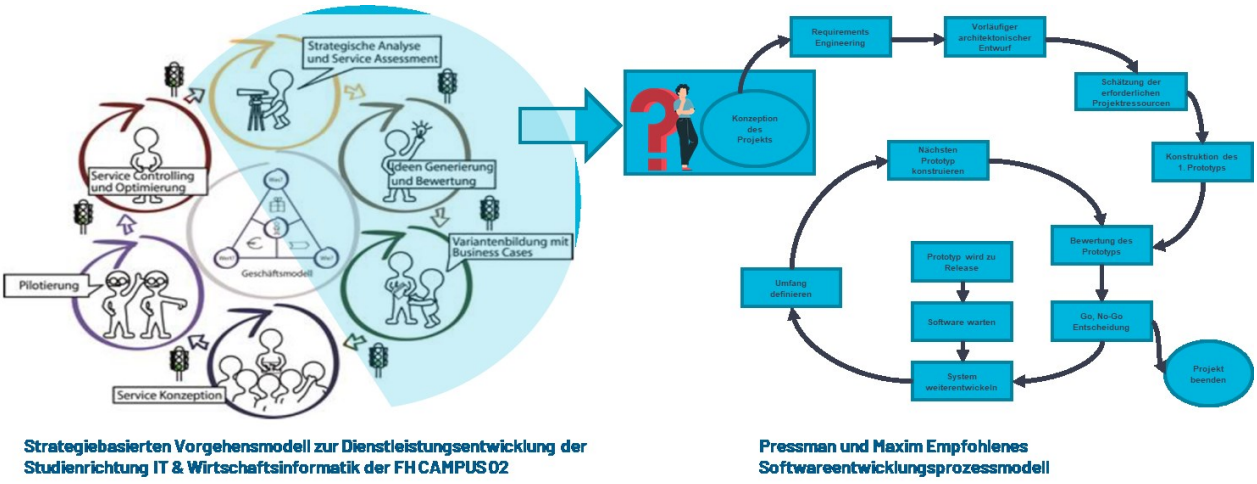


Abbildung 10: Kombination von Software- und Serviceengineering

Zusätzlich gilt für die Betrachtung von Software als eine IT-basierte Dienstleistung, dass die Entwicklung von Software eine Verbindung zwischen Software- und Service Engineering erfordert.

Wie von Meyer und van Husen (2007) berichtet, sind für die Entwicklung hochgradig verzahnter, hybrider Dienstleistungs- und Informationsprodukte traditionelle Vorgehensmodelle und Methoden nicht ausreichend, um den Ansprüchen eines systematischen Engineering gerecht zu werden. Was in Forschung und Praxis bisher nicht entwickelt und eingesetzt wurde, sind Vorgehensmodelle, die die integrierte Entwicklung von Software und Dienstleistungen explizit behandeln.

Ein Vorgehensmodell zur Entwicklung komplexer IT-basierter Dienstleistungen muss daher folgende Anforderungen erfüllen (Meyer und van Husen 2007):

- Allgemeingültigkeit – das Vorgehensmodell muss den Erstellungsprozess einer IT-basierten Dienstleistung umfassend beschreiben.
- Anwendbarkeit – der Entwicklungsprozess des Produktes muss durch das Vorgehen konstruktiv begleitet werden können. Insbesondere müssen Elemente des Software-Engineerings und des Service-Engineerings Beachtung finden.
- Anpassbarkeit – das Vorgehensmodell muss für eine Vielzahl möglicher Entwicklungsprojekte mit unterschiedlicher Ausprägung im Spannungsfeld zwischen Dienstleistungs- und Softwareentwicklung verwendbar sein.
- Zielorientierung – das Ergebnis des Entwicklungsprozesses ist eine IT-basierte Dienstleistung.
- Wiederverwertbarkeit – sowohl das Vorgehensmodell selbst als auch die während des Erstellungsprozesses einer Dienstleistung erstellten Informationen müssen geeignet gespeichert werden und bei Bedarf verfügbar sein.

Sind diese Anforderungen erfüllt, kann das Vorgehensmodell im Rahmen der Entwicklung einer IT-basierten Dienstleistung eingesetzt werden (Meyer und van Husen 2007).

Somit gelangen wir durch Zusammenführen des FH CAMPUS 02 Dienstleistungsentwicklungsmodells mit einem Software-Entwicklungsmodell zum Vorschlag des folgenden Modells:

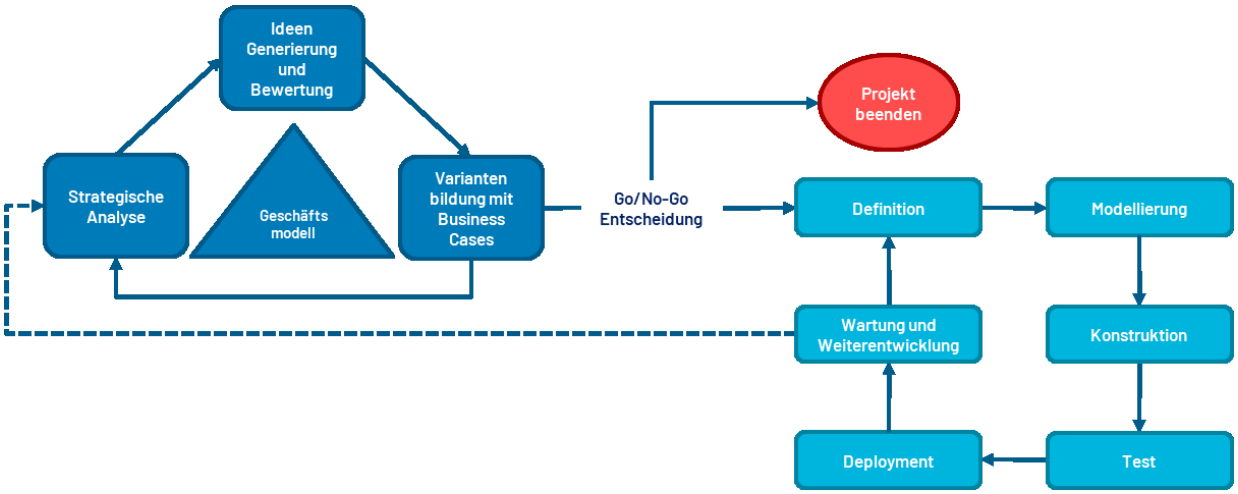


Abbildung 11: Das vorgeschlagene strategiebasierte Vorgehensmodell zur Softwareentwicklung

Dieses Modell richtet sich für die Konzeptionsphase nach dem strategiebasierten Vorgehensmodell zur Dienstleistungsentwicklung der Studienrichtung IT & Wirtschaftsinformatik der FH CAMPUS 02 und wird ergänzt durch ein iteratives, inkrementelles und agiles Softwareentwicklungsvorgehensmodell.

4.3.1 Strategische Analyse

Es ist wichtig, mit einer strategischen Sichtweise zu beginnen. Die strategische Analyse ermöglicht die Standortbestimmung und auch die Identifikation strategischer Lücken.

Die zentrale Frage ist, ob das Leistungsportfolio wirklich zu den Zielen der Organisation passt, und ob die geplante Dienstleistung in den aktuellen Produktmix passt.

Zur Analyse der strategischen Position gibt es verschiedene Methoden und Werkzeuge wie Systemanalyse, Kundenkontaktkreise, DL-Portfolio, Business Model Canvas, Konkurrenzanalyse und SWOT-Analyse.

Die Auswahl von Tools ist der Firma überlassen. Kreuzer et al. (2011) empfehlen, dass aus den verschiedenen möglichen Tools solche auszusuchen, die bereits in der Firma verwenden wurden. Der große Vorteil ist, dass man mit diesen Methoden vertraut ist und entsprechende Fähigkeiten bereits vorhanden sind. Daher geht es schneller, als eine völlig unbekannte und neue Methode anzuwenden.

Diese Phase endet mit der SWOT-Analyse, die die internen und externen Aspekte, die das Serviceangebot beeinflussen, integriert. Die SWOT-Analyse ermöglicht den Unternehmen eine strategische Ausrichtung zu entwickeln und schafft ein klar abgegrenztes Suchfeld für neue Services.

4.3.2 Ideen-Generierung und Bewertung

Nach der Identifikation des Suchfeldes für strategische Innovationen geht es im darauffolgenden Schritt darum, Ideen systematisch zu generieren und zu bewerten. In dieser Phase können Brainstorming, Brainwriting, Sticking Dots, Paarvergleich und andere kreative Methoden eingesetzt werden.

Die zentrale Aufgabe ist es, zu entscheiden, welche Ideen für Innovation (neue oder bestehende Software) Priorität haben und mit der Unternehmensstrategie im Einklang stehen.

Es ist wichtig, dass Ideen nicht kritisiert werden, egal wie unkonventionell sie sind. Eine systematische Bewertung aller Ideen ermöglicht den Vergleich von Konzepten.

Laut Kreuzer und Aschbacher (2011) ist die zentrale Frage, ob die Service-Idee wirklich zu den Zielen der Organisation passt und ob die geplante Dienstleistung in den aktuellen Produktmix passt.

4.3.3 Variantenbildung mit Business Cases

Sowohl in Phase 1 als auch in Phase 2 werden ausgewählte Business Cases für die Entscheidungsfindung von Innovationsprojekten erarbeitet. Phase 3 – Variantenbildung mit Business Cases – hat als Hauptaufgabe die Erstellung eines Mini-Businessplanes. Diese „Business Opportunity Description“ umfasst Markt-, Wettbewerbs- und Kundeneigenschaften sowie die Analyse des potenziellen Umsatz-, Kosten- und Gewinnbeitrags.

Am Ende der Phase 3 ergibt sich ein grob formuliertes Geschäftsmodell für das neue Service oder für die Weiterentwicklung bestehender Services, das als Grundlage für die Geschäftsführung dient, um eine Go- oder No-Go-Entscheidung zu erwirken.

Falls die Geschäftsführung sich für ein No-Go entscheidet, wird diese konkrete Idee verworfen und eine andere Business Opportunity Description erarbeitet. Bei Bedarf wird erneut mit Phase 1 oder Phase 2 gestartet, um neue Ideen zu sammeln.

In fall eine Go-Entscheidung geht es weiter mit der Softwareentwicklung. Zu beachten ist, dass der Teil des Softwareentwicklungsvorgehensmodells selbst iterativ und inkrementell ist, d. h. dieser Zyklus kann sich so oft wie nötig wiederholen. Es kann sein, dass einer der Zyklen eine große Veränderung in der Weiterentwicklung erzeugt, sei es durch den Einsatz neuer Technologien, die Implementierung neuer Funktionalitäten oder die Anpassung der Software an einen neuen Zielmarkt. Wenn dies der Fall ist, kann der Prozess mit der strategischen Analyse fortfahren, in der erneut gründlich reflektiert wird, ob eine so große Änderung mit dem Geschäftsmodell des Unternehmens übereinstimmt, ob sie strategisch sinnvoll ist, ob sie Wert für das Unternehmen und seine Kunden generiert und welche Risiken mit der Umsetzung verbunden sind.

4.3.4 Definition

In der Definitionsphase der Softwareentwicklung werden die geschäftlichen Anforderungen strukturiert erfasst. Dazu werden alle Stakeholder miteinbezogen.

Es geht um die Spezifikation der Software Anforderungen – Requirements Engineering – und umfasst folgende Aktivitäten:

- Anforderungen von allen Beteiligten sammeln
- Anforderungen dokumentieren
- Akzeptanzkriterien für die Anforderungen der Stakeholder beschreiben lassen
- Anforderungsverwaltung

4.3.5 Modellierung

Aus den gesammelten Anforderungen wird die passende Softwarearchitektur modelliert und die gesamte Software Struktur wird entwickelt.

Es geht um das Erfassen der gesamten Softwarelösung einschließlich Daten- und Datenbankstruktur und Design.

Dafür erstellen Softwarearchitekten und -ingenieure Modelle, bewerten Lösungsalternativen anhand nichtfunktionaler Anforderungen und dokumentieren die Architekturentscheidungen.

Nachdem die Softwarearchitektur erstellt wurde, ist es Zeit, die erforderlichen Projektressourcen zu schätzen.

Dafür werden historische Daten verwendet, um die Zeit für die Fertigstellung jeder User Story zu schätzen. Eine Teilmenge der für die Stakeholder wichtigsten User Stories wird für die erste Iteration in der Konstruktionsphase ausgewählt.

4.3.6 Konstruktion

In der Konstruktionsphase geht es um die Implementierung der in der vorherigen Phase spezifizierten Softwarearchitektur.

Das Entwicklungsteam erstellt Code, um die modellierten Anforderungen in Software zu übersetzen.

Die ausgewählte Teilmenge der User Stories werden realisiert, um eine funktionsfähige Software an die Testphase zu übergeben.

4.3.7 Test

Testen ist eine wichtige Phase die in der Softwareerstellung nie übersprungen werden soll. Sie ist eine entscheidende Phase für die Entwickler, denn wenn Fehler in Code festgestellt werden, ist eine Wiederholung der Konstruktionsphase nötig.

Funktionalen Tests wie Integrationstests, Unit-Tests, Systemtests und Akzeptanztests sowie nicht-funktionale Tests werden durchgeführt, um sicher zu gehen, dass die Software fehlerfrei und wie von allen Stakeholdern erwartet funktioniert.

Die Testfälle werden schon während der Konstruktion entwickelt, und nun wird die Software mit geeigneten Benutzern getestet, um Feedback zu erfassen.

Die Bestimmung der Qualität der aktuellen Interaktion ermöglicht eine Go- oder No-Go-Entscheidung über die Fortsetzung der Entwicklung in der Deployment Phase.

4.3.8 Deployment

In der vorherigen Phase wurde die Qualität der bisher erstellten Software bestimmt. In der Deployment Phase geht es um die Bereitstellung der Software in Produktion, der fertige Code der aktuellen Iteration wird in die Software integriert.

Wenn die bestehende Software schon funktionsfähig ist, wird das Deployment zu einem Release, d. h. die Software wird veröffentlicht und kann den Anwendern zur Verwendung freigegeben werden.

4.3.9 Wartung und Weiterentwicklung

Die Software Entwicklungsprozess endet nicht mit der Veröffentlichung an die Endbenutzer.

Sobald mehrere Benutzer beginnen, mit der Software zu interagieren, können zuvor nicht erkannte Probleme und Änderungswünsche auftreten.

In dieser Phase wird Support, Wartung und Weiterentwicklung geleistet, um die Software funktionsfähig und auf dem neuesten Stand zu halten. Basierend auf realen Eingaben der Benutzer wird das Software-Service aktualisiert und verbessert.

Es ist sehr wichtig, die Software in Reaktion auf das Feedback der Endbenutzer stets zu warten und weiterzuentwickeln.

Es kann sein, dass neue Funktionen oder zusätzliche Anforderungen entstehen. Für die Softwareentwicklung heißt das, eine neue Iteration zu starten.

Falls der Umfang dieser Weiterentwicklung sehr groß und aufwändig ist, ist es zu empfehlen, den Prozess wieder mit der strategischen Analyse zu starten, um zu überlegen, ob solch eine Weiterentwicklung in Sinne der Firmenstrategie ist und Wert schaffen wird, nicht nur für die Kunden, sondern auch für die Firma selbst.

5 ANWENDUNG DES STRATEGIEBASIERTEN VORGEHENSMODELLS ZUR SOFTWAREENTWICKLUNG IN DER PRAXIS

Die Diskussion um eine Entwicklungssystematik für Dienstleistungen findet nicht nur auf akademischer Ebene statt, sondern wird auch verstärkt zwischen Praktikern geführt. Die Forderung der Unternehmen nach praxisorientierten Lösungsansätzen wird umso lauter, je stärker sich Wertschöpfung und Gewinnmargen zu Gunsten des Dienstleistungsbereichs verschieben (Bullinger und Scheer 2006).

5.1 Ausgangssituation

Die Firma Bartelt GmbH zählt zu den führenden Unternehmen in Österreich in den Bereichen Laborgeräte, Laborverbrauchsmaterialien und Laboreinrichtungen. Der Hauptsitz befindet sich in Graz, weiteres bestehen Standorte in Wien, Linz und Innsbruck. Das Unternehmen wurde 1936 als Familienbetrieb gegründet und wird heute von Fritz Bartelt in dritter Generation geführt (Bartelt GmbH.)

Die Firma liefert Produkte und Dienstleistungen rund um Laborgeräte und -einrichtungen in Österreich.

Die Software-Abteilung der Bartelt GmbH arbeitet seit über 30 Jahren an Laborsoftwarelösungen für verschiedene Labors. Die Abteilung bietet Webapplikationen für die Optimierung der täglichen Laborarbeit, individuelle Laborsoftwarelösungen für spezielle Anforderungen oder Laborinformationssysteme (LIMS) für klinisch-diagnostische Krankenhauslabore an (Bartelt G.m.b.H.).

Die Abteilung genießt eine relative Autonomie und Unabhängigkeit in Bezug auf strategische Entscheidungen und die zu entwickelnden Produkte.

Die Abteilung besteht aus 11 Mitarbeitern und ist wie folgt strukturiert:

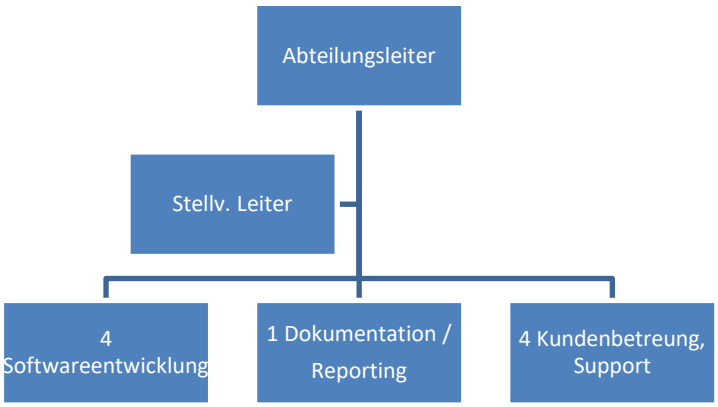


Abbildung 12: Organigramm Softwareabteilung

Neben den kundenspezifischen Individual-Softwarelösungen entwickelt die Abteilung folgende Softwarelösungen:

- **datalabX –** ein Laborinformationssystem für Krankenhäuser, das in über 20 Krankenhäusern in Österreich täglich im Einsatz ist.
- **chemXPERT –** Softwarelösung zur Verwaltung und Dokumentation von Chemikalien.
- **smplXPERT –** Softwarelösung für das Lagern und Verwalten von Proben

Die Entwicklung des Produkts datalabX war der Auslöser für die Gründung der Softwareabteilung. Es entstand aus der Anregung eines Kunden.

Im Laufe der Zeit entstand das Bedürfnis einer größeren Vielfalt der angebotenen Produkte. Die Abteilung war bis dahin ausschließlich von datalabX abhängig, und es war wünschenswert, andere Standbeine aufzubauen, und so wurden die Produkte smplXPERT und chemXPERT ins Leben gerufen.

Da die neuen Softwarelösungen entwickelt wurden, um eine Lücke im Markt zu adressieren, wurde erwartet, dass sie sich von selbst verkaufen würden. Allerdings wurden keine strukturierten strategischen Überlegungen angestellt. Zwar erkennen viele potenzielle Kunden den Nutzen dieser Produkte, allerdings handelt es sich häufig um kleine Labore oder Universitätslabore, die jeweils ein sehr enges Budget zu Verfügung haben und daher diese Produkte am Ende selten kaufen.

5.2 Praktische Anwendung

Der Praxisteil wird in der Softwareabteilung der Firma Bartelt durchgeführt.

Nach Phase 3 (Variantenbildung mit Business Cases) kann bereits eine Go/NoGo-Entscheidung für jede Idee getroffen werden. Damit ist es klar, ob das Vorgehensmodell eine Entscheidungsgrundlage bietet. Ebenso zeigt sich bereits, ob der Prozess selbständig durchgeführt werden kann. Der reine Softwareentwicklungsteil des Vorgehensmodells ist bereits bekannt und wird in der Firma praktiziert.

Aus diesen Gründen und angesichts der Knappheit von Zeit und Ressourcen konzentriert sich der praktische Teil auf die Umsetzung der Schritte 1 bis 3 des Prozesses.

Folgende Aktivitäten wurden durchgeführt:

- Kickoff Meeting
- Strategische Analyse
- Ideen-Generierung und Bewertung
- Variantenbildung mit Business Cases
- Feedback-Fragebogen

5.2.1 Kickoff Meeting

Als erstes wurden Thema und Kontext der Masterarbeit, verschiedene theoretische Begriffe sowie die drei geplanten Phasen präsentiert.

Als nächstes stand die Auswahl der Tools für die strategische Analyse auf der Agenda. Wie bei Kreuzer et al. (2011) empfohlen, wurden Tools aus den diversen möglichen Tools ausgesucht, welche den Anwesenden bereits bekannt waren oder bereits in der Firma verwendet wurden.

Somit wurde gemeinsam entschieden, folgende Tools für die strategische Analyse einzusetzen:

- Konkurrenzanalyse
- DL-Portfolio
- Business Model Canvas
- SWOT-Analyse

5.2.2 Strategische Analyse

Ziel dieser Phase ist es die Identifikation strategischer Lücken, in großen und ganzen wird eine Standortbestimmung der Abteilung durchgemacht.

5.2.2.1. Konkurrenzanalyse

In der Konkurrenzanalyse werden die Stärken und Schwächen der Mitbewerber mit den eigenen nach selbst definierten Kriterien bewertet und zeigen Aufholbedarf bzw. Vorsprung.

Der Abteilungsleiter erstellte eine Liste der Mitbewerber. Daraufhin erfolgte ein Desk Research über diese Konkurrenten.

Da nicht besonders viele Informationen über Kennzahlen dieser Unternehmen im Internet zu finden waren, wurde beschlossen, statt einer Konkurrenzanalyse eine Konkurrenzübersicht zu erstellen, in der verschiedene Informationen gesammelt wurden; siehe ANHANG A.

5.2.2.2. Dienstleistungsportfolio

Das Dienstleistungsportfolio (DL-Portfolio) stellt alle Dienstleistungen der Abteilung dar inklusive einer Kategorisierung nach verschiedenen Kriterien.

Die Grundbestandteile eines DL-Portfolio sind:

- Name der Dienstleistung
- Standardisierung / Einzigartigkeit: Standardprodukt versus individuell angepasste Leistung
- Betroffene Phasen aus dem Kundenkontaktkreis: Informationsphase, Angebots-/Kaufphase, Auftragsausführungsphase, Nutzungsphase

- Aktivität/Dienstleistung: z.B. Website, Smart Service
- Strategische Bedeutung: z.B. Höherer Preis für Grundleistung; Prozess Optimierung, Kostenreduzierung; Verbesserung der Zahlungsbereitschaft
- Wie berechnet? Fixer Preis, Preis wird verhandelt, Leistung wird verschenkt, ist versteckt.
- Kurzbeschreibung

Diese Grundbestandteile können mit folgenden Kriterien ergänzt werden:

- Zahlungsbereitschaft (gering/mittel/hoch)
- Nachfrage (gering/mittel/hoch)
- Erbringungskosten (gering/mittel/hoch)
- Kano-Leistungsmerkmal (Basis-/Leistungs-/Begeisterungsmerkmal)
- BCG-Matrix (Question Marks/Stars/Cashcows/Poor Dogs)
- Erbringungspotenzial (alleine/Kooperation/nur extern)

Die Unterteilung und Klassifizierung der Softwarelösungen und deren Services ist nach intensiven Diskussionen entstanden. Siehe ANHANG B.

5.2.2.3. Business Model Canvas

Ein Geschäftsmodell beschreibt wie eine Organisation Wert schafft, liefert und erfasst. Ein Business Model Canvas ist eine Methode, mit der verschiedene Aspekte eines Geschäftsmodells abgebildet und analysiert werden können (Osterwalder und Pigneur 2013).

Osterwalder und Pigneur (2013) glauben, dass ein Geschäftsmodell am besten durch neun grundlegende Bausteine beschrieben werden kann, die die Logik zeigen, wie ein Unternehmen beabsichtigt, Geld zu verdienen. Die neun Blöcke decken die vier Hauptbereiche eines Unternehmens ab: Kunden, Angebot, Infrastruktur und Finanzierbarkeit. Das Geschäftsmodell ist wie ein Blueprint für eine Strategie, die durch organisatorische Strukturen, Prozesse und Systeme umgesetzt werden soll.

Die neun Blöcke sind:

- Kundensegmente – das sind die Gruppen von Personen und/oder Organisationen, die ein Unternehmen oder eine Organisation mit einem dedizierten Wertversprechen erreichen und Werte schaffen möchte
- Wertversprechen – basierend auf einem Bündel von Produkten und Dienstleistungen, die für ein Kundensegment Wert schaffen
- Marktkanäle – beschreiben, wie ein Wertversprechen kommuniziert und einem Kundensegment über Kommunikations-, Vertriebs- und Verkaufskanäle geliefert wird

- Kundenbeziehungen – skizzieren, welche Art von Beziehung mit jedem Kundensegment aufgebaut und gepflegt wird, und sie erläutern, wie Kunden gewonnen und gehalten werden
- Schlüsselpartner – zeigt das Netzwerk von Lieferanten und Partnern, die externe Ressourcen und Aktivitäten einbringen
- Schlüsselaktivitäten – sind die wichtigsten Aktivitäten, die eine Organisation braucht, um gut zu funktionieren
- Schlüsselressourcen – sind die wichtigsten Vermögenswerte, die erforderlich sind, um die zuvor beschriebenen Elemente anzubieten und zu liefern
- Kosten – beschreibt alle Kosten, die für den Betrieb eines Geschäftsmodells anfallen
- Einnahmen – Ergebnis eines Wertversprechens, das einem Kundensegment erfolgreich angeboten wurde. Auf diese Weise erfasst ein Unternehmen den Wert mit einem Preis, den die Kunden zu zahlen bereit sind.

Die Abteilung hatte schon ein bestehendes Business Model Canvas aus dem Jahr 2014, dieses wurde analysiert und aktualisiert. Siehe ANHANG C.

5.2.2.4. SWOT-Analyse

Die strategische Analyse-Phase endet mit der SWOT-Analyse, die die internen und externen Aspekte der Abteilung integriert.

Die SWOT-Analyse dient der Strategiefindung. Dazu werden die Systemumwelt und die Ist-Situation der Abteilung betrachtet, um Maßnahmen abzuleiten, die wiederum der Definitionsfindung von strategischen Zielen dienen.

Da die Ideen aller Mitarbeiter wichtig sind, wurde beschlossen, dass diese in der SWOT-Analyse teilnehmen sollen.

Als Vorbereitung für die SWOT-Analyse wurde an alle Mitarbeiter ein Dokument mit SWOT-Fragen zur Beantwortung übermittelt (siehe ANHANG D). Die Absicht war dabei, jedem Teilnehmer genügend Zeit zu geben, um alleine (ohne Einfluss von Kollegen) reflektieren zu können.

Einmal im Jahr nimmt sich die Abteilung einen Tag Zeit für eine interne Klausur, um gemeinsam über das vergangene Jahr zu reflektieren und das nächste Jahr zu planen.

In der Klausur gab es eine Präsentation der strategischen Analyse, währenddessen die Ergebnisse der bereits implementierten Werkzeuge präsentiert, offen diskutiert und aktualisiert wurden. Dabei wurden auch alle Antworten auf die SWOT-Fragen vorgestellt, diskutiert und priorisiert.

Am Ende dieser Session konnte man sich gemeinsam auf jeweils 5 Stärken, Schwächen, Chancen und Risiken einigen. Daraufhin wurde ein Termin für die SWOT-Analyse vereinbart, um

Stärken, Schwächen, Chancen und Risiken zu kombinieren und daraus Maßnahmen sowie strategische Ziele abzuleiten.

In einer dreistündigen Einheit mit allen Mitarbeitern der Abteilung wurde die SWOT-Analyse finalisiert (siehe ANHANG E).

5.2.3 Ideen-Generierung und Bewertung

Aus der SWOT-Analyse wurden im Rahmen einer Brainstorming-Session konkrete Maßnahmen, Projekte und Aktivitäten abgeleitet.

SWOT Matrizen	Maßnahme/Projekt	Verantwortliche	Termin	Überprüfbares Ergebnis
S4-O1; S5-O1; W2-O1; W2-T2	Bartelt für Vertrieb von chemXPERT	LÖRO	Laufend	1 chemXPERT bis ende des Jahres verkauft
S4-O1; S5-O1; W2-O1; W2-T2	Partnerschaft für Vertrieb von DLX im DACH-Raum	PS	Ende März	Partner Firma gefunden
S3-O4	Skalierbares dlX inklusive Mandantenfähigkeit umsetzen	PS	2025	
S3-O3	Testautomatisierung für dlX	PAU	Ende 2022	10 testcases automatisiert
S3-O3	Testautomatisierung für OX	STE	Ende 2022	Selenium für Routine Testcase funktioniert
S4-O4	Installer für DLX	PIR ?	Ende 2022	Schnittstellendienste automatisch installieren
S4-O4	Online Formular für Hotline in dlX	KAD/PIR?	Ende Q2/2022	
S3-T1; S4-T1; W1-T1	Sukzessive Approximation an ISO 13845	PS	Ende 2022	Aktionsplan formuliert (mit OBA)
S3-T4	dlX Risikoanalyse	PS	Mitte 2022	Alle gefährdeten Teile identifiziert
W2-O2	Auftrag.at richtige Schlüsselwörter eintragen	PS für dlX; LÖRO für Ox	Mitte 2022	
W5-O1-O2-O5	Bartelt-Anwalt: Lizenzvereinbarung entwickeln lassen; Externes know-how für die Verträge	LÖRO	Ende 2022	Aktualisierte Verträge; AGB;

Tabelle 2: Abgeleitet Projekte basierend auf der SWOT-Analyse

Zusätzlich wurden andere strategische Vorhaben notiert, die in der Zukunft abgearbeitet und geplant werden sollen.

SWOT Matrizen	Strategie
S1-O2:	Gute Referenzen für Ausschreibungen von Kunden bekommen
S1-T2; W1-T2	Großkunden pflegen; Kundenbasis auf breitere Beine stellen
S2-O1; S2-O2	Kunden präsentieren unsere Produkte und könnten bei Verkauf eventuell Provision bekommen
S2-O4	Wo es Bedarf gibt, gemeinsam mit Kunden, Anforderungen für große und komplexe Labors erarbeiten. Kunden Unterstützen der Projektumsetzung
S3-T3	Konsequente Weiter- und Fortbildung
S4-O1; S5-O1; S4-T2; W2-O1, W2-T2	Vertrieb aufbauen intern/extern
S1-T4; S1-T4; S4-T3	Technologie laufend aktualisieren und als Zusatzleistung verkaufen
W2-O2	Laufend und ständig an Ausschreibungen mitmachen
W2-O3	Laufenden automatisierten Vertrieb, Marketing in Social Media
W4-O1	Wo es Bedarf gibt Auftragsentwicklung extern
W4-O3; W4-O4; W3-T3-T4; W4-T1-T3-T4	Mehr Personal Ressourcen
W3-T2	LIS Partner für große Labors

Tabelle 3: Andere strategische Vorhaben

Die Teilnehmer haben sich geeinigt, dass der Verkauf von chemXPERT eine Priorität ist, die in der aktuellen Phase betrachtet werden soll.

5.2.4 Variantenbildung mit Business Cases

Variantenbildung mit Business Cases dient dazu, ein Grobkonzept zu erstellen und vorab zu bewerten. Dafür wird eine „Business Opportunity Description“ erstellt, die Markt-, Wettbewerbs- und Kundeneigenschaften und die Analyse des potenziellen Umsatz-, Kosten- und Gewinnbeitrags umfasst.

Für dieses Vorhaben war eine Zusammenarbeit mit der Marketingabteilung und den Chemikalien-Verkäufern erforderlich. Vor allem war es notwendig zu verstehen, wie der Verkaufsprozess funktioniert und in welchen Touch Points es möglich wäre, den Verkauf von chemXPERT einzubinden. Dafür wurde ein Service Blue Print erstellt (siehe ANHANG F).

Folgende Business Opportunity Description wurde als Basis für eine Go/No-Go-Entscheidung der Abteilungsleitung erstellt:

Business Opportunity Description	
Erfasser/Verantwortlicher	NP/LÖRO
Name des Projekts	chemXPERT-Verkauf mittels Chemikalien-Verkauf
Beschreibung	Der Verkauf der Softwarelösung chemXPERT soll übernommen werden vom bereits bestehenden Inhouse-Vertrieb in Zusammenhang mit dem Chemikalien-Verkaufsprozess. Die Firma hat viele Kunden, die Chemikalien kaufen, und viele davon kaufen auch Chemikalien, die als Gift klassifiziert sind (fortan als Gift bezeichnet). Es gibt verschiedene Arten von Kunden: Kunden mit jährlichen Verträgen Kunden, die mittels Order Management-System bestellen Kunden die via Email oder Telefon bestellen Sporadische Kunden via Web Shop Potenzielle Kunden für den chemXPERT sind diejenigen, die in erster Linie Gifte bestellen, sowie große Kunden mit Vertrag oder jene über das Order Management-System. Grob geschätzt handelt es sich um 10 bis 20 Vertragskunden und weiteren Kunden über das Order Management System. Folgende Werbebotschaft ist zu empfehlen („Pain reliever“): *Laut ChemG 1996 § 43; Giftverordnung 2000 § 9 Abs. 3 und 4, müssen Verwender von Giften folgende Aufzeichnungen über Herkunft und Verbleib jedes Gifts führen:* • *Bezeichnung des Gifts* • *erworbene Menge* • *Verweis auf den Beleg über den Erwerb (Lieferschein, Rechnung...)* • *Datum des Erwerbs* • *Name des Abgebers* • *verwendete Menge und Verwendungszweck, bei einer weiteren Verarbeitung auch Bezeichnung der dabei entstandenen Produkte und dafür jeweils eingesetzte Menge jedes einzelnen Gifts* *Einmal pro Jahr ist die verbleibende Menge jedes Gifts auszuweisen. Die Aufzeichnungen müssen nach der letzten Eintragung mindestens 7 Jahre aufbewahrt werden.* *Bartelt empfiehlt diese Aufzeichnungen zu führen nicht nur für Gifte, sondern für alle verwendeten Chemikalien.*

Business Opportunity Description	
	Unser chemXPERT deckt alle diese Vorschriften und viel mehr ab! Mit chemXPERT schaffen Sie jederzeit und unkompliziert einen Überblick über ihren gesamten Chemikalienbestand und erstellen aussagekräftige Auswertungen.

Grober Ablauf des Prozesses	Im Chemikalien-Verkaufsprozess (siehe Service Blueprint unten) gibt es Momente, in die Marketing/Vertrieb von chemXPERT integriert werden kann:
	• Beim ersten Kontakt mit potenzielle Kunden kann das chemXPERT-Flyer mitgegeben werden.
	• Die Werbebotschaft kann in folgenden Momenten transportiert werden:
	  ○ Bei der Ermittlung der Auftragsbestätigung.
	  ○ Bei der Chemikalien-Lieferung und Rechnungserstellung.
	  ○ Bei der Ermittlung der Auftragsbestätigung.
	  ○ Im Danke-Email nach dem Kauf.
	• Bei Kundenbesuchen von Verkaufs-Außendienstmitarbeitern in verschiedenen Phasen.

Projekt-Potenzial	Kundennutzen: • Automatisiert und gesetzeskonform • Ordnung im Labor (Einlagerung, Entnahme, Lagerbestand, Entsorgung und Bestellung dokumentieren) • Gesamter Bestand auf einen Blick • Koppelbar mit Waagen • Historiendokumentation und Daten-Export als PDF und CSV Mehrwert für Bartelt: • Umsatzsteigerung • Steigerung der Kundenbindung (Chemikalienkunden nutzen chemXPERT und kaufen mehr Chemikalien bei Bartelt direkt aus chemXPERT) • Steigerung des Images als Chemikalien-Spezialist durch die verbreitete Nutzung von chemXPERT
Finanzielle Betrachtung	Kosten: • Personal (Chemikalienverkäufer, Software Abteilung, Marketing) • Mögliche chemXPERT-Erweiterungen • IT-Infrastruktur (Hosting/Cloud Service, Lizenzen) Voraussetzungen: • Marketing-Plan erstellt (Zusammenarbeit zwischen Software-Abteilung, Marketing und Chemikalienverkauf) Grobe Umsatz Schätzung: Angenommen 10 Kunden kaufen chemXPERT um 77,90 Euro pro Monat, das ergibt ca. 10.000 Euro pro Jahr.
Empfehlung an die Geschäftsführung	Sofortige Umsetzung des Projekts, chemXPERT ist fix fertig einsetzbar, Kunden können schon bestellen.
Entscheidung der Geschäftsführung zum weiteren Vorgehen	Go, Marketing Campaign zusammen mit der Marketingabteilung erstellen und Verkaufsstrategie mit Chemikalien Verkäufer besprechen und realisieren.

Tabelle 4: Business Opportunity Description

5.2.5 Feedback Fragebogen

Es ist wichtig zu wissen, ob der bisher durchgeführte Prozess einen Mehrwert für die Abteilung bringt und ob er bei zukünftigen Gelegenheiten verwendet wird. Um dies zu beurteilen, wurde ein Fragebogen entwickelt, um das entsprechende Feedback zu erfassen.

Die meisten Fragen im Fragebogen wurden vom Technology Acceptance Model (TAM) adaptiert. TAM konzentriert sich auf zwei theoretische Konstrukte: die wahrgenommene Nützlichkeit

(perceived usefulness) und die wahrgenommene Benutzerfreundlichkeit (perceived ease of use), die als grundlegende Determinanten der Systemnutzung theoretisiert werden (Davis 1989).

Fragen 1 bis 8 betreffen die Nützlichkeit und Fragen 9 bis 13 die Benutzerfreundlichkeit. Siehe Fragebogen in ANHANG G.

Dieser Fragebogen ermöglicht die Beantwortung von zwei Masterarbeits-Hypothesen.

> ***Hypothese 2****: Es ist möglich, diesen Prozess eigenständig durchzuführen, d.h. selbst zu steuern.*

Das Konstrukt „perceived ease of use“ gibt Hinweise, ob der Prozess benutzerfreundlich ist, ob die Unternehmen den Prozess also eigenständig durchführen können.

Die wahrgenommene Benutzerfreundlichkeit bezieht sich auf das Ausmaß, in dem eine Person glaubt, dass die Verwendung eines bestimmten Systems mühelos wäre. Unter sonst gleichen Bedingungen wird ein System, das als einfacher zu verwenden wahrgenommen wird, wahrscheinlicher von Benutzern akzeptiert als ein anderes, weniger einfach zu verwendendes System (Davis 1989).

> ***Hypothese 3****: Eine Kombination zwischen Service- und Software Engineering ermöglicht es Unternehmen, Grundlagen für eine fundierte Entscheidung zu erhalten, um die Strategie einer oder mehrerer Software-Lösungen zu gestalten*

Hypothese 3 impliziert, dass der Prozess für den Entscheidungsfindungsprozess nützlich ist, somit würde das Konstrukt „perceived usefulness“ eine Antwort auf diese Hypothese liefern.

Wahrgenommene Nützlichkeit wird hier definiert als das Ausmaß, in dem eine Person glaubt, dass die Verwendung eines bestimmten Systems ihre Arbeitsleistung verbessern würde. Ein System mit hoher wahrgenommener Nützlichkeit wiederum ist eines, für das ein Benutzer an die Existenz eines positiven Nutzens-Leistungsverhältnis glaubt (Davis 1989).

Nach dem Prozess wurden der Abteilungsleiter und stellvertretende Abteilungsleiter gebeten, den Fragebogen zu beantworten, weil sie erstens aktiv in allen Phasen des Prozesses und an der Implementierung sämtlicher Werkzeuge mitwirkten, und weil sie zweitens die Entscheidungsträger der Abteilung sind.

6 ERGEBNISSE

6.1 Auswertung

Die drei strategischen Phasen wurden über einen Zeitraum von drei Monaten durchgeführt, siehe ANHANG H.

Die strategische Denkweise war durchaus neu für die Abteilung. Alle waren eingeladen, mitzumachen. Daraus entstanden viele Ideen für Aktionen und Projekte. Manche waren mehr an der Dienstleistungsentwicklung ausgerichtet und andere hatten einen stärkeren Fokus auf Softwareentwicklung.

Sehr positiv war, dass durch diese drei Schritte strategische Schwächen und auch die dazu gehörigen Lösungen identifiziert werden konnten.

Priorität hatte das Projekt „Verkauf von chemXPERT mittels Chemikalien-Verkäufern“, obwohl dieses Projekt nicht direkt Softwareentwicklung beinhaltet. Durch die Business Opportunity Description (BOD) konnte ein möglicher Lösungsweg identifiziert werden, welcher sich derzeit in Umsetzung befindet.

Das zeigt, dass das Model auch für bestehende Softwarelösungen wertvoll sein kann. Es ist aber auch ein Hinweis, dass eine frühere strategische Betrachtung wichtig gewesen wäre. Denn die Abteilung hat mit dem Produkt chemXPERT zwar eine komplette und nützliche Softwarelösung kreiert, diese war jedoch zu wenig strategisch gedacht. Dazu gehört eine zu geringe Orientierung am Markt, also die Beantwortung derartiger Fragen: Wer sind die Kunden? Wie werden sie auf das Produkt aufmerksam gemacht werden? etc.

In der Abteilung fehlt noch das Bewusstsein, strategische Überlegungen mit Softwareentwicklung in einem iterativen Prozess zu verbinden. Das ist daran erkennbar, dass viele Vorschläge zur Ausarbeitung in einer BOD entweder rein strategisch oder rein softwaretechnisch gesehen wurden.

Als direkte Folge der Erarbeitung der BOD hat sich im Gespräch mit dem Chemikalien-Verkäufer ein Feedback für die Weiterentwicklung der chemXPERT Software ergeben. Nämlich, dass die aktuelle Softwarelösung zu komplex und schwer verständlich sei. Ziel ist es nun, die Software modularer zu gestalten.

Somit beginnt in unmittelbarer Zukunft ein Softwareentwicklungs-Zyklus, und es kann durchaus sein, dass es später wieder zu einer adaptierten strategischen Ausrichtung kommt.

Um Hypothesen 2 und 3 validieren zu können, war es wichtig, Feedback einzuholen. Zur Erfassung des entsprechenden Feedbacks wurde ein Fragebogen entwickelt.

Der Fragebogen nutzt das Technology Acceptance Model (TAM) von (Davis 1985) und konzentriert sich auf zwei Konstrukte:

- Die wahrgenommene Benutzerfreundlichkeit (perceived ease of use): um Hypothese 2 – *Es ist möglich, diesen Prozess eigenständig durchzuführen, d.h. selbst zu steuern* – zu überprüfen.
- Die wahrgenommene Nützlichkeit (perceived usefulness): um Hypothese 3 - *Die Ergebnisse des strategiebasierten Vorgehensmodells ermöglicht den Firmen, Grundlagen für eine fundierte Entscheidung zu erhalten, um die Strategie einer oder mehrerer Software-Lösungen zu gestalten* – zu überprüfen.

Der Fragebogen wurde an 2 Personen geschickt, die folgende Voraussetzungen erfüllten:

- Aktive Teilnahme an allen Aktivitäten
- Mitwirkung bei der Erstellung und Implementierung sämtlicher Werkzeuge
- Entscheidungsträger der Abteilung

Somit wurde der Fragebogen vom Abteilungsleiter (AL) und dem stellvertretenden Abteilungsleiter (SA) beantwortet.

Die Roh-Ergebnisse sind in ANHANG G dargestellt.

Hinsichtlich der Benutzerfreundlichkeit wurden die folgenden Ergebnisse erzielt (siehe Tabelle 5):

…ist leicht zu verstehen und verwenden	SA	AL	Durchschnitt
DL-Portfolio	1	1	1
Business Model Canvas	1	1	1
Konkurrenzüberblick	1	1	1
SWOT-Analyse	1	1	1
Business Opportunity Description	1	1	1

Tabelle 5: Ergebnisse – Benutzerfreundlichkeit. Skala: 1 (trifft sehr zu) bis 5 (trifft gar nicht zu)

Das Ergebnis von 1 für jedes benutzte Werkzeug bedeutet, dass für die Fragebogenteilnehmer die Aussage sehr zutreffend ist, dass die Werkzeuge leicht zu verstehen und zu verwenden sind. Daraus können wir schließen, dass die verwendeten Werkzeuge sehr benutzerfreundlich sind.

Sicherlich trägt die Besonderheit des Prozesses, bei dem die Teilnehmer die für sie geeigneten Tools frei wählen können, dazu bei, dass sie relativ vertraut und leicht umsetzbar sind.

Jedes der verwendeten Werkzeuge sollte entsprechend seiner Funktion ein bestimmtes Ziel erreichen. Um die Nützlichkeit der Werkzeuge zu bestimmen, vergleichen die Fragen des Fragebogens die Verwendung mit dem spezifischen Ziel jedes Werkzeugs, d. h. die Fragen sollen überprüfen, ob die Tools die Ziele erreicht haben, für die sie verwendet wurden.

Die Ergebnisse zur Nützlichkeit lauten wie folgt (siehe Tabelle 6):

Werkzeug	Untersucht nach dem Ziel…	SA	AL	Durchschnitt
DL-Portfolio	Produkte und Services abbilden und analysieren	3	2	2.5
Business Model Canvas	Aspekte des Geschäftsmodells abbilden und analysieren	2	1	1.5
Konkurrenzüberblick	Konkurrenten besser kennen	1	2	1.5
SWOT-Analyse	Strategische Stoßrichtungen ableiten	1	2	1.5
Business Opportunity Description	Grobdesign erstellen, um ein Go/NoGo-Entscheidung zu erwirken	2	1	1.5
Alle drei Phasen	Aktuelle Abteilungs-Situation besser verstehen	2	2	2
Alle drei Phasen	Projekte, Maßnahmen und Aktionen planen	3	1	2
Alle drei Phasen	Informationen sammeln, um eine Go/NoGo-Entscheidung zu treffen	3	1	2
Gesamtdurchschnitt		2.125	1.5	**1.81**

Tabelle 6: Ergebnisse – Nützlichkeit. Skala: 1 (trifft sehr zu) bis 5 (trifft gar nicht zu)

Mit einem Durchschnitt von 1,81 können wir feststellen, dass die Instrumente die Ziele erreicht haben, für die sie eingesetzt wurden. Das DL-Portfolio wurde von einem der Befragten neutral (Note 3) bewertet, alle anderen Instrumente wurden jedoch positiv bewertet. Insbesondere die Nutzung der drei strategischen Phasen wurde mit der Note 2 bewertet, was bedeutet, dass die Befragten deren Nutzung für sinnvoll halten um die aktuelle Situation der Abteilung besser zu verstehen, Projekte, Maßnahmen und Aktionen zu planen und schlussendlich wichtige Informationen zu erhalten, um eine fundierte Go/NoGo-Entscheidung zu treffen.

6.2 Betrachtung der Hypothesen

Hypothese 1: Eine Kombination von Service- und Software-Engineering ist möglich, um ein strategiebasiertes Softwareentwicklungsmodell zu formulieren.

Die Hypothese wurde in Kapitel 4 theoretisch ausgearbeitet. Durch Zusammenführen des FH CAMPUS 02 Dienstleistungsentwicklungsmodells mit einem Software-Entwicklungsmodell gelangen wir zu einem strategiebasierten Vorgehensmodell zur Softwareentwicklung.

Das kombinierte Vorgehensmodell orientiert sich für die Konzeptionsphase am strategiebasierten Vorgehensmodell zur Dienstleistungsentwicklung der Studienrichtung IT und Wirtschaftsinformatik der FH CAMPUS 02 und wird durch ein iteratives, inkrementelles und agiles Softwareentwicklungsvorgehensmodell ergänzt.

Das strategische Denken zur bewussten Entscheidungsfindung bezüglich Softwarelösungen, welches im Softwareentwicklungsprozessmodell zu fehlen scheint, wird in den Dienstleistungsentwicklungsvorgehensmodellen sehr gut konzipiert und kann sich mit dem Softwareentwicklungsprozessmodell gut ergänzen (siehe Abbildung 12).

Die Praxis bestätigte, dass der strategiebasierte Teil des Modells einen Mehrwert für die Softwareentwicklung brachte.

Hypothese 2: Es ist möglich, diesen Prozess eigenständig durchzuführen, d.h. selbst zu steuern.

Das Konstrukt „perceived ease of use“, welches im Fragebogen durch Fragen 9 bis 13 (siehe ANHANG G) abgefragt wurde, zeigt, dass der Prozess benutzerfreundlich ist, und dass das untersuchte Unternehmen den Prozess eigenständig durchführen konnte.

Die strategische Schritte 1 bis 3 sind einfach zu verstehen, und es ist genau beschrieben, welche Ziele in jedem Schritt erreicht werden sollen. Es gibt in jedem Schritt eine Menge verschiedener Tools, aus denen ausgewählt werden kann. Und es handelt sich in meisten Fällen um bekannte Werkzeuge, die im betriebswirtschaftlichen Umfeld häufig eingesetzt werden.

Das Ergebnis von 1 für jedes benutzte Werkzeug bedeutet, dass für die Fragebogenteilnehmer die Aussage sehr zutreffend ist, dass die Werkzeuge leicht zu verstehen und zu verwenden sind. Daraus können wir schließen, dass die verwendeten Werkzeuge sehr benutzerfreundlich sind.

Die Softwareentwicklungsschritte sind für ein Software-Unternehmen ohnehin bekannt.

Hypothese 3: Die Ergebnisse des strategiebasierten Vorgehensmodells ermöglicht es Unternehmen, Grundlagen für eine fundierte Entscheidung zu erhalten, um die Strategie einer oder mehrerer Software-Lösungen zu gestalten.

Hypothese 3 impliziert, dass der Prozess für den Entscheidungsfindungsprozess nützlich ist, somit liefert das Konstrukt „perceived usefulness“ eine Antwort auf diese Hypothese.

Im Wesentlichen sind die Erwartungen an die Nützlichkeit der Tools erfüllt worden (Fragen 1 bis 8, siehe ANHANG G).

die Nutzung der drei strategischen Phasen wurde im Feedback Fragebogen mit der Note 2 bewertet, was bedeutet, dass die Befragten deren Nutzung für sinnvoll halten, um die aktuelle Situation der Abteilung besser zu verstehen, Projekte, Maßnahmen und Aktionen zu planen und schlussendlich wichtige Informationen zu erhalten, um eine fundierte Go/NoGo-Entscheidung zu treffen.

Das Dienstleistungsportfolio wurde etwas kritisch gesehen, da es schwerfiel und mehrere Anläufe benötigte, um das Tool auf die Softwarelösungen der Firma anzuwenden. Es war nicht immer einfach zu differenzieren zwischen Softwarelösungen und den dazugehörigen Dienstleistungen, was es erschwerte, die Bewertungskategorien sinnvoll anwenden zu können.

Die SWOT-Analyse war besonders effektiv. Einerseits war es positiv, dass alle Mitarbeiter der Abteilung aktiv beteiligt waren und deshalb gab es reichhaltigen Input. Andererseits flossen die Ergebnisse aller anderen Tools in die SWOT-Analyse ein. Damit war die SWOT-Analyse besonders aussagekräftig.

Für die Nützlichkeit sprechen auch die Antworten auf die Frage: „Würden Sie die 3 Phasen wiederverwenden?“, die mit einem klaren Ja geantwortet würde.

Wie im Feedback-Fragebogen rückgemeldet wurde, könnte es eine Bereicherung sein, wenn auch Leute von außerhalb der Abteilung bei der SWOT-Analyse dabei wären, die eine bessere Kundensicht haben oder vielleicht sogar Kunden sind. Das entspricht der Förderung nach Multidisziplinarität für einen erfolgreichen Innovationsprozess.

Das strategiebasierte Modell hat der Abteilung geholfen, einen Gesamt-Überblick über wichtige strategische Lücken zu erreichen. In weiterer Folge hat die Ausarbeitung des BOD einen konkreten Nutzen geschaffen.

6.3 Betrachtung der Forschungsfrage

Welche Kombination von Service-Engineering und Software-Engineering ist geeignet, um Softwarelösungen mit Mehrwert für Kunden und Softwareunternehmen zu generieren?

Für die Forschungsfrage können zwei Teil-Aspekte betrachtet werden:

1. Welche Kombination von Service-Engineering und Software-Engineering ist geeignet für die Entwicklung von Softwarelösungen?
2. Generiert diese Kombination Mehrwert für Kunden und Softwareunternehmen?

In Kapitel 4 wurde ein Vorgehensmodell als eine Kombination aus Software- und Service Engineering entwickelt. Dieses Vorgehensmodell erweitert das Softwareentwicklungsmodell um strategische Aspekte in der Konzeptions-Phase, die in Softwareentwicklungsmodellen zu fehlen scheinen.

Erkenntnisse aus dem Service-Engineering zeigen, dass die Konzeption-Phase ein umfangreicher und strukturierter Vorgang ist. Insbesondere wird das am Vorgehensmodell des FH CAMPUS 02 deutlich – die ersten 3 Phasen beschäftigen sich intensiv und strukturiert mit der Projekt-Konzeption, basierend auf dem Geschäftsmodell und der Firmenstrategie.

Durch Zusammenführen des FH CAMPUS 02 Dienstleistungsentwicklungsmodells mit einem Softwareentwicklungsmodell gelangen wir zum Vorschlag des strategiebasierten Vorgehensmodells zur Softwareentwicklung, das sich für die Konzeptionsphase nach dem strategiebasierten Vorgehensmodell zur Dienstleistungsentwicklung der Studienrichtung IT & Wirtschaftsinformatik der FH CAMPUS 02 richtet und es durch ein iteratives, inkrementelles und agiles Softwareentwicklungsvorgehensmodell ergänzt.

Somit wird der erste Teil der Forschungsfrage mit dem vorgeschlagenen Modell beantwortet, welches als Kombination von Service- und Software Engineering formuliert ist, die für die umfassende, strategiegetriebene Erstellung von Softwarelösungen geeignet ist.

Mit der Prüfung der Anwendbarkeit dieses Modells in der Praxis beschäftigte sich Kapitel 5.

Angesichts der Knappheit von Zeit und Ressourcen und unter Berücksichtigung, dass der reine Softwareentwicklungsteil des Vorgehensmodells in vielen Unternehmen schon etabliert ist und

praktiziert wird, konzentriert sich der praktische Teil lediglich auf die Umsetzung der strategischen Schritte 1 bis 3 des Prozesses.

Die Ergebnisse zeigen, dass das Vorgehensmodell mit eigenen Mitteln in der Praxis anwendbar ist, und dass daraus wichtige Information erhalten wurden, um strategische Entscheidungen für bestehende oder neue Software-Lösungen zu treffen.

Außerdem sind als direkte Folge der Erarbeitung der ersten drei Phasen neue Erkenntnisse ans Licht gekommen, die einen neuen Softwareentwicklungs-Zyklus auslösten, und ist durchaus möglich, dass es zu einem späteren Zeitpunkt wieder zu einer adaptierten, strategischen Ausrichtung der Softwareabteilung kommt.

Für eine abschließende Beantwortung der Frage, ob diese Kombination Mehrwert für Kunden und Softwareunternehmen generiert, ist es mit den Einschränkungen dieser praktischen Anwendung noch zu früh. Dafür werden weitere Untersuchungen in mehreren Firmen mit multidisziplinären Teams und über einem längeren Zeitraum benötigt.

7 CONCLUSIO

In den letzten 60 Jahren hat sich Software als eigenständige Industrie etabliert. Heutzutage entwickelt sich fast alles um Digitalisierung herum. Dennoch haben Software-Unternehmen noch immer Schwierigkeiten, qualitativ hochwertige Software rechtzeitig und innerhalb des Budgets zu entwickeln. Die Zahl von Softwareprojekten, die scheitern, ist noch immer hoch.

Software-Engineering beschäftigt sich mit der Verwendung eines systematischen, disziplinierten und quantifizierbaren Ansatzes für die Entwicklung, den Betrieb und die Wartung von Software.

Erkenntnisse aus der Literaturrecherche zeigen, dass ein Softwareentwicklungs-Vorgehensmodell hochgradig interaktiv, inkrementell und agil sein soll. Die meisten Softwareunternehmen verwenden bereits derartige Prozesse wie etwa Scrum.

Startpunkt jedes Softwareentwicklungsvorgehensmodells ist die Projekt-Konzeption. Wie aber dieser erste Schritt systematisch und strukturiert erarbeitet und umgesetzt werden soll, erklärt die Literatur von Software-Engineering nicht.

Viele IT-Unternehmen führen lediglich eine Ad-hoc-Entwicklung ihrer Services mit unzureichenden Entwicklungsprozessen zur effizienten Entwicklung ihrer IT-basierten Services durch.

Die Entwicklung von Software erfordert eine Verbindung zwischen Software- und Service-Engineering. Die bisher untersuchten Softwareentwicklungsvorgehensmodelle verdeutlichen nicht ausreichend, wie die Projekt-Konzeption gestaltet werden soll.

Ein Vorgehensmodell wird benötigt, welches die eingegliederte Entwicklung von Software und Dienstleistungen ausdrücklich behandelt.

Erkenntnisse aus dem Service-Engineering, insbesondere betreffend IT-basierter Dienstleistungen zeigen, dass die Entwicklung von Softwarelösungen nicht ohne strategischen Kontext passieren soll. Die untersuchten Vorgehensmodelle legen besonderes Augenmerk auf die Projekt-Konzeption unter Berücksichtigung des strategischen Kontexts des Unternehmens, was wiederum bei den Softwareentwicklungsmodellen zu fehlen scheint.

Die zentrale Frage ist daher: Welche Kombination von Service Engineering und Software Engineering ist geeignet, um Softwarelösungen mit Mehrwert für Kunden und Softwareunternehmen zu generieren?

Ziel dieser Arbeit ist es, ein solches strategiebasierte Vorgehensmodell für Softwareentwicklung zu entwickeln, welches Softwareunternehmen mit eigenen Mitteln einsetzen können und daraus strategische Entscheidungen für bestehende oder neue Software-Lösungen zu treffen.

Eine strukturierte und strategische Vorgehensweise, die für die Software Entwicklung benutzt werden kann, bietet das strategiebasierte Vorgehensmodell zur Dienstleistungsentwicklung der Studienrichtung IT & Wirtschaftsinformatik der FH CAMPUS 02.

Durch die Zusammenführung des FH CAMPUS 02-Vorgehensmodells mit einem Software-Entwicklungsmodell entstand das vorgeschlagene strategiebasierte Vorgehensmodell zur Softwareentwicklung als Antwort zur Forschungsfrage dieser Masterarbeit.

Das Modell richtet sich für die Konzeptionsphase nach dem strategiebasierten Vorgehensmodell zur Dienstleistungsentwicklung der Studienrichtung IT & Wirtschaftsinformatik der FH CAMPUS 02 und wird ergänzt um ein iteratives, inkrementelles und agiles Softwareentwicklungs-Vorgehensmodell.

Die Anwendung in der Praxis war auf eine einzelne Firma beschränkt, und es wurden die ersten drei Schritte des Modells ausgeführt. Dieser Prozess hat der Abteilung geholfen, ihre Softwarelösungen strategisch zu betrachten und hat konkreten Nutzen geschaffen.

Das entwickelte Modell und seine Anwendung in der Praxis lieferten notwendige Grundlagen, um die Hypothesen zu validieren.

Die Ergebnisse liefern uns einen Ansatzpunkt, um zu sagen, dass erstens eine Kombination von Service- und Software-Engineering zur Formulierung eines strategiebasierten Vorgehensmodells zur Softwareentwicklung möglich ist. Zweitens, dass das Vorgehensmodell durchaus in der Praxis eigenständig anwendbar ist. Und drittens, dass Unternehmen Grundlagen erhalten, um fundierte Entscheidungen zur Gestaltung der Strategie ihrer Softwarelösungen treffen zu können.

Allerdings weist die bisherige Anwendung in der Praxis Einschränkungen auf, die erst in zukünftigen Forschungsaktivitäten zu klären sind. Weitere Untersuchungen in mehreren Firmen mit multidisziplinären Teams und über einen längeren Zeitraum sind nötig. Ebenso ist die Einbeziehung von Kunden-Feedback wichtig, um das Modell zu validieren und endgültig zu dem Schluss kommen, inwieweit das Modell geeignet ist, um Softwarelösungen mit Mehrwert für Kunden und Softwareunternehmen zu generieren.

Die Entwicklung dieses Vorgehensmodells ist ein Schritt in Richtung strategiebasierter Softwareentwicklung. Der Prozess alleine kann kein Garant für Erfolg sein. Es gibt andere Faktoren, die den Erfolg beeinflussen, wie integrierte Entwicklungsteams und Kunden-Co-Kreation. Aber es ist zumindest ein wichtiger Maßstab für den richtigen Weg zu einer erfolgreichen Integration strategischer Aspekte in den Softwareentwicklungsprozess und kann somit ein Tool sein, um die Entwicklung und Weiterentwicklung von Softwarelösungen zu verbessern.

8 EINSCHRÄNKUNGEN UND ZUKÜNFTIGE FORSCHUNG

Diese Masterarbeit ist ein erster Schritt in Richtung strategiebasierter Vorgehensmodelle für die Softwareentwicklung. Der vorgeschlagene Prozess wurde theoretisch begründet und zeigt, dass eine Kombination aus Software- und Service-Engineering ein möglicher Ansatz ist, um ein strategiebasiertes Vorgehensmodell zur Softwareentwicklung zu kreieren.

Allerdings konnte aus Zeit- und Verfügbarkeitsgründen nur ein Ausschnitt des gesamten Prozesses (Schritte 1 bis 3) in einem Unternehmen durchgeführt werden. Die Anwendung in der Praxis lieferte vielversprechende Ergebnisse, die die Anwendbarkeit des Modells demonstrieren, und wichtige Inputs, um die Forschungsfrage zu beantworten und die Hypothesen zu validieren.

Ob das Vorgehensmodell tatsächlich Softwareunternehmen hilft, Softwarelösungen mit Mehrwert für Kunden und Softwareunternehmen zu generieren, bleibt vorerst noch offen.

Größere empirischen Untersuchungen sollten durchgeführt werden, und zwar in verschiedenen Unternehmen unterschiedlicher Größe und unterschiedlichem Produkt-Sortiment. Dabei sollte auch ein größerer Untersuchungszeitraum gewählt werden, um den Prozess in mindestens einer gesamten Prozess-Iteration untersuchen zu können. Damit sind weitere Erkenntnisse zur Beantwortung der Frage möglich, auf welche Art und in welchem Maß das Modell wirklich praktikabel und nützlich ist.

Zukünftige Studien könnten das Forschungsgebiet erweitern, indem sie andere Aspekte untersuchen wie:

- Experten-Interviews, um das Modell zu validieren
- Langzeit-Studien, um den gesamten Prozess zu verfolgen mit größere empirische Untersuchungen
- Das Vorgehensmodell in Unternehmen mit verschiedene großen und mit unterschiedlichen Arbeitsweisen durchführen, um dieselbe zu validieren
- Integration von Kunde-Co-Kreation und multidisziplinären Teams in das Modell, da diese in mehreren Studien als Erfolgsfaktoren in Innovationsprozessen aufgeführt werden

ANHANG A Konkurrenzübersicht

Firmenname	Website	Werbebotschaft		Produkte	Services	Mitarbeiter	Medienkanäle
Labene Medizin-software Gmb	https://www.labene.at/index.php	Entspannen Sie sich! Unsere Software unterstützt Ihren Laboralltag zuverlässig, effizienzsteigernd und perfekt an den Ablauf in Ihrem Labor angepasst. Konzentrieren Sie sich nur mehr auf Ihre verantwortungsvolle Tätigkeit im Labor. Wir kümmern uns um die Software und auf Ihren Wunsch auch um Hardware für Ihr Labor als Komplettpaket.	1	LISLAB MEDIZIN		16	
			2	LISLAB BAKTERIOLOGIE			
			3	LISLAB UMWELT			
			4	LISQC			
Dorner Health IT Solutions	https://www.dorner.de/	Flexibel, effizient und sicher. Sie haben Ihre Patienten im Blick – DORNER Ihre IT. Wo immer im Gesundheitswesen Informationen entstehen, verwaltet, aufbereitet und kommuniziert werden, bietet DORNER effiziente IT-Lösungen. Individuell angepasst und modular kombinierbar.	1	LIMS für die Labormedizin - [X/Lab]	Integration	51-200 Mitarbeiter Laut XING und Linkedin	https://www.linkedin.com/company/dorner-health-solutions/
			2	LIMS für die Mikrobiologie - [M/Lab]	Hotline und Instandhaltung		https://www.xing.com/pages/dornerhealthitsolutions
			3	LIMS für die Transfusionsmedizin - [B/Lab]	Updates		https://twitter.com/dorner_it
			4	LIMS für die Genetik – [i/med] Genetik	Schulungen		https://www.youtube.com/channel/UCsbYvPmNcx3Sw06gwtP8LcQ
			5	LIMS für das Hygienelabor – [i/med] Hygiene	und viele weitere Support-Leistungen		
			6	Die elektronische Auftragserfassung - [i/med] Order Entry			
			7	Mobile, webbasierte Befundanzeige – [i/med] Info			
			8	Die Software für Ihre Probenverwaltung - [S/Lab]			
			9	Die elektronische Patientenakte - [i/med] EPA			
			10	Digitaler Online-Anamnesebogen - [i/med] Anamnese			
			11	IT-Lösung für die Sportmedizin und Trainings¬wissenschaft – [i/med] Sport			
			12	Das gerichtsmedizinische KAS – [i/med] Gerichtsmedizin			
			13	Workflowbasierte Anwendersoftware - [i/med] Workflow			
			14	Die Software für die stationäre und ambulante Abrechnung - [i/med] Billing			
			15	Qualitäts¬management und Dokumenten¬verwaltung - [i/med] Q/Docs			
			16	Termin- und Ressourcenplanung - [i/med] Organizer			

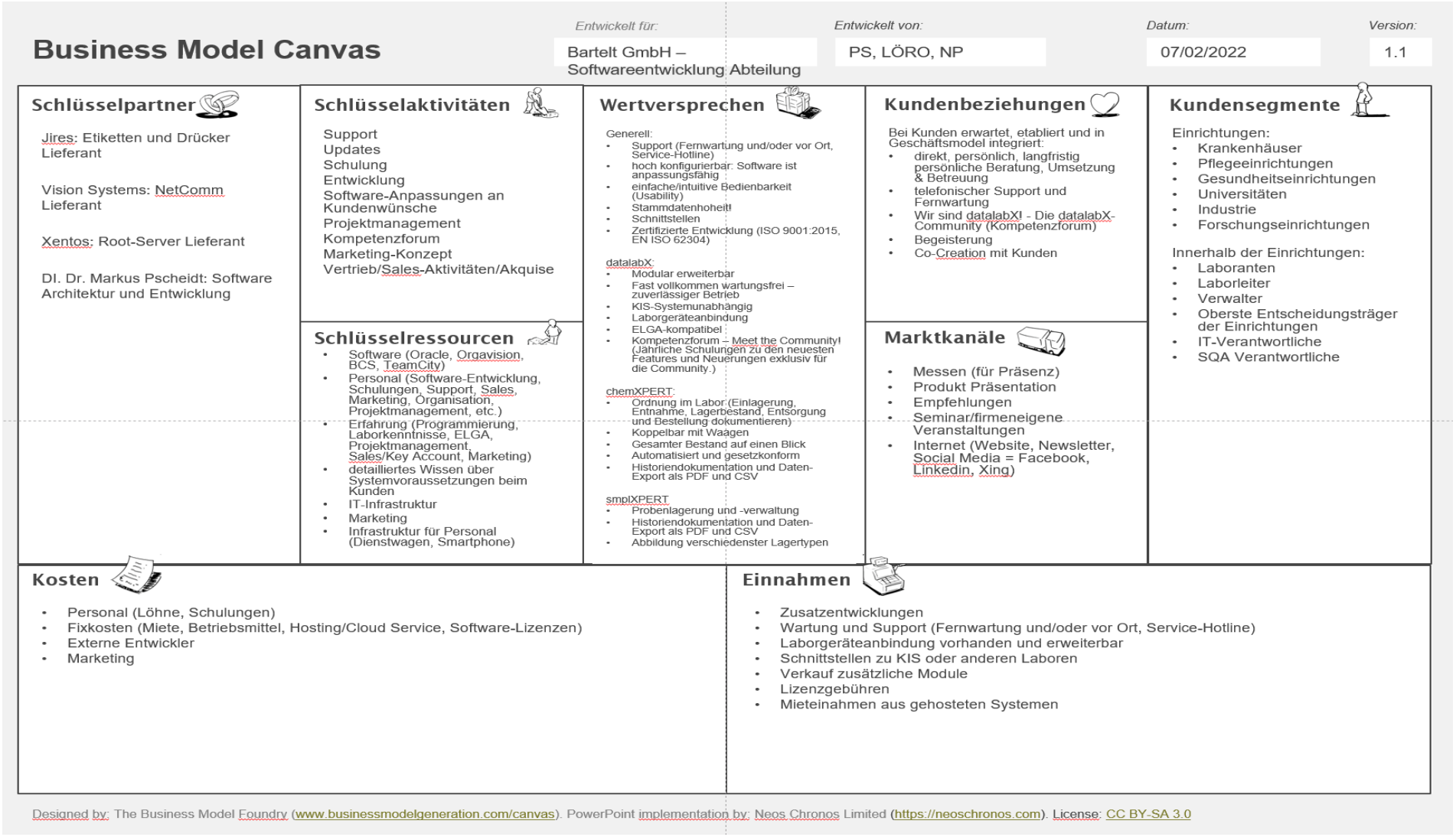
CGM LAB+A25:H43 Deutschlan	https://www.cgm.com/deu_de/produkte/labor/molis.html	Mehr Sicherheit durch Ihr Labor CGM vernetzt das Gesundheitswesen. CompuGroup Medical (CGM) ist Österreichs führender eHealth-Anbieter. CGM ist als einziger Softwarehersteller im Markt ausschließlich auf die Entwicklung Softwarelösungen für unterschiedliche Institutionen des Gesundheitsbereichs Unsere digitalen Lösungen helfen heilen in Österreich und weltweit.	1	Molis - Das multidisziplinäre und hochskalierbare LIS für Privat- & Kliniklabore		>5.000 Mitarbeitende 201-500 Mitarbeitende in Österreich	https://www.facebook.com/CompuGroup-Medical-SE-Co-KGaA-112275907098431
			2	CGM CHANNEL – Das cloudbasierte Order Entry Portal			https://www.instagram.com/compugroup_medical/
			3	MED REQUEST: elektronische Labor-anforderung im niedergelassenen Bereich			https://twitter.com/CGMeHealth
							https://www.xing.com/pages/compugroupmedicalag
							https://www.linkedin.com/company/compugroup-medical-ag/
CliniSys MIPS	https://www.clinisysgroup.com/at/de/	Expertenwissen rund um IT, Medizin und Wissenschaft für bewährte Verfahren zur Patientenversorgung. Komplettlösungen für den gesamten Diagnoseprozess. Nahtlos nutzbare Software für eine bessere Patientenversorgung	1	GLIMS - Das meistgenutzte Laborinformationssystem in Europa	Support	350	https://www.linkedin.com/company/clinisys-mips/
			2	GLIMS Genetics	Schulungen		https://www.xing.com/pages/mipsdeutschlandgmbh-co-kg
			3	DaVinci - Umfassende und benutzerfreundliche Lösung, die von Pathologen für Pathologen entwickelt wurde			https://twitter.com/CliniSysGroup
			4	CyberLab - Effiziente Kommunikation zwischen Gesundheits-Dienstleistern und Laboren			
OSM AG (Von Dedalus Labor GmbH übernommen)	https://www.osm-ag.de/	Qualität, Preis, Geschwindigkeit – die Anforderungen an ein effektives Labormanagement. Die besten Voraussetzungen diese Anforderungen zu meistern und ein zukunftssicheres Preis-/Leistungsverhältnis zu generieren, haben Laboratorien und Krankenhäuser, die über effiziente Informations-, Kommunikations- und Managementsysteme verfügen.	1	Opus::L - Gesamtheitliches und integratives Labormanagement - integriert die Bereiche klinische Chemie, Mikrobiologie, Immunhämatologie und Transfusionsmedizin.		über 186 Mitarbeitern	
			2	Opus::L/M - Das Modul für die Mikrobiologie			
			3	Opus::L/IT - das Modul für die Immunhämatologie und Transfusionsmedizin.			
			4	Opus::L/Faktura - elektronische Rechnungsstellung.			
			5	Opus::L/Blutspende			
			6	Opus::L/Populationsspezifische Referenzintervalle			
			7	Opus::L/Business Analytics - Tool für die Analyse und statistische Auswertung der Labordaten aus Opus::L			
			8	Mirth Connect – HL7 Kommunikationsserver, um die Anforderungen der Kommunikation im klinischen Umfeld abzubilden.			
			9	ixserv - webbasierte Integrations- und Kommunikationsplattform für klinisches Befund- und Auftragsmanagement.			
			10	ix.connect - ergänzung zu ixserv um die Anbindung externer Partner an Laboratorien und Kliniken zu vereinfachen.			
Softwarehaus Kerth	https://www.kerth.at/	Wir sind seit 30 Jahren erfolgreich im Bereich der Softwarelösungen für medizinisch technische Labors tätig. Unsere langjährig bestehende und laufend weiterentwickelte Laborsystem Kerth Software, erfüllt alle Anforderungen eines modernen Laborbetriebes, von der Probenannahme bis zur Befundausgabe.	1	Laborsystem	Hotline		
			2	Blutgruppen- und Konservenverwaltung	Vor-Ort Einsätze		
			3	Hostanbindungen	Kostenloses Update (im Fehlerfall oder bei neuen Programmteilen) unter dem selben Betriebssystem		
			4	Interfaceanbindungen	Kostengünstiges Upgrade bei Wechsel auf andere Betriebssysteme		
			5	DatenDrehScheiben	Schnittstellenprobleme		
					Erreichbarkeit nach Vereinbarung		
					Fernwartung / permanent		

ANHANG B Dienstleistungsportfolio

Dienstleistungsportfolio	Bartelt GmbH	Datum: 07.02.2022
	Softwareentwicklung	PS, LÖRO, NP

lfd. Nr.	Name	DL Typ		Phase aus Kundenkontakt/ Auftragsdurchlauf					Aktivität / Dienstleistung		Strategische Bedeutung für Bartelt					Wie berechnet?				Erbringungspotenzial			Einordnug nach BCG Matrix				Kurzbeschreibung
		Standard	Individuell	Entwicklungsphase	Informationsphase	Angebots-/Kaufphase	Auftragsausführungsphase	Nutzungsphase	Technologie	Services	Höherer Preis für Grundleistung	Prozeßoptimierung / Kostenreduzierung	Erschließung neuer Zielgruppen	Erhöhung der Kundenbindung	Verbesserung der Zahlungsbereitschaft	hat Preis	wird verhandelt	wird verschenkt	ist versteckt	Alleine	Kooperation	Nur Extern	Question Mark	Star	Cashcow	Poor Dog	
1	datalabX	x		x	x	x	x	x	Softwarelösung	Entwicklung, Installation, Updates, Support, Laborgeräteeinbindung, Schulung, Kompetenzforum	x			x	x	x	x			x	?				x		Laborinformationssystem für Krankenhäuser
3	datalabXdepot	x		x	x	x	x	x	Softwarelösung	Entwicklung,Installation, Updates, Support, Laborgeräteeinbindung, Schulung, Kompetenzforum	x			x	x	x	x			x	?				x		Unterstütz Bluttransfusion und Blutgruppenserologie
4	datalabXpoct	x		x	x	x	x	x	Softwarelösung	Entwicklung,Installation, Updates, Support, Laborgeräteeinbindung, Schulung, Kompetenzforum	x			x	x	x	x			x	?				x		Point of care Testing
5	datalabXmibi	x		x	x	x	x	x	Softwarelösung	Entwicklung,Installation, Updates, Support, Laborgeräteeinbindung, Schulung, Kompetenzforum	x			x	x	x	x			x	?				x		Mikrobiologie
6	datalabXorderentry	x		x	x	x	x	x	Softwarelösung	Entwicklung,Installation, Updates, Support, Laborgeräteeinbindung, Schulung, Kompetenzforum	x			x	x	x	x			x	?				x		Anforderungsverwaltung und Befunddarstellung
7	chemXPERT	x		x	x	x			Softwarelösung / SaaS	Entwicklung, Installation, Hosting, Updates, Support, Schulung	x		x	x	x	x	x	x		x	?		x				Softwarelösung zur Verwaltung und Dokumentation von Chemikalien
8	smplXPERT	x		x	x	x		x	Softwarelösung / SaaS	Entwicklung, Installation, Hosting, Updates, Support, Schulung	x		x	x	x	x	x	x		x	?		x				Softwarelösung für das Lagern und Verwalten von Proben
10	OxMM		x	x				x	Softwarelösung	Entwicklung, Updates, Support	x		x				x			x					x		Laborinformationssystem für Mayr-Melnhof Karton AG
11	OxKost	x		x				x	Softwarelösung	Entwicklung, Updates, Support	x		x	x			x			x					x		Software für Weinverkostung
12	Menarini		x	x			x		Softwarelösung	Entwicklung, Hardware/Installation, Updates	x		x			x	x				x				x		Middleware für Geräteeinbindung
13	AimConnector (Waage Data Colector)	x						x	Softwarelösung	Entwicklung, Updates, Support		x	x			x			x	x			?				Middleware für Geräteeinbindung
14	Interface Manager	x		x					Softwarelösung	Entwicklung, Updates, Support		x	x			x			x	x			?				Middleware für Geräteeinbindung
16	Etiketten	x	x					x		Vertrieb				x		x						x			x			Etiketten
17	Verdünnungsreihe	x						x	Online Tool	Hosting			x	x				x		x								Berechnung der arithmetische oder geometrische Verdünnungsreihe
18	Verdünnungsrechner	x						x	Online Tool	Hosting			x	x				x		x								Ermittlung die Mischungsverhältnisse zweier Substanzen
19	Zentrifugationswerte	x						x	Online Tool	Hosting			x	x				x		x								Ermittlung die relative Zentrifugalbeschleunigung oder die Umdrehungsgeschwindigkeit

ANHANG C Business Model Canvas



ANHANG D SWOT Fragen

adaptiert aus Dr. Jürgen Fleig (2021)

Strengths – Stärken

Stärken sind solche Merkmale des Unternehmens, die im Wettbewerb ein Vorteil sind oder die das Unternehmen besser beherrscht als die Konkurrenz.

- Worin liegen Gründe für vergangene Erfolge?
- Worauf kann die Abteilung stolz sein?
- Was kann die Abteilung besser als seine Wettbewerber?
- Welche Ressourcen lassen sich intern noch besser nutzen?

Weaknesses – Schwächen

Schwächen sind alle Merkmale des Unternehmens, die im Wettbewerb ein Nachteil sind und sich negativ auf den Erfolg auswirken können.

- Was fiel bislang schwer?
- Was fehlt in unserer Abteilung?
- Warum gehen Aufträge an den Wettbewerber verloren? Wo punktet die Konkurrenz?
- Was sollte besser vermieden werden?

Opportunities – Chancen

Chancen sind Faktoren und Entwicklungen im Umfeld oder Markt, die für das Unternehmen ein Vorteil sein können oder aus denen Potenziale erwachsen können.

- Welche Möglichkeiten bieten sich?
- Welche Veränderungen in der Branche können Vorteile bringen? Welche Zukunftschancen sind absehbar?
- Was lässt sich nutzen, was der Wettbewerb nicht kann?
- Wie entwickelt sich das Umfeld?

Threats – Risiken

Risiken sind solche Faktoren und Entwicklungen im Umfeld oder im Marktgeschehen eines Unternehmens, aus denen Nachteile oder Gefahren entstehen können, die das Unternehmen schwächen oder die zu Verlusten führen können.

- Wo lauern Gefahren für das bisherige Geschäftsmodell?
- Wie wirken sich Technologie- oder Politikwechsel aus?
- Was macht die Konkurrenz, wo baut sie aus?

ANHANG E SWOT-Matrix

Stärken

S1. **Gute Kundenbeziehung** durch kompetentes, persönliches und personalisiertes Service

S2. **Kundenmitwirkung** und **Zusammenarbeit**, wodurch sich die Kundenbindung und Zufriedenheit/Treue der Kunden verbessert.

S3. Kompetenz und Fachwissen der **Mitarbeiter,** was stark beiträgt zur Weiterentwicklung der Produkte und zur Anpassungsfähigkeit an Bedürfnisse der Kunden.

S4. Attraktive, individuell erweiterbar und kundenorientierte **Produkte**

- a. Langlebiges dIX mit langjährigem Kundenbestand und einer großen Kundenbasis für österreichische Verhältnisse
- b. Innovative und technologisch moderne neue Produkte

S5. **HL7 Mitgliedschaft** was die Mitarbeit an der Weiterentwicklung des Standards ermöglicht.

Schwächen

W1. **Fehlende Strategie**, die dazu führt:

- a. Richtungsstreits innerhalb der Abteilung,
- c. reaktives Handeln
- d. Unzureichendes **Projekt Management/Planung,** um ordentlichen finanziellen Gewinn zu machen und die Vision zu erreichen.
- e. Fehlendes Commitment zu herausfordernden Zielen und Innovation.

W2. Keinen überzeugenden **Vertrieb und Marketing,** um neue Kunden zu gewinnen, unsere Produkte zu verkaufen und finanziellen Gewinn zu machen.

W3. **Fehlende Skalierbarkeit, Instabilität des Produkts** nach größeren Weiter/Neu-Entwicklungen.

W4. **Personal Knappheit**

- a. Führt zu Überstunden und lässt keine Möglichkeiten für Tätigkeiten abseits der Routinearbeit (Strategie bauen, Weiterbildung und Weiterentwicklung, Code Refactoring und Verbesserung, etc.)
- b. Unsere **Abteilung ist zu klein** um am Markt zu expandieren (Internationale Märkte erobern, große Installationen in sehr großen Labors vorweisen)
- c. Kritische Auswirkungen von Mitarbeitern-Ausfällen

W5. Unzureichende **rechtliche Absicherung**, z.B. Unzureichende Lizenzverträge.

Chancen

O1. **Partnerschaften** für Vertrieb und Entwicklung

O2. **Ausschreibungen**

O3. Prozess-**Automatisierung** (Updates, Installation, etc.)

O4. **Große und komplexe Labors**

- a. Labor 4.0 (Ar-Brille, KI)
- b. Laborstraßen
- c. Mandantenfähiges Laborsystem mit gemeinsamen Stammdaten
- d. Molekularbiologische Funktionalitäten
- e. Genetik

O5. **Hosting** (Cloud-Dienste)

Risiken

T1. **Medizinprodukte**-Gesetz auch für **Laborsoftware**: Vorrausetzungen für Medizinprodukt (Zertifizierung - ISO 13485 statt ISO 9001:2015)

T2. **Unsicherheit** und Abhängigkeit von **Großkunden** (KIS Hersteller wollen ihr LIS bei unseren Kunden reinbringen)

T3. **Softwareentwicklung** wegen **komplexer** und somit **teuer**, Risiko, dass es finanziell schwerer geht.

T4. **Technologiewechsel** (Aufwand alte Technologien umzustellen, z.B. FOP) und Instabilität des Produkts

S-O Ausbauen Strategie

Welche Stärken passen zu welchen Chancen? Wie können Stärken dazu beitragen, die Chancen (besser) zu nutzen? Welche Geschäftsfelder oder Produktbereiche kann das Unternehmen erweitern?

Chancen/Stärken	**S1: Gute Kundenbeziehung**	**S2: Kundenmitwirkung** und **Zusammenarbeit**	**S3:** Kompetenz und Fachwissen der **Mitarbeiter**	**S4:** Attraktive, individuell erweiterbar und kundenorientierte Produkte	**S5: HL7 Mitgliedschaft**
O1: Partnerschaften für Vertrieb und Entwicklung	S1-O1:	S2-O1: Provisionssystem	S3-O1: Schulung für Partner	S4-O1: Markt im Ausland für DLX Partner f. Ox-Vertrieb im In-/Ausland	S5-O1: Partner suchen für Ox/Dlx-Vertrieb
O2: Ausschreibungen	S1-O2: Gute Referenzen bekommen	S2-O2: Kunde präsentiert unsere Produkte	S3-O2: Ausschreibungsunterlagen ausfüllen; Produkt präsentieren	S4-O2:	S5-O2:
O3: Prozess-**Automatisierung**	S1-O3:	S2-O3: Kunden sollen es als Vorteil betrachten, wenn sie selbst Dlx-Updates installieren können (wie in Ox)	S3-O3: Update-Maske in Dlx; LG/IO-Module automatisieren; Testautomatisierung für Produkte	S4-O3: Installer für DLX; Online Formular für Hotline	S5-O3:
O4: Große und komplexe Labors	S1-O4:	S2-O4: Mit Kunden gemeinsam Anforderungen für große und komplexe Labors erarbeiten. Unterstützung von Kunden in der Projektumsetzung	S3-O4: DLX-Mandantenfähigkeit	S4-O4: Erweiterung unserer Produkte	S5-O4:
O5: Hosting (Cloud-Dienste)	S1-O5:	S2-O5: Pilot-Kunde	S3-O5:	S4-O5:	S5-O5:

S-T Absichern Strategie

Welchen Risiken kann das Unternehmen mit welchen Stärken begegnen? Wie können Stärken den Eintritt bestimmter Risiken abwenden? In welchen technischen oder organisatorischen Bereichen muss sich das Unternehmen absichern?

Risiken/Stärken	**S1: Gute Kundenbeziehung**	**S2: Kundenmitwirkung** und **Zusammenarbeit**	**S3:** Kompetenz und Fachwissen der **Mitarbeiter**	**S4:** Attraktive, individuell erweiterbar und kundenorientierte Produkte	**S5: HL7 Mitgliedschaft**
T1: Medizinprodukte-Gesetz auch für **Laborsoftware**	S1-T1:	S2-T1:	S3-T1: Vorbereitung, DLX mit ISO 13485 zu zertifizieren: alle Voraussetzungen erfüllen	S4-T1: Sich an ISO 13485/GAMP annähern, d.h. auch wenn noch nicht verpflichtend, kann freiwillig schrittweise das Niveau unserer Produkte erhöht werden	S5-T1: Bei HL7-Kollegen laufend erkundigen, wie real die Medizinprodukt-Gefahr ist; Lobby gegen Medizinprodukt
T2: Abhängigkeit von **Großkunden**	S1-T2: Weiter gute Kontakte pflegen, um sicherzustellen, dass Kunden bei uns bleiben	S2-T2: gleich wie S1-T2	S3-T2:	S4-T2: Abhängigkeit von DLX reduzieren durch Verkauf von Ox-Produkten	S5-T2:
T3: Softwareentwicklung wird **komplexer** und somit **teuer**	S1-T3: Kostenbeteiligung der Kunden	S2-T3:	S3-T3: Konsequente Weiterbildung; Outsourcing	S4-T3: Technologie laufend aktualisieren	S5-T3:
T4: Technologiewechsel und Instabilität des Produkts	S1-T4: Technologiewechsel als Zusatzleistung verkaufen (wie bei Web-Clients); Kunden akzeptieren	S2-T4:	S3-T4: Zeitbudget von Entwicklern für Technologie-Umstellung reservieren; Risikoanalyse durchführen	S4-T4:	S5-T4: Lobby für Feststellung eines Standards

W-O Aufholen Strategie

In welchen Geschäftsfeldern oder Märkten muss das Unternehmen aufholen? Welche Schwächen müssen beseitigt werden? Wodurch lassen sich Schwächen ausgleichen? Welche Chancen stecken in einer Schwäche?

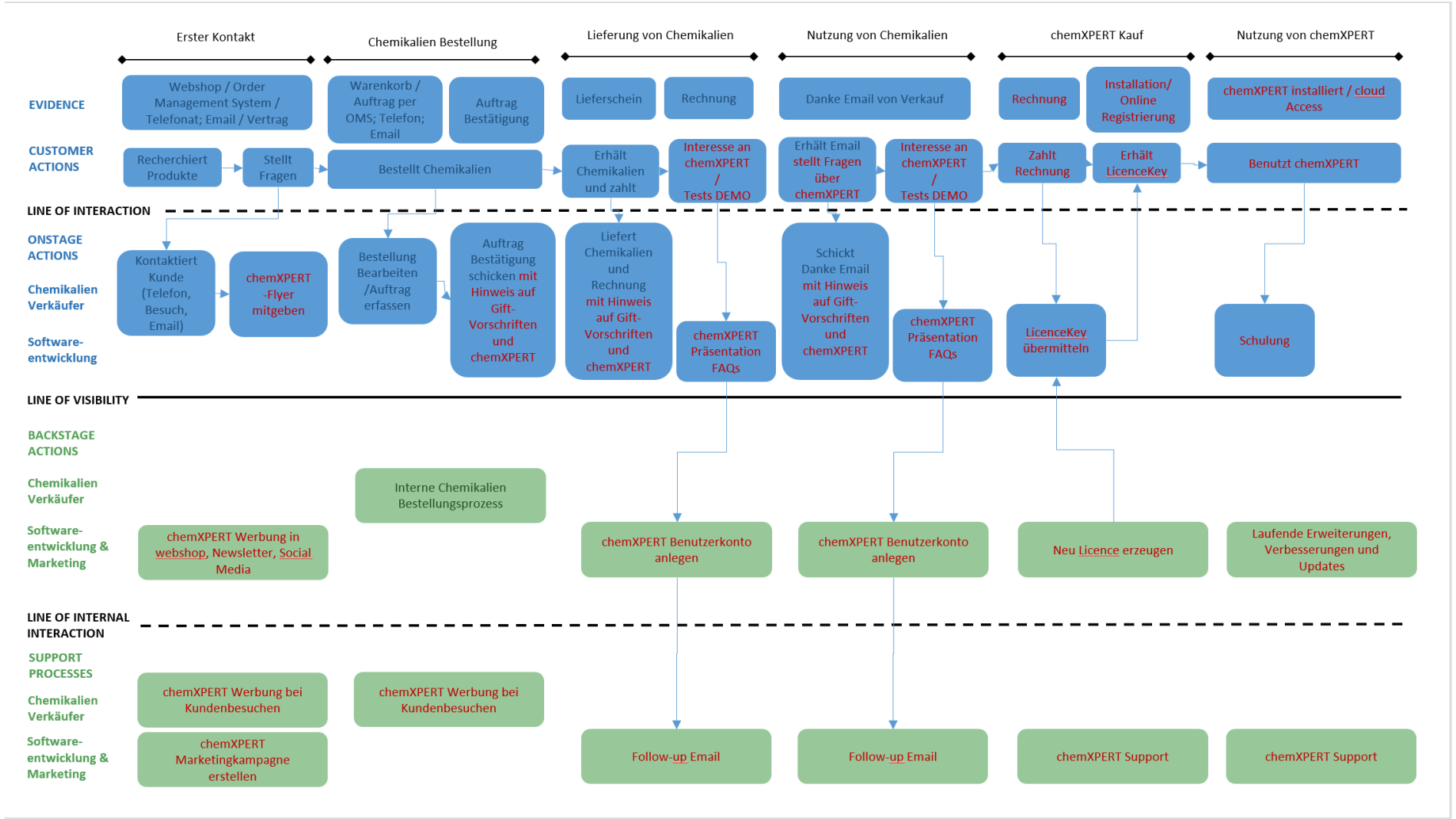
Chancen/Schwächen	**W1:** Fehlende Strategie	**W2:** Keinen überzeugenden **Vertrieb und Marketing**	**W3: Fehlende Skalierbarkeit, Instabilität des Produkts** nach größeren Weiter/Neu-Entwicklungen.	**W4: Personal Knappheit**	**W5:** Keine **rechtliche Absicherung**
O1: Partnerschaften für Vertrieb und Entwicklung	W1-O1: Partner gibt Strategie vor	W2-O1: Partner für Vertrieb	W3-O1:	W4-O1: Auftragsentwicklung extern	W5-O1:Externes know-how für die Verträge
O2: Ausschreibungen	W1-O2:	W2-O2: An vielen Ausschreibungen mitmachen; Auftrag.at richtige Schlüsselwörter eintragen	W3-O2:	W4-O2:	W5-O2: Bartelt-Anwalt: Lizenzvereinbarung entwickeln lassen
O3: Prozess-**Automatisierung**	W1-O3:	W2-O3: Automatisierter Vertrieb, Marketing in Social Media	W3-O3: Testautomatisierung	W4-O3: Personal einstellen	W5-O3:
O4: Große und komplexe Labors	W1-O4:	W2-O4:	W3-O4:	W4-O4: Personal einstellen	W5-O4:
O5: Hosting (Cloud-Dienste)	W1-O5:	W2-O5:	W3-O5:	W4-O5:	W5-O5: Externes know-how für die Verträge

W-T Vermeiden Strategie

Wo treffen Schwächen auf Risiken? Welche Gefahren erwachsen dadurch dem Unternehmen? Wie kann das Unternehmen dennoch vor Schaden geschützt werden? Welche Aktivitäten sollte das Unternehmen vermeiden oder nicht mehr ausüben?

Risiken/Schwächen	**W1:** Fehlende Strategie	**W2:** Keinen überzeugenden **Vertrieb und Marketing**	**W3: Fehlende Skalierbarkeit, Instabilität des Produkts** nach größeren Weiter/Neu-Entwicklungen.	**W4: Personal Knappheit**	**W5:** Keine **rechtliche Absicherung**
T1: Medizinprodukte-Gesetz auch für **Laborsoftware**	W1-T1: Sukzessive Approximation an ISO 13845	W2-T1:	W3-T1:	W4-T1:Mehr Personal Ressourcen	W5-T1:
T2: Abhängigkeit von **Großkunden**	W1-T2: Großkunden pflegen; Kundenbasis auf breitere Beine stellen	W2-T2: Vertrieb aufbauen intern/extern	W3-T2: LIS Partner für große Labors	W4-T2:	W5-T2:
T3: Softwareentwicklung wird **komplexer** und somit **teuer**	W1-T3:	W2-T3:	W3-T3: Mehr Personal Ressourcen	W4-T3:Mehr Personal Ressourcen	W5-T3:
T4: Technologiewechsel und Instabilität des Produkts	W1-T4:	W2-T4:	W3-T4: Mehr Personal Ressourcen	W4-T4:Mehr Personal Ressourcen	W5-T4:

ANHANG F Service Blueprint



ANHANG G Feedback Fragebogen

Die Strategische Analyse, die Ideen-Generierung und die Variantenbildung sollten uns einen Überblick über alle Produkte/Services in der Abteilung geben und uns helfen, strategischer Lücken zu identifizieren, ein klar abgegrenztes Suchfeld für neue oder bestehende Projekte und schlussendlich ein grob formuliertes Projekt-Design für die Geschäftsführung, um eine Go/NoGo Entscheidung zu erwirken. Dafür haben wir verschiedene Tools verwendet.

Bitte klassifiziere folgende Aussagen von 1 bis 5, wobei 1 für sehr zutreffend steht und 5 für trifft gar nicht zu.

1. Die Verwendung des DL-Portfolios hat es leichter gemacht, unsere verschiedenen Produkte und die dazu gehörigen Services abzubilden und zu analysieren.

2 Antworten

ID ↑	Name	Antworten
1	anonymous	3
2	anonymous	2

2.Die Verwendung des Business Model Canvas hat es leichter gemacht, die unterschiedlichen Aspekte unseres Geschäftsmodells zu analysieren und abzubilden.

2 Antworten

ID ↑	Name	Antworten
1	anonymous	2
2	anonymous	1

3.Die Verwendung der Konkurrenzüberblick hat es uns ermöglicht, unsere direkten Konkurrenten besser zu kennen.

2 Antworten

ID ↑	Name	Antworten
1	anonymous	1
2	anonymous	2

4.Die Verwendung der SWOT-Analyse hat es leichter gemacht, die Abteilung zu analysieren, um Stärken, Schwächen, Chancen und Risiken zu entdecken und entsprechende strategische Stoßrichtungen abzuleiten.

2 Antworten

ID ↑	Name	Antworten
1	anonymous	1
2	anonymous	2

5.Die Verwendung der Business Opportunity Description hat es ermöglicht, ein Grobdesign eines Projektes zu erstellen, um eine Go/NoGo-Entscheidung zu erwirken.

2 Antworten

ID ↑	Name	Antworten
1	anonymous	2
2	anonymous	1

6.Die 3 Phasen haben mir geholfen die aktuelle Situation der Abteilung besser zu verstehen.

2 Antworten

ID ↑	Name	Antworten
1	anonymous	2
2	anonymous	2

7.Diese 3 Phasen haben mir geholfen Projekte, Maßnahmen und Aktionen zu planen.

2 Antworten

ID ↑	Name	Antworten
1	anonymous	3
2	anonymous	1

8.Diese 3 Phasen haben mir schlussendlich wichtige Informationen geliefert um ein Go/NoGo Entscheidung über Projekte zu treffen.

2 Antworten

ID ↑	Name	Antworten
1	anonymous	3
2	anonymous	1

9.DL-Portfolio ist leicht zu verstehen und verwenden.

2 Antworten

ID ↑	Name	Antworten
1	anonymous	1
2	anonymous	1

10.Business Model Canvas ist leicht zu verstehen und verwenden.

2 Antworten

ID ↑	Name	Antworten
1	anonymous	1
2	anonymous	1

11.Konkurrenzüberblick ist leicht zu verstehen und verwenden.

2 Antworten

ID ↑	Name	Antworten
1	anonymous	1
2	anonymous	1

12.SWOT Analyse ist leicht zu verstehen und verwenden.

2 Antworten

ID ↑	Name	Antworten
1	anonymous	1
2	anonymous	1

13.Business Opportunity Description ist leicht zu verstehen und verwenden.

2 Antworten

ID ↑	Name	Antworten
1	anonymous	1
2	anonymous	1

14.Würden Sie die 3 Phasen wiederverwenden?

- o Ja, einmal im Jahr (in unserer Klausur)
- o Ja, anlassbezogen: z.B. wenn ein neues Projekt oder eine große Erweiterung eines bestehenden Projekts ansteht
- o Nein, es braucht viel Zeit und bringt nicht viel
- o Nein, die Tools sind schwer zu verstehen und/oder zu verwenden

2 Antworten

ID ↑	Name	Antworten
1	anonymous	Ja, anlassbezogen: z.B. wenn ein neues Projekt oder eine große Erweiterung eines bestehenden Projekts ansteht
2	anonymous	Ja, einmal im Jahr (in unserer Klausur)

15.Wenn Sie den Prozess noch einmal durchgehen, würden Sie etwas anders machen? Wenn ja, was?

2 Antworten

ID ↑	Name	Antworten
1	anonymous	Nein
2	anonymous	Es wäre gut, auch Leute außerhalb der Abteilung dabei zu haben, die eine bessere Kundensicht haben oder vielleicht sogar Kunden sind.

ANHANG H Aktivitäten den strategischen Phasen Beschreibung

Aktivität	Werkzeuge	Teilnehmer	Termin	Dauer - Vorbereitung / Termin (in Stunden)	Ziel	Ergebnisse
Kickoff Meeting	Brainstorming	AL, SA, PL*	23.01.202	10/1	Projektplanung und Werkzeug-Auswahl	Tools ausgesucht und nächste Schritte geplant
Strategische Analyse	Konkurrenzübersicht	AL, PL	25.01.2022	5/-	Informationen über direkte Konkurrenten darstellen	Konkurrenzübersicht erstellt und dokumentiert
	DL-Portfolio	AL, SA, PL	07.02.2022	6/2	Darstellung alle Softwarelösungen und dazugehörigen Services und Evaluierung in den verschiedene Kategorien	DL-Portfolio der Abteilung erstellt
	Business Model Canvas	AL, SA, PL	07.02.2022	2/1	Abbildung und Analyse den verschiedene Aspekte der Abteilungsgeschäftsmodell	Business Model Canvas erstellt
	SWOT-Fragen	AM	07.02.2022 – 10.02.2022	12/-	Reflektion über SWOT der Abteilung	S.W.O.T gelistet
	Präsentation	AM	15.02.2022	4/2	Bisherige Tools präsentiert und S.W.O.T. diskutiert, ergänzt und reduziert	S.W.O.T auf jeweils 5 reduziert
	SWOT-Analyse	AM	09.03.2022	2/3	Kombination von S.W.O.T.	SWOT-Matrix erstellt
Ideen-Generierung und Bewertung	Brainstorming Abstimmung	AM	09.03.2022	-/1	Ableitung von Maßnahmen und strategische Ziele von SWOT-Matrix Priorisierung der abgeleitet Aktionen	Projekte, Maßnahmen, strategische Ziele abgeleitet und gelistet
Variantenbildung mit Business Cases	Service Blueprint	PL, CV	24.03.2022	4/2	Beschreibung der Verkaufsprozess von Chemikalien und Integration von chemXPERT verkauf Überlegungen	Service Blueprint erstellt
	Business Opportunity Description	AL, SA, PL	01.04.2022	5/1	Grobkonzept den Verkauf von chemXPERT mittels Chemikalien-Verkauf erstellt für eine Go/NoGo Entscheidung	BOD - chemXPERT-Verkauf mittels Chemikalien-Verkauf Go-Entscheidung getroffen
Feedback	Fragebogen	AL, SA, PL	08.05.2022	3/0.25	Feedback erfassen	Feedback-Fragebogen ausgefüllt

* AL = Abteilungsleiter; SA = stellvertretender Abteilungsleiter; PL = Projektleiterin; AM = Alle Mitarbeiter; CV = Chemikalienverkäufer

ABKÜRZUNGSVERZEICHNIS

AL ..... *Abteilungsleiter*
BOD ..... *Business Opportunity Description*
DevOps ..... *Development and Operations*
DL ..... *Dienstleistung*
FH ..... *Fachhochschule*
GmbH..... *Gesellschaft mit beschränkter Haftung*
IEEE..... *Institute of Electrical and Electronics Engineers*
IT*Informationstechnologien*
KMU ..... *Kleine und mittlere Unternehmen*
LIMS..... *Laborinformationssystem*
NSD ..... *New Service Development*
PBI ..... *Product Backlog Item*
RAD ..... *Rapid Application Development*
SA ..... *stellvertretende Abteilungsleiter*
SD-Logik ..... *Service Dominant Logik*
SLAs ..... *Service Level Agreements*
SWEBOK ..... *Software Engineering Body of Knowledge*
SWOT ..... *Strengths, Weaknesses, Opportunities und Threats*
TAM ..... *Technology Acceptance Model*

ABBILDUNGSVERZEICHNIS

TABELLENVERZEICHNIS

LITERATURVERZEICHNIS

Andrea Ovans (2015): What Is a Business Model? Hg. v. Harvard Business Review. Online verfügbar unter https://hbr.org/2015/01/what-is-a-business-model, zuletzt aktualisiert am 23.01.2015, zuletzt geprüft am 21.05.2022.

Aschbacher, Helmut (2014): Framework für das agile Entwickeln von IKT-basierten Dienstleistungen unter Nutzung von Smart Services.

Bartelt G.m.b.H.: Bartelt Website. Online verfügbar unter https://www.bartelt.at/, zuletzt geprüft am 15.05.2022.

Broy, Manfred (2018): The Leading Role of Software and Systems Architecture in the Age of Digitization. In: Volker Gruhn und Rüdiger Striemer (Hg.): The Essence of Software Engineering. Cham: Springer International Publishing, S. 1–23.

Bullinger, Hans-Jörg; Scheer, August-Wilhelm (2006): Service Engineering — Entwicklung und Gestaltung innovativer Dienstleistungen. In: Hans-Jörg Bullinger und August-Wilhelm Scheer (Hg.): Service Engineering. Berlin/Heidelberg: Springer-Verlag, S. 3–18.

Bullinger, Hans-Jörg; Scheer, August-Wilhelm; Schneider, Kristof (2006): Service Engineering. Entwicklung und Gestaltung innovativer Dienstleistungen. 2., vollst. überarb. und erw. Aufl. Heidelberg: Springer Berlin Heidelberg.

Bullinger, Hans-Jörg; Schreiner, Peter (2006): Service Engineering: Ein Rahmenkonzept für die systematische Entwicklung von Dienstleistungen. In: Hans-Jörg Bullinger und August-Wilhelm Scheer (Hg.): Service Engineering. Berlin/Heidelberg: Springer-Verlag, S. 53–84.

Cooper, Robert G. (1983): A process model for industrial new product development. In: *IEEE Trans. Eng. Manage.* EM-30 (1), S. 2–11. DOI: 10.1109/TEM.1983.6448637.

Davis, Fred D. (1985): A Technology Acceptance Model for Empirically Testing New End-User Information Systems 1985.

Davis, Fred D. (1989): Perceived Usefulness, Perceived Ease of Use, and User Acceptance of Information Technology. In: *MIS Quarterly* 13 (3), S. 319. DOI: 10.2307/249008.

Dr. Jürgen Fleig (2021): SWOT-Analyse. So wird eine SWOT-Analyse erstellt. Hg. v. Redaktion business-wissen.de. Online verfügbar unter https://www.business-wissen.de/artikel/swot-analyse-so-wird-eine-swot-analyse-erstellt/#:~:text=Die%20SWOT%2DAnalyse%20ist%20noch,nur%20grobe%20strategische%20Sto%C3%9Frichtungen%20sichtbar., zuletzt aktualisiert am 26.08.2021, zuletzt geprüft am 15.05.2022.

Edvardsson, Bo; Meiren, Thomas; Schäfer, Adrienne; Witell, Lars (2013): Having a strategy for new service development – does it really matter? In: *Journal of Service Management* 24 (1), S. 25–44. DOI: 10.1108/09564231311304170.

Edvardsson, Bo; Olsson, Jan (1996): Key Concepts for New Service Development. In: *The Service Industries Journal* 16 (2), S. 140–164. DOI: 10.1080/02642069600000019.

Ehrenhofer, Christoph; Kreuzer, Ernst (2012): The Role of Business Model Design in the Service Engineering Process: A Comparative Case Study in the Field of Cloud Computing to Join Service Engineering with Business Model Design. In: 2012 Annual SRII Global Conference (SRII). 2012 Annual SRII Global Conference (SRII). San Jose, CA, USA, 7/24/2012 - 7/27/2012: IEEE / Institute of Electrical and Electronics Engineers Incorporated, S. 283–292.

Fähnrich, Klaus-Peter; Opitz, Marc (2006): Service Engineering — Entwicklungspfad und Bild einer jungen Disziplin. In: Hans-Jörg Bullinger und August-Wilhelm Scheer (Hg.): Service Engineering. Berlin/Heidelberg: Springer-Verlag, S. 85–112.

Ghezzi, Carlo (2018): Formal Methods and Agile Development: Towards a Happy Marriage. In: Volker Gruhn und Rüdiger Striemer (Hg.): The Essence of Software Engineering. Cham: Springer International Publishing, S. 25–36.

Gruhn, Volker; Striemer, Rüdiger (2018): The Essence of Software Engineering. Cham: Springer International Publishing.

Höber, Angelika; Pergler, Elisabeth; Weitlaner, Doris; Grahsl, Hans-Peter (2015): Performance journey mapping: a service performance assessment framework. In: *TQM Journal* 27 (2), S. 231–246. DOI: 10.1108/TQM-12-2014-0112.

IEEE Computer Society: Guide to the Software Engineering Body of Knowledge Version 3.0 (SWEBOK Guide V3.0).

Johannes Kepler Universität Linz (2017): Sektoren der Wirtschaft. Hg. v. Johannes Kepler Universität Linz (JKU), Institut für Digital Business. Online verfügbar unter https://oesterreich.com/de/wirtschaft/branchen-und-industriezweige/sektoren-der-wirtschaft, zuletzt aktualisiert am 2017, zuletzt geprüft am 21.05.2022.

Kreuzer, Ernst; Aschbacher, Helmut (2011): Strategy-Based Service Business Development for Small and Medium Sized Enterprises. In: Mehdi Snene, Jolita Ralyté und Jean-Henry Morin (Hg.): Exploring Services Science, Bd. 82. Berlin, Heidelberg: Springer Berlin Heidelberg (Lecture Notes in Business Information Processing), S. 173–188.

Kreuzer, Ernst; Schäfer, Adrienne; Aschbacher, Helmut (2011): The Concept of Service Strategy Scorecard - an Integrated Approach for Lean Service Engineering and Service Improvement: Theoretical framework and implications for Service Science. In: *The 2011 Naples Forum on Service - Service Dominant logic, Network & Systems Theory and Service Science*.

Laqua, Roland (2007a): Das Projekt ServCASE. In: Frederick W. Taylor (Hg.): Entwicklung it-basierter dienstleistungen. Co design von software und. Heidelberg: Physica Springer, S. 3–10.

Laqua, Roland (2007b): ServCASE Service Metamodell. In: Frederick W. Taylor (Hg.): Entwicklung it-basierter dienstleistungen. Co design von software und. Heidelberg: Physica Springer, S. 127–142.

Marquardt, Katrin (2017): Smart services – characteristics, challenges, opportunities and business models. In: *Proceedings of the International Conference on Business Excellence* 11 (1), S. 789–801. DOI: 10.1515/picbe-2017-0084.

Memminger, Andrea; Wäsch, Jürgen (2007): Entwicklung von E-Business Dienstleistungen für die Produktkommunikation. In: Frederick W. Taylor (Hg.): Entwicklung it-basierter dienstleistungen. Co design von software und. Heidelberg: Physica Springer, S. 207–223.

Meyer, Kyrill; Böttcher, Martin; van Husen, Christian (2007): Software-Service-Co-Design — Zusammenfassung der Breitenerhebung. In: Frederick W. Taylor (Hg.): Entwicklung it-basierter dienstleistungen. Co design von software und. Heidelberg: Physica Springer, S. 41–52.

Meyer, Kyrill; van Husen, Christian (2007): Die ServCASE-Methode im Überblick. In: Frederick W. Taylor (Hg.): Entwicklung it-basierter dienstleistungen. Co design von software und. Heidelberg: Physica Springer, S. 11–25.

Ng, Irene C.L.; Vargo, Stephen L. (2018): Service-dominant (S-D) logic, service ecosystems and institutions: bridging theory and practice. In: *JOSM* 29 (4), S. 518–520. DOI: 10.1108/JOSM-07-2018-412.

Osterwalder, Alexander; Pigneur, Yves (2013): Business Model Generation. A handbook for visionaries, game changers, and challengers. Weinheim [u.a.]: (Wiley)-VCH.

Osterweil, Leon J. (2018): What is software? In: Volker Gruhn und Rüdiger Striemer (Hg.): The Essence of Software Engineering. Cham: Springer International Publishing, S. 59–76.

Pikkarainen, Minna; Codenie, Wim; Boucart, Nick; Heredia Alvaro, José Antonio (2011): The Art of Software Innovation. Berlin, Heidelberg: Springer Berlin Heidelberg.

Pressman, Roger S.; Maxim, Bruce R. (2020): Software engineering. A practitioner's approach. Ninth edition. New York, NY: McGraw Hill.

Prestes Joly, Maíra; Teixeira, Jorge Grenha; Patrício, Lia; Sangiorgi, Daniela (2019): Leveraging service design as a multidisciplinary approach to service innovation. In: *JOSM* 30 (6), S. 681–715. DOI: 10.1108/JOSM-07-2017-0178.

Reichwald, Ralf; Schaller, Christian (2003): Innovationsmanagement von Dienstleistungen — Herausforderungen und Erfolgsfaktoren in der Praxis. In: Hans-Jörg Bullinger und August-Wilhelm Scheer (Hg.): Service Engineering. Berlin, Heidelberg: Springer Berlin Heidelberg, S. 171–198.

Reichwald, Ralf; Schaller, Christian (2006): Innovationsmanagement von Dienstleistungen — Herausforderungen und Erfolgsfaktoren in der Praxis. In: Hans-Jörg Bullinger und August-Wilhelm Scheer (Hg.): Service Engineering. Berlin/Heidelberg: Springer-Verlag, S. 167–194.

Scheer, August-Wilhelm; Grieble, Oliver; Klein, Ralf (2006): Modellbasiertes Dienstleistungsmanagement. In: Hans-Jörg Bullinger und August-Wilhelm Scheer (Hg.): Service Engineering. Berlin/Heidelberg: Springer-Verlag, S. 19–51.

Stepanek, George (2005): Software Project Secrets. Why Software Projects Fail. Berkeley, CA: Apress (Expert's Voice).

The Standish Group (2015): Chaos Report. Online verfügbar unter https://www.standishgroup.com/sample_research_files/CHAOSReport2015-Final.pdf, zuletzt aktualisiert am 2014, zuletzt geprüft am 01.10.2021.

van Husen, Christian; Fuchs, Bettina; Böttcher, Martin; Meyer, Kyrill (2007): Ermittlung von Problemfeldern bei der Entwicklung IT-basierter Dienstleistungen. In: Frederick W. Taylor (Hg.): Entwicklung it-basierter dienstleistungen. Co design von software und. Heidelberg: Physica Springer, S. 29–39.

Vargo, Stephen L.; Lusch, Robert F. (2016): Institutions and axioms: an extension and update of service-dominant logic. In: *J. of the Acad. Mark. Sci.* 44 (1), S. 5–23. DOI: 10.1007/s11747-015-0456-3.

Vargo, Stephen L.; Lusch, Robert F.; Koskela-Huotari, Kaisa (Hg.) (2019): The SAGE handbook of service-dominant logic. London: SAGE Publications. Online verfügbar unter http://sk.sagepub.com/reference/the-sage-handbook-of-service-dominant-logic.