Software-Referenzarchitekturen für Geschäftsmodelle des e-Business unter besonderer Beachtung ihrer Erlösmodelle

Von der Fakultät für Mathematik und Informatik
der Universität Leipzig
angenommene

DISSERTATION

zur Erlangung des akademischen Grades

DOCTOR INGENIEUR
(Dr.-Ing.)

im Fachgebiet Informatik

vorgelegt von:

Dipl. Wirt.-Inf. Thorsten Weber
geboren am 25.10.1972 in Nürnberg

Die Annahme der Dissertation haben empfohlen:
1. Professor Dr. Stefan Eicker (Universität Duisburg-Essen)
2. Professor Dr. Volker Gruhn (Universität Leipzig)
3. Professor Dr. Helmut Krcmar (Techn. Universität München)

Die Verleihung des akademischen Grades erfolgt auf Beschluss des Rates der Fakultät für Mathematik und Informatik vom 17.Juli 2006 mit dem Gesamtprädikat cum laude.

Bibliographische Daten

Weber, Thorsten

Software-Referenzarchitekturen für Geschäftsmodelle des e-Business unter besonderer Beachtung ihrer Erlösmodelle

Universität Leipzig, Diss., 319 S., 155 Lit., 174 Abb., 3 Tab.

Das e-Business hat aufgrund der technologischen Möglichkeiten durch den kommerziellen Einsatz des Internet zahlreiche neue Geschäftsmodelle hervorgebracht. Dabei wurden zum einen bestehende Geschäftsmodelle der traditionellen Geschäftswelt auf die neuen Rahmenbedingungen des e-Business angepasst und zum anderen gänzlich neue Formen von Geschäftsmodellen erfunden. Ein Grund für das Entstehen neuer Geschäftsideen und deren Konkretisierung durch Geschäftsmodelle ist die Möglichkeit, im e-Business neue, zuvor nicht realisierbare Erlösmodelle umzusetzen. Die Erlösmodelle sind somit nicht nur ein wesentlicher Teil der Geschäftsmodelle, sondern bilden in einigen Fällen darüber hinaus das entscheidende Element einer neuartigen Geschäftsidee im e-Business. Um von einem Geschäftsmodell zu einer lauffähigen Anwendung zu gelangen, werden Softwarearchitekturen benötigt. Sie stellen den Bauplan der Softwareanwendung dar, die im e-Business als Realisierung des Geschäftsmodells interpretiert werden kann. Um die Gestaltung dieser Architekturen zu erleichtern und zu beschleunigen, können entsprechende Referenzarchitekturen als abstrakte Entwurfsvorlagen verwendet werden, sofern sie existieren. Mit der Aufstellung eines Kataloges von Software-Referenzarchitekturen für Erlösmodelle des e-Business als wesentlicher Bestandteil eines Geschäftsmodells befasst sich diese Arbeit.

Dazu wird zunächst eine Klassifikation der Erlösmodelle des e-Business geliefert, die den Übergang vom *fachlich* orientierten Geschäftsmodell auf die zugrundeliegende *technisch* orientierte Softwarearchitektur unterstützt. Aufbauend auf dieser Klassifikation wird ein Katalog der Software-Referenzarchitekturen hergeleitet. Der Katalog enthält eine Sammlung von Komponenten, die gemeinsam als Bausatz der Software-Referenzarchitekturen angesehen werden können.

Da die Klassifikation ein breites Spektrum potenzieller Erlösmodelle berücksichtigt, stehen mit den Referenzarchitekturen Entwurfsvorlagen für zahlreiche unterschiedliche Geschäftsmodelle zur Verfügung. Die detaillierte Granularität der Software-Referenzarchitekturen mit den aufgeführten Komponenten und Klassen kann dabei sehr gut verwendet werden, um einfacher und schneller eine konkrete Architektur abzuleiten, als wenn diese von Grund auf neu entwickelt werden müsste. Die Ergebnisse dieser Arbeit leisten einen wesentlichen Beitrag, um die Realisierung von Geschäftsmodellen des e-Business in vereinfachter und beschleunigter Form durchzuführen und unterschiedliche, daran beteiligte Personengruppen zu unterstützen.

Inhaltsverzeichnis

1 Forschungsgegenstand

Die überschwängliche Euphorie in Wirtschaft und Technik im Zusammenhang mit der New Economy und ihren angeblich alles verändernden Kräften ist weitgehend verflogen. Zahlreiche Firmen, die in den letzten Jahren gegründet wurden, sind bereits wieder verschwunden und der Börsenwert des Neuen Marktes, der wirtschaftliche Inbegriff der Erwartungshaltung gegenüber der New Economy, ist auf ein Bruchteil dessen geschrumpft, was er einmal war.

Dennoch ist unbestreitbar, dass die Entwicklung des electronic Business (kurz e-Business) der letzten Jahre das Geschäftsleben verändert hat und weiterhin prägen wird. Die Rolle des Internet ist aus unternehmerischen Aktivitäten nicht mehr wegzudenken und das e-Business wird weiterhin eine bedeutende Stellung einnehmen. So werden, wie in der Vergangenheit umfangreich geschehen, auch in Zukunft bestehende Geschäftsmodelle aufgrund der Möglichkeiten des e-Business neu ausgerichtet oder völlig neuartige Ideen entstehen.

Ein sehr bedeutender Bestandteil eines Geschäftsmodells ist sein Erlösmodell. Das Erlösmodell legt die Art und Weise fest, wie das Unternehmen Einkommen erwirtschaften will. Es gibt Antworten auf die Fragen, *womit* und *von wem* Geld eingenommen werden soll. Damit bildet es das Kernelement des wirtschaftlichen Überlebens einer Unternehmung. Für Geschäftsmodelle des e-Business erhöht sich die Gewichtung ihrer Erlösmodelle darüber hinaus noch im Vergleich zu traditionellen[1] Geschäftsmodellen. Hintergrund ist die Tatsache, dass die Besonderheiten des e-Business vollkommen neuartige Möglichkeiten zum Erzielen von Erlösen ermöglichen, die in der traditionellen Geschäftswelt nicht anzuwenden sind[2]. Neben der Optimierung bestehender Geschäftsmodelle durch ihre Adaption für das e-Business sind aufgrund neuer Einnahmemöglichkeiten auch vollkommen neuartige Geschäftsmodelle realisierbar.

Während der Umsetzung von Geschäfts- mitsamt ihren Erlösmodellen im *e*-Business kommt der Softwarearchitektur stets eine besondere Bedeutung zu, da es sich um *elektronisch* abzuwickelnde Geschäftsaktivitäten handelt. Geschäftsmodell und zugrundeliegende Softwarearchitektur sind dabei untrennbar miteinander verbunden. Die Architektur muss derart gestaltet sein, dass sie die Anforderungen des Geschäftmodells während der Durchführung der Prozesse vollständig erfüllen kann. Die Softwarearchitektur stellt sozusagen den Bauplan der konkreten Softwareanwendung dar, die im Produktionsbetrieb die Abwicklung der Prozesse zur Realisierung des Geschäftsmodells übernimmt. Es ist insofern offensichtlich, dass die Softwarearchitektur für alle Personengruppen, die an der Umsetzung eines Geschäftsmodells beteiligt sind, von großem Interesse ist. Zu diesen Gruppen gehören neben technisch ausgebildeten und orientierten Personen, die für die Umsetzung der Anforderungen in eine lauffähige Softwareanwendung verantwortlich sind, ebenso fachlich und kaufmännisch orientierte Personen.

1. *traditionell* wird hier als Abgrenzung zum e-Business verstanden
2. vgl. hierzu [SL00]

Aufgrund dieser heterogenen Personengruppen mit ihren unterschiedlichen thematischen Kenntnissen und Schwerpunkten ergeben sich bei der Herleitung von Softwarearchitekturen für Geschäftsmodelle des e-Business somit folgende Fragestellungen: Wie können fachlich und kaufmännisch orientierte Personen in die Konzeption von Softwarearchitekturen mit einbezogen werden und anhand welcher Instrumente kann mit ihnen über wichtige Designentscheidungen diskutiert werden? Wie können neuartige Geschäftsmodelle umgesetzt werden, ohne dass die Softwarearchitektur gänzlich neu entworfen, sondern indem auf bestehende Ansätze zurückgegriffen wird? Inwiefern können fachlich orientierte Beschreibungen der Geschäftsmodelle dazu verwendet werden, technische Komponenten zu ermitteln, die für die umsetzende Softwareanwendung benötigt werden? Wie können Bestandteile der Geschäftsmodelle identifiziert werden, deren Funktionalität mittels Komponenten in anderen Softwareanwendungen bereits umgesetzt wurde und die somit wiederverwendet werden können?

Die vorliegende Arbeit befasst sich mit der Beantwortung dieser Problemstellungen. Dazu wird den folgenden zwei Fragestellungen detailliert nachgegangen und es wird versucht, geeignete Lösungen zu finden:

- Inwiefern lässt sich trotz der weitreichenden Vielfältigkeit unterschiedlicher Geschäftsmodelle des e-Business eine begrenzte Klassifikation der potenziell enthaltenen Erlösmodelle aufstellen?
- Inwiefern können für diese Domäne der Erlösmodelle des e-Business entsprechende Software-Referenzarchitekturen aufgestellt werden?

Sofern es gelingt, diesen Katalog von Software-Referenzarchitekturen aufzustellen, bilden sie eine abstrakte Entwurfsvorlage für die Realisierung konkreter Anwendungsfälle. Somit würde für einen wesentlichen Bestandteil eines Geschäftsmodells des e-Business eine Vorlage für einen wesentlichen Bestandteil der gesamten Softwarearchitektur zur Verfügung stehen. Die Software-Referenzarchitekturen können den beteiligten Personen während des Herleitungsprozesses der konkreten Softwarearchitektur dazu dienen, auf bestehende Ansätze zurückzugreifen und bestehende Komponenten wiederzuverwenden, anstatt sie vollständig neu zu entwickeln. Entsprechend der Legostein-Metapher[3] können diese identifizierten Komponenten dazu verwendet werden, als Bausteine des Gesamtsystems zu dienen. Die Gestaltung der Softwarearchitektur wird somit erleichtert und beschleunigt. Aufgrund der Klassifikation der Erlösmodelle auf Basis einer *fachlichen* Charakterisierung der sie umgebenden Geschäftsmodelle dient der Ansatz darüber hinaus dazu, nicht-technisch orientierte Personengruppen in den Gestaltungsprozess der Software-Architektur umfassend mit einzubeziehen.

3. vgl. hierzu [GT00]

1.1 Zielsetzung

Das wesentliche Ziel dieser Arbeit ist, einen Katalog von Software-Referenzarchitekturen für Erlösmodelle von Geschäftsmodellen des e-Business aufzustellen. Als Teilziel soll zunächst eine geeignete Klassifikation der Erlösmodelle des e-Business aufgestellt werden.

Die Konzentration auf die Erlösmodelle des e-Business hat den Vorteil, dass die Referenzarchitekturen für eine Vielzahl unterschiedlichster Geschäftsmodelle wiederverwendet werden können, sofern sich ihr Erlösmodell mit einem der hier behandelten Typen deckt. Damit kann ein wesentliches Element der vollständigen Softwarearchitektur ausgehend von dieser Vorlage vereinfacht und in kurzer Zeit gestaltet werden.

Zunächst muss demnach eine geeignete Typologie der Erlösmodelle vorgenommen werden, um anschließend für jeden Typ eine Referenzarchitektur herzuleiten. Diese Klassifizierung der Modelle erfolgt anhand von Klassifikationskriterien, die identifiziert werden müssen.

Die Erlösmodelle werden in der Realität niemals isoliert, sondern immer nur in Zusammenhang mit einem realen Geschäftsmodell angewandt. Man kann also sagen, dass ein Erlösmodell im Rahmen seiner Realisierung immer ein *umgebendes* Geschäftsmodell aufweist. Die Herleitung der Klassifikationskriterien berücksichtigt diese Abhängigkeit, indem Einflussfaktoren der Geschäftsmodelle auf ihre Erlösmodelle analysiert werden. Es wird also untersucht, inwiefern eine Wechselwirkung zwischen den Geschäfts- und ihren Erlösmodellen besteht und anhand welcher Kriterien diese Beeinflussung beschrieben werden kann.

Mit Hilfe der Klassifikationskriterien werden Erlösmodelle und ihre Varianten identifiziert und beschrieben. Die Kriterien sollen dabei vor allem softwaretechnische Rückschlüsse ermöglichen. Man könnte Erlösmodelle beispielsweise auch hinsichtlich betriebswirtschaftlicher Aspekte klassifizieren, jedoch wird im Rahmen dieser Arbeit eine Klassifikation benötigt, die Schlussfolgerungen über die Gestaltung der zugrundeliegenden Softwarearchitektur ermöglicht. Dementsprechend sind auch die Klassifikationskriterien auszusuchen. Es müssen darüber hinaus im Rahmen der Klassifikation ausreichend feingranulare Varianten der Erlösmodelle gefunden werden, damit möglichst viele verschiedene Geschäftsmodelle diese Erlösmodelle und die zugehörigen Referenzarchitekturen in ihren Systemen einsetzen können.

Die Typologie der Erlösmodelle stellt somit ein erstes Teilziel dieser Arbeit dar. Aufbauend auf diesem Teilziel wird im weiteren Verlauf das wesentliche Ziel der Arbeit hergeleitet. Dazu wird zu jeder identifizierten Erlösmodellvariante eine geeignete Software-Referenzarchitektur hergeleitet und detailliert beschrieben, so dass der angestrebte Katalog entsteht.

1.2 Vorgehen

Das Vorgehen wird in die folgenden Schritte unterteilt.

Zunächst wird in Kapitel 2 eine thematische Abgrenzung der Arbeit vorgenommen. Dazu werden die relevanten Begriffe dieser Arbeit einführend erläutert und abgegrenzt.

Anschließend widmet sich Kapitel 3 der Betrachtung existierender Arbeiten zum Thema Geschäftsmodelle und Erlösmodelle im e-Business. Es wird dabei untersucht, inwiefern die bestehenden Ansätze anderer Autoren das Vorhaben dieser Arbeit unterstützen können. Es werden die bereits existierenden Klassifikationen der Modelle und die verwendeten Klassifikationskriterien untersucht. Kritisch wird vor allem analysiert, inwiefern die vorhandenen Arbeiten den Übergang von Geschäfts- und Erlösmodellen auf die realisierenden Softwarearchitekturen berücksichtigen.

In Kapitel 4 wird anschließend ein eigenes Klassifikationsschema für Erlösmodelle des e-Business aufgestellt und angewandt. Dabei werden zunächst die Kriterien mitsamt ihren potenziellen Ausprägungen vorgestellt. Danach werden diese angewendet, um die Typologie der Erlösmodelle durchzuführen. Somit bildet das Ergebnis dieses Kapitels das zuvor erwähnte erste Teilziel dieser Arbeit.

Kapitel 5 widmet sich dem Thema Softwarearchitekturen. Es werden bestehende Arbeiten und Auffassungen bezüglich Softwarearchitekturen untersucht. Dabei wird den Fragen nachgegangen, wozu Softwarearchitekturen grundsätzlich nützlich sind und warum sie eingesetzt werden, welche Anforderungen Softwarearchitekturen des e-Business berücksichtigen müssen und wie sie zu gestalten sind. Vor allem wird untersucht, inwiefern Ansätze der komponentenbasierten Softwareentwicklung geeignet sind. Das Kapitel bildet somit die Grundlage für das nächste Kapitel.

In Kapitel 6 wird schließlich der Katalog der Software-Referenzarchitekturen hergeleitet. Aufbauend auf der Typologie aus Kapitel 4 werden für jedes Erlösmodell zunächst die Anforderungen an die Softwarearchitektur betrachtet. Diese werden im weiteren Verlauf jeweils berücksichtigt, wenn die Architekturen aufgestellt und beschrieben werden. Dieses Kapitel stellt den Schwerpunkt und das wesentliche Ergebnis der Arbeit dar.

Kapitel 7 enthält die Validierung. Es wird anhand eines realen Anwendungsfalls gezeigt, wie die Softwareachitektur für das Geschäftsmodell einer Internetlotterie-Plattform anhand der Referenzarchitekturen aus dem vorherigen Kapitel hergeleitet wird.

Abschließend werden in Kapitel 8 die Schlussfolgerungen der Ergebnisse diskutiert.

2 Fokussierung der Arbeit

Der Titel der vorliegenden Arbeit lautet *Software-Referenzarchitekturen für Geschäftsmodelle des e-Business unter besonderer Beachtung ihrer Erlösmodelle*. Er enthält somit vier wesentliche Begriffe: *e-Business*, *Geschäftsmodelle*, *Erlösmodelle* und *Software-Referenzarchitekturen*. Diese Begriffe werden in unterschiedlichen Kapiteln dieser Arbeit jeweils definiert und detailliert erläutert.

Zunächst wird jedoch in diesem Kapitel bereits vorab eine grobe Begriffsklärung erfolgen, anhand derer ein grundsätzliches Verständnis vermittelt und die Fokussierung der Arbeit ersichtlich wird. Wie gezeigt wird, existiert zu keinem der Begriffe in der Literatur ein einheitliches, allgemein akzeptiertes Verständnis. Aus diesem Grund erscheint es sinnvoll, die Verwendung der Begriffe im Kontext dieser Arbeit einleitend zu erläutern.

Zunächst wird auf den Begriff *e-Business* eingegangen. Anders als die anderen drei erwähnten Begriffe wird dieser im weiteren Verlauf der Arbeit nicht zusätzlich untersucht, sondern nur hier betrachtet. Wie bereits erwähnt, gibt es zahlreiche abweichende Auffassungen über die Bedeutung des Begriffs e-Business. Eine häufig zitierte Arbeit liefert Wirtz ([Wir01]), in der er e-Business wie folgt definiert:

Definition 1: e-Business nach Wirtz
Electronic Business bezeichnet die Anbahnung sowie die teilweise respektive vollständige Unterstützung, Abwicklung und Aufrechterhaltung von Leistungsaustauschprozessen mittels elektronischer Netzwerke.

Diese sehr allgemeine Definition gibt nur unpräzise Auskunft über die Art der Leistungsaustauschprozesse sowie über die Art der elektronischen Netze. In der hier vorliegenden Arbeit wird jedoch auf den Einsatz des Internet fokussiert. Eine Definition, die diese Auffassung des e-Business stärker hervorhebt, wurde von der Firma IBM geliefert:

Definition 2: e-Business nach IBM
A secure, flexible and integrated approach to delivering differenitated business value by combining the systems and processes that run core business operations with the simplicity and reach made possible by internet technology.

Abgesehen von der nur in der zweiten Definition enthaltenen Betonung des Internet eignen sich beide Erläuterungen, das Verständnis des e-Business in dieser Arbeit wiederzuspiegeln. Da der Begriff des e-Business sehr häufig im Zusammenhang mit dem Begriff des e-Commerce fällt, wird das Verhältnis kurz untersucht.

Einige Autoren grenzen beide Begriffe voneinander explizit ab. So betrachtet Wirtz [Wir01] das e-Business als den umfassenderen Begriff, der eine Sammlung von weiteren Begriffen zusammen beschreibt. Neben dem e-Commerce unterscheidet der Autor auch e-Communication, e-Education, e-Collaboration und e-Information und fasst diese fünf Begriffe unter dem Oberbegriff des e-Business zusammen. Ohne hier auf die übrigen Begriffe einzugehen, wird die von Wirtz aufgefasste Bedeutung des e-Commerce vorgestellt:

Definition 3: e-Commerce nach Wirtz
Electronic Commerce beinhaltet die elektronische Unterstützung, die in direkten Zusammenhang mit dem Kauf und Verkauf von Gütern und Dienstleistungen via elektronischer Netze in Verbindung stehen.

Auch Hermanns und Sauter [HS99] betonen, dass „häufig [...] jedoch unter Electronic Commerce lediglich verschiedene Aspekte subsumiert [werden], die den Bereich des Electronic Shopping oder Online-Shopping betreffen. Der Begriff *Shopping* signalisiert allerdings eine Beschränkung auf den Vertrieb bzw. den Handel mit Waren.

Der gleichen Auffassung sind auch Bartelt und Lamersdorf, die e-Commerce ebenfalls mit dem „elektronisch unterstützten Handel“ gleichsetzen [BL00]. Auch sie sehen das e-Business als Oberbegriff der Teildisziplinen e-Commerce, e-Information und e-Cooperation.

Wenn demnach e-Commerce eine Teildisziplin des e-Business darstellt, die auf den reinen Handel beschränkt ist, so würde dieser Begriff für die vorliegende Arbeit nicht ausreichen. Hier werden neben dem Handel auch andere Formen von Geschäftsaktivitäten untersucht, wie später noch gezeigt wird. Dass eine solche Trennung der Begriffe jedoch kein allgemeingültiges Verständnis ist, zeigen Definitionen anderer Autoren. Stellvertretend wird hier die Definition von Thome und Schinzer [TS97] vorgestellt:

Definition 4: e-Commerce nach Thome und Schinzer
Electronic Commerce ermöglicht eine umfassende, digitale Abwicklung der Geschäftsprozesse zwischen Unternehmen und zu deren Kunden über öffentliche (Internet) und private Netze.

Diese Definition ist wiederum sehr nah an den zwei zuvor gelieferten Definitionen des Begriffs e-Business und zeigt, dass kein einheitliches Verständnis über die einzelnen Begriffe und über die Wechselwirkung beider Begriffe existiert. Eine solche klare Unterscheidung ist für diese Arbeit hier auch nicht notwendig. Wichtig für die vorliegende Arbeit ist lediglich, dass sich die Reichweite des e-Business nicht nur auf den reinen Kauf und Verkauf von Gütern und Dienstleistungen beschränkt. Da einige Definitionen von e-Commerce diese Einschränkung vorsehen, wird hier der Begriff e-Business zugrunde gelegt.

Abschließend wird noch ein weiterer Aspekt betrachtet: die *interne* beziehungweise *externe* Ausrichtung der Aktivitäten, die durch das e-Business abgedeckt werden. Einige Autoren beziehen explizit beide Arten der Aktivitäten in das e-Business mit ein. So legen beispielsweise Jarvenpaa und Tiller in ihrer Definiton fest, dass „[...] the definition of e-business includes inter-organisational systems such as Internet technology [...] or internal computing which is in support of commmercial online exchange“ [JT99].

Der Ansatz der internen Aktivitäten wird hier nicht verfolgt. Der Fokus in dieser Arbeit liegt auf den Aktivitäten zwischen Unternehmen und Kunden sowie zwischen Unternehmen und Geschäftspartnern.

Nachdem nun die Sichtweise des Begriffs e-Business für diese Arbeit erläutert wurde, werden im folgenden einige einführende Erklärungen der übrigen drei relevanten Begriffe geliefert.

Zunächst wird der Begriff des *Geschäftsmodells* vorgestellt und im Sinne dieser Arbeit eingegrenzt. Da eine detaillierte Untersuchung vorhandener Definitionen des Geschäftsmodells ausführlich in Kapitel 3 erfolgt, wird hier nur eine erste, recht allgemeine Erläuterung des Begriffs geliefert. Wichtiger an dieser Stelle ist die Abgrenzung eines Geschäftsmodells zu einem *Geschäftsplan*.

Ein Geschäftsmodell ist eine Beschreibung der grundsätzlichen Geschäftsidee eines Unternehmens und der daran beteiligten Einflussfaktoren. Es ist ein Modell der wesentlichen Komponenten, die an der Abwicklung der Geschäftsaktivitäten beteiligt sind. Nicht zu verwechseln ist das Geschäftsmodell mit einem Geschäftsplan.

Während das Geschäftsmodell sich inhaltlich mit der Geschäftsidee und deren detaillierter Darstellung auseinandersetzt, hat ein Geschäftsplan vor allem ein umfangreiches Zahlenwerk zum Inhalt, bei dem die konkrete Umsetzung des Geschäftsmodells in Beträgen und Zeitpunkten ausgedrückt wird. Es dokumentiert notwendige Investitionen, anfallende Kosten, geplante Umsätze, Gewinne und weitere finanzielle sowie zeitliche Rahmendaten. Der Geschäftsplan baut somit auf dem qualitativen Geschäftsmodell auf und erweitert dies um eine Vielzahl detaillierter Finanz-, Zeit- und weiterer Pläne.

Er ist ein Schlüsseldokument für die Beurteilung und das Controlling der Geschäftstätigkeit und besitzt zwei wesentliche Funktionen: Eine Orientierungsfunktion zur Selbstorientierung des Existenzgründers oder Geschäftsmodellbetreibers und eine Informationsfunktion für außenstehende Dritte, z.B. Banken, das Mutterunternehmen oder Stellen zur Genehmigung öffentlicher Fördermittel.

Der Inhalt eines Geschäftsplans umfasst nach Rasner, Füser und Faix [RFF99] in weitgehender Übereinstimmung mit Timmons [Tim94] und Struck [Str90] vor allem die folgenden Bestandteile:

- Eine Marktanalyse, die u.a. eine Konkurrenzanalyse umfassen sollte, dokumentiert Größe, Wachstum, Segmentierung des Marktes, den Wettbewerb und die Positionierung gegenüber der Konkurrenz.
- Der Realisierungsfahrplan dokumentiert die Arbeitspakete und Meilensteine in der Entwicklung des Unternehmens. Er sollte eine realistische Fünf-Jahresplanung umfassen und enthält Personal- und Abschreibungsplanungen.
- Der Finanzplan ist der wichtigste und umfangreichste Teil eines Geschäftsplans und bildet den Schwerpunkt bei der Beurteilung der Tragfähigkeit einer Geschäftsidee. Er besteht aus drei Hauptteilen:
    - der Darlegung der Investitions- und Finanzierungsplanung
    - dem Aufbau der Erfolgsrechnung
    - der Ableitung des Liquiditätsplans.
- Ein Marketingkonzept enthält einen schlüssigen Plan der Marketing- und Vertriebsaktivitäten. Im Einzelnen sind dies eine Markteintrittsstrategie, das Absatzkonzept und Maßnahmen zur Absatzförderung.

- Ein Zukunftskonzept formuliert Vision, Ziele, Strategie, Erfolgsfaktoren, Aktionsplanung und Unternehmensgrundsätze.

Geschäftspläne unterscheiden sich somit von Geschäftsmodellen und werden im Rahmen dieser Arbeit nicht näher betrachtet. Stattdessen ist ein weiterer Begriff von Bedeutung, der ebenfalls einleitend erläutert werden soll - der Begriff des *Erlösmodells*.

Ein Erlösmodell ist sehr allgemein ausgedrückt eine Beschreibung, *wie* ein Unternehmen seine Erlöse erzielt. Es beschreibt, *wofür* die Kunden bezahlen. Somit legt es die Finanzierung der Geschäftsidee fest.

Erlösmodelle werden in Kapitel 3 ebenfalls detaillierter untersucht. Die dabei betrachteten Arbeiten anderer Autoren werden zeigen, dass für den Begriff des Erlösmodells einige weitere Begriffe existieren, die im Rahmen dieser Arbeit synonym zu verstehen sind. Beispiele sind Erlösquellen, Ertragsmodelle, Erlösformen oder Einkommensquellen.

Darüber hinaus gibt es jedoch auch Begriffe, die sich mit den Preisen von Produkten und Dienstleistungen befassen. Beispiele sind *Preismodelle*, *Preisstrategien* und *Preisfindungsmechanismen*. Diese sind nicht mit den Erlösmodellen zu verwechseln, da sie sich damit befassen, *wieviel* ein Unternehmen für sein Angebot verlangen kann und wie dieser möglichst optimale Preis zu finden ist.

Als letzter der vier Begriffe wird nun die *Software-Referenzarchitektur* im Sinne dieser Arbeit vorgestellt. Im Detail wird er in Kapitel 5 untersucht, weshalb hier nur kurz darauf eingegangen wird. Der Begriff setzt sich zusammen aus Softwarearchitektur und Referenzarchitektur[1].

Eine Definition des Begriffs Softwararchitektur liefern Bass, Clements und Kazman [BCK97]:

Definition 5: Softwarearchitektur nach Bass, Clements, Kazman

The software architecture of a program or computing system is the structure or structures of the system, which comprise software components, the externally visible properties of those components, and the relationships among them.

Softwarearchitekturen sind also Modelle von Softwaresystemen, die aus Komponenten bestehen und deren Eigenschaften sowie Beziehungen zueinander beschrieben sind. In dieser Arbeit werden Softwarearchitekturen nicht allgemein, sondern für eine bestimmte Domäne betrachtet. Diese Domäne umfasst Geschäfts- beziehungsweise Erlösmodelle des e-Business. Die im Rahmen dieser Arbeit hergeleiteten Softwarearchitekturen dienen somit als Referenzlösungen für diesen Anwendungsbereich und werden deshalb als *Software-Referenzarchitekturen* bezeichnet.

1. In dieser Arbeit wird der Begriff ’Software-Referenzarchitektur’ dem Begriff der ’Referenz-Softwarearchitektur’ vorgezogen.

Eine allgemeiner gefasste Definition von Referenzmodellen liefert [Goe03]:

Definition 6: Referenzmodell nach Goeken[2]
Ein Referenzmodell ist eine Empfehlung, die für die Entwicklung konkreter Modelle nützlich ist. Es stellt allgemein gültige Lösungsvorschläge für eine (abstrakte) Klasse von Problemen dar und unterstützt die auf konkrete Aufgabenstellungen bezogene Problemlösung, indem es einen Ausgangspunkt bietet und als Modellmuster für eine Klasse zu modellierender Sachverhalte herangezogen werden kann.

Diese Definition von Referenz*modellen* wird hier auf Referenz*architekturen* übertragen. Dies erscheint legitim, zumal eine Architektur ebenfalls ein Modell darstellt, wie die vorherige Definition von Softwarearchitekturen verdeutlicht hat.

Auf Grundlage der Software-Referenzarchitekturen können konkret umzusetzende Softwarearchitekturen entwickelt und entsprechend den anwendungsfallspezifischen Anforderungen angepasst werden. Der Begriff wird noch einmal ausführlicher in Kapitel 5 im Zusammenhang mit Softwarearchitekturen behandelt.

Nachdem nun die wesentlichen Begriffe dieser Arbeit einleitend vorgestellt und ihre Bedeutungen im Sinne dieser Arbeit eingegrenzt wurden, sollte die Absicht der Arbeit, *Software-Referenzarchitekturen für Geschäftsmodelle des e-Business unter besonderer Beachtung ihrer Erlösmodelle* zu entwickeln, verständlich geworden sein. Mit diesem Vorhaben befassen sich die nun folgenden Kapitel.

2. Goeken bezieht sich in seiner Definition auf [FL02], [Sch97] und [WAB02]

3 Geschäfts- und Erlösmodelle im e-Business

Untersuchungsgegenstand dieses Kapitels sind existierende wissenschaftliche Arbeiten zu Geschäfts- und Erlösmodellen des e-Business. Es wird dabei untersucht, inwiefern geeignete Definitionen und Beschreibungen existieren und inwiefern sich diese Ansätze dazu eignen, Klassifikationen zu ermöglichen, die anschließend als Ausgangspunkt der zugehörigen Softwarearchitekturen dienen können.

Die Literaturrecherche hat ergeben, dass zahlreiche Autoren die Erlösmodelle als einen Teil der Geschäftsmodelle betrachten und in ihren Arbeiten behandeln. Darüber hinaus gibt es jedoch auch Autoren, die sich dem Thema Erlösmodelle im e-Business losgelöst vom Kontext der Geschäftsmodelle gewidmet haben. Aus diesem Grund werden zunächst in Abschnitt 3.1 die Arbeiten analysiert, die sich auf Geschäftsmodelle konzentrieren, bevor in Abschnitt 3.2 auf Ansätze der isolierten Betrachtung von Erlösmodellen eingegangen wird.

Die analysierten Arbeiten werden in beiden Abschnitten unter den folgenden Fragestellungen diskutiert:

1. Inwiefern liefern die Arbeiten Aussagen über die Beziehung und Wechselwirkung von Geschäfts- und Erlösmodellen?

2. Welche Klassifikationen für Geschäfts- und Erlösmodelle werden angeboten?

3. Inwiefern lassen sich diese Klassifikationen und die verwendeten Klassifikationskriterien dazu verwenden, geeignete Softwarearchitekturen herzuleiten?

Die wichtigste der drei Fragen im Zusammenhang dieser Arbeit ist sicherlich die dritte. Nur, wenn sich aus den gelieferten Klassifikationen auch softwaretechnische Aussagen herleiten lassen, kann das Ziel der Arbeit in Form von Architekturmustern erreicht werden.

Abschließend befasst sich Abschnitt 3.3 mit den Schlussfolgerungen der Literaturanalyse.

3.1 Verwandte Arbeiten über Geschäftsmodelle im e-Business

Die wissenschaftliche Behandlung von Geschäftsmodellen ist noch relativ neu und der Begriff wird in der Literatur sehr unterschiedlich verwendet. Je nach betriebswirtschaftlicher, informationstechnischer oder populärwissenschaftlicher Sichtweise ist der Begriff des Geschäftsmodells verschieden oder gar nicht exakt definiert [Rap01]. Eine Untersuchung von [SDL03] im Dezember 2003 hat herausgefunden, dass 33 verschiedene Autoren eine Begriffserklärung des Geschäftsmodells geliefert haben, wobei 25 der Arbeiten davon erst in den Jahren 2000 und später entstanden sind. Die im folgenden Verlauf analysierten Arbeiten stellen einen Auszug einiger früher Arbeiten dar, die von vielen Autoren anschließend referenziert wurden und die sich darüber hinaus für das Ziel dieser Arbeit eignen.

3.1.1 Das Geschäftsmodell nach Timmers

Einer der ersten Autoren, die im Umfeld von e-Business und e-Commerce eine explizite Definition des Begriffes Geschäftsmodell geliefert haben, ist Timmers. Nahezu alle folgenden Autoren haben seine Arbeit referenziert und übernommen, angepasst oder erweitert. Der Schwerpunkt seiner Arbeit ist der Handel im Business-to-Business unter der Verwendung elektronischer Marktplätze ([Tim98], [Tim99]). Aufgrund der Relevanz von Timmers' Arbeit für weitere Autoren wird sein Ansatz hier relativ ausführlich wiedergegeben und diskutiert.

Die folgende Definition von Timmers beschreibt, welche Aspekte ein vollständiges Geschäftsmodell zu berücksichtigen hat. Sie dient vielen anderen wissenschaftlichen Arbeiten über internetbasierte Geschäftsmodelle als Grundlage.

Definition 4: Geschäftsmodell nach Timmers
Ein Geschäftsmodell umfasst folgende Elemente:

- *Eine Architektur[1] für das Produkt, den Service und die Informationsflüsse*
- *Eine Beschreibung der beteiligten Akteure und ihrer Rollen*
- *Eine Beschreibung des potenziellen Nutzens der Akteure*
- *Eine Beschreibung der Einkommensquellen für den oder die Betreiber des Geschäfts.*

Timmers verwendet den Begriff der Einkommensquellen (original: sources of revenue) und definiert diese als einen integralen Bestandteil von Geschäftsmodellen. Timmers spezifiziert in seiner Arbeit die Einkommensquellen nicht explizit, liefert aber anhand existierender Geschäftsmodelle einige Beispiele, die weiter unten in Tabelle 3-1 aufgelistet sind. Aufgrund dieser Beispiele und aufgrund der Tatsache, dass Timmers die *sources of revenue* darüber hinaus nicht näher behandelt, wird für diese Arbeit davon ausgegangen, dass die Einkommensquellen dem hier verwendeten Begriff der Erlösmodelle gleichgesetzt werden können. Insofern kann behauptet werden, dass nach Timmers' Definition ein Geschäftsmodell immer auch ein Erlösmodell beinhaltet.

Bevor die von Timmers identifizierten elf Geschäftsmodelle vorgestellt werden, wird die systematische Vorgehensweise behandelt, mit der Timmers *Architekturen* für Geschäftsmodelle identifiziert. Dabei ist zu beachten, dass er den Begriff Architektur nicht im Sinne von Softwarearchitektur verwendet, sondern mit Organisation gleichsetzt (vgl. [Tim99], S.31). Dennoch wird untersucht, inwiefern dieser Ansatz grundsätzlich hilfreich ist für die Herleitung von Softwarearchitekturen.

Die Vorgehensweise basiert auf einer Wertschöpfungsketten-Dekonstruktion und -Rekonstruktion in Verbindung mit der Suche nach möglichen Wegen, um Informationsverarbeitung entlang der Kette zu integrieren. Die Vorgehensweise umfasst drei Schritte:

1. Timmers setzt den Begriff *Architektur* mit *Organisation* gleich.: „A business model is defined as the organization (or 'architecture') of product, service and information flows, and the sources of revenues and benefits for supplier and customer." [Tim99], S.31

1. Wertschöpfungsketten-Dekonstruktion
   Dies erfolgt durch Identifikation der einzelnen Glieder der Wertschöpfungskette eines Unternehmens. Timmers bedient sich dabei Porters [Por99] Ansatz, für den diese Wertschöpfungskette eines Unternehmen allgemein aus den primären Elementen Eingangslogistik, Operationen, Ausgangslogistik, Marketing & Vertrieb, Kundendienst und den unterstützenden Elementen Beschaffung, Technologieentwicklung, Personalwirtschaft und Unternehmensinfrastruktur besteht (siehe Abbildung 3-1).

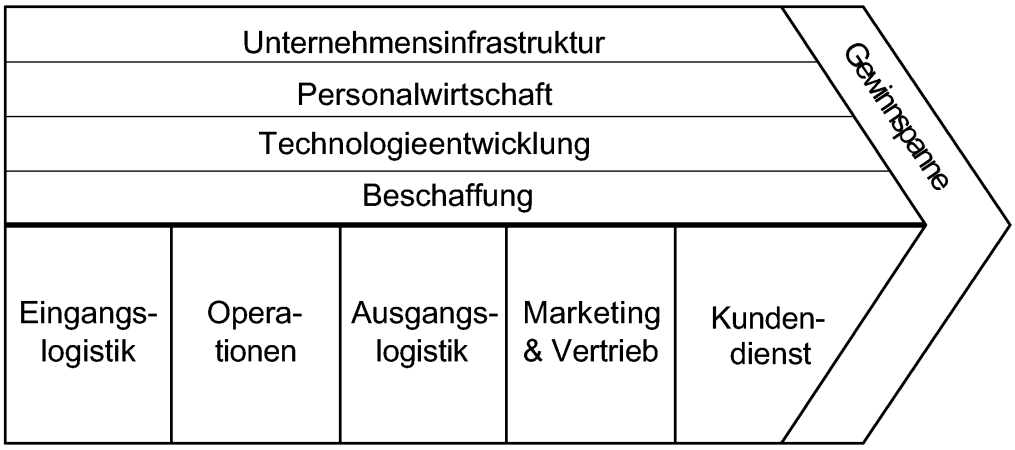


Abbildung 3-1: Wertschöpfungskette nach [Por99]

2. Interaktionsmuster
   Die Interaktionsmuster werden anhand der Anzahl der beteiligten Parteien gebildet und können in den Formen 1:1, 1:n, m:1 oder n:m auftreten. Timmers gibt nicht explizit an, wer mit wem konkret in welcher Form interagiert, jedoch geht aus seinen Beispielen hervor, dass er allgemein den Informationsaustausch und die Transaktionen zwischen den Beteiligten einer oder mehrerer Elemente der Wertschöpfungskette betrachet (vgl. [Tim99], S. 34).

3. Wertschöpfungsketten-Rekonstruktion
   Dieser letzte Schritt des Vorgehensmodells umfasst die Integration der Informationsverarbeitung in einen oder mehrere Schritte der Wertschöpfungskette. Je nach Interaktionsmuster aus Schritt 2 werden ein oder mehrere Glieder der Wertschöpfungskette je Beteiligten zusammengefasst und unter Berücksichtigung der Informationsverarbeitung „rekonstruiert“.

Vor allem dieser letzte Schritt erscheint sehr undurchsichtig und wird von Timmers auch nicht detailliert erläutert. Er gibt lediglich an, dass „mögliche Architekturen von Geschäftsmodellen dadurch konstruiert werden, indem die Interaktionsmuster mit der Wertschöpfungsketten-Rekonstruktion kombiniert“ werden. Demnach sei z.B. ein e-Shop eine 1:1 Verknüpfung von Anbietern und Käufern im Bereich (=Glied der Wertschöpfungskette) Marketing & Vertrieb. Eine Auktion würde dementsprechend eine 1:n Relation von Anbietern und Käufern des gleichen Wertschöpfungskettengliedes entsprechen.

Auch wird offensichtlich, dass Architekturen in keinster Weise technologisch orientiert zu verstehen ist. Timmers unterscheidet nicht zwischen Architekturen von Geschäftsmodellen und den Geschäftsmodellen selbst. Er setzt beide Begriffe auch aufgrund seiner Definition gleich. Bezogen auf technologische Aspekte gibt Timmers lediglich an, dass die grundsätzliche Machbarkeit, eine Architektur für ein Geschäftsmodell zu implementieren, stark von der verfügbaren Technologie abhängt. Er kommt zu folgenden Erkenntnissen:

- Informations- und Kommunikationstechnologie ermöglicht eine große Vielzahl an Geschäftsmodellen.
- Die Tauglichkeit der verfügbaren Technologie ist nur ein Kriterium zur Auswahl eines Geschäftsmodells.
- Anstöße zur Technologieentwicklung können aus der Definition von neuen Geschäftsmodellen entstehen.
- Viele der theoretisch vorstellbaren Geschäftsmodelle wurden noch nicht kommerziell ausgetestet.

Während Timmers' systematischer Ansatz zur Identifikation von Architekturen zu einer Vielzahl potenzieller Geschäftsmodelle führt, identifiziert er durch Marktbeobachtung und Studien letztendlich elf verschiedene Typen von praxisrelevanten Geschäftsmodellen für elektronische Märkte, die in Tabelle 3-1 wiedergegeben werden. Neben dem Geschäftsmodellnamen und einer erläuternden Beschreibung werden die von Timmers ermittelten Einnahmequellen explizit mit aufgeführt.

Geschäftsmodell	Beschreibung	Einnahmequellen
e-Shop (suchende Nachfrage)	Dieses Geschäftsmodell verbindet traditionelle Handelsformen mit e-Commerce. Durch Web Marketing wirbt das Unternehmen für sich und seine Güter und Dienstleistungen. (Beispiel: Amazon, www.amazon.com, Fleurop, www.fleurop.com)	Einnahmen werden aufgrund reduzierter Ausgaben in Verbindung mit erhöhten Einnahmen erzielt. Zusätzlich besteht die Möglichkeit, Werbeeinnahmen zu erzielen.
e-Procurement (Lieferantensuche)	Dies ist die elektronische Ausschreibung und Beschaffung von Gütern und Dienstleistungen, meist durch große Unternehmen oder öffentliche Einrichtungen implementiert. (Beispiel: Beschaffungsplattform der Automobilindustrie für Zulieferer Convisint, www.convisint.de)	Einnahmen basieren auf Reduzierung der Kosten (automatisierte Angebotsabwicklung und kosten-effektivere Angebote).

Tabelle 3-1: Geschäftsmodelle nach Timmers [Tim98]

Geschäftsmodell	Beschreibung	Einnahmequellen
e-Auction (elektronische Abgabe von Geboten)	Neben der multimedialen Präsentation der Güter integriert diese elektronische Version der Versteigerung Dienstleistungen zur Vertragserstellung, Zahlung und Lieferung. (Beispiel: eBay, www.ebay.com)	Für den Betreiber bieten sich Einnahmequellen durch den Verkauf der technischen Plattform sowie durch Transaktionsgebühren und Werbeeinnahmen.
e-Mall (Zusammenfassung mehrerer e-Shops)	Elektronische Malls fassen mehrere e-Shops unter einem Dach zusammen, häufig mit zentraler Anmeldung und Bezahlung. Die weiteren e-Shops können unter dem Dach einer bekannten Marke auftreten oder sich auf ein Marktsegment spezialisieren (Industriemarktplätze). (Beispiel: shopping24, www.shopping24.de)	Einnahmen durch Mitgliedsbeiträge, Werbeeinnahmen und Transaktionsgebühren möglich.
Third Party Marketplace (fremdbetriebener Marktplatz als zusätzlicher Vertrieb)	Hierbei handelt es sich um Marktplätze, die Firmen von Dritten fremdbetreiben lassen, um dadurch einen zusätzlichen Absatzkanal zu erhalten. Der Betreiber stellt häufig Leistungen wie Bestellung, Bezahlung, Logistik und Transaktionsunterstützung zur Verfügung. (Beispiel: Marktplatz der Chemie-Branche, www.chemconnect.com)	Erlöse werden durch einmalige Mitgliedsbeiträge, Dienstleistungsgebühren, Transaktionsgebühren oder Anteile am Transaktionswert erzielt.
Virtual Communities (Mehrwert durch Kommunikation zwischen Mitgliedern)	Hier entsteht vor allem durch die Information ein Mehrwert, die die Mitglieder durch ihre Kommunikation dem Betreiber der Plattform liefern. Häufig sind Virtual Communities Ergänzungen zu anderen Geschäftsmodellen, um Kunden zu binden (Beispiel: Ciao, www.ciao.com, Dooyoo, www.dooyoo.de)	Erlöse werden durch Mitgliedsbeiträge und Werbung erzielt.
Value Chain Service Provider (Spezialisierung auf ein Glied der Wertschöpfungskette)	Hierbei handelt es sich meist um Dienstleister, die sich auf ein bestimmtes Element der Wertschöpfungskette spezialisieren, z.B. elektronisches Bezahlen oder Logistik. (Beispiel: UPS, www.ups.com).	Sie finanzieren sich über eine Gebühr oder einen prozentualen Anteil am Warenwert.
Value Chain Integrator (Integration mehrerer Elemente der Wertschöpfungskette)	Diese Dienstleister integrieren mehrere Schritte einer Wertschöpfungskette um den Informationsfluss zwischen den einzelnen Schritten auszunutzen. (Beispiel: HotDispatch, www.hotdispatch.com oder Amazons zShops).	Sie finanzieren sich durch Beratungsleistungen oder Transaktionsgebühren.

Tabelle 3-1: Geschäftsmodelle nach Timmers [Tim98]

Geschäftsmodell	Beschreibung	Einnahmequellen
Collaboration Platforms (Unterstützung der Zusammenarbeit zwischen Unternehmen)	Diese Plattformen stellen ein Set von Werkzeugen bereit und liefern eine Informationsumgebung, die die Zusammenarbeit von Unternehmen unterstützt. Dabei können sie sich auf bestimmte Funktionen wie z.B. den kollaborativen Entwurf und kollaborative Technologien konzentrieren oder auf Projektunterstützung wie z.B. in Form eines virtuellen Teams von Beratern. (Beispiel: experimentelle Projekte für kollaboratives 3D-Design und Simulation).	Erlöse können durch die Plattformverwaltung oder durch den Verkauf der Werkzeuge entstehen.
Information Brokers (Informationsdienste)	Hierbei handelt es sich um eine Vielzahl von Diensten, die z.B. geschäftliche Informations- und Beratungsleistungen liefern. (Beispiel: Yahoo, www.yahoo.com).	Sie finanzieren sich nicht nur durch Werbung, sondern können darüber hinaus durch Mitgliedsbeiträge, einmalige Dienstleistungszahlungen oder Beratungsgebühren Einnahmen erzielen.
Trust Services (Treuhanddienste)	Diese Vertrauens- oder Treuhanddienste werden von Zertifikationsbehörden, elektronischen Notaren und anderen vertrauenswürdigen Dritten angeboten. (Beispiel: Verisign, www.verisign.com).	Sie kombinieren Mitgliedsbeiträge mit einmaligen Dienstleistungszahlungen.

Tabelle 3-1: Geschäftsmodelle nach Timmers [Tim98]

Die Einnahmequellen zu den einzelnen Geschäftsmodellen werden vom Autor nur kurz angegeben und nicht näher begründet oder definiert.

Timmers mischt bei der Nennung der Erlösquellen Einnahmen, die durch Zahlungen externer Akteure an das Unternehmen zustande kommen, mit Kostenreduzierungen und Effektivitätssteigerung. So gibt er für das Geschäftsmodell e-Procurement als Einnahmequelle eine *Reduzierung der Kosten* an[2]. Betrachtet man nur die Möglichkeiten der Erlösgenerierung durch die Zahlung von externen Akteuren, so lässt sich aus den elf Geschäftsmodellen die folgende Liste herausfiltern:

- WERBUNG

  Timmers sieht dieses Erlösmodell in vielen Geschäftsmodellen als mögliche Quelle des Einkommens an.

- VERKAUF VON PRODUKTEN UND DIENSTLEISTUNGEN

  Dieses Erlösmodell wird aus den Angaben zu den Geschäftsmodellen *e-Shop* und *Information Broker* abgeleitet. Auch der Verkauf von Werkzeugen (Softwaresysteme) wie beim Geschäftsmodell der *Collaboration Platform* gehört hierzu.

2. „The main source of income is reduction of cost (automated tender processing, more cost-effective offers)“, [Tim98]. Neben dieser *main source* werden keine weiteren angegeben.

- TRANSAKTIONSGEBÜHREN

  Beispielsweise aus den Geschäftsmodellen *e-Auction*, *e-Mall*, *Value Chain Service Provider* und *Value Chain Integrator*

- MITGLIEDSBEITRÄGE

  Timmers nennt beispielsweise *Virtual Communities* und *Information Broker* als Geschäftsmodelle.

Wie bereits erwähnt, geht der Autor nicht näher auf diese Erlösmodelle (oder Erlösquellen) ein und liefert sie auch nicht als grundsätzlich mögliche Formen, sondern verwendet sie ausschließlich in seinen identifizierten Beispielen. Er liefert somit nicht wirklich eine Klassifikation der Erlösmodelle. Dennoch dient die Liste als ein erster Ansatz.

Timmers liefert dagegen explizit ein Klassifikationsschema für Geschäftsmodelle. Dieses berücksichtigt aber nicht seine zuvor identifizierten Bestandteile wie Produkte, Akteure, Nutzenaspekte und Einkommensquellen. Statt dieser Bestandteile der Geschäftsmodelle in die Klassifikation mit einfließen zu lassen, verwendet er ein zweidimensionales Schema.

Dieses umfasst die zwei Dimensionen: Innovationsgrad und funktionale Integration. Der Innovationsgrad reicht von elektronischen Versionen bisheriger Geschäfte bis hin zu völlig neuen Geschäften, die nur durch den Einsatz der Internettechnologien entstehen können. Die funktionale Integration reicht von monofunktionalen Geschäftsmodellen (z.B. e-Shop mit reinem Online-Marketing) bis zu vollintegrierter Funktionalität wie Wertschöpfungsketten-Integration. Die folgende Abbildung 3-2 zeigt die Einordnung der Geschäftsmodelle innerhalb dieses Schemas.

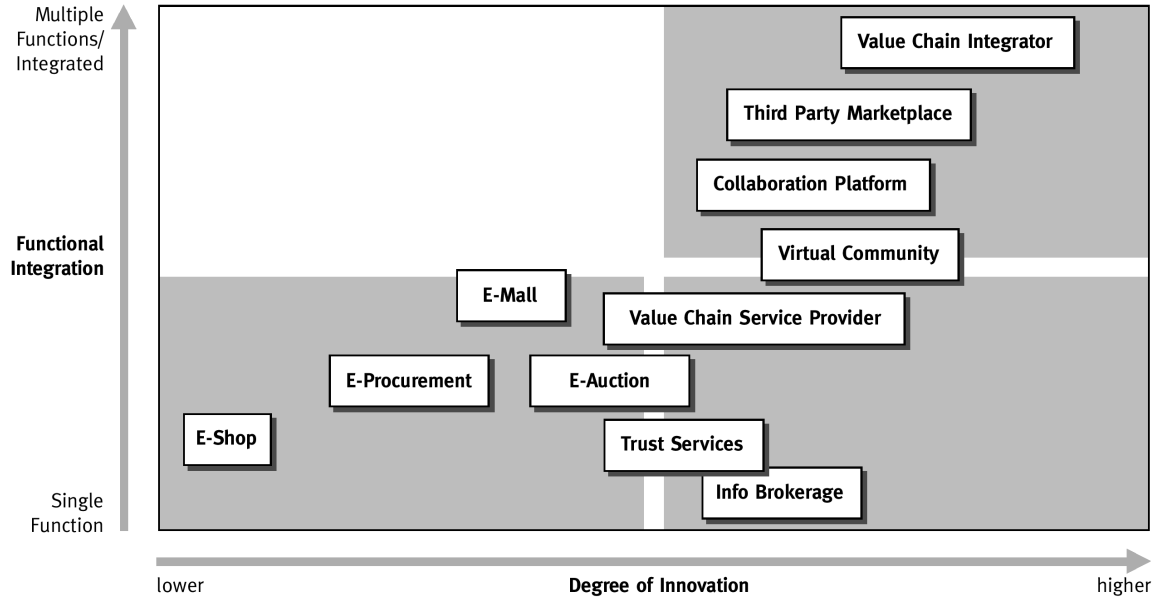


Abbildung 3-2: Klassifizierung von e-Business Geschäftsmodellen nach [Tim98]

Die Abbildung zeigt in der unteren linken Ecke einfache e-Shops, die lediglich elektronische Versionen traditioneller Absatzkanäle darstellen. Demgegenüber steht die Wertschöpfungsketten-Integration (Value Chain Integrator), die in traditioneller Form nicht möglich wäre, da sie in hohem Maße von der Informations- und Kommunikationstechnologie abhängig ist, um Informationen über Netzwerke fließen zu lassen und einen Mehrwert durch die Integration dieser Informationsflüsse zu generieren.

Anhand der Anordnung im Diagramm liest Timmers zwei Trends für Geschäftsmodelle ab: Der eine ist ein Übergang hin zu verstärkter Integration von Informationsflüssen (im oberen rechten Quadranten), der andere die Entwicklung spezialisierter, sehr innovativer Dienste (der untere rechte Quadrant).

Eine Betrachtung der Arbeit Timmers' unter den eingangs aufgeführten beiden hauptsächlichen Gesichtspunkten führt zu den folgenden Ergebnissen.

Timmers sieht ein Erlösmodell als integralen Bestandteil eines Geschäftsmodells an. Dies geht aus der Definition hervor. Ebenso sind die Produkte, Services und Informationsflüsse, die Akteure in ihren Rollen und die Nutzen für die Akteure Bestandteile von Geschäftsmodellen. Es wird nicht darauf eingegangen, inwiefern eine Wechselwirkung zwischen den enthaltenen Elementen besteht. Es wird zum Beispiel nicht untersucht, inwiefern unterschiedliche Produkte und Services oder aber unterschiedliche Akteure die Wahl des Erlösmodells beeinflussen.

Timmers liefert eine Vorgehensweise, wie Geschäftsmodelle zu identifizieren sind. Er zerlegt dazu Aktivitäten des Unternehmens in Wertschöpfungsketten-Elemente und betrachtet jedes einzelne Element hinsichtlich der Möglichkeit, *Informationsverarbeitung* zu integrie-

ren. Auch äußert der Autor, dass die *grundsätzliche Machbarkeit* eines Geschäftsmodells stark von der Informationstechnologie abhängt und dass aus der Definition neuer Geschätfsmodelle *Anstöße für Technologieentwicklungen* entstehen können. Diese Beispiele verdeutlichen Timmers' Auffassung einer engen Verbindung zwischen den Geschäftsmodellen des e-Business und der Informationstechnologie. Insofern deckt sich die Auffassung von Timmers mit dem Vorhaben der vorliegenden Arbeit.

Die elf von Timmers identifizierten Geschäftsmodelltypen werden anhand ihrer Bestandteile (Akteure, Nutzenaspekte und Erlösmodelle) erläutert[3]. Diesen ordnet er einige Erlösmodelle zu, wobei sie zum Teil unbrauchbar sind (eher Einsparpotenziale als Erlösmodelle) und nicht systematisch klassifiziert werden. Die eigentliche Klassifikation seiner Geschäftsmodelle führt Timmers anhand zweier Dimensionen vor, die keinerlei Aussagen zu Softwareachitekturen ermöglichen und auch einen gänzlich anderen Anspruch haben.

Die Klassifikationskriterien sind somit ungeeignet für die Herleitung von Softwarearchitekturen. Ebenso ist das gelieferte Vorgehensmodell nicht verwertbar. Der Begriff der Architektur wird gänzlich von technologischen Aspekten abweichend aufgefasst und Timmers' Erkenntnisse über Technologien sind ebenfalls nicht einzusetzen. Dennoch sind die Bestandteile seines Geschäftsmodells sowie die gelieferten Erlösmodelle ein erster Ansatz für eine eigene Herleitung geeigneter Klassifikationskriterien und anschließende Klassifikation von Erlösmodellen.

Timmers Arbeit ist Grundlage vieler weiterer Arbeiten, die sich mit einer Betrachtung von Geschäftsmodellen im Bereich des e-Commerce und e-Business wissenschaftlich beschäftigen. In den folgenden Kapiteln werden weitere Arbeiten vorgestellt, die aufbauend auf Timmers Definition zusätzliche Aspekte berücksichtigen.

3.1.2 Das Geschäftsmodell nach Mahadevan

Mahadevan [Mah00] liefert eine weitere Definition eines Geschäftsmodells. Demnach setzt sich ein Geschäftsmodell im e-Business aus insgesamt drei sogenannten *Flüssen (streams)* zusammen.

Definition 4: Geschäftsmodell nach Mahadevan
Ein Geschäftsmodell besteht nach Mahadevan aus folgenden Bestandteilen:

- *Wertefluss (value stream)*
  Dieser identifiziert das Wertangebot für Käufer, Verkäufer, Marktmacher und Portale.
- *Ertragsfluss (revenue stream)*
  Dieser dient als Plan zur gesicherten Einkommenserzeugung für das Geschäft.
- *Logistischer Fluss (logistic stream)*
  Hier werden verschiedene Ausprägungen möglicher Lieferketten behandelt.

3. In der Tabelle wurden nur die Erlösmodelle in einer eigenen Spalte aufgeführt. Timmers liefert in seinen Erläuterungen zu den Modellen jedoch häufig auch die Nutzenaspekte und die Akteure. Teilweise sind die Informationen in der Tabelle in den Beschreibungen wiederzufinden.

Ein reales Geschäftsmodell entsteht demnach aus der individuellen Zusammenstellung von Ausprägungen dieser drei Flüsse.

Anhand dieser Aufzählung wird bereits ersichtlich, dass auch Mahadevans Ansatz Übereinstimmungen mit Timmers Modell aufweist, da der Wertefluss im Wesentlichen auf die von Timmers erwähnte Beschreibung des Nutzens für die beteiligten Akteure zurückzuführen ist und der Ertragsfluss den verwendeten Einkommensquellen ähnlich ist. Lediglich der Logistische Fluss ist in dieser Art eine Besonderheit von Mahadevans Definition. Im Folgenden wird auf die einzelnen Flüsse detaillierter eingegangen, wobei der Schwerpunkt der Betrachtungen auf den Ertragsflüssen liegt.

Zunächst werden die Werteflüsse betrachtet. Mahadevan leitet vier mögliche Werteflüsse in internetbasierten Unternehmen ab. Sie geben an, welche Werte für die Akteure generiert werden können. Diese sind gegeben durch...

- eine gemeinsame Interessensvertretung in Virtual Communities
- eine Reduktion der Transaktionskosten für Informationsaustausch, Preis- und Produktangebote
- eine gewinnbringende Ausnutzung der Informationsasymmetrie, wobei ein Mehrwert durch die Verknüpfung von einem Kaufgesuch mit mehreren Anbietern geschaffen werden kann
- die zusätzliche Bereitstellung von Mehrwertdiensten wie beispielsweise Sicherheits- und Vertrauensdienste, Anonymitäts- und Liefergarantien oder finanzielle Instrumente für Transaktionen.

Diese sogenannten Werteflüsse schließen sich dabei nicht gegenseitig aus. Organisationen bauen ihr Geschäftsmodell meist auf Basis eines dominanten Werteflusses auf, der durch zusätzliche Werteflüsse ergänzt wird.

Der Ertragsfluss zeigt, wie in der Definition erwähnt, mögliche Einkommensquellen eines Unternehmens auf und kann somit zunächst grundsätzlich dem Begriff des Erlösmodells gleichgesetzt werden. Mahadevan identifiziert sechs unterschiedliche Ertragsflüsse im e-Business:

- R1. Erhöhte Gewinnmargen gegenüber traditionellen Geschäften

  Diese kommen zustande durch eine Reduzierung der Transaktionskosten, eine vereinfachte Kundensuche und durch Kostenreduzierung mittels Disintermediation (die Reduktion von Vermittlung und Zwischenhandel).

- R2. Erträge von Online-Verkäufer-Communities

  Marktbetreiber bieten Kunden freie Mitgliedschaft in einer Community, die sie mit einer Vielzahl von Hintergrundinformationen zu gewünschten Produkten versorgen (unabhängige und neutrale Informationen, zusätzliche Vergleichsmöglichkeiten). Anbieter, die dieser Community beitreten, treffen auf einen umfangreichen, kaufinteressierten Kundenstamm. Der Markbetreiber generiert einen Ertragsfluss durch Mitglieds- und Transaktionsgebühren von den Anbietern.

- R3. Werbung

  Viele Internetdienste haben als Haupteinnahmequelle die Einblendung von Werbung als Banner oder durch andere Formen, die Kunden auf gewünschte Ziel-Webseiten lenken. Diese Einnahmequelle unterliegt zur Zeit allerdings einem sehr starken Einbruch.

- R4. Variable Preisstrategien

  Unternehmen, die elektronisch auslieferbare Produkte verkaufen, stehen neue Preisstrategien offen. Aufgrund der hohen Vorkosten (Produktentwicklung), aber nahezu vernachlässigbaren Grenzkosten (Produktion und Auslieferung) sind variable Preise und eine besondere Preisgebung für Optionen und gebündelte Produkte möglich. Eine Preisstaffelung in Abhängigkeit vom Zahlungswillen des Kunden ist darüber hinaus ebenfalls denkbar. Dazu müssen entsprechende Kundenprofile auswertbar sein.

- R5. Ertragsflüsse, die die Informationsasymmetrie ausnutzen

  Ein Intermediär (Mittler) kann einen Ertragsfluss generieren, indem er die Informationsasymmetrie (unterschiedlich ausgeprägte Möglichkeiten zur Nutzung und Auswertung von Informationen) zwischen Käufer und Verkäufer ausnutzt. Diese Auktionsformen generieren Einnahmen meist in Abhängigkeit von den Einsparungen des Käufers.

- R6. Freie Angebote

  Hierbei können zukünftige Erträge durch die Rückgabe aktueller Erträge an den Kunden erwirtschaftet werden. Beispielsweise werden Teilprodukte kostenlos abgegeben, die den Bedarf nach weiteren Produkten in der Zukunft wecken. Das Angebot kostenloser Dienste ermöglicht den schnellen Aufbau großer Communities, die dann weitervermarktet werden können, oder deren Feedback der Weiterentwicklung von Produkten oder Diensten zugute kommen kann.

Eine Betrachtung dieser Ertragsflüsse zeigt, dass nicht alle sechs Typen als Erlösmodelle betrachtet werden können. Der erste Ertragsfluss entspricht eher einer Kostenreduzierung und damit verbundenen Erhöhung der Gewinnmarge im Vergleich zu traditionellen Geschäften. Es ist somit lediglich eine Abgrenzung des e-Business, nicht jedoch ein alleinstehendes Erlösmodell. Der vierte Ertragsfluss (Variable Preisgestaltung) entspricht eher einem Modell zur Preisgestaltung als einem Erlösmodell. Das sechste Modell (freie Angebote) geht lediglich darauf ein, die Nutzergemeinde zu vergrößern, ohne jedoch auf die (irgendwann notwendige) Einnahmequelle des Unternehmens einzugehen. Somit lassen sich lediglich die folgenden Erlösmodelle nach Mahadevan identifizieren:

- MITGLIEDSGEBÜHREN

  Aus dem zweiten Ertragsfluss geht dieses Erlösmodell hervor, wobei die Gebühren nicht von den (End-)Kunden getragen werden, sondern von den (unternehmerischen) Anbietern, die sich über eine Community mit ihrem Angebot an die Kundschaft wenden.

- TRANSAKTIONSGEBÜHREN von Anbietern

  Auch diese Gebühren werden von den Anbietern innerhalb einer Community und nicht von den Kunden gezahlt.

- WERBUNG

  Einnahmen durch die Schaltung von Werbebannern.

- TRANSAKTIONSGEBÜHREN von Kunden

  Diese Form der Transaktionsgebühren werden von Kunden gezahlt, wobei sie sich anteilig an den Ersparnissen ihrer Einkäufe berechnen (fünfter Ertragsfluss)

Als dritter Bestandteil eines Geschäftsmodells werden von Mahadevan drei mögliche Lieferketten in Form von Logistischen Flüssen aufgeführt[4]:

- Disintermediation bedeutet die Verkürzung der Lieferkette
- Infomediation bedeutet eine an die Anforderungen der Benutzer angepasste Auswahl, Speicherung und Verbreitung von Informationen
- Meta-Mediation ist ein Prozess, der über die Aggregation von Lieferanten und Produkten hinausgeht und zusätzliche Dienste zur Transaktionsunterstützung umfasst

Nach Mahadevan setzt sich nun ein Geschäftsmodell durch die Mischung von Ausprägungen innerhalb der drei genannten Arten von Flüssen zusammen. Der Prozess der Erstellung eines Geschäftsmodells besteht somit hauptsächlich in der Auswahl einer passenden Kombination aus den möglichen Alternativen.[5]

Betrachtet man die Arbeit von Mahadevan unter den drei relevanten Fragestellungen, so lässt sich zunächst festhalten, dass auch er Erlösmodelle als einen Bestandteil von Geschäftsmodellen ansieht. Dabei muss jedoch darauf hingewiesen werden, dass seine Definition von Ertragsflüssen nicht absolut gleichzusetzen ist mit Erlösmodellen. Er liefert insgesamt sechs Ertragsflüsse, die er ausreichend beschreibt, wodurch eine gute Klassifikation gegeben ist. Aufgrund des abweichenden Charakters der Ertragsflüsse von den hier betrachteten Erlösmodellen können aus dieser Klassifikation jedoch nur vier Erlösmodelle abgeleitet werden, so dass der Ansatz von Mahadevan nicht vollständig übernommen werden kann.

Zur übergeordneten Klassifikation von Geschäftsmodellen liefert er ein dreidimensionales Modell (Mahadevan nennt es *Framework*), dass sich aus den möglichen Werteflüssen, Ertragsflüssen und Logistischen Flüssen zusammensetzt. Diese Flüsse stellen somit die Klassifikationskriterien dar. Die Frage nach der Wechselwirkung zwischen Erlös- und Geschäftsmodell lässt sich lediglich insofern beantworten, als dass die Wahl eines Ertragsflusses die Klassifikation des Geschäftsmodells innerhalb des Frameworks ändert. Eine Präzisierung der Wechselwirkungen wird leider nicht geliefert.

4. Die Wiedergabe der Logistischen Flüsse erfolgt hier sehr stark reduziert.
5. Mahadevan geht noch auf Faktoren ein, die die Auswahl geeigneter Flüsse und somit den Prozess der Modellierung beeinflussen. Dies wird hier jedoch nicht näher betrachtet, da es für diese Arbeit nicht relevant ist.

Auch liefert Mahadevans Arbeit keinerlei Ansätze für die Herleitung von Softwarearchitekturen für die klassifizierten Geschäfts- oder Erlösmodelle, so dass als Fazit im Wesentlichen die vier detailliert charakterisierten Erlösmodelle für den weiteren Verlauf dieser Arbeit von Bedeutung sind und später erneut aufgegriffen werden.

3.1.3 Das Geschäftsmodell nach Buchholz

Buchholz [Buc01] untersucht ebenfalls wie Timmers Business-to-Business Marktplätze und fokussiert dabei auf die Beschaffungsseite. Er führt für diese Einkaufsplattformen den Begriff des *Netsourcing* ein. Auch er referenziert die Arbeiten von Timmers und Mahadevan, die in den vorherigen Abschnitten diskutiert wurden, jedoch bevorzugt er als Definition eines Geschäftsmodells die Version von Amit und Zott [AZ00], auf die in einem nächsten Abschnitt separata eingegangen wird. Als Besonderheit des Ansatzes von Amit und Zott sieht Buchholz deren Differenzierung von Geschäfts- und Erlösmodell (Revenue model[6]). Buchholz erweitert diesen Ansatz und definiert ein Geschäftsmodell als „Klammer von vier konstituierenden Komponenten“ ([Buc01], S. 41).

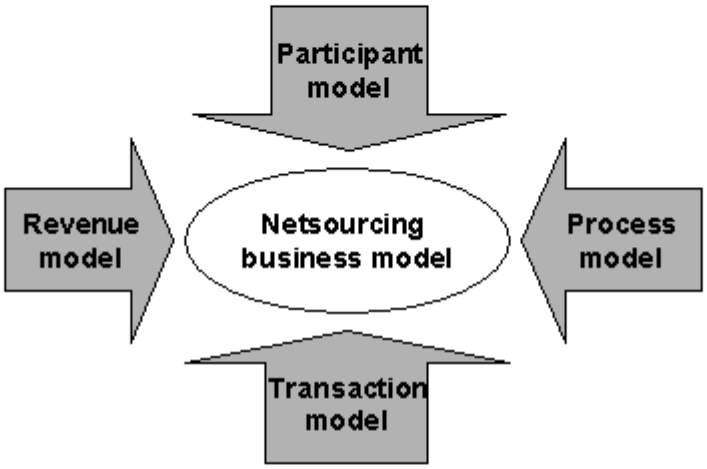


Abbildung 3-3: Geschäftsmodell nach Buchholz [Buc01]

Abbildung 3-3 zeigt die vier konstituierenden Komponenten *Participant Model*, *Process Model*, *Transaction Model* und *Revenue Model*.

Das Participant Model führt die Marktplatzbeteiligten auf, für die ein positiver Wertschöpfungsbeitrag generiert werden soll. Die Beteiligtenstruktur ähnelt dabei den möglichen Interaktionsmustern bei Timmers (siehe Abschnitt 3.1.1).

6. Amit und Zott verstehen unter einem Revenue Model „the spezific modes in which a business model enables revenue generation“.

Das Process Model beschreibt die Geschäftsprozesse, die über den elektronischen Marktplatz durchgeführt werden sollen. Neben der Beschaffungsabwicklung (Procurement) als Schwerpunkt nennt er Supply Chain Management, Product Development, Production und Financial Processes als Prozesse, die durch einen elektronischen Markplatz unterstützt werden können.

Die Art und Weise, wie diese Geschäftsprozesse durch das Internet optimiert werden, detailliert in Buchholz' Ansatz das Transaction Model, das „die Methoden und Instrumente definiert, welche die Transaktion zwischen Käufer und Verkäufer auf der Basis des Internets möglich machen". Im Wesentlichen nennt der Autor hierbei zum einen Kataloge mit statisch definierten Preisen und zum anderen unterschiedliche Auktionsformen zur dynamischen Preisfestlegung. An dieser Stelle zieht Buchholz eine Verbindung zur Softwareanwendung, indem er feststellt, dass durch das Transaction Modell „auch maßgeblich die informationstechnologische Lösung determiniert" wird ([Buc01], S.45). Darüber hinaus geht er jedoch nicht weiter auf diese Thematik ein.

Das Revenue Model (Buchholz nennt es Ertragsmodell) schließlich geht der Frage nach, „wie überhaupt mit dem Markplatz Geld verdient werden soll". Buchholz liefert die folgenden sieben Varianten der Ertragsmodelle, die sich seiner Meinung nach in Theorie und Praxis herauskristalisiert haben:

- TRANSAKTIONSGEBÜHR

  Diese kann mengen- oder wertabhängig sein und fällt für Käufer und/oder Verkäufer bei jeder Transaktion an

- SUBSKRIPTIONSGEBÜHREN

  Sind unabhängig von der Einzeltransaktion und werden wie Mitgliedsbeiträge von den Benutzern für den Marktzugang und die Nutzung bezahlt.

- GAIN SHARING

  Hier erhält der Betreiber einen Anteil der Einsparungen der Käufer und Verkäufer.

- WERBEEINNAHMEN

  - selbsterklärend -

- INFORMATIONSVERKAUF

  Hierbei werden „anonymisierte Marktplatzinformationen" erhoben und veräußert. Buchholz geht nicht näher auf die Art der Informationen ein.

- LIZENSIERUNG der Marktplatz-Technologie

  Hier wird die Software der Plattform für andere Betreiber vermarktet.

- GEBÜHREN für zusätzliche DIENSTLEISTUNGEN

  Als Beispiel einer zusätzlichen Dienstleistung nennt Buchholz das Consulting.

Wichtig ist an dieser Stelle, noch einmal darauf hinzuweisen, dass sich Buchholz Analysen auf elektronische Marktplätze konzentrieren und somit nicht zwangsläufig das gesamte Spektrum der Erlösmodelle des e-Business abdecken. Die meisten der hier vorgestellten Ertragsmodelle sind auch schon in den Arbeiten von Timmers und Mahadevan zu finden. Neu ist an dieser Stelle der Informationsverkauf. Ohne dass die Informationen näher spezifiziert werden, ist ersichtlich, dass die Verhaltensweisen der Marktplatzteilnehmer erhoben, ausgewertet und anonymisiert aufbereitet werden, um sie anschließend zu verkaufen.

Buchholz liefert als Klassifikation von Geschäftsmodellen eine Typologie, die lediglich den „Wertschöpfungsbeitrag des e-markets“ als Differenzierungsmerkmal enthält. Anhand dieses einen Kriteriums liefert er fünf *Netsourcing* Geschäftsmodelle, auf die hier nicht näher eingegangen wird, da sie aufgrund des Klassifikationskriteriums für diese Arbeit nicht verwendet werden können.

Auch Buchholz interpretiert also ein Erlösmodell als einen integralen Bestandteil von Geschäftsmodellen. Er betrachtet es als eine von vier *Komponenten*. Bei der Beschreibung einer weiteren Komponente, dem Transaction Model, zieht er explizit eine Relation zur Softwaretechnik, indem er sagt, dass dieses Modell „maßgeblich die informationstechnische Lösung determiniert“. Leider erläutert der Autor diese Beziehung nicht im Detail und überträgt sie auch nicht auf die übrigen Komponenten. Er nennt beispielhaft Funktionsbausteine wie Kataloge und Auktionsmechanismen, die man aus softwaretechnischer Sicht als Komponenten betrachten kann, wie im weiteren Verlauf dieser Arbeit noch gezeigt wird. Über diese Beispiele hinaus behandelt Buchholz diesen Zusammenhang zwischen den Bestandteilen von Geschäftsmodellen einerseits und dem zugrundeliegenden Softwaresystem andererseits jedoch nicht weiter. Dennoch ist bei Buchholz ersichtlich, dass ausgehend vom Geschäftsmodell die zugrundeliegende Softwareanwendung beeinflusst wird. Somit bekräftigt auch die Auffassung von Buchholz die Ausgangsüberlegung der vorliegenden Arbeit.

Darüber hinaus liefert Buchholz insgesamt sieben verschiedene Ausprägungen von Erlösmodellen, wobei mit dem INOFRMATIONSVERKAUF ein bisher noch nicht erwähntes Modell aufgeführt wurde. Er berücksichtigt diese Erlösmodelle aber (ebensowenig wie die übrigen drei Komponenten) leider nicht bei seiner Typologie von Geschäftsmodellen. Die von ihm klassifizierten Modelle sind für diese Arbeit unbrauchbar, zumal sie sich lediglich anhand eines Kriteriums ergeben, das keinerlei Aussagekraft für softwaretechnische Schlussfolgerungen enthält.

3.1.4 Das Geschäftsmodell nach Dubosson-Torbay, Osterwalder, Pigneur

Die Autoren Dubosson-Torbay, Osterwalder und Pigneur [DOP01][7] liefern in ihrer Arbeit ein Modell (sie nennen es *e-Business Model Framework*), das zur Definition und zur Klassifikation von Geschäftsmodellen herangezogen werden soll. Sie liefern als verbale Definition eines Geschäftsmodells die folgende Version:

7. vgl. auch [OP02]

Definition 5: Geschäftsmodell nach Dubosson-Torbay, Osterwalter, Pigneur
A business model is nothing else than the architecture of a firm and its network of partners for creating, marketing and delivering value and relationship capital to one or several segments of customers in order to generate profitable and sustainable revenue streams.

Entsprechend dieser Definition besteht ihr *e-Business Model Framework* aus vier Komponenten:

- Product Innovation

  Die Produkte und Services eines Unternehmen, die einen wesentlichen Wert für einen Kunden darstellen und für den dieser bereit ist zu zahlen

- Customer Relationship

  Das Kundenbeziehungskapital (customer relationship capital), das das Unternehmen erzeugt und pflegt, um den Kunden zufrieden zu stellen und nachhaltig Einnahmen zu erzielen.

- Infrastructure Management

  Eine Infrastruktur und ein Netzwerk von Partnern, die notwendig sind, um Werte zu erzeugen und die Kundenbeziehung aufrecht zu halten.

- Financial Aspects

  Finanzielle Aspekte wie Kosten und Einkommensstrukturen, die innerhalb der drei vorherigen Komponenten gefunden werden können.

Die erste und dritte Komponente sind ähnlich den Elementen, die auch Timmers in seiner Arbeit als Bestandteile eines Geschäftsmodells definiert hat[8]. Die besondere Herausstellung des Customer Relationships dagegen ist eine Besonderheit dieser Definition. Die Finanziellen Aspekte dagegen sind ebenfalls aus anderen Arbeiten bekannt, wo sie ebenfalls als ein Baustein eines Geschäftsmodells in ähnlicher Form verwendet wurden. Diese Komponente wird nun näher betrachtet.

Die Autoren betonen explizit, dass es nicht ihr Anliegen ist, die finanziellen Aspekte in Form eines Erlösmodells oder Preismodells als das einzige und wichtigste Element eines Geschäftsmodells anzusehen, sondern als ein viertes Element und als die Konsequenz aus den vorherigen drei Komponenten. Die Finanziellen Aspekte umfassen nach Ansicht der Autoren...

- die Kosten für die Infrastruktur, um Werte zu erzeugen,
- die Einnahmen der verkauften Werte,
- und den Profit als Differenz der Einnahmen und Kosten.

8. Das *Infrastructure Management* entspricht weitgehend den *Akteuren in ihren Rollen* von Timmers.

Für die Generierung der Einnahmen geben die Autoren folgende Möglichkeiten an:

- SUBSKRIPTIONSGEBÜHREN der Kunden
- Einnahmen aus WERBUNG und SPONSORING von anderen Firmen
- PROVISIONEN und TRANSAKTIONSANTEILE (transaction cuts) von angebotenen Services
- EINNAHMEAUFTEILUNG (revenue sharing) mit anderen Firmen
- VERKAUF von Produkten

Wie zu erkennen ist, entsprechen diese Möglichkeiten der Generierung von Einnahmen weitgehend den bisher bereits aufgelisteten Erlösmodellen anderer Autoren. Neu sind in dieser Auflistung die Nennung des SPONSORINGS sowie die EINNAHMEAUFTEILUNG mit anderen Firmen. Die Autoren gehen jedoch nicht näher auf diese Erlösmodelle ein und geben auch keine Beispiele, die eine Anwendung in der Praxis verdeutlichen würden.

Inwiefern nun der Ansatz der Autoren, die finanziellen Aspekte als Schlussfolgerung der drei zuvor betrachteten Komponenten zu sehen, von bisherigen Ansätzen anderer Autoren abweichen soll, wird aus der Arbeit nicht ersichtlich. Stattdessen scheint es, dass [DOP01], wie andere Autoren auch, einige potenzielle Erlösmodelle losgelöst von den restlichen Komponenten eines Geschäftsmodells angeben.

Das *e-Business Model Framework* wird von den Autoren dazu verwendet, bestehende Geschäftsmodelle anhand der vier Komponenten strukturierter beschreiben und vergleichen zu können. Die Autoren gehen jedoch weiter und liefern für die Klassifikation von Geschäftsmodellen zusätzliche Kriterien, die sie als Zusammenfassung ihrer Literaturrecherchen aufstellen. Einige der Kriterien übernehmen sie dabei identisch von Timmers [Tim98] und Tapscott/Ticoll/Lowy [TTL99]. Insgesamt liefern sie die folgenden zwölf Klassifikationskriterien:

- Nutzerrolle

  Ist der Kunde/Interessent ein Käufer oder selbst ein Anbieter von Produkten? Oder werden ihm nur Informationen und Dienste kostenlos angeboten in Gegenleistung zu Nutzerinformationen?

- Interaktionsmuster

  Wird das Angebot von einem oder mehreren Anbietern an einen oder mehrere Kunden offeriert?

- Art des Angebots

  Bietet das Unternehmen Informationen, Dienstleistungen oder Produkte an?

- Preissystem

  Wie bezahlt der Kunde? Entsprechend seiner Nutzungsrate, in Form einer festen Subskription oder einer festen oder prozentualen Gebühr? Oder zahlt er einen festen oder dynamisch ermittelten (z.B. mittels Auktion) Preis? Oder bezahlt der Kunde gar nicht?

- Grad der Anpassung (level of customization)

  Das Ausmaß reicht von *mass content* zu *customized content*.

- Wirtschaftliche Kontrolle (economic control)

  Dieses Kriterium reicht von selbst-organisierend (keine einzelne Stelle steuert den Inhalt der Transaktionen oder das wirtschaftliche Ergebnis) bis zu hierarchisch (es gibt eine zentrale Stelle, die den Inhalt, den Preis und die Transaktionsflüsse steuert und kontrolliert).

- Grad der notwendigen Sicherheit

  Das Ausmaß an erforderlicher Sicherheit, um die Käufe zu monitoren und zu verifizieren.

- Grad der Werte-Integration (level of value integration)

  Einige Angebote umfassen zahlreiche ursprüngliche Einzelkomponenten unterschiedlicher Quellen, andere Angebote werden unverändert weitergereicht.

- Wert/Preis Angebot

  Ist das Angebot eher auf hohe oder zusätzliche Werte ausgerichtet oder auf einen billigen Preis?

- Maß des Traffic (scale of traffic)

  Benötigt das Angebot hohe Zugriffszahlen auf die Website oder nicht?

- Innovationsgrad (degree of innovation)

  Gemäß der Definition von Timmers gibt dieses Kriterium an, ob das Angebot eine elektronische Form eines traditionellen Geschäftsmodells ist oder aber es innovativer, indem es Funktionen anbietet, die zuvor nicht existierten.[9]

- Ausmaß der Machtverteilung auf Käufer- oder Verkäuferseite

Die Autoren ordnen diese zwölf Klassifikationskriterien ihren vier Komponenten des Frameworks zu und sind auf diese Weise in der Lage, die Geschäftsmodelle detaillierter zu beschreiben und zu klassifizieren. Der für diese Arbeit interessanteste Komponente Financial Aspects weisen sie jedoch mit dem *Preissystem* nur ein Kriterium zu.

Im weiteren Verlauf ihrer Arbeit wird das Framework mitsamt den zugeordneten Klassifikationskriterien dazu verwendet, um jeweils Kennzahlen zu identifizieren, anhand derer „der Erfolg des Unternehmens und seines Geschäftsmodells gemessen werden kann“[10]. Als Beispiel liefern sie für eine Auktionsplattform die Kennzahlen Anzahl der Kunden, Anzahl der registrierten Nutzer, Marktanteil und Anzahl der Besucher pro Tag. Diese Kenngrößen sind wie bereits erwähnt dafür hergeleitet worden, um den Erfolg eines Geschäftsmodells

9. vgl. Seite 25

10. „Designing and managing a business model requires a measurement system which identifies the key factors and indicators by which the success of the company and its business model can be assessed.“ [DOP01]

zu messen, wobei der Begriff *Erfolg* in der Arbeit nicht näher erläutert wird. Auf jeden Fall zielen die Kennzahlen nicht darauf ab, Aussagen über softwaretechnische Aspekte herleiten zu können, sondern sind betriebswirtschaftlich orientiert.

Betrachtet man die Arbeit der drei Autoren unter den relevanten drei Fragestellungen, so erkennt man zunächst, dass auch sie das Erlösmodell als Bestandteil eines Geschäftsmodells ansehen. Obwohl sie sich äußern, dass ihre Komponente *Financial Aspects* weniger das wichtigste Element eines Geschäftsmodells sondern vielmehr ein Resultat der übrigen drei Komponenten sei, wird dieser Ansatz nicht ersichtlich. Anstatt auf eine Wechselwirkung oder Beeinflussung zwischen den Komponenten einzugehen, liefern die Autoren wie andere zuvor auch einige mögliche Erlösmodelle, die die Basis der Einkommensströme bilden können. Diese können grundsätzlich für das weitere Vorhaben der vorliegenden Arbeit verwendet werden.

Das Modell (beziehungsweise das sogenannte *e-Business Model Framework*) der vier Komponenten wird um zwölf Klassifikationskrierien ergänzt, so dass eine sehr feingranulare strukturierte Beschreibung und damit auch Klassifikation von Geschäftsmodellen möglich ist. Dieser Ansatz unterscheidet die Arbeit von Dubosson-Torbay, Osterwalder und Pigneur von übrigen Arbeiten, da sie ein wirklich umfangreiches Beschreibungsschema liefern, dass nicht nur ein oder zwei, sondern zahlreiche Kriterien umfasst. Obwohl die Autoren die meisten Kriterien nicht den Erlösmodellen (bwz. den Finanziellen Aspekten) zuordnen, werden sie im weiteren Verlauf dieser Arbeit noch berücksichtigt. Es ist dabei noch zu untersuchen, inwiefern sie auch Potenziale aufweisen, die Erlösmodelle näher zu beschreiben und eine Klassifikation ermöglichen, die die Herleitung von Softwarearchitekturen erlaubt. Denn auch die Arbeit dieser drei Autoren zielt nicht darauf ab, Aussagen bezüglich der zugrundezulegenden Softwarearchitektur herleiten zu können. Stattdessen konzentrieren sich die Autroen auf betriebswirtschaftliche Aspekte von Geschäftsmodellen, wie auch die Verwendung der Kennzahlen zur Messung des Erfolgs der Geschäftsmodelle zeigt.

3.1.5 Das Geschäftsmodell nach Wirtz

Wirtz [Wir01] befasst sich ebenfalls mit Geschäftsmodellen im e-Business. Er versteht unter einem Geschäftsmodell das Folgende:

Definition 6: Geschäftsmodell nach Wirtz
Ein Geschäfstmodell bildet ab, welche Ressourcen in die Unternehmen fließen und wie diese durch den innerbetrieblichen Leistungserstellungsprozess in marktfähige Informationen, Produkte und/oder Dienstleistungen transformiert werden.

In dieser Definition geht der Autor nicht direkt auf Erlöse oder sonstige finanzielle Aspekte ein. Dennoch spaltet er ein Geschäftsmodell in insgesamt sechs Partialmodelle auf, wobei dann das Erlösmodell berücksichtigt wird. Abbildung 3-4 zeigt diese Teilmodelle.

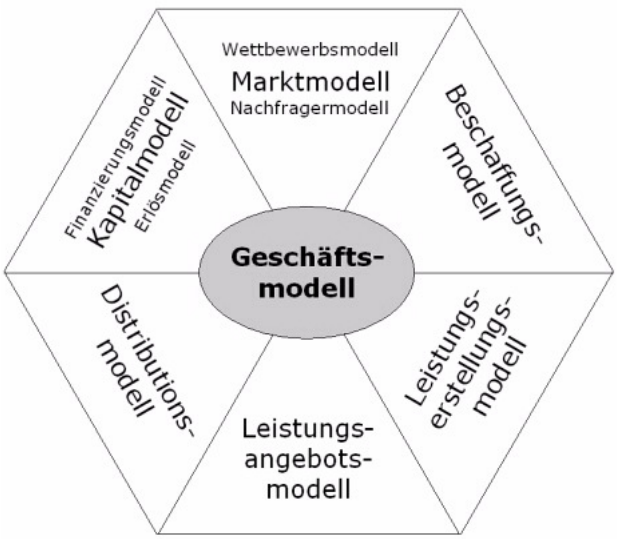


Abbildung 3-4: Geschäftsmodell und Partialmodelle nach Wirtz [Wir01]

Die Modelle sind im einzelnen:

- Das Marktmodell beschreibt, welchen Akteuren das Unternehmen in welchen Märkten gegenübersteht und wie diese Märkte strukturiert sind.
- Das Beschaffungsmodell bildet das Verhältnis des Unternehmens zu seinen Lieferanten ab.
- Das Leistungserstellungsmodell bildet den Prozess ab, in dem aus den einfließenden Ressourcen das Angebot erstellt wird.
- Das Leistungsangebotsmodell geht der Frage nach, welche Leistungen den Kunden angeboten werden sollen.
- Das Distributionsmodell legt fest, welche Produkte in welcher Form zu welchem Zeitpunkt zum Kunden transportiert werden.
- Das Kapitalmodell besteht aus einem Finanzierungs- und einem Erlösmodell. Das Finanzierungmodell zeigt die Quellen auf, die das für die Finanzierung der Tätigkeiten notwendige Kapital liefern. Beim Erlösmodell übernimmt der Autor den Ansatz von Zerdick et al., der in in Abschnitt 3.2.1 beschrieben wird. Dabei werden je nach Herkunft der Erlöse *direkte* (direkt vom Kunden) und *indirekte* (von Dritten) Erlösformen unterschieden. Da dieses Erlösmodell später ausführlich diskutiert wird, wird hier auf weitere Erläuterungen verzichtet.

Im weiteren Verlauf liefert Wirtz eine Typologie von Geschäftsmodellen im e-Business. Er grenzt die Geschäftsmodelle anhand eines einzigen Kriteriums ab, dem *Leistungsangebot*. Er nennt das Modell entsprechend der Namen der Geschäftsmodelltypen das *4C-Net-Business-Model*. Danach lassen sich vier Basisgeschäftsmodelle voneinander abgrenzen:

- Content

  Dieser Typ umfasst die Sammlung, Selektion, Systematisierung, Kompilierung (Packaging) und Bereitstellung von Inhalten auf einer eigenen Plattform.

- Commerce

  Dieser Geschäftsmodelltyp umfasst die Anbahnung, Aushandlung und/oder Abwicklung von Geschäftstransaktionen.

- Context

  Dieser Typ liefert dem Nutzer einen Überglick und eine Orientierung über die online verfügbaren Informationen, indem diese klassifiziert und systematisiert werden.

- Connection

  Dieses Basisgeschäftsmodell ermöglicht die Interaktion von Akteuren in virtuellen Netzwerken, die wegen der Höhe der Transaktionskosten oder aber aufgrund on Kommunikationsbarrieren in der physischen Welt nicht realisierbar wären.

In einer näheren Betrachtung der vier Basismodelle identifiziert der Autor keine konkreten Erlösmodelle, sondern gibt lediglich zu jedem Typ an, ob direkte oder indirekte Erlösmodelle vorkommen.

In einer weiteren Arbeit von Wirtz und Becker [WB02] werden im Rahmen einer näheren Analyse des Marktes insgesamt elf Geschäftsmodellvarianten und dazu wiederum insgesamt knapp 30 Untergruppen von Geschäftsmodellen identifiziert. Obwohl diese Klassifikation, die die Autoren *Mehr-Ebenen-Betrachtung* nennen, sehr feingranular aufgezogen ist, liefern die Autoren leider keine durchgängige Betrachtung der jeweiligen Erlösmodelle. Lediglich ein einziges Mal wird auf diese Thematik eingegangen.

Unter Betrachtung der Fragestellungen lässt sich festhalten, dass auch Wirtz das Erlösmodell als einen integralen Bestandteil eines Geschäftsmodells ansieht. Er übernimmt den Ansatz von Zerdick et al. zur näheren Typisierung der Erlösmodelle. Sein Vorgehen, Geschäftsmodelle konsequent nach ihrem Leistungsangebot zu klassifizieren, erscheint zunächst interessant und vielversprechend für die Betrachtung von Erlösmodellen, da die Nutzer genau für diese Leistungen in der Regel auch bereit sind zu bezahlen. Auch die Identifikation von insgesamt 30 Ausprägungen von Geschäftsmodelltypen und -subtypen ermöglicht eine feingranulare Betrachtung existierender Modelle im e-Business. Leider berücksichtigen der Autor und seine Kollegen jedoch in ihren Untersuchungen die jeweiligen Erlösmodelle nicht. Der exemplarische Ansatz, direkte und indirekte Erlösformen zu unterscheiden, wurde während der Identifikation der Geschäftsmodelltypen leider nicht vollständig angewendet und steht somit nicht zur Verfügung. Darüber hinaus bleiben in den Arbeiten beider Autoren die softwaretechnischen Aspekte gänzlich unberücksichtigt.

3.1.6 Das Geschäftsmodell nach Bartelt et al.

Bartelt et al. untersuchen in ihren Arbeiten ([BL00], [BWL00]und [BZF01]) Geschäftsmodelle im e-Business, wobei sie dabei auch die Herleitung von Softwaresystemen mit berücksichtigen:

„Um diese [Geschäftsmodelle] zu konkretisieren und die Erstellung entsprechender Softwarelösungen zu fördern, werden adäquate Modellierungsmethoden benötigt, die jedoch derzeit nur in geringem Maße vorhanden sind“ (aus [BWL00]).

Der Fokus der vorliegenden Analyse in diesem Abschnitt liegt auf dem Papier von Bartelt und Lamersdorf ([BL00]), in dem sie zum einen ein Klassifikationsschema und zum anderen ein Instrument zur Modellierung von Geschäftsmodellen liefern.

Die beiden Autoren orientieren sich in ihrer Arbeit an der Geschäfstmodelldefinition von Timmers. Sie liefern ein zweidimensionales Klassifikationsschema von Geschäftsmodellen. Die erste Dimension wird dabei durch das Wirtschaftssubjekt (Anbieter, Nachfrager, Mittler) gebildet, das das Modell initiiert oder trägt. In der zweiten Dimension wird unterschieden, ob sich das Wirtschaftssubjekt aktiv oder passiv bei der Kommunikation mit seinen Handelspartnern verhält. Beispielsweise wartet ein Anbieter in einem e-Shop passiv auf einen Nachfrager. Ein e-Newsletter[11] wird vom Anbieter aktiv per Push-Verfahren dem Abonnenten zugestellt. e-Procurement und e-Tendering sind Beispiele für vom Nachfrager initiierte, aktive und passive Beschaffungen. Ein Mittler steht zwischen Anbieter und Nachfrager und kann sich beiden gegenüber aktiv oder passiv verhalten. Damit kann der Mittler in allen vier Bereichen positioniert sein. Die e-Mall ist ein Beispiel für ein üblicherweise passives Verhalten sowohl den Nachfragern als auch den Anbietern gegenüber. Diese Klassifikation verdeutlicht also die Positionierung eines Geschäftsmodells in Bezug zum Geschäftsakteur.

Nachdem Bartelt und Lamersdorf dieses Klassifikationsschema vorgestellt haben, um mit ihm unterschiedlichste Geschäftmodelle nach ihren Besonderheiten zu gruppieren, gehen sie der Frage nach, wie einzelne Geschäftsmodelle zu modellieren sind. Insgesamt umfasst ihr Ansatz zur Modellierung von Geschäftsmodellen vier Methoden. Laut Aussage der Autoren soll mit diesen Methoden auch die Grundlage zur Erstellung geeigneter Softwarelösungen (s.o.) gegeben sein.

Als erste Methode werden die von Timmers (vgl. Abschnitt 3.1.1) definierten Akteure und ihre Rollen sowie deren gegenseitige Beziehungen durch Diagramme grafisch dargestellt. Abbildung 3-5 zeigt als Beispiel ein Szenario für das Geschäftsmodell eines Online Shops, das von einem Anbieter getragen wird, der seine Produkte und Dienstleistungen über das Netz mehreren aktiven Nachfragern zur Verfügung stellt. Die Legende auf der rechten Seite erläutert die Knoten- und Kantentypen, aus denen diese Diagramme bestehen.

11. Die Autoren sehen ein eNewsletter als ein Geschäftsmodell an.

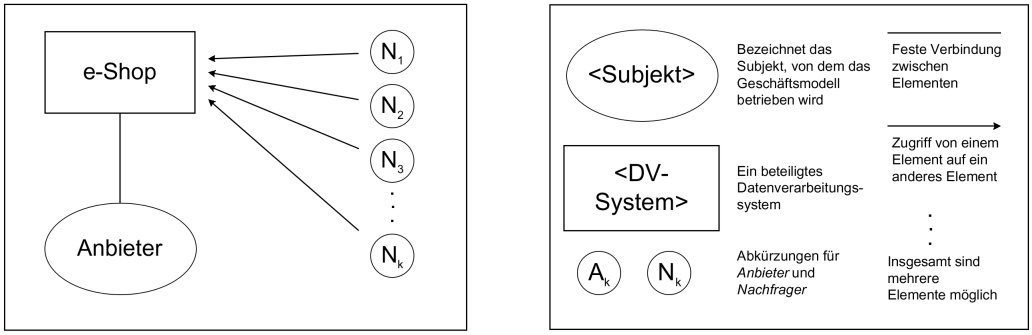


Abbildung 3-5: Modellierung der Akteure und Rollen von Geschäftsmodellen nach [BL00]

Als zweiten Bestandteil der Modellierung liefern Bartelt und Lamersdorf eine Darstellung für die von den Geschäftsmodellen unterstützten Phasen einer Handelstransaktion. Dieses Phasenmodell beruht auf der Einteilung von Guttman et al. [GMM98] in die sechs Phasen Bedarfsidentifikation, Produktvermittlung, Händlervermittlung, Verhandlung, Kauf & Lieferung und Produkt-Service & Evaluation. Anhand eines Blockdiagramms wird die Unterstützung hervorgehoben, die ein Geschäftsmodell für eine bestimmte Phase erbringt. Je breiter und dunkler der Balken, umso intensiver ist die Unterstützung dieser Phase durch das entsprechende Geschäftsmodell (siehe Abbildung 3-6). Das Beispiel zeigt die von einem e-Shop typischerweise unterstützten Phasen. Schwerpunkt des e-Shops ist Kauf und Lieferung, unterstützt durch Produktvermittlung und Verhandlung. Die Phasen Bedarfsidentifikation, Händlervermittlung und Produkt-Service und Evaluation sind nur geringfügig oder gar nicht unterstützt.

Nr.	Phase	Abk.
1.	Bedarfsidentifikation	BI
2.	Produktvermittlung	PV
3.	Händlervermittlung	HV
4.	Verhandlung	Vh
5.	Kauf und Lieferung	KL
6.	Produkt-Service und Evaluation	SE

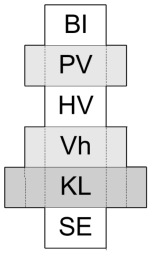


Abbildung 3-6: Phasen der Bedarfsdeckung und Grad ihrer Unterstützung nach [BL00]

Als weiteren Bestandteil der Modellierung eines Geschäftsmodells betrachten sie Erlösquellen. Sie übernehmen dabei den Ansatz von Skiera und Lambrecht [SL00], die die drei Varianten *Produkte*, *Kontakte* und *Informationen* identifizierten. Da die Arbeit von Skiera und Lambrecht detailliert im Abschnitt 3.2.2 vorgestellt wird, erfolgt hier neben dieser Auflistung keine nähere Betrachtung.

Der gemäß der Ankündigung, Softwarelösungen für die Geschäftsmodelle herzuleiten, interessanteste Aspekt sind die von Bartelt und Lamersdorf vorgestellten *Funktionsmodule*. Die Autoren definieren diesen Begriff nicht konkret, gehen jedoch auf die potenzielle

Wiederverwertbarkeit dieser Funktionsmodule in anderen Geschäftsmodellen ein und behaupten, dass durch Austausch und durch Kombination der Module neue Geschäftsmodelle geschaffen werden können. Als Beispiel nennen sie das Geschäftsmodell eines eCatalog, das die Funktionsmodule *Katalogfunktion*, *Produktpräsentation*, *Suchfunktion* und *Lieferbarkeitsauskunft* enthält. Ein e-Shop enthält darüber hinaus als weitere Module beispielsweise einen *Warenkorb*, die *Zahlungsabwicklung*, eine *Logistik-Verfolgun*g oder *Sicherheitsfunktionen*. Die Autoren sagen, dass die „softwaretechnische Realisierung dieser Funktionsmodule [...] gegebenenfalls durch Komponenten erfolgen“ kann [BL00].

An dieser Stelle gibt es eine sehr weitreichende Übereinstimmung mit dem Vorhaben der vorliegenden Arbeit. Bartelt und Lamersdorf nennen explizit einige Funktionsmodule, aus denen sich Geschäftsmodelle zusammensetzen und sagen darüber hinaus, dass diese durch Komponenten umzusetzen sind. Würden die Autoren ihre identifizierten Geschäftsmodelle konsequent in diese Funktionsmodule zerlegen oder ein Verfahren anbieten, diese Funktionsmodule zu identifizieren, könnten daraus die zugrundeliegenden Softwarearchitekturen gebildet werden. Leider liefern die Autoen diese Dekomposition der Geschäftsmodelle nicht. Dennoch deckt sich diese Modellierungsmethode der beiden Autoren mit der Idee der vorliegenden Arbeit.

Aufbauend auf ihrer Modellierungsmethode identifizieren und beschreiben die Autoren sieben relevante Geschäftsmodelle des e-Business. Dabei wenden sie jedoch leider nicht durchgängig alle vier Modellierungsmethoden an, so dass auch nicht durchgängig die einzusetzenden Erlösmodelle geliefert und begründet werden. Darüber hinaus erscheint teilweise fraglich, ob es sich bei den identifizierten Modellen um vollständige Geschäftsmodelle oder aber nur um Funktionsmodule handelt. Beispielsweise wird ein eNewsletter als Geschäftsmodell definiert, ohne jedoch auf die Erlösquelle dieses Modells einzugehen oder die enthaltenen Funktionsmodule explizit zu nennen. Auch eine eAuction kann sowohl als vollständiges Geschäftsmodell wie auch als Funktionsmodul betrachtet werden, wenn man die „eAuction auf die Preisbildung“ reduziert.

Eine Betrachtung der Arbeit unter den drei Fragestellungen zeigt, dass die Autoren die Erlösmodelle als wesentlichen Bestandteil von Geschäftsmodellen betrachten, was durch ihre Anlehnung an die Arbeit von Timmers erfolgt. Sie übernehmen dabei vollständig den Ansatz von Skiera und Lambrecht, die die drei Grundformen Produkte, Kontakte und Informationen als Erlösmodelle (bzw. Erlösquellen) definiert haben.

Die Klassifikation der Geschäftsmodelle kann anhand der Modellierungsmethoden erfolgen, die die Autoren zusammentragen. Vor allem der Einsatz der Funktionsmodule verspricht zunächst, softwaretechnische Schlussfolgerungen durch die Klassifikation zu ermöglichen. Dies gelingt jedoch nur in Ansätzen, da nur einige wenige Funktionsmodule explizit identifiziert werden und die Abgrenzung zwischen Geschäftsmodell und enthaltenen Funktionsmodulen nicht immer eindeutig nachvollziehbar ist. Es werden auch keine Funktionsmodule in Abhängigkeit der Erlösquellen aufgezeigt. Diese könnten anderenfalls eventuell als Bestandteil von Softwarearchitekturen verwendet werden.

3.1.7 Das Geschäftsmodell nach Fugmann, Heinrich, Leist, Winter

Das Institut für Wirtschaftsinformatik (IWI) der Universität St. Gallen befasst sich mit Geschäftsmodellen des Informationszeitalters und Architekturkonzepten für Banken (vgl. z.B. [Öst99], [Öst00]). Es geht dabei um die Erarbeitung eines Vorgehensmodells für die konsistente Gestaltung von Geschäftsmodellen, Prozessen und Applikationen auf Grundlage von Bankenarchitekturen, insbesondere im Hinblick auf Änderungen in der Unternehmensstruktur und in der Zusammenarbeit von Banken im Informationszeitalter. Auch wenn die Grundlage dabei Geschäftsmodelle der Banken bilden, werden die Ansätze hier auf ihre Übertragbarkeit hin untersucht.

Die Arbeiten von Fugmann et al. [FHL99], Leist und Winter [LW00] sowie Heinrich und Leist [HL02] betonen die Bedeutung des Ansatzes, Geschäftsmodell, Prozessmodell und Softwarearchitektur aufeinander aufbauend und eng miteinander verzahnt zu entwickeln.

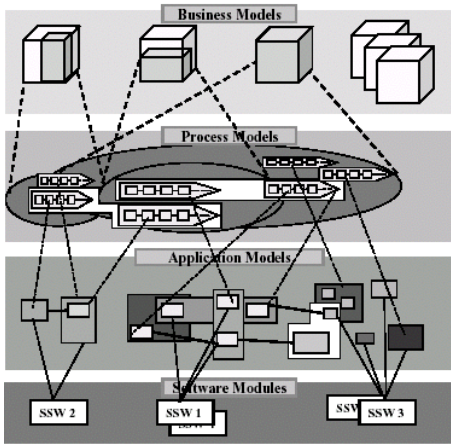


Abbildung 3-7: Zusammenhang von Geschäftsmodell und Softwarearchitektur nach [FHL99]

Wie Abbildung 3-7 zeigt, folgt der Ansatz sehr stark der Idee der vorliegenden Arbeit, ausgehend vom Geschäftsmodell passende Softwarearchitekturen herzuleiten. Entsprechend der Grafik können *Geschäftsmodelle* zunächst in *Prozessmodelle* zerlegt werden, die mittels *Applikationsmodelle* realisiert werden können. Die einzelnen Applikationsmodelle setzen sich dabei aus *Softwaremodulen* zusammen, die auch als Komponenten interpretiert werden können. Betrachtet man ein Erlösmodell als einen Bestandteil eines Geschäftsmodells, können die während der Umsetzung anfallenden Aktivitäten in einem Prozessmodell zusammengefasst werden. Dieses lässt sich durch Applikationsmodelle softwaretechnisch realisieren. Diese Applikationsmodelle setzten sich wiederum aus einzelnen Softwarekomponenten (Softwaremodulen) zusammen. Es gibt also eine große Überschneidung in dem Vorgehen der Autoren und dem Vorgehen in der hier vorliegenden Arbeit.

Für das zu entwerfende Vorgehensmodell benötigen die Autoren zunächst einige Charakterisierung der Geschäftsmodelle. Sie liefern ein Beschreibungsschema, dass eine Außen- und eine Innensicht eines Unternehmens unterscheidet. Beide Sichten enthalten eine Reihe von Kriterien, die zur Charakterisierung von Unternehmen und damit auch zur Klas-

sifikation verwendet werden können. Aufgrund der Focussierung auf den Bankensektor fehlt allerdings die Allgemeingültigkeit der Elemente, weshalb sie hier nicht verwendet werden können.

Während die Autoren sich das Ziel setzen, ein Vorgehensmodell zu liefern, das die durchgängige Gestaltung von einem Geschäftsmodell hin zu einer Softwarearchitektur ermöglichen soll, so liefern sie diesbezüglich jedoch keine verwertbaren Ergebnisse. Die vorhandenen Arbeiten enden bei der Beschreibung von Geschäftsmodellen. Insofern können trotz des äußerst vielversprechenden Ansatzes keine konkreten Überlegungen übernommen werden.

Eine Betrachtung der Arbeiten unter den relevanten Fragestellungen wurde somit bereits implizit geliefert. Die Autoren liefern ein Schema zur Klassifikation von Geschäftsmodellen, das jedoch sehr stark auf die Finanzwirtschaft zugeschnitten ist und somit nur Anregungen liefern kann. Bezogen auf eine Modellierungsmethode hinsichtlich einer Realisierung von Softwarearchitekturen liefern sie leider keinerlei Hinweise, obwohl sie dieses Ziel verfolgen. Insgesamt sind die Arbeiten somit bis auf Anregungen nicht verwertbar.

3.1.8 Das Geschäftsmodell nach Amit und Zott

Auf diese Arbeit von Amit und Zott wird abschließend eingegangen, da die Autoren explizit daruaf hinweisen, dass ein Geschäftsmodell nicht mit einem Erlösmodell zu verwechseln sei. Es wird deshalb untersucht, inwiefern diese Trennung zusätzliche Erkenntnisse für diese Arbeit liefert. Darüber hinaus wurde die Arbeit von anderen Autoren aufgegriffen, so dass sie hier kurz analysiert wird.

Amit und Zott [AZ00] gehen mit einem sehr strategisch ausgerichteten Ansatz an eine Untersuchung von Geschäftsmodellen heran. Sie entwickeln dabei ein sogenanntes *Value Driver Model*, das sowohl ein Geschäfts- als auch ein Erlösmodell umfasst. Sie weisen in ihrem Ansatz explizit darauf hin, dass beide Modelle nicht miteinander zu verwechseln sind, sondern eigenständige Modelle sind. Sie definieren ein Geschäftsmodell als eine „architekturelle Konfiguration von Transaktionskomponenten zur Ausschöpfung von Geschäftsmöglichkeiten“[12] und bleiben damit sehr allgemeingültig.

Die Transaktionskomponenten sind dabei die speziellen Informationen, Dienste oder Produkte, die ausgetauscht werden bzw. die daran beteiligten Parteien. Die architekturelle Konfiguration beschreibt die Verbindung und Anordnung der Transaktionskomponenten.

12. Ein Geschäftsmodell ist „the architectural configuration of the components of transactions designed to exploit business opportunities“ [AZ00].

In Abgrenzung zum Geschäftsmodell adressiert ein Erlösmodell dagegen die spezielle Art und Weise, wie ein Geschäftsmodell die Erzeugung von Erlösen ermöglicht. Möglichkeiten dieser Werterzeugung und damit potenzielle Erlösmodelle sehen die Autoren in einer Kombination von...

- SUBSKRIPTIONSGEBÜHREN,
- WERBUNG
- sowie transaktionsabhängigem Einkommen wie fixe TRANSAKTIONSGEBÜHREN,
- VERMITTLUNGSGEBÜHREN
- und fixe oder variable PROVISIONEN.

Zusammenfassend lässt sich aus Sicht der Autoren festhalten, dass sich ein Geschäftsmodell mit der *Werterzeugung* (value creation) befasst und ein Erlösmodell mit der *Wertzuteilung* (value appropriation).

Trotz der Trennung beider Modelle liefern die Autoren aber, wie die Liste der von ihnen identifizierten Erlösmodelle zeigt, ähnliche oder identische potenzielle Erlösmodelle wie andere Autoren, die die Erlösmodelle als Bestandteil von Geschäftsmodellen angesehen haben. Insofern liefert ihr Ansatz keine verwertbaren neuen Erkenntnisse für die vorliegende Arbeit. Es werden auch keine softwaretechnischen Gesichtspunkte von Geschäftsmodellen von den Autoren betrachtet, die hier aufgegriffen werden könnten.

3.1.9 Fazit

Es wurde gezeigt, dass von den zahlreichen Autoren, die Geschäftsmodelle des e-Business definiert und untersucht haben, ein Großteil ein Erlösmodell als integralen Bestandteil eines Geschäftsmodells ansieht. Die gelieferten Definitionen unterscheiden sich nicht grundsätzlich, sondern stimmen weitgehend überein, wobei sie teilweise einige Aspekte unterschiedlich stark hervorheben. Im Rahmen dieser Arbeit wird die Definition von Timmers als die geeignetste angesehen und dem eigenen Verständnis eines Geschäftsmodells zugrundegelegt, was auch mit der Tatsache zusammenhängt, dass sie eine der ältesten Definitionen ist und von anderen Autoren häufig referenziert oder als Basis genommen wird.

Als ein weiteres Fazit in Bezug auf die Klassifikation von Erlösmodellen wird festgehalten, dass nicht jeder Autor eigene, neue Typen definiert, sondern dass weitgehend ähnliche oder gar identische Erlösmodelle identifiziert werden. Einige Autoren greifen bei ihrer Auflistung möglicher Erlösmodelltypen direkt auf Arbeiten anderer Autoren zurück, die Erlösmodelle des e-Business eigenständig und nicht im Zusammenhang mit Geschäftsmodellen betrachtet haben. Diese Arbeiten werden im nächsten Abschnitt untersucht, so dass hier keine abschließende Betrachtung der bisher zusammengetragenen Erlösmodelle erfolgt.

Ein weiteres Fazit ist die Tatsache, dass abgesehen von Dubosson-Torbay, Osterwalder und Pigneuer die übrigen Autoren nur ein- oder zweidimensionale Klassifikationsschema von Geschäftsmodellen vorgenommen haben. Dubosson-Torbay, Osterwalder und Pig-

neuer dagegen haben insgesamt zwölf Klassifikationskriterien geliefert, auf die im nächsten Kapitel zurückgegriffen wird, wenn ein eigenes Klassifikationsschema für Erlösmodelle hergeleitet wird.

Abschließend kann festgehalten werden, dass die existierenden Arbeiten kaum Aussagen zu softwaretechnischen Realisierungen der Geschäfts- oder ihrer Erlösmodelle liefern. Wenn dieser Anspruch in einigen Arbeiten vorliegt, so sind die Ergebnisse für die Absicht dieser Arbeit nicht ausreichend, da keine Aussagen über die Herleitung oder über die Art und Weise von Softwarearchitekturen für diese Modelle geliefert wurden.

3.2 Verwandte Arbeiten über Erlösmodelle im e-Business

In diesem Abschnitt werden Arbeiten vorgestellt, die sich ausschließlich mit Erlösmodellen im e-Business befassen und diese nicht nur als Bestandteil von Geschäftsmodellen behandeln. Sie grenzen sich somit von den bisher analysierten Arbeiten ab, in denen die Erlösmodelle stets als ein Bestandteil von Geschäftsmodellen angesehen wurden und in denen der Fokus auf der Betrachtung dieser Geschäftsmodelle lag.

3.2.1 Erlösmodelle nach Zerdick et al.

Zerdick et al. [ZPS99] konzentrieren sich in ihrer Arbeit auf die Medienindustrie. Sie unterteilen die Entscheidung, die ein Unternehmen hinsichtlich der Frage über die Art und Weise sowie über die Höhe der Erlöse treffen muss, in zwei Teilbereiche:

- Grundsatzentscheidung über Erlöstypen und Erlösmodelle
- Entscheidung über die Preispolitik

Die Preispolitik (Entgeltpolitik) trifft dabei die grundsätzliche Entscheidung über die Festsetzung der Preise für das Angebot des Unternehmens. Diese ist hier jedoch weniger interessant. Der Festlegung auf die Preispolitik voraus geht die Entscheidung über das zu wählende Erlösmodell, das hier näher analysiert wird.

Die Autoren legen bei ihrer Systematisierung der Erlösmodelle auf oberster Ebene die Herkunft der Erlöse zugrunde und nicht die Gegenleistung, die für die Erlöse gebracht wird. Bezieht man diesen Ansatz auf die Bestandteile eines Geschäftsmodells nach der Interpretation von Timmers, so kann man sagen, dass die Autoren die Erlöse in Abhängigkeit der Akteure und nicht in Abhängigkeit der Produkte oder Dienstleistungen auflisten. Insofern unterteilen sie die Erlösmodelle zunächst in *direkte* und *indirekte* Erlöse.

Die direkten Erlösformen werden unmittelbar vom Nutzer der Leistung bezogen. Die indirekten Erlöse dagegen werden nicht direkt von den Nutzern gezahlt, sondern von Dritten, die ein Interesse daran haben, dass der Konsument die Leistungen des Anbieters nutzt. Die folgende Grafik zeigt diese grundsätzliche Unterteilung der Erlösformen:

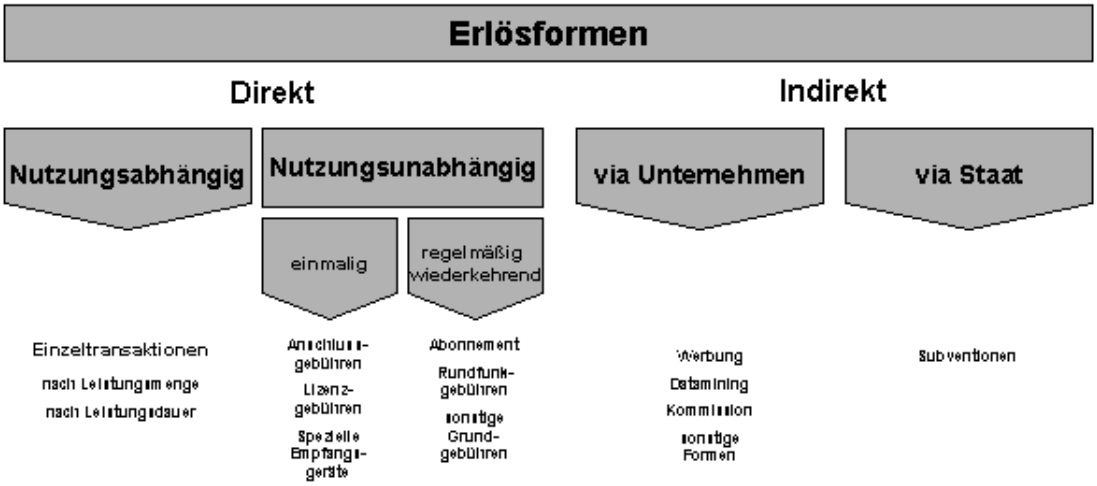


Abbildung 3-8: Erlösformen nach Zerdick et al. [ZPS99]

Es ist zu erkennen, dass sowohl die direkten als auch die indirekten Erlösformen weitere Untergliederungen aufweisen. Die direkten Formen lassen sich demnach voneinander in *nutzungsabhängige* sowie *nutzungsunabhängige* Arten abgrenzen. Nutzunsabhängig kann auch als transaktionsabhängig bezeichnet werden und bedeutet, dass der Nutzer in Abhängigkeit der Menge und/oder Dauer der Nutzung eine finanzielle Gegenleistung erbringt. Jede einzelne Transaktion wird also direkt vom Nutzer bezahlt (EINZELTRANSAKTION). Die nutzungs- oder transaktionsunabhängigen Erlöse werden dagegen entweder einmalig oder periodisch wiederkehrend entrichtet. Einmalige Gebühren sind ANSCHLUSS- und LIZENZGEBÜHREN oder Kosten für SPEZIELLE EMPFANGSGERÄTE. Verallgemeinernd können Anschlussgebühren auch als Zugangs- oder Vertragsabschlussgebühren interpretiert werden, wodurch sie nicht nur auf den Mediensektor beschränkt bleiben. Die wiederkehrenden Erlösformen können als SUBSKRIPTIONS-, RUNDFUNK oder SONSTIGE GRUNDGEBÜHREN vorkommen. Die letzten Beispiele zeigen sehr deutlich die Fokussierung der Autoren auf den Mediensektor. Sie werden im weiteren Verlauf dieser Arbeit nicht mehr betrachtet.

Die indirekten Erlösformen sind gemäß der Herkunft zu unterscheiden in Erlöse, die andere, dritte Unternehmen entrichten sowie Erlöse, die vom Staat gezahlt werden. Dritte Unternehmen können Gelder für WERBUNG, DATAMINING (Profilbildung und -handel) sowie für die Vermittlung von Transaktionen in Form von KOMMISSION entrichten. Der Staat kann als Zahler durch SUBVENTIONEN aktiv werden.

Die Erlösformen, die in der Abbildung 3-8 dargestellt werden, können im Rahmen dieser Arbeit als Erlösmodelle betrachtet werden, wobei hier aufgrund der allgemeineren Betrachtung über die Medienindustrie hinaus die Gebühren für spezielle Empfangsgeräte, die Rundfunkgebühren sowie die staatlichen Subventionen nicht weiter berücksichtigt werden. Die Autoren liefern aufgrund ihres Ansatzes keine Aussagen zu den Wechselwirkungen der Erlösmodelle mit Geschäftsmodellen und liefern auch keine Hinweise auf softwaretechnische Realisierungen, jedoch erscheint die Klassifikation für das Ziel der Arbeit ein sehr

guter Ausgangspunkt zu sein. Inwiefern der Ansatz weiter verfolgt wird, zeigt das nächste Kapitel. Zunächst wird jedoch abschließend eine weitere Arbeit über Erlösmodelle des e-Business analysiert.

3.2.2 Erlösmodelle nach Skiera und Lambrecht

Skiera und Lambrecht [SL00] untersuchen Erlösmodelle im Internet und greifen auf die Arbeit von Zerdick et al. zurück, wobei sich jedoch absichtlich weniger auf die Herkunft der Erlöse und stattdessen auf die Gegenleistung für die Erlöse konzentrieren. Nimmt man auch hier Timmers Bestandteile eines Geschäftsmodells als Grundlage, richtet sich ihr Ansatz somit eben nicht nach den Akteuren, sondern nach den Produkten und Dienstleistungen, aber auch nach dem Nutzen für die Akteure.

Die folgende Abbildung zeigt die drei Erlösquellen *Produkte*, *Kontakte* und *Informationen*, die nach Meinung der Autoren im Internet zur Verfügung stehen:

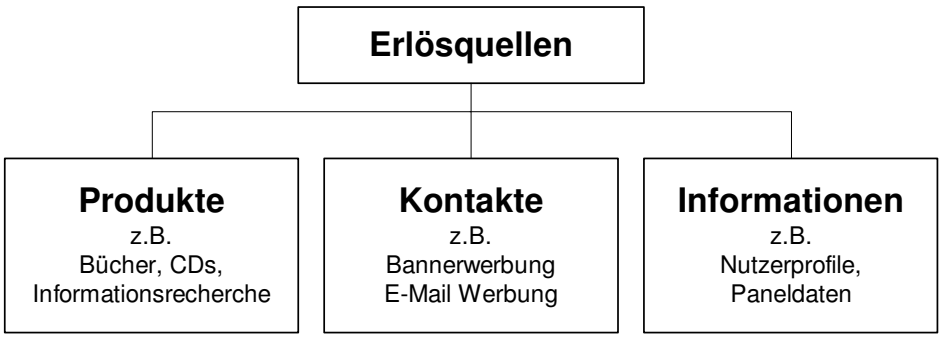


Abbildung 3-9: Erlösquellen nach Skiera und Lambrecht [SL00]

PRODUKTE dienen dann als Erlösquelle, wenn durch ihren Verkauf direkt die Erlöse generiert werden. Skiera und Lambrecht verwenden den Begriff des Produkts zusammenfassend für Güter und Dienstleistungen. Beispiele sind e-Shops, die Güter wie Bücher oder DVDs oder aber Dienstleistungen wie Recherchen verkaufen.

KONTAKTE können als Erlösquellen in Form von Werbung und Vermittlungsprovisionen auftreten. Das Unternehmen nutzt also über seinen Webauftritt den Kontakt mit dem Kunden dazu, Erlöse zu erzielen. Unterschieden wird offensichtlich, ob der Kunde die Kontakte nur sieht (Werbebanner) oder ob er auch aktiv einen Kontakt in Form einer Vermittlung herstellt (Clicks auf Werbebanner oder eingebundene Links, die zu Websites anderer Unternehmen führen).

INFORMATIONEN können dann als Erlösquelle eingesetzt werden, wenn über den Nutzer Informationen gesammelt und veräußert werden können. Beispiele sind Nutzerprofile oder Paneldaten. So können Informationen darüber ermittelt werden, welche Produkte von Kunden betrachtet, aber nicht gekauft werden, oder welche Produkte ein Käufer eines bestimmten Produktes zusätzlich betrachtet.

Die Autoren weisen besonders darauf hin, dass in der Regel die Erlösquellen nicht isoliert, sondern in Kombination auftreten. Sie untersuchen im weiteren Verlauf, welche Erlösquelle sich auf welcher Stufe der Wertschöpfungskette[13] anbietet. Diese Betrachtung ist hier jedoch nicht von Relevanz und wird deshalb nicht näher vorgestellt.

Der Ansatz der beiden Autoren richtet sich anders als bei Zerdick et al. nicht nach der Herkunft der Erlöse, sondern nach der Gegenleistung für die Erlöse. Er stellt somit eine gute Ergänzung zur ersten Arbeit dar und wird im nächsten Kapitel mit in die Herleitung der Erlösmodelltypen einfließen, auch wenn verständlicherweise keine Aspekte der Wechselwirkung mit Geschäftsmodellen diskutiert wurden. Ebensowenig gehen die Autoren auf Softwarearchitekturen oder sonstige informationstechnologische Anhaltspunkte zur Umsetzung der Erlösmodelle ein.

3.2.3 Fazit

Die beiden Arbeiten, die in diesem Abschnitt untersucht wurden, analysieren Erlösmodelle des e-Business losgelöst von Geschäftsmodellen. Insofern konnten die eingangs des Kapitels aufgelisteten Fragestellungen nicht vollständig angewandt werden. Stattdessen war lediglich von Interesse, welche Klassifikationen von Erlösmodellen die Arbeiten liefern.

Vor allem die Arbeit von Zerdick et al. wird hier als sehr geeignet angesehen. Die dort gelieferte Unterscheidung in direkte und indirekte Erlösmodelle kann im Rahmen dieser Arbeit gut angewandt werden, um die unterschiedlichen Geldgeber als die Herkunftsquellen der Erlöse zu unterscheiden. Diese Unterscheidung ist aus softwaretechnischer Sicht sehr wichtig, da die zugrundeliegenden Softwaresysteme von Geschäftsmodellen entsprechend dieser Erlösmodelle auf den direkten Bezug (also vom Endkunden) oder aber auf den indirekten Bezug (also von Geschäftspartnern) ausgerichtet sein müssen. Somit müssen die Softwarearchitekturen diesen Sachverhalt ebenfalls berücksichtigen, wie im weiteren Verlauf der Arbeit noch detailliert aufgezeigt und verdeutlicht wird. Darüber hinaus ist aber auch der Ansatz von Skiera und Lambrecht von Bedeutung für diese Arbeit, da die Unterteilung in Produkte, Kontakte und Informationen eine geeignete Strukturierung der Erlös-Gegenleistungen darstellt. Sie gibt darüber hinaus Auskunft über die Beschaffenheit der Leistungen, die ebenfalls für die Abwicklung der Erlösmodelle und damit auch für die Softwaresysteme von Bedeutung sind. Dies wird im weiteren Verlauf ebenfalls ausführlicher dargestellt.

3.3 Schlussfolgerung

In dem Kapitel wurde umfassend auf existierende themenverwandte Arbeiten eingegangen, um einerseits einen Überblick über die vorhandenen Forschungsarbeiten zu vermitteln und andererseits zu untersuchen, indwiefern bestehende Ergebnisse für das Ziel dieser Arbeit herangezogen werden können. Es hat sich dabei gezeigt, dass im Wesentlichen eine Übereinstimmung darin besteht, dass Erlösmodelle ein Bestandteil von

13. Wertschöpfungskette nach Porter [Por99]

Geschäftsmodellen im e-Business sind. Auch die Definitionen der Geschäftsmodelle sowie der Erlösmodelle decken sich sinngemäß weitgehend, wenn auch verschiedene Autoren unterschiedliche Schwerpunkte definiert und auch unterschiedliche Ausprägungen von Geschäftsmodellen identifiziert haben. Letzteres liegt vor allem daran, dass voneinander abweichende Klassifikationskriterien herangezogen wurden und somit die Granularität der ermittelten Modelltypen stark variiert.

Die Variation der Erlösmodelle hält sich dagegen in Grenzen. Es werden sehr häufig übereinstimmende Formen aufgezählt, wie beispielsweise die Transaktions- oder Subskriptionsgebühren. Einige Autoren fügen zusätzliche Erlösmodelle hinzu, die teilweise nur in sehr speziellen Marktsegmenten Verwendung finden können. Rundfunkgebühren sind hierfür ein Beispiel. Grundsätzlich lässt sich jedoch anhand der gelieferten Erlösmodelltypen eine sehr umfassende Liste möglicher Ausprägungen aufstellen. Die folgende Tabelle 3-2 liefert eine Übersicht über die in diesem Kapitel zusammengetragenen Erlösmodelle.

Timmers	Mahadevan	Buchholz	Durbosson, Osterw., Pigneur	Amit / Zott	Zerdick et al. (Wirtz)	Skiera / Lambrecht (Bartelt / Lamersdorf)
Verkauf von Produkten und Dienstleistungen		Informationsverkauf	Verkauf von Produkten		Einzeltransaktion	Produkte
Mitgliedsbeiträge	Mitgliedsgebühren	Subskriptionsgebühr	Subskriptionsgebühr	Subskriptionsgebühr	Abonnement, sonstige Grundgebühren	Dienstleistung (Teil von Produkten)
Transaktionsgebühren	Transaktionsgebühr von Anbietern und Käufern	Transaktionsgebühr	Transaktionsanteile	Transaktionsgebühr	Lizenz-, Anschluss- und Rundfunkgebühren	Kontakte
Werbung	Werbung	Gain sharing	Sponsoring		Datamining	Informationen
		Lizensierung der Technologie	Provision	Vermittlungsgebühren, Provision	Kommission	
		Werbung	Werbung	Werbung	Werbung	
		Gebühren für Consulting	Einnahmeaufteilung (revenue sharing)		Subventionen	
					spezielle Empfangsgeräte	
					sonstige Formen	

Tabelle 3-2: Übersicht der zusammengetragenen Erlösmodelle

Ausgehend von diesen Erlösmodellen wird im folgenden Kapitel eine eigene Klassifikation von Erlösmodellen hergeleitet. Bei der Identifikation und Charakterisierung der Erlösmodelle soll vor allem ein Aspekt berücksichtigt werden, der in den bestehenden Arbeiten nahezu gänzlich außerachtgelassen wurde: Eine Klassifikation der Erlösmodelle zum Zweck der Identifikation von Softwarearchitekturen, die diese Modelle im Kontext ihrer Geschäftsmodelle realisieren. Dazu werden zunächst geeignete Klassifikationskriterien definiert. Diese bauen auf der Liste von Dubosson-Torbay, Osterwalder und Pigneuer auf, wie bereits im Fazit in Abschnitt 3.1.9 betont wurde.

4 Erlösmodelltypologie als Grundlage der Softwarearchitekturen

Aufbauend auf den zusammengetragenen Erlösmodellen als Ergebnis der Literaturanalyse des vorherigen Kapitels werden hier diejenigen Erlösmodelle identifiziert und beschrieben, für die im weiteren Verlauf die Software-Referenzarchitekturen hergeleitet werden. Zu den einzelnen Erlösmodelltypen werden dabei jeweilige Subtypen identifiziert, sofern eine feinere Unterscheidung notwendig ist. Ausschlaggebend für eine solche Unterscheidung ist die Tatsache, dass die Subtypen Charakteristika aufweisen, die bei der Herleitung entsprechender Softwarearchitekturen berücksichtigt werden müssen. Entsprechend diesem Ziel weisen die Erlösmodelltypen und -subtypen Eigenschaften auf, die softwaretechnische Relevanz besitzen. Deshalb erfolgt die Identifikation der Erlösmodelle anhand von Kriterien, die ebensolche technischen Aspekte berücksichtigen. Würde ein Erlösmodell beispielsweise in Anlehnung an Buchholz[1] anhand des Kriteriums *Wertschöpfungsbeitrag* mit den möglichen Ausprägungen *niedrig* bis *hoch* charakterisiert, so hätte diese Aussage vielleicht betriebswirtschaftliche, nicht jedoch softwaretechnische Relevanz. Solche Klassifikationskriterien werden hier nicht berücksichtigt. Wie bereits im Fazit des vorhergehenden Kapitels erwähnt, wird hier als Ausgangspunkt der Kriteriendefinition auf die Liste der Klassifikationskriterien für Geschäftsmodelle von Dubosson-Torbay, Osterwalter und Pigneur[2] zurückgegriffen, die diese Autoren als Ergebnis ihrer eigenen Literaturrecherche wiedergegeben haben.

Da ein Erlösmodell immer im Zusammenhang mit einem es umgebenden Geschäftsmodell reale Anwendung findet, sollen die Kriterien darüber hinaus die Wechselwirkung mit diesen Geschäftsmodellen aufzeigen. An dieser Stelle wird auf die Definition Timmers' zurückgegriffen, der die folgenden vier Bestandteile eines Geschäftsmodells analysiert hat ([Tim98], vgl. Abschnitt 3.1.1):

- Eine Beschreibung der beteiligten Akteure und ihrer Rollen
- Eine Architektur für das Produkt, den Service und die Informationsflüsse
- Eine Beschreibung des potenziellen Nutzens der Akteure
- Eine Beschreibung der Einkommensquellen für den oder die Betreiber des Geschäfts.

Entsprechend dieser vier Bestandteile werden in Abschnitt 4.1 zunächst allgemeine Klassifikationskriterien der Erlösmodelle hergeleitet. Timmers' erster Punkt bezüglich der Akteure und ihren Rollen wird als *Akteurbezogene Kriterien* zusammengefasst. Die Produkte, Services und Informationsflüsse werden als *Angebotsbezogene Kriterien* berücksichtigt. *Nutzenaspekte* werden ebenfalls diskutiert, jedoch sind hier wenige allgemeingültige Kriterien zu identifizieren, wie im entsprechenden Abschnitt erläutert wird. Der vierte Bestandteil in Form der Einkommensquellen stellt hier eine Besonderheit dar, da dieses Element mit den hier zu identifizierenden Erlösmodellen gleichgesetzt wird. Somit beziehen sich die in dieser Kategorie identifizierten Kriterien direkt auf das Erlösmodell und nicht auf die übrigen

1. [Buc01], vgl. Abschnitt 3.1.3 auf Seite 31, Das Geschäftsmodell nach Buchholz
2. [DOP01], vgl. Abschnitt 3.1.4 auf Seite 33, Das Geschäftsmodell nach Dubosson-Torbay, Osterwalder, Pigneur

Bestandteile von Geschäftsmodellen. Die Kategorie wird *Erlösmodellbezogene Kriterien* benannt. In Abschnitt 4.2 werden dann die Erlösmodelltypen und -subtypen identifiziert und beschrieben.

4.1 Kriterien zur Klassifikation von Erlösmodellen

4.1.1 Akteurbezogene Kriterien

In diesem Abschnitt werden Kriterien aufgeführt, die die Akteure eines Unternehmens adressieren. Aus Sicht des Erlösmodells sind vor allem die Kunden[3] relevant, so dass sich die Kriterien auf diese Gruppe beschränken.

4.1.1.1 Kundenrolle

Die Kundenrolle beschreibt die möglichen Funktionen, die ein Kunde innerhalb der Geschäftsbeziehung zu dem Unternehmen einnimmt.

Unterschieden wird zwischen folgenden Kundenrollen:

- Informant

  Informanten sind dadurch gekennzeichnet, dass sie sich anhand des Angebotes des Unternehmens informieren, ohne jedoch an einer direkten Handelsbeziehung teil zu nehmen, d.h. der Informant zahlt kein Geld für die Nutzung des Angebotes. In der Regel konsumiert der Kunde Content, der vom Unternehmen bereitgestellt wird. Um die Attraktivität zu erhalten, handelt es sich dabei entweder um aktuellen Content wie Nachrichten oder aber um sehr fachbezogene Informationen wie beispielsweise Informationen rund um das Thema Versicherungen.

  Kunden von Portalen oder Online-Magazinen treten zumeist als Informanten auf. Da die Unternehmen von ihren Kunden kein direktes Geld für ihr Angebot erhalten, steht zumeist die Vermarktung von Werbeflächen im Vordergrund, so dass eine große Anzahl von Kunden sowie gute Kenntnisse über die Kundschaft für das Unternehmen von Interesse sind. Es ist also zu unterscheiden, ob der Informant anonym oder identifiziert ist, was jedoch übergreifend in einem weiteren Kriterium berücksichtigt wird.

3. Es wird in dieser Arbeit nicht unterschieden zwischen Kunden und externen Nutzern. Man könnte argumentieren, dass ein externer Nutzer erst durch den Kauf eines Angebotes zum Kunden des Unternehmens wird, aber diese Unterscheidung ist hier nicht notwendig und wird somit vernachlässigt.

- Käufer

  Ein Käufer zahlt direkt einen Beitrag für die Nutzung des Angebotes. Bietet das Unternehmen Güter an, zahlt der Kunde in der Regel einen einmaligen Preis. Die Kundschaft von e-Shops sind typische Vertreter dieser Rolle. Aber auch Subskriptionsgebühren für die Nutzung eines Services, wie z.B. die Bereitstellung eines Internetzugangs, kann von dieser Kundenrolle gezahlt werden.

- Verkäufer

  Sofern der Kunde innerhalb der Handelsbeziehung nicht (nur) als Konsument, sondern (auch) als Anbieter auftritt, nimmt er die Rolle des Verkäufers ein. Marktplätze und Tauschbörsen sind Beispiele, in denen die Kunden des Unternehmens (zumeist die Betreiber des Marktes oder der Börse) nicht direkt Waren von ihm kaufen, sondern ihre eigenen Güter und Dienstleistungen anbieten oder von anderen Verkäufern beziehen. Als Gegenleistung für die Nutzung der Plattform entrichtet der Verkäufer in der Regel eine transaktionsabhängige Gebühr (z.B. ein prozentualer Anteil des Umsatzes).

- Value-Integrator

  Ein Value-Integrator ist direkt am Produktionsprozess des Angebotes beteiligt. Er trägt mit einem persönlichen Beitrag dazu bei, dass ein Wert entsteht, den das Unternehmen anderen zur Verfügung stellen kann. In Communities treten die Kunden beispielsweise als Value-Integrator auf, indem sie in Foren und Diskussionsrunden eigene Frage stellen oder Antworten liefern, die dann für eine breite Nutzerschicht von Interesse sind und somit den Informationsgehalt des Angebotes erhöhen.

  Die Kundenrolle des Value-Integrator tritt somit in Geschäftsmodellen nicht als alleinige Beziehungsform auf, da es kaum vorstellbar ist, dass alle Kunden einen Beitrag leisten, ohne dass andere Kunden diesen Wert in der Rolle des Informanten oder Käufers konsumieren. Diese Rolle existiert nur zusätzlich optional.

4.1.1.2 Kundenbindung

Die Kundenbindung gibt Auskunft über die Identifikation des Kunden und kann in folgenden Ausprägungen auftreten:

- Anonym

  Eine anonyme Kundenbindung liegt dann vor, wenn sich der Kunde zur Nutzung des Angebotes nicht am System des Unternehmens anmeldet. Content ist in der Regel anonym zugänglich, wobei einige Anbieter von Content zusätzliche Services oder ergänzende Informationen bereithalten, deren Nutzung eine Identifikation voraussetzt. Um über eine anonyme Kundschaft dennoch Informationen zu erlangen, die z.B. für die zielgruppengerechte Schaltung von Werbung notwendig ist, können Cookies eingesetzt werden, die auf Seiten der Kunden gespeichert werden und Informationen bereit halten, die bei einem erneuten Kontakt ausgelesen und ausgewertet werden können. Der unseriöse Charakter dieser Technik führt jedoch dazu, dass ein Einsatz nicht grundsätzlich möglich und von weiteren Faktoren abhängig ist.

- Identifiziert

  Bei der identifizierten Kundenbindung meldet sich der Kunde mit Hilfe von Zugangsdaten am System des Unternehmens an. Die verbreitetste Form im e-Business ist die Kombination aus Login und Passwort, die einen Kunden eindeutig zu erkennen gibt. Auf diese Weise können Profile über den Kunden verwaltet werden, in denen Informationen über ihn gespeichert werden und somit die Voraussetzung für ein Rechte- und Rollenmanagement bilden oder eine umfangreiche Personalisierung ermöglichen.

  Hohe Anforderungen an die Sicherheit eines Geschäftsmodells können des Weiteren dazu führen, alternative Methoden zur Identifizierung der Kunden einzusetzen. Die Erkennung von biometrischen Merkmalen, wie z.B. ein Fingerabdruck, das Gesicht oder die Iris des Auges, werden in der Praxis bereits zur Absicherung von Räumen oder Gebäuden eingesetzt; allerdings erfordert ihre Verwendung zusätzliche Hardware, wodurch diese Verfahren im e-Business derzeit keine Anwendung finden und im weiteren Verlauf dieser Arbeit nicht weiter betrachtet werden.

Je nach Geschäftsmodell kann unterschieden werden, ob eine Identifizierung mit oder ohne Validierung erforderlich ist oder ob eine anonyme Kundenbindung vorgezogen wird. Oftmals ist besonders der anonyme Charakter des Internet ein wesentlicher Mehrwert für den Kunden, so dass eine eventuell zwingend notwendige Identifizierung auf der Erhebung geringster persönlicher Daten beruht.

Für die Architekturgestaltung ist es von großer Bedeutung, welche Form der Kundenbindung realisiert wird, da entsprechende Profilinformationen verwaltet werden müssen, falls der Kunde identifiziert auftritt.

4.1.2 Angebotsbezogene Kriterien

Diese Kriterien richten sich nach den Produkten und Dienstleistungen, die das Unternehmen anbietet.

4.1.2.1 Beschaffenheit des Angebotes

Die Beschaffenheit des Angebotes beeinflusst zusammen mit anderen Kriterien die Auswahlmöglickeiten von Erlösmodellen, wie im weiteren Verlauf zu sehen sein wird. Unterschieden werden folgende Ausprägungen:

- Physikalisches Gut

  Physikalische Güter können auch als traditionelle Güter bezeichnet werden, die in greifbarer Form vorkommen. Die Lagerung sowie der Vertrieb benötigen einen realen[4] Raum und werden sehr stark von den Produkteigenschaften wie Größe, Volumen, Gewicht, Robustheit und Haltbarkeit beeinflusst. Je einfacher physikalische Güter aufgrund dieser Eigenschaften während des Vertriebs zu handhaben sind, desto geeigneter erscheinen sie für einen direkten Verkauf an die Konsumenten. Die Zustellung physikalischer Güter erfordert Adressdaten, die in Kundenprofilen zu hinterlegen sind.

- Digitales Gut

  [Mer99] prägt den Begriff der *Soft Goods*, mit denen er digitale Güter bezeichnet. [Hoq00] verwendet den Begriff der *Virtual Goods* für diese Waren und verweist darauf, dass es bei einigen Gütern gelungen ist, aus einem ursprünglichen physikalischen Gut (oder *Hard Good*) ein digitales zu erzeugen. Der wesentliche Vorteil von digitalen Gütern liegt in ihren Verarbeitungseigenschaften. Einmalig hergestellte Waren lassen sich nahezu kostenfrei vervielfältigen und vor allem problemlos über die gegebenen Infrastrukturen elektronisch vertreiben. Eine direkte Zustellung vom Hersteller bis zum Endverbraucher ist möglich und die klassischen Vertriebswege über Zwischenhandel und Handel können umgangen werden. Sofern es das Zahlungssystem zulässt, kann eine vollständig anonyme Abwicklung einer Kauftransaktion durchgeführt werden.

4. *real* ist hier als Abgrenzung zu *virtuell* zu verstehen

- Vertragsgut

  Versicherungen stellen ein typisches Vertragsgut dar. Während der Träger des Vertrages (in der Regel Papierdokument) als physikalisches Gut bezeichnet werden kann, ist der eigentliche Inhalt nicht greifbar, aber auch nicht digitalisiert. Spezielle Eigenschaften von Vertragsgütern sind zumeist eine definierte Laufzeit und eine damit verbundene periodische Vertragsgebühr, sowie die Sicherstellung der Authentizität der Vertragspartner.

  Im Gegensatz zu Dienstleistungen (s.u.) erhält der Kunde bei Vertragsgütern keine unmittelbare Leistung, für die er bezahlt, sondern es wird mit dem Unternehmen eine Vereinbarung über eine Leistungserbringung getroffen. Die unmittelbare Leistung (und damit die tatsächliche Dienstleistung) eines Versicherungsvertrags wäre z.B. die Zahlung der Versicherungssumme, die aber nur im Schadensfall eintritt, so dass der Vertrag nur die verbindliche Vereinbarung über diesen Sachverhalt zwischen Versicherungsnehmer und Versicherungsgesellschaft regelt und fest hält.

- Dienstleistung

  [Mer99] prägt zusätzlich den Begriff der *Soft-Services*, womit keine Produkte, sondern Dienstleistungen gemeint sind, die vollständig über das Internet abgewickelt werden können. Beispiele hierfür sind das Online-Banking, Reisebuchung, Übersetzungsdienste, Suchdienste oder die Entwicklung von Softwarekomponenten.

- Information

  Obwohl im e-Business Informationen meist frei zugänglich sind und bis auf wenige Ausnahmen[5] kein Geschäftsmodell direktes Entgeld für Content erhebt, bilden Informationen sehr häufig das eigentliche Angebot von Geschäftsmodellen oder aber sind als zusätzliche Services Bestandteil des Angebots.

Wie die unterschiedlichen Ausprägungen zeigen, beeinflusst die physikalische Beschaffenheit die notwendige Identität des Kunden, die Abwicklung der Zustellung, die möglichen Zahlungsvarianten und die Dauer einer Geschäftsbeziehung.

4.1.2.2 Preisniveau des Angebotes

Der monetäre Wert des Angebotes wird hier als Preisniveau bezeichnet und steht in direktem Zusammenhang mit zu verwendenden Bezahlverfahren im e-Business.

5. Nach [Rad01] gelang es dem Wall Street Journal als einem der wenigen Geschäftsmodelle, für einen Teil des aktuellen Online-Inhaltes Geld einzunehmen.

[MTL99] liefern eine Unterteilung der Handelstätigkeiten in Mikrotransaktionen und Makrotransaktionen. Mikrotransaktionen umfassen demnach Beträge von wenigen Cent bis etwa 5 €, Makrotransaktionen den Bereich von etwa 5 € bis zu einigen 1000 €. Diese Unterteilung ist aufgrund der Ausrichtung der Kriterien auf die spätere Herleitung einer Softwarearchitektur zu grobgranular, so dass sie hier nicht verwendet wird.

Stattdessen wird die Unterteilung von [Rei01][6] aufgegriffen, der die Kategorien wie folgt abgrenzt:

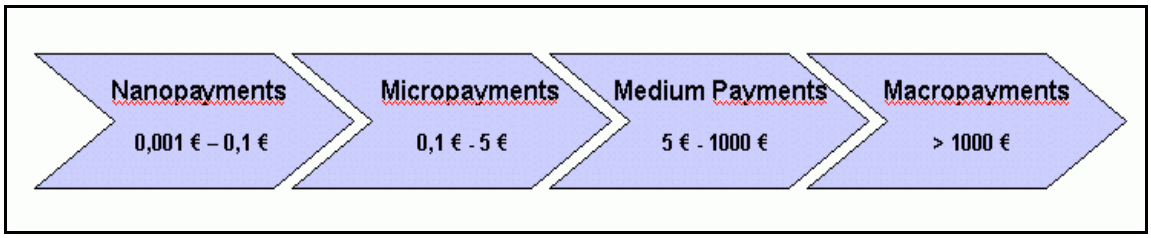


Abbildung 4-1: Vier Stufen des Preisniveaus in Anlehnung an [Rei01]

- Nanopayments (0,001 € - 0,1 €)

  Einzelne Nachrichtenartikel oder einfache Auskunftsdienste sind Beispiele für Angebote, die in dieser Kategorie anzusiedeln sind. Eine Bezahlung einer einzelnen Einheit ist nur mit digitalen Zahlungsverfahren realisierbar. Für andere Zahlungsverfahren sind die anfallenden Fixkosten der Transaktion zu hoch.

- Micropayments (0,1 €- 5 €)

  Vertreter dieser Kategorie sind Zeitschriften oder umfangreiche Testberichte. Auch hier müssen digitale Zahlungsverfahren berücksichtigt werden. Darüber hinaus können jedoch auch konventionelle Verfahren wie Überweisung oder Lastschrift unterstützt werden, wenn Transaktionen der oberen Kategoriegrenze durchgeführt werden.

- Mediumpayments (5 €- 1000 €)

  Das Angebot von e-Shop umfasst in der Regel Produkte in diesem finanziellen Bereich. Beispiele sind Bücher, CDs, Konzertkarten, Kleidung, HiFi-Artikel, Photocameras, Flüge und Reisen. Als potenzielle Zahlungsmittel stehen in dieser Kategorie sowohl digitale als auch konventionelle Verfahren zur Verfügung.

6. Während [Rei01] von Payments spricht, wird hier das Angebot als Grundlage der Unterteilung gewählt. Sofern ein Unternehmen von einem bestimmten Kaufvolumen ausgeht oder einen Mindestumsatz für eine Transaktion vorgibt und dieser vom einzelnen Angebot abweicht, ist die entsprechende Kategorie des Gesamtwertes zu berücksichtigen. Insofern kann die Abweichung hier vernachlässigt werden.

- Macropayments (> 1000 €)

  Der direkte Verkauf von Waren dieser Preiskategorie wie z.B. Autos an die Endkunden ist im e-Business nur sehr schwer vollständig durchzuführen. Kunden verhalten sich sehr vorsichtig beim Erwerb dieser Güter und sehen stationäre Vertriebswege immer noch als sicherer an als die elektronischen Kanäle [DPC00].

  Dennoch können geeignete Verfahren eingesetzt werden, um auch Angebote dieser Kategorie im e-Business zu handeln. Als konventionelles Zahlungsverfahren kann z.B. die Zustellung mit Nachnahme unterstützt werden. Auch eine zusätzliche vertragliche Vereinbarung kann eine Maßnahme sein, Transaktionen in dieser Kategorie zu ermöglichen.

Der monetäre Wert des Angebotes hat direkt ableitbare Auswirkungen auf die Gestaltung der Softwarearchitektur, wie die Beispiele anhand der Zahlungsabwicklung zeigen.

4.1.2.3 Herkunft des Angebotes

Anhand dieses Kriteriums wird beschrieben, ob die Erstellung des Angebotes des Unternehmens außerhalb des eigenen Einflussbereiches liegt. Es werden folgende Ausprägungen unterschieden:

- fremdbestimmt

  Unternehmen, die weitgehend keinen redaktionellen Einfluss auf ihr Angebot haben, weisen einen hohen Fremdbestimmungsgrad des Angebotes auf. Ein Syndicator (=Mittler)[7] bezieht sein Anbebot weitestgehend von zahlreichen Zulieferern und stellt sie in gebündelter Form seinen Kunden zur Verfügung. Auf den Inhalt selbst nimmt er dabei keinerlei verändernden Einfluss; es findet folglich kein eigener redaktioneller Eingriff statt. Das Angebot ist somit fremdbestimmt. Ein anderes Beispiel bietet ein Marktplatzbetreiber, der lediglich eine Plattform zur Verfügung stellt, auf der die Teilnehmer das Angebot erstellen und handeln. Der Betreiber stellt dieses Angebot ohne weiteren redaktionellen Eingriff zur Verfügung[8].

- selbstbestimmt

  Für Geschäftsmodelle, die ihr Angebot redaktionell selber erstellen oder die Erstellung überwachen, trifft diese Ausprägung zu. Ein Beispiel sind e-Shops, die ihre gehandelten Produkte zwar von Zulieferern beziehen, die Auswahl jedoch selbstständig bestimmen und auch die Aufbereitung des Angebotes kontrollieren. Sie greifen also aktiv in die Angebotserstellung ein.

7. Laut einer Definition von [Kro02] handelt es sich bei Syndication um das Zusammenstellen und Vermitteln digitaler Inhalte.
8. Eine etwaige Kontrolle des Angebotes durch den Betreiber wird hier vernachlässigt.

4.1.3 Nutzenbezogene Kriterien

Entsprechend der Definition von Timmers ist die Beschreibung des *potenziellen Nutzens für die Akteure* ein Bestandteil von Geschäftsmodellen. Aus Sicht des Erlösmodells ist der Nutzen für die Akteure deshalb interessant, weil er eine Antwort gibt auf die Frage „Wofür sind die Akteure bereit zu zahlen?“.

Betrachtet man also den Ansatz von Skiera und Lambrecht (vgl. Abschnitt 3.2.2 auf Seite 48ff), Erlösmodelle entsprechend der Gegenleistung zu klassifizieren, so kann dies auch mit dem Nutzenaspekt gleichgesetzt werden. Darüber hinaus richtet sich die Frage aber auch an die Herkunft der Erlöse, da sie die Akteure anspricht. Somit kann der Nutzen auch als Kriterium dafür verwendet werden, Erlösmodelle nach dem Ansatz von Zerdick et al. (Abschnitt 3.2.1 auf Seite 46ff) zu beschreiben.

An dieser Stelle werden aufgrund der Vielfältigkeit möglicher Nutzenarten keine Ausprägungen zu diesem Klassifikationskriterium angegeben. Stattdessen wird diese Frage durch den Begriff des *Primärnutzens* bei der Diskussion der Erlösmodelle gestellt.

4.1.3.1 Primärnutzen

Das Kriterium beantwortet die Frage „Wofür sind die Akteure bereit zu zahlen?“.

4.1.3.2 Weitere Nutzenarten

Neben diesem primären Nutzen für den Akteur können weitere Nutzenarten durch ein Geschäftsmodell gegeben sein. Beispiele für Nutzenarten, die sich für Kunden im e-Business häufig ergeben:

- Verfügbarkeit eines umfassenden Informationspools
- Zeitersparnis durch schnelle Informationsbeschaffung mit umfangreichen Recherchemöglichkeiten
- Kostenvorteile, die Händler durch Einsparungen in der Vertriebsgestaltung an die Käufer weitergeben
- Bequemlichkeit, die sich durch die ubiquitären Zugangsmöglichkeiten des e-Business ergibt

Diese Vorteile für den Kunden sind direkt aus den Eigenschaften des e-Business abzuleiten, ohne dass das Geschäftsmodell speziell darauf ausgerichtet sein muss. Sie können daher als *e-Business-inherente* Nutzenarten bezeichnet werden. Darüber hinaus gibt es aber auch Nutzenarten, deren Umsetzung explizit von einem Geschäftsmodell berücksichtigt wird. Diese können hier jedoch aufgrund der vielfältigen Möglichkeiten nicht vollständig oder weitgehend vollständig klassifiziert werden. Häufig wird der Einsatz umfangreicher Dienstleistungen in Form von Mehrwerten für die Kunden als wichtiger Aspekt für die Nutzung eines Angebotes angesehen[9]. Dieser Nutzentyp kann jedoch nicht allgemein beschrieben werden.

Dennoch werden im Rahmen dieser Arbeit die weiteren Nutzenarten *Personalisierung* und *Anonymität* näher betrachtet.

- Personalisierung

  Personalisierung im e-Business beruht wesentlich auf den Ideen des One-to-One-Marketing von Peppers und Rogers (vgl. hierzu [PR97]). Demnach wird trotz der Anonymität des Internets und trotz des Überangebotes an Informationen versucht, jeden einzelnen Kunden gezielt anzusprechen und personalisierte Angebote bereitzustellen. Um jeweils nur die Informationen anzuzeigen, die den persönlichen Interessen des Kunden entsprechen, wird das Angebot des Unternehmens kategorisiert und mit den Interessen des Kunden abgeglichen. Diese Interessen werden in Nutzerprofilen gespeichert, nachdem sie direkt oder indirekt erhoben wurden.

- Anonymität

  Dieser Nutzenaspekt ergibt sich nicht direkt durch das Angebot eines Unternehmens, sondern ist eine Begleiterscheinung des Internet. Anonymität kann ein wichtiger Nutzen für einen Kunden darstellen und ein Grund sein, ein Angebot in Anspruch zu nehmen. [Ott01] weist darauf hin, dass Produkte, deren Beschaffung nicht nach außen bekannt werden soll, sich für eine (weitgehend anonyme) Beschaffung über das Internet besonders eignen. Daraus folgt auch, dass in diesem Fall die Anonymität soweit wie möglich im Interaktionsprozess zwischen Unternehmen und Kunden zu berücksichtigen ist.

Inwiefern die Nutzenaspekte für einzelne Erlösmodelle relevant sind, wird in den entsprechenden Abschnitten innerhalb der Typologie betrachtet.

4.1.4 Erlösbezogene Kriterien

Dieser Abschnitt enthält Kriterien, die sich auf die Erlöse beziehen.

4.1.4.1 Erlösherkunft

Der Ansatz von Zerdick et al. [ZPS99], die Herkunft der Erlöse als Klassifikationskriterium zu verwenden, wird auch hier aufgegriffen. Deshalb wird unterschieden in eine *direkte* und *indirekte* Erlösherkunft.

- direkt

  Die direkten Erlösformen werden unmittelbar vom Nutzer der Leistung bezogen. Der Nutzer entspricht hier dem Kunden.

9. Laut [Öst00] ist zu erwarten, dass die Unternehmen im e-Business dazu übergehen, nicht nur für einen Bedarf, sondern für eine ganze Reihe damit verbundener Bedürfnisse des Kunden ein zentraler Ansprechpartner zu sein. Das Vorhandensein solcher zusätzlicher Services, die einen Mehrwert darstellen, beeinflusst direkt die Nutzung des Internet als Verkaufskanal [Mef01]. Dies ist auch darauf zurückzuführen, dass im Zuge eines Wertewandels ein gestiegenes Service-Bewusstsein beim Kunden vorliegt (vgl. [HL02] und [Cho97])

- indirekt

  Die indirekten Erlöse dagegen werden nicht direkt von den Kunden gezahlt, sondern von Dritten, die ein Interesse daran haben, dass der Konsument die Leistungen des Anbieters nutzt.

4.1.4.2 Zahlungsverfahren

Je nach Erlösmodell können unterschiedliche Zahlungsverfahren ausgewählt werden. Es stehen grundsätzlich folgende Möglichkeiten zur Verfügung:

- Rechnung

  Dem Kunden wird eine Rechnung zugestellt, so dass dieser nach Erhalt der Ware den Betrag über seine Bank an das Unternehmen überweist. Der Kunde muss somit identifizierbar sein und seine postalische Adresse angeben.

- Lastschrift

  Der Kunde räumt dem Unternehmen eine Vollmacht ein, dass dieses den Betrag von seinem Bankkonto abbuchen kann. Der Kunde muss somit ebenfalls identifizierbar sein und seine Bankdaten, nicht jedoch seine postalische Anschrift angeben.

- Nachnahme

  Dieses Verfahren erhöht die Sicherheit für die Handelspartner, da die Ware zugestellt wird, jedoch vom Zusteller erst ausgehändigt wird, wenn der Rechnungsbetrag übergeben wurde. Der Kunde kann also sicher sein, die Ware zu erhalten; das Unternehmen kann sicher sein, das Geld zu bekommen. Nachteil dieses Verfahrens sind die zusätzlichen Gebühren, die der Zusteller erhebt.

- Kreditkarte

  Dieses Verfahren entspricht weitgehend einer Vorauskasse. Der Kunde teilt dem Unternehmen seine Kreditkartennummer mit, bevor dieses die Produkte zustellt oder die Dienstleistung ausführt. Das Unternehmen kann die Echtheit der Daten und die Gültigkeit der Karte direkt überprüfen und damit sicherstellen, den Betrag vom Kunden zu erhalten. Der Kunde muss neben diesen Angaben keine zusätzliche Identifikation preisgeben[10].

10. Zumindest nicht aus Sicht der Zahlungsabwicklung. Die Zustellung der Produkte oder Dienstleistung kann dagegen weitere persönliche Daten erfordern.

- Externer Dienstleister zur Zahlungsabwicklung

  Hierbei verfügt der Kunde über ein Konto bei einem separaten Dienstleister, mit dem auch der Händler einen Vertrag abgeschlossen hat. Bezahlungen werden über dieses Konto getätigt. Dem Kunden wird beim Kauf der Waren direkt sein Konto beim Dienstleister belastet. Dieser sammelt die einzelnen Transaktionkosten und zieht den Betrag periodisch mittels Lastschrift vom Bankkonto des Kunden ein. Der Dienstleister kann einen periodischen Mindestumsatz vertraglich vereinbaren, so dass es sich für den Kunden und für die Anbieter auch lohnt, Produkte im Preissegment der Nanopayments (vgl. 4.1.2.2 Preisniveau des Angebotes) zu handeln. Ein wesentlicher Vorteil dieses Verfahrens liegt darin, dass der Kunde gegenüber dem Anbieter anonym auftreten und die Zahlung durchführen kann.

4.2 Erlösmodelltypologie

Es werden insgesamt sechs Basiserlösmodelle identifiziert. Die ersten drei können gemäß des Klassifikationskriteriums *Erlösherkunft*[11] als *direkte* Erlösmodelle, die letzten drei als *indirekte* Erlösmodelle charakterisiert werden. Eine detaillierte Betrachtung der Modelle erfolgt in den nächsten Abschnitten.

4.2.1 Einzeltransaktion

Das Erlösmodell EINZELTRANSAKTION ist dadurch charakterisiert, dass der Kunde die Produkte oder Dienstleistungen bei jeder einzelnen Transaktion bezahlt. Dieses Erlösmodell wird hauptsächlich im Geschäftsmodell eines e-Shops angewandt, wo eine Transaktion einer Bestellung der Waren entspricht, die der Kunde direkt zu bezahlen hat.

11. und damit auch gemäß des Ansatzes von Zerdick et al. ([ZPS99]).

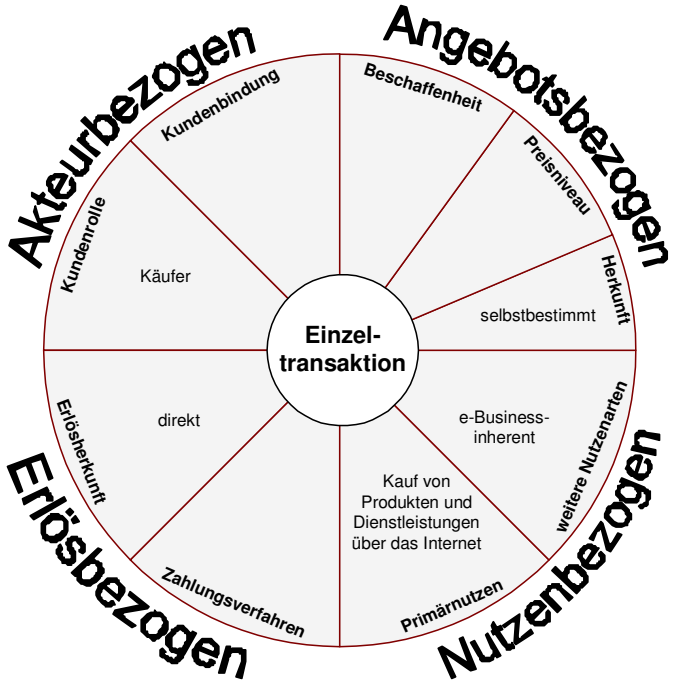


Abbildung 4-2: Erlösmodell Einzeltransaktion in der Basisvariante

Die Darstellung in Abbildung 4-2 zeigt, dass einige Kriterien für das Erlösmodell Einzeltransaktion stets unterstellt werden können. Der Kunde tritt immer als *Käufer* auf und er bezahlt das Angebot *direkt*. Der Primärnutzen ist für den Kunden dadurch gegeben, dass er die Waren direkt über das Internet bestellen kann. Neben den *e-Business-inherenten* Nutzenarten gibt es keine speziellen weiteren, die grundsätzlich bei der Verwendung dieses Erlösmodells identifiziert werden können. Aus Sicht des Unternehmens kann festgehalten werden, dass das Angebot *selbstbestimmt* ist, da ein e-Shop die Wahl und Darstellung seiner Waren kontrolliert und beeinfusst.

Die Kriterien Kundenbindung, Beschaffenheit und Preisniveau des Angebotes sowie die Zahlungsmittel können dagegen in abweichenden Ausprägungen vorkommen, so dass sie gesondert betrachtet werden. Es muss aus Sicht des Erlösmodells zwischen drei Arten von e-Shops unterschieden werden. Es können physikalische Produkte, digitale Produkte oder Dienstleistungen angeboten werden. Die Auswirkungen auf die übrigen Kriterien werden in den zugehörigen Varianten erläutert. Diese Varianten bilden die Subtypen des Erlösmodells EINZELTRANSAKTION.

4.2.1.1 Einzeltransaktion Variante physikalisches Gut

Sofern physikalische Produkte verkauft werden, müssen diese dem Kunden postalisch zugestellt werden, so dass seine Identität zumindest in Form einer Liefer- und Rechnungsanschrift erhoben werden muss. Die Interaktion des Kunden mit dem Unternehmen erfolgt also *identifiziert*. Als Zahlungsverfahren kommen die Varianten *Rechnung*, *Lastschrift*,

Nachnahme und *Kreditkarte* in Frage. Die Kreditkarte kann auch deshalb eingesetzt werden, da das Preisniveau der physikalischen Produkte im *Medium*- oder sogar im *Macro*-Segment liegt. Der Vertrieb physikalischer Produkte im Wert von unter 5 € lohnen sich dagegen aufgrund der Bearbeitungsgebühren und Verwaltungskosten in der Regel nicht für diese Erlösmodell.

Das vollständige Erlösmodell ist in Abbildung 4-3 zu sehen. Die speziell in dieser Variante vorkommenden Ausprägungen der Kriterien sind zur besseren Erkennung unterstrichen dargestellt worden.

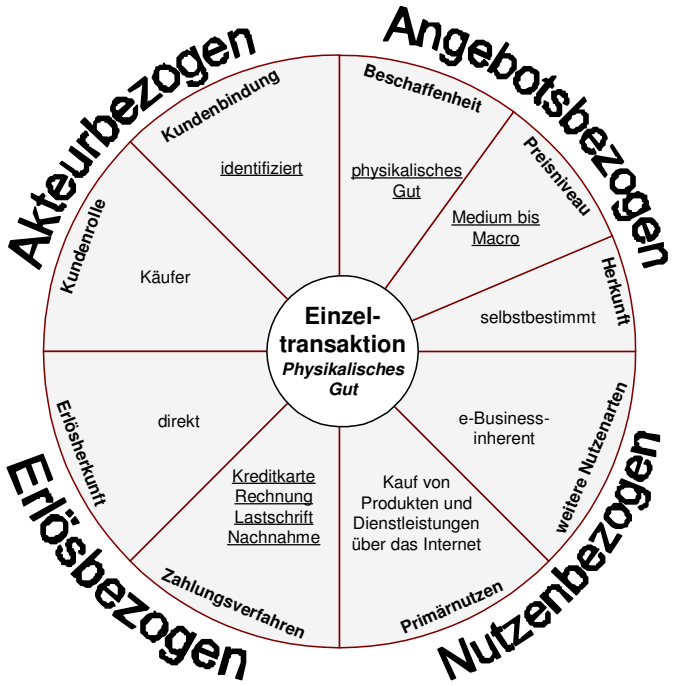


Abbildung 4-3: Erlösmodellklassifikation Einzeltransaktion Variante physikalisches Gut

Aus Sicht einer Softwarearchitekturgestaltung lassen sich folgende grundsätzliche Rückschlüsse anhand dieser Ausprägungen der Klassifikationskriterien herleiten:

- die identifizierten Kunden müssen durch Profile abzubilden und persistent zu speichern sein
- die physikalischen Produkte müssen mittels Produktkatalog verwaltet werden, sie müssen vom Kunden zu bestellen sein und bezahlt werden
- die Zahlungsverfahren müssen unterstützt und vom Kunden zu wählen sein
- die Erlöse werden direkt vom Kunden erhoben; somit muss die Abrechnung berücksichtigt werden

Diese Rückschlüsse ersetzen keine detaillierte Anforderungsanalyse, liefern aber bereits erste Anhaltspunkte für die Architekturgestaltung. Auf dieser Basis kann während der Herleitung der Software-Referenzarchitekturen aufgebaut werden. Diese Herleitung erfolgt in Kapitel 6 und wird dort umfassend beschrieben.

4.2.1.2 Einzeltransaktion Variante digitales Gut

Der Vertrieb digitaler Güter über das Internet hat vor allem hinsichtlich seiner Vervielfältigung, Lagerung und Zustellung Vorteile gegenüber physikalischen Produkten. So kann der Kunde die digitalen Produkte wie beispielsweise elektronische Dokumente, Videos oder Musikfiles direkt von der Website des Anbieters herunterladen. Es bedarf also keine Einbindung eines Lieferservices. Ein großer Vorteil aus Sicht des Käufers kann dabei in dem potenziellen Nutzen der *Anonymität* liegen, die in dieser Variante unterstellt wird. Das Geschäftsmodell muss also eine anonyme Kaufabwicklung unterstützen.

Neben der digitalen Form des Angebotes ist darüber hinaus vor allem ein geeignetes Zahlungsverfahren erforderlich. Dieses ist durch die Einschaltung eines *externen Dienstleisters* gegeben. Durch die Nutzung dieses Zahlungsverfahrens muss sich der Kunde nur gegenüber dem Dienstleister, nicht jedoch gegenüber dem Betreiber des Geschäftsmodells identifizieren. Er kann also *anonym* agieren.

Ein weiterer Vorteil dieses Verfahrens besteht darin, dass grundsätzlich auch Kleinstbeträge abgerechnet werden können. Der externe Dienstleister sammelt in der Regel die Belastungen des Kunden über einen gewissen Zeitraum und bucht anschließend den Gesamtbetrag vom Bankkonto des Kunden ab. Der Dienstleister kann darüber hinaus einen monatlichen Mindestumsatz mit seinem Kunden vereinbaren, um am Ende der Abrechnungsperiode einen kostendeckenden Betrag abzubuchen. Durch die Einbeziehung des externen Dienstleisters führt also nicht jede einzelne Transaktion zu einer Bankbewegung und es können Produkte mit sehr geringen Beträgen gehandelt werden. Theoretisch sind auch Macropayments möglich, jedoch sind praktische Anwendungsfälle für digitale Güter in diesem Bereich unwahrscheinlich. Deshalb werden die Bereiche *Nano-*, *Micro-* und *Mediumpayments* berücksichtigt.

Die folgende Abbildung 4-4 zeigt zusammenfassend alle Ausprägungen dieser Variante.

Abbildung 4-4: Erlösmodellklassifikation Einzeltransaktion Variante Digitales Gut

Es ergeben sich die folgenden softwaretechnischen Rückschlüsse:

- der Auftritt ist derart zu gestalten, dass der Kunde anonym interagieren und Produkte kaufen kann
- die Einbeziehung des externen Dienstleisters zur Durchführung der Zahlung ist notwendig
- die digitalen Güter sind zu verwalten, jedoch besteht keine Anforderung an eine postalische Zustellung

Auch diese Anforderungen sind zunächst nur sehr grobgranular und müssen im Rahmen der Softwarearchitekturgestaltung noch detaillierter ergänzt und angepasst werden.

4.2.1.3 Variante Dienstleistung

Des Weiteren besteht die Möglichkeit, Dienstleistungen zu erwerben. Hier muss die Art der Dienstleistung genauer betrachtet werden. Zum einen kann ein Unternehmen Leistungen anbieten, die keine Identifizierung des Auftraggebers benötigen. Ein Beispiel wäre eine Recherche zu einem speziellen Sachgebiet, dessen Ergebnisse digital an eine email-Adresse geschickt werden kann. In diesem Fall ist die Dienstleistung einem digitalen Gut gleichzusetzen. Die Zahlung könnte ebenfalls anonym wie in der eben beschriebenen Variante mittels eines externen Dienstleisters erfolgen.

Im anderen Fall will oder kann der Auftraggeber dageben nicht anonym bleiben. Ein Kunde kann beispielsweise auf die Zustellung einer Rechnung bestehen oder aber die Dienstleistung selber benötigt zur Ausführung die Identifizierung des Kunden. Eine Reise oder ein Flug sind hierfür Beispiele. In diesem Fall kann die Dienstleistung der Variante physikalisches Gut gleichgesetzt werden. Eventuell ist die Produktverwaltung (Flüge, Reisen) weniger aufwendig zu gestalten, aber die sonstigen Schlussfolgerungen gelten unverändert.

Da die Variante Dienstleistung durch die bereits behandelten Varianten je nach konkreter Ausprägung abgedeckt wird, wird sie im weiteren Verlauf dieser Arbeit nicht mehr berücksichtigt.

4.2.2 Subskription

Das zweite Erlösmodell ist die SUBSKRIPTION. Hierbei besteht ein Vertrag zwischen dem Anbieter einer Leistung und dessen Kunden, der einerseits eine periodische Bezahlung eines bestimmten Betrages durch den Kunden regelt und als Gegenleistung den Bezug der Leistung in einer bestimmten Menge oder Zeit vereinbart. Übersteigt der Leistungsbezug den vereinbarten Rahmen, sind die zusätzlichen Einheiten extra zu bezahlen.

Die folgende Abbildung 4-5 zeigt die grundsätzlich gültigen Klassifikationskriterien für dieses Erlösmodell.

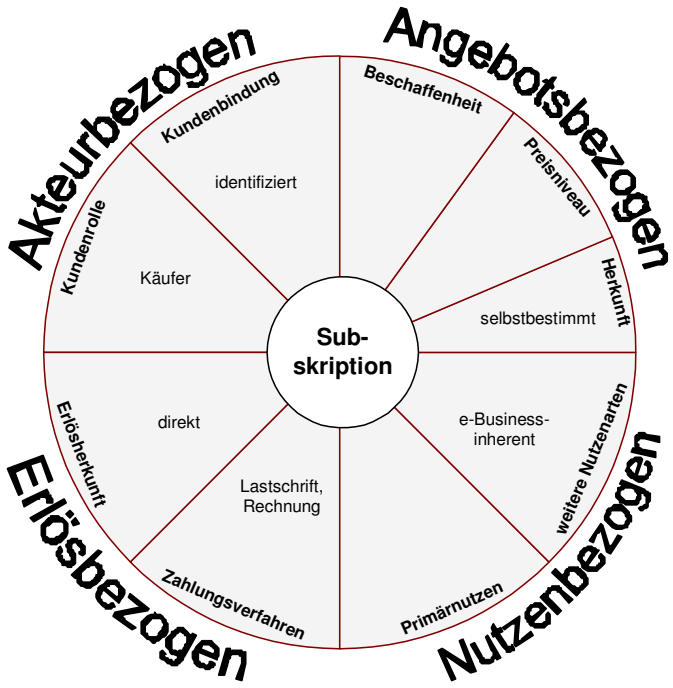


Abbildung 4-5: Erlösmodell Subskription in der Basisvariante

Da eine vertragliche Bindung zwischen dem Anbieter und dem Kunden besteht, tritt dieser *identifiziert* auf. Darüber hinaus führt diese vertragliche Bindung dazu, dass die Zahlungsverfahren *Lastschrift* und *Rechnung* eingesetzt werden. Da durch die Kreditkarte zusätzliche Kosten entstehen und es aufgrund der Bindung keine Notwendigkeit gibt, scheidet dieses Verfahren ebenso aus wie die Nachnahme oder die Einbeziehung eines externen Dienstleisters zur Zahlungsabwicklung. Der Kunde agiert in der Rolle des *Käufers*, der die Produkte oder Dienstleistungen im Rahmen der vereinbarten Leistung beziehen kann und als Gegenleistung einen Betrag bezahlt. Es handelt sich somit ebenfalls um ein *direktes* Erlösmodell. Das Angebot wird vom Unternehmen selbstständig festgelegt und ist somit *selbstbestimmt*. Es liegen grundsätzlich nur die *e-Business-inherenten* Nutzenarten vor.

Daraus folgt, dass die Angebotsbeschaffenheit, das Preisniveau und der Primärnutzen offenbar unterschiedliche Ausprägungen aufweisen und gesondert zu betrachten sind. Die abweichende Beschaffenheit des Angebotes stellt dabei das Unterscheidungskriterium der folgenden drei Subtypen dar.

4.2.2.1 Subskription Variante Dienstleistung

Diese Variante betrachtet die Subskription einer Dienstleistung. Ein typisches Beispiel für ein Geschäftsmodell im e-Business, das dieses Erlösmodell einsetzt, ist ein Internet Service Provider, der seinen Kunden den Zugang zum Internet als Dienst anbietet. Dieses Geschäfstmodell wird hier als Referenzanwendung betrachtet.

Den Kunden des Unternehmens stehen in der Regel einige alternative Tarife zur Auswahl zur Verfügung. Diese richten sich entweder nach der maximalen Datenmenge, die innerhalb einer Periode (meist ein Monat) abgerufen werden kann, oder aber nach einer maximalen Verbindungsdauer, die dem Kunden innerhalb der Periode zur Verfügung steht. Je nach Intensität der Nutzung werden mehrere Alternativen angeboten.

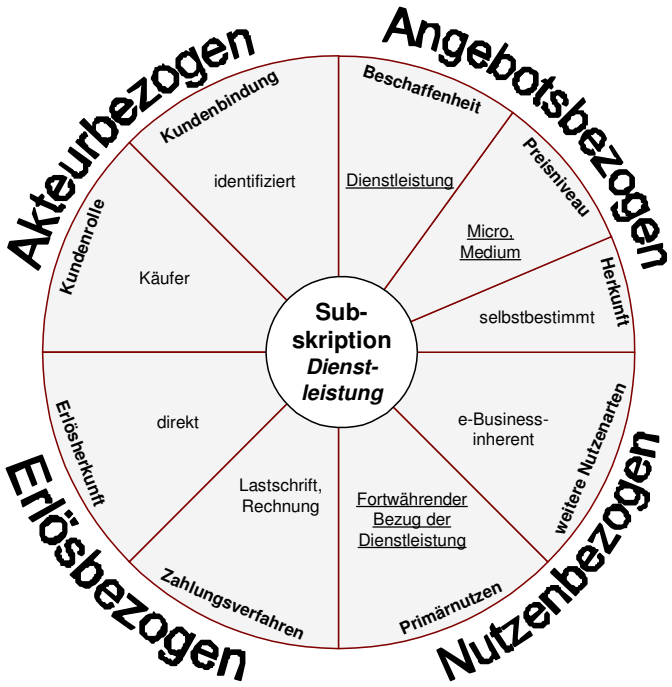


Abbildung 4-6: Erlösmodellklassifikation Subskription Variante Dienstleistung

Als Gegenleistung erbringt der Kunde periodisch einen Betrag, der weitgehend im Segment der oberen Mirco- und der Mediumpayments anzusiedeln ist. Wie bereits erwähnt regelt der Vertrag auch die Kosten für zusätzliche Einheiten, die der Kunde über den vereinbarten Rahmen hinaus verbraucht. Diese werden dann separat in Rechnung gestellt. Der Primärnutzen des Kunden lässt sich somit als fortwährender Bezug der Dienstleistung beschreiben.

Für die Softwarearchitekturgestaltung lässt sich darauf folgendes ableiten:

- die Dienstleistungen müssen in Form von Verträgen und Tarifen abgebildet werden
- der Kunde verfügt über Profile, die seine persönlichen Daten umfassen sowie die vertragliche Bindung belegen
- die tatsächlich in Anspruch genommenen Dienstleistungen (Zugang zum Internet) müssen je nach Einheit (mengen- oder zeitabhängig) protokolliert und am Ende der Rechnungsperiode gegen den Vertrag abgerechnet werden
- der Einzug der Erlöse erfolgt über eine Rechnung oder Lastschrift

Die Verwaltung der Dienstleistungen und die entsprechende Protokollierung des Bezugs stellt eine Besonderheit dieser Variante dar, die im nächsten Subtyp des Erlösmodells SUBSKRIPTION anders umzusetzen ist.

4.2.2.2 Subskription Variante digitales und physikalisches Gut

Als Referenzgeschäftsmodell für diese Erlösmodellvariante wird ein spezialisierter Contentanbieter betrachtet, der sowohl Fachbücher als physikalische Produkte als auch Artikel in digitaler Form im Rahmen eines Abonnements anbietet. Ein Kunde schließt dementsprechend einen Vertrag ab, der ihm innerhalb einer Periode eine bestimmte Anzahl an physikalischen und/oder digitalen Produkten ermöglicht. Die Klassifikation des Erlösmodells ist weitgehend identisch mit der vorherigen Variante, wie die Darstellung in Abbildung 4-7 zeigt.

Abbildung 4-7: Erlösmodellklassifikation Subskription Variante digitales und/oder physikalisches Gut

Auch die Anforderungen an die Softwarearchitekturgestaltung sind weitgehend identisch, weisen jedoch eine spezielle Unterscheidung auf, die sich aufgrund der veränderten Beschaffenheit des Angebotes ergibt:

- die Anzahl der tatsächlich abgerufenen Produkte muss protokolliert und am Ende der Rechnungsperiode gegen den Vertrag abgerechnet werden
- die Produkte müssen über einen Produktkatalog verwaltet werden,

- der Umgang mit physikalischen Produkten erfordert darüber hinaus auch ein Bestell- und ein Versandwesen
- die Rahmenverträge über den Bezug der Produkte müssen mit den unterschiedlichen Tarifen abgebildet werden
- der Kunde verfügt über Profile, die seine persönlichen Daten umfassen sowie die vertragliche Bindung belegen
- der Einzug der Erlöse erfolgt über eine Rechnung oder Lastschrift

4.2.2.3 Subskription Variante Vertragsgut

Bisher wurden für das Erlösmodell SUBSKRIPTION die Varianten für Dienstleistungen sowie für digitale und physikalische Produkte betrachtet. Eine weitere Variante dieses Erlösmodells ist der Sonderfall der *Vertragsgüter*. Ein Beispiel für ein Erlösmodell im e-Business, in dem Vertragsgüter gehandelt werden, ist eine Versicherung, die im Internet agiert. Der Kunde schließt einen Vertrag mit dem Versicherer in Form einer Subskription über den Versicherungsschutz ab. Dadurch bezieht der Kunde für die Laufzeit des Vertrages als Leistung die *Absicherung eines Risikos*, was als Primärnutzen angesehen werden kann. Der Kunde abonniert also den Schutz der Police. Der Charakter der Subskription wird zusätzlich dadurch bekräftigt, dass sich die Verträge in der Regel automatisch um einen festgelegten Zeitraum verlängern, sofern der Kunde nicht explizit kündigt. Die folgende Abbildung zeigt die vollständige Klassifikation dieser Variante:

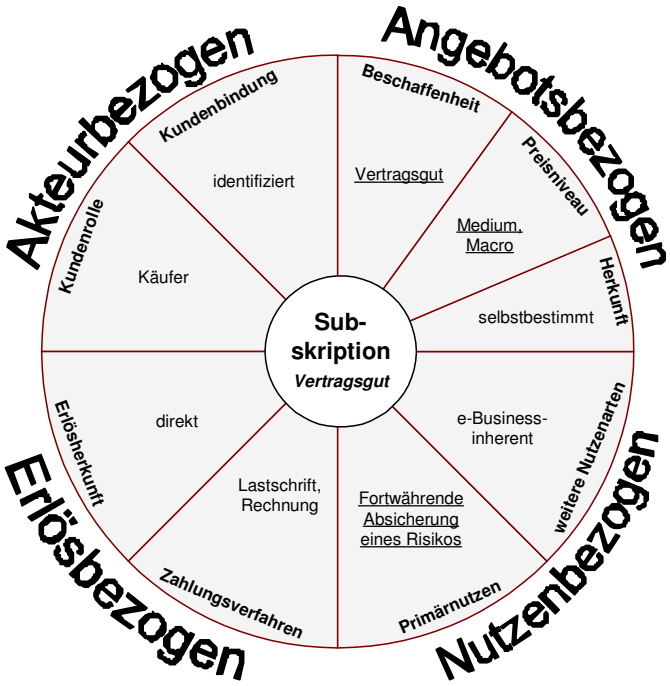


Abbildung 4-8: Erlösmodellklassifikation Subskription Variante Vertragsgut

Die Besonderheit dieser Version liegt darin, dass der Kunde die Leistung (Absicherung des Risikos) nicht in einzelnen Einheiten abruft oder bezieht und der Vertrag somit keine Maximalbezugsgrößen enthält. Eine Protokollierung der Inanspruchnahme einzelner Einheiten entfällt somit ebenso wie eine Berechnung von Einzelpositionen. Beide Aspekte müssen also von der Softwarearchitektur nicht berücksichtigt werden.

Die Verwaltung der Produkte, also der Policen, einer Versicherung sowie die Zuordnung der Policen zu den Kunden erfolgt über ein eigenständiges komplexes Bestandsführungssystem, dass das zentrale Element der Architektur darstellt. Da das Bestandsführungssystem einer Versicherung jedoch nur am Rand mit dem Erlösmodell zu tun hat, wird diese Variante im Rahmen dieser Arbeit nicht weiter betrachtet.

4.2.3 Transaktionsgebühren

Transaktionsgebühren fallen immer dann an, wenn die Kunden für die Durchführung einer Transaktion ein Entgelt an den Betreiber einer Plattform entrichten. Da die Kunden auch die handelnden Akteure innerhalb des Geschäftsmodells sind, wird der Erlös also direkt von ihnen erhoben. Somit ist auch das Erlösmodell TRANSAKTIONEGEBÜHREN den *direkten* Erlösmodellen zugeordnet. Als Referenz dient das Geschäftsmodell eines e-Marktes. Das Unternehmen ist der Marktplatzbetreiber und stellt seinen Kunden eine Plattform zur Verfügung, die diese zum Handeln nutzen können. Abbildung 4-9 zeigt das Erlösmodell mit seinen Klassifikationskriterien und deren Ausprägungen.

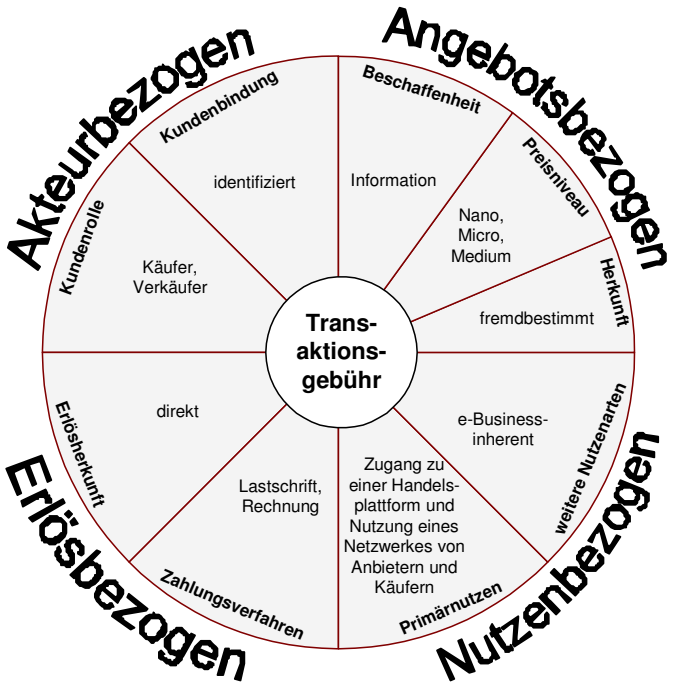


Abbildung 4-9: Erlösmodell Transaktionsgebühr in der Basisvariante

Wie zu erkennen ist, sind bereits alle Klassifikationskriterien in der Basisvariante mit Ausprägungen angegeben. Die Kunden treten als *Käufer* und als *Verkäufer* auf. Für den Zugang zum Marktplatz müssen sie sich registrieren, so dass sie *identifiziert* agieren. Gegebenenfalls können Interessenten zunächst auch anonym auftreten, jedoch müssen spätestens zur Transaktionsabwicklung beide Parteien identifizierbar sein. Die Kunden (genauer: die Verkäufer) stellen das Angebot zusammen, so dass es aus Sicht des Unternehmens *fremdbestimmt* ist. Das Unternehmen selber verwaltet auch keine Produkte - es stellt lediglich die Informationen über diese Produkte der Anbieter in aufbereiteter Form dar. Somit ist die Beschaffenheit des Angebotes als *Informationen* klassifiziert. Die Höhe der Transaktionsgebühren an das Unternehmen kann fix oder prozentual berechnet werden. Die Spannweite wird dadurch von *Nano-* bis *Mediumpayments* reichen. Als Primärnutzen für die Anwender kann der *Zugang zu der Handelsplattform und die damit verbundene Nutzung des Netzwerkes von Anbietern und Käufern* identifiziert werden. Als Zahlungsmittel bieten sich aufgrund der Identifizierung der Kunden und der Spannweite des Preisniveaus auch hier die *Rechnung* und *Lastschrift* an.

Aus Sicht der Softwarearchitektur sind demnach die folgenden Mindestanforderungen zur Realisierung dieses Erlösmodells zu berücksichtigen:

- die Verkäufer müssen eine Möglichkeit haben, das Angebot fremdbestimmt zu erstellen
- der Betreiber konzentriert sich in Bezug auf das Angebot auf die Strukturierung des Produktkatalogs sowie auf die Kontrolle der enthaltenen Produkte
- sowohl die Käufer als auch die Verkäufer müssen über Profile verfügen, die je nach Rolle gegebenenfalls spezifische Daten enthalten
- die Zahlungsverfahren müssen berücksichtigt werden

Trotz dieser vollständigen Angabe der Klassifikationskriterien werden Subtypen unterschieden. Diese richten sich danach, *wer* konkret für die Transaktionskosten aufkommt. Es können entweder nur die Verkäufer, nur die Käufer oder beide Parteien zur Finanzierung des Geschäftsmodells in Frage kommen. Die folgenden Abschnitte zeigen die Unterschiede.

4.2.3.1 Transaktionsgebühr Variante Verkäufer als Kostenträger

Diese Variante unterstellt, dass lediglich der Verkäufer für die Kosten aufkommt. Dies bietet sich aus Sicht des Geschäftsmodellbetreibers dann an, wenn der Marktplatz über eine große Anzahl von Kaufinteressenten verfügt und es somit für den Verkäufer attraktiv ist, diese Interessensgruppe ansprechen zu können.

Vom Verkäufer können zwei Arten von Gebühren erhoben werden: Zugangs- und Verkaufsgebühren. Die Zugangsgebühr kann dabei beispielsweise als Fixbetrag vorgegeben sein, und die Verkaufsgebühr als Transaktionsgebühr prozentual vom erzielten Verkaufspreis berechnet werden.

Die vollständige Darstellung der Erlösmodellklassifikation in Abbildung 4-10 hebt den *Verkäufer* sowie die *direkte* Erlösherkunft hervor. Durch diese Kombination wird das spezifische Merkmal dieses Subtypen ausgedrückt.

Abbildung 4-10: Erlösmodellklassifikation Transaktionsgebühr in der Variante Verkäufer als Kostenträger

Als Besonderheit der Softwarearchitektur muss berücksichtig werden, dass die Transaktionen der Verkäufer protokolliert und die Gebühren ermittelt sowie in Rechnung gestellt werden müssen.

4.2.3.2 Transaktionsgebühr Variante Verkäufer und Käufer als Kostenträger

In dieser Variante kommen zusätzlich zu den Gebühren des Verkäufers auch Gebühren für den Käufer vor. Diese können ebenfalls in Form von Zugangs- und Verkaufsgebühren anfallen. Auch die Gestaltung in fixe und prozentuale Beträge kann frei gewählt werden.

Die Softwarearchitektur muss sicherstellen, dass in dieser Variante beide Handelspartner einer Transaktion protokolliert werden und die entsprechenden Gebühren berechnet werden. Die Profilinformationen beider Akteure sind demnach hier auch umfangreicher als in der vorherigen Variante, da der Käufer dort keine rechnungsrelevante Daten angeben musste. Diese sind hier erforderlich. Abbildung 4-11 zeigt die zugehörige Klassifikation dieses Subtypen des Erlösmodells TRANSAKTIONSGEBÜHR.

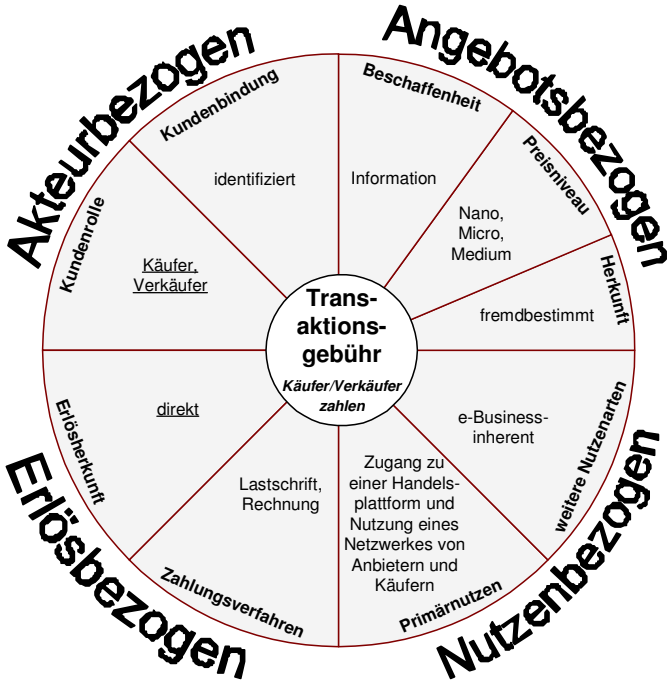


Abbildung 4-11: Erlösmodellklassifikation Transaktionsgebühr in der Variante Verkäufer und Käufer als Kostenträger

Der folgende Abschnitt behandelt den letzten der drei Subtypen.

4.2.3.3 Transaktionsgebühr Variante Käufer als Kostenträger

Analog zur ersten Variante dieses Erlösmodells kann ein Geschäftsmodell auch darauf ausgerichtet sein, dass lediglich die Käufer die Erlöse in Form von Zugangs- und Kaufgebühren beisteuern. Dies ist beispielsweise dann denkbar, wenn die Käufer hohe Einsparungen über den Markplatz erzielen und deshalb ein großes Interesse daran haben, Zugang zu erhalten und Käufe durchzuführen.

Die technischen Aspekte wurden bereits vollständig in der ersten Variante betrachtet, so dass sie hier nicht wiederholt werden. Es muss lediglich die Rolle des Verkäufers durch den Käufer ausgetauscht werden.

4.2.4 Profilhandel

Nachdem bisher drei Erlösmodelle mit *direkter* Erlösherkunft vorgestellt wurden, stellt der PROFILHANDEL das erste *indirekte* Erlösmodell dar. Beim Profilhandel werden zunächst demographische und psychographische Informationen[12] über die Kunden eines Unternehmens gesammelt und ausgewertet, um sie anschließend in aufbereiteter Form an andere

12. vgl. [PR99]

Unternehmen zu verkaufen. In der Regel werden die Daten dabei anonymisiert, so dass nicht das Profil einer einzelnen real existierenden Person weitergegeben wird. Aus Sicht der bezahlenden Unternehmen ist es vielmehr wichtig zu erfahren, wie das Kaufverhalten oder die Interessen bestimmter Kundengruppen sind.

An dieser Stelle kann kein Geschäftsmodell als Referenz angegeben werden, da die Einsatzmöglichkeiten des Erlösmodells zu umfassend sind. Es kann sowohl von e-Shops angewandt werden, die Produkte an ihre Kunden verkaufen, von Communities, deren registrierte Teilnehmer sich je nach Interessensgebieten in Foren zusammenschließen, oder von Nachrichtendiensten, die den anonymen Nutzern ein breites Spektrum an Informationen bereitstellen. Aus diesem Grund sind hier die angebotsbezogenen Klassifikationskriterien nicht zu spezifizieren. Ebensowenig kann die Kundenrolle oder der Primärnutzen allgemeingültig angegeben werden. Abbildung 4-12 zeigt die übrigen Kriterien mit einigen Ausprägungen, die für dieses Erlösmodell grundsätzlich gelten.

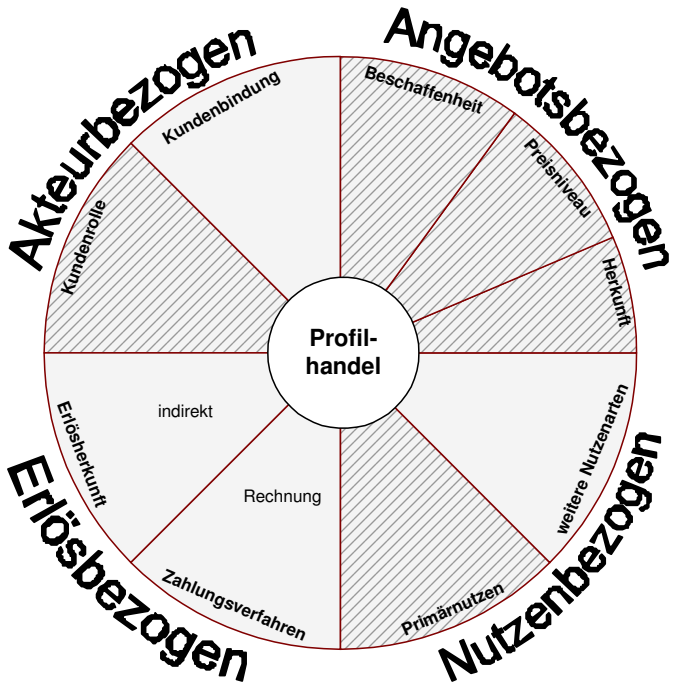


Abbildung 4-12: Erlösmodell Profilhandel in der Basisvariante

Es ist zunächst zu erkennen, dass die Kriterien, die hier nicht betrachtet werden können, schraffiert dargestellt wurden. Die erlösbezogenen Kriterien Erlösherkunft und Zahlungsverfahren sind dagegen bestimmbar. Die Profile werden wie bereits erwähnt von Dritten erworben, die ein irgendwie geartetes Interesse an den Informationen besitzen. Somit bildet nicht direkt das Angebot die Basis der Erlöse, sondern es dient nur dazu, die Kunden zu binden und die Informationen zu sammeln. Die Herkunft der Erlöse ist somit als *indirekt* zu klassifizieren. Da die Interessenten keine Privatpersonen, sondern Unternehmen sind, wird als Zahlungsverfahren lediglich die *Rechnung* zu unterstützen sein.

Die Kundenbindung stellt das entscheidende Kriterium der Subtypenbildung dar. Es gibt somit zwei Varianten; einmal eine Form mit identifizierten und einmal mit anonymen Kunden. Beide werden in den folgenden Abschnitten betrachtet.

4.2.4.1 Profilhandel Variante identifizierter Kunde

Aus Sicht des Unternehmens ist es verständlicherweise erheblich einfacher, Profilinformationen über *identifizierte* Kunden zu erheben. Diese verfügen über ein persistentes Nutzerprofil, dass über die Dauer der Beziehung permanent angereichert werden kann. Geschäftsmodelle, deren Erlöse auf dem Handel mit Profilen ausgelegt sind, haben darüber hinaus ein großes Interesse, so viele Daten und so konkrete Daten wie möglich über den Kunden zu erfahren. Aus diesem Grund kann hier unterstellt werden, dass die Geschäftsmodelle im Rahmen ihrer Angebotsgestaltung als Mehrwert für den Kunden Möglichkeiten zur Personalisierung anbieten.

> Als Beispiel dient ein Geschäftsmodell eines spezialisierten Contentanbieters, der umfassende Börseninformationen bereitstellt. Um die detaillierten Informationen abrufen zu können, muss sich ein Kunde zunächst registrieren. Bereits während der Registrierung wird er nach personenbezogenen Angaben wie Geschlecht, Alter, Ausbildung und Beruf gefragt. Anschließend hat er die Möglichkeit, einen personalisierten Newsletter zusammenzustellen, der ihm jedesmal automatisch zugesandt wird, sobald Ad-hoc Nachrichten über die selektierten Firmen eintreffen. Darüber hinaus kann der Kunde ein Musterdepot mit Wertpapieren einrichten, dessen Wertentwicklung ihm nach jeder Anmeldung direkt angezeigt wird.
>
> Der Kunde hat ein hohes Interesse daran, die Personalisierungsdienste in Anspruch zu nehmen und liefert dem Unternehmen als Gegenleistung umfangreiche Informationen über sich.

Das hier angegebene Beispiel befasst sich mit der *expliziten* Erhebung von Benutzerinformationen. Explizit bedeutet hier, dass die Angaben von dem betroffenen Kunden selber angegeben wurden. Es sind also zum Beispiel Antworten auf Fragen oder Angaben zu bestimmten Interessensgebieten, die der Nutzer von sich aus über die GUI mitteilt.

Eine weitere Möglichkeit, Daten zur Anreicherung der Profile zu sammeln, besteht in der *impliziten* Datenerhebung. Hierbei wird das Verhalten des Nutzers während einer Sitzung aufgezeichnet und anschließend ausgewertet. Ein Beispiel bilden sogenannte Web-Chains[13]. Darunter wird die Schrittfolge der Bewegungen eines Benutzers verstanden. Sie protokolliert die Navigationsschritte des Nutzers anhand dessen Seitenaufrufe. Auf dieser Grundlage können anschließend beispielsweise Rückschlüsse über die Interessen gezogen werden, wenn nicht nur die reine Seitenfolge, sondern zusätzlich auch die Verweildauern auf den einzelnen Seiten ausgewertet werden.

13. vgl. [Han00]

Das Klassifikationskriterium weitere Nutzenarten erhält deshalb die Ausprägung *Personalisierung*. Die Darstellung in Abbildung 4-13 zeigt die vollständige Erlösmodellklassifikation.

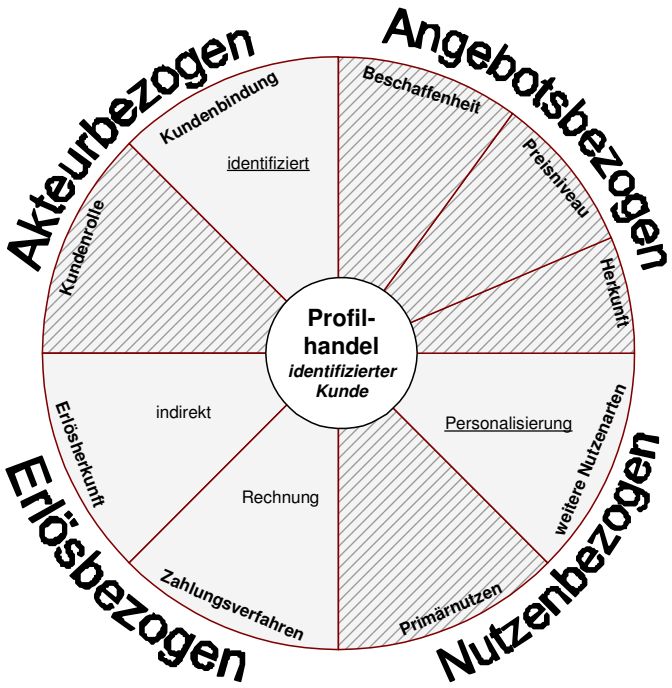


Abbildung 4-13: Erlösmodellklassifikation Profilhandel Variante identifizierter Kunde

Eine Betrachtung der Klassifikationskriterien mitsamt ihren Ausprägungen führt zu folgenden Mindestanforderungen an eine geeignete Softwarearchitektur zur Realisierung dieser Erlösmodellvariante:

- das Kundenprofil als Basis der Einkommensgenerierung muss umfangreich und flexibel erweiterbar sein
- es müssen sowohl Verhaltensweise des Kunden durch implizite Auswertungen der Sitzungen als auch explizite Angaben durch den Kunden gesammelt werden
- es müssen Tools zur Auswertung der erhobenen Daten und zur Aufbereitung der Profilinformationen bereitgestellt werden

Eine genauere Betrachtung der Anforderungen findet im Rahmen der Herleitung der Referenzarchitekturen statt.

4.2.4.2 Profilhandel Variante anonymer Kunde

Alternativ zur vorherigen Variante werden nun Geschäftsmodelle betrachtet, die lediglich eine anonyme Kundenbindung aufweisen. Es wird dabei nicht unterstellt, dass diese Anonymität als expliziter Nutzenaspekt herausgestellt und unterstützt wird. Das zugehörige Klassifikationskriterium bleibt entsprechend ohne Ausprägung, wie in der folgenden Abbildung zu sehen ist.

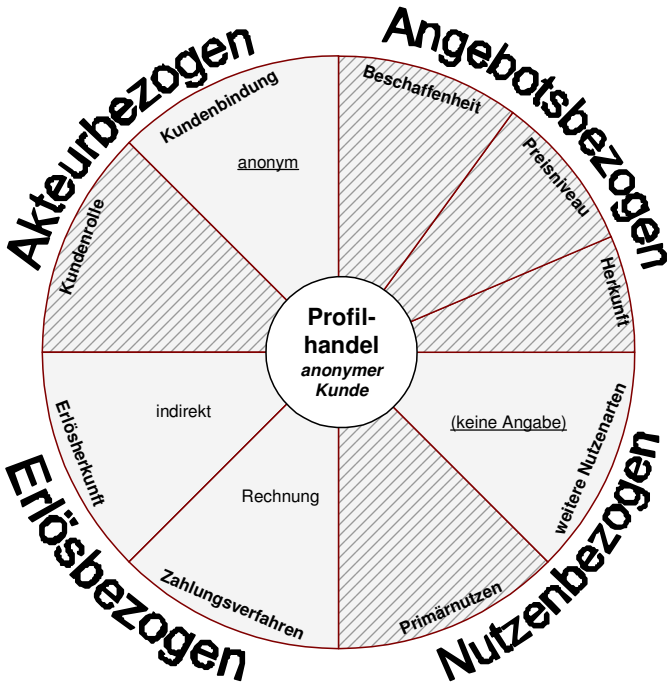


Abbildung 4-14: Erlösmodellklassifikation Profilhandel Variante anonymer Kunde

Aus Sicht der Softwarearchitektur ändert sich nicht grundlegend etwas. Es ist immer noch das Bestreben des Geschäftsmodellbetreibers, Informationen über den Kunden zu gewinnen. Da dieser nun jedoch anonym agiert, kann das Profiling des Kunden nur anhand seines Verhaltens innerhalb der Sitzung erfolgen. Die erhobenen Profile werden demnach sehr wahrscheinlich einen weitaus geringeren Umfang aufweisen als bei identifizierten Kunden und weitestgehend auf der impliziten Erhebung von Daten beruhen.

Es ist aber auch möglich, explizite Angaben des Nutzers zu erheben. Beispielsweise können anonyme Umfragen durchgeführt werden, die anhand der Fragestellung und Antwortmöglichkeiten eine bestimmte Meinung oder ein Interesse des Kunden zum Ausdruck bringen können.

Darüber hinaus besteht die Möglichkeit, unter Verwendung von Cookies einen wiederkehrenden Kunden zu identifizieren und somit sein bereits bestehendes Profil zu ergänzen. Für die Herleitung der Referenzarchitektur wird diese Option jedoch nicht weiter betrachtet.

Zusammenfassend ergeben sich somit die selben grundlegenden Anforderungen an die Softwarearchitektur wie in der vorherigen Variante. Wie bereits erwähnt, wird allerdings die Datenerhebung in dieser Variante nicht sessionübergreifend und darüber hinaus überwiegend implizit erfolgen.

4.2.5 Provision

Die Begriffe Provision und Kommission werden häufig synonym verwendet. Die folgenden Begriffsdefinitionen zeigen jedoch den Unterschied.

Definition 7: Provision
Vergütung für die Besorgung oder Vermittlung von Geschäften. Wird in der Regel in Prozenten des Gewinns oder des Umsatzes berechnet.[14]

Aus der gleichen Quelle ist darüber hinaus zu erfahren, dass eine Kommission „ein Auftrag [ist], der im eigenen Namen für Rechnung eines anderen ausgeführt wird“.

Aus Sicht eines Erlösmodells ist die Vergütung interessant. Insofern bezeichnet das Erlösmodell PROVISION in dieser Arbeit die Zahlung von Erlösen als Gegenleistung von Vermittlungen. Die Vermittlungen beziehen sich dabei auf die Kunden eines Unternehmens, dass an ein anderes Unternehmen weitergeleitet wird. Der Sonderfall einer Abschlussprovision wird separat in einem eigenen Abschnitt ebenfalls betrachtet. Es ist erkenntlich, dass das andere Unternehmen als Herkunft der Erlöse angesehen werden kann, und es sich somit um ein *indirektes* Erlösmodell handelt.

Wichtig ist an dieser Stelle festzuhalten, dass die Vermittlung von Kunden über Werbebanner auch diesem Erlösmodell zugeordnet werden kann. Da aber WERBUNG aufgrund seiner Gewichtung im e-Business als eigenständiges Erlösmodell im nächsten Abschnitt separat betrachtet wird, grenzt sich das Erlösmodell PROVISION hier von dieser Option ausdrücklich ab.

Die folgende Abbildung zeigt die Klassifikationskriterien, die grundsätzlich identifiziert werden können.

14. aus: Bertelsmann Lexikon, Lexikographisches Institut, München (Hrsg.), 1996

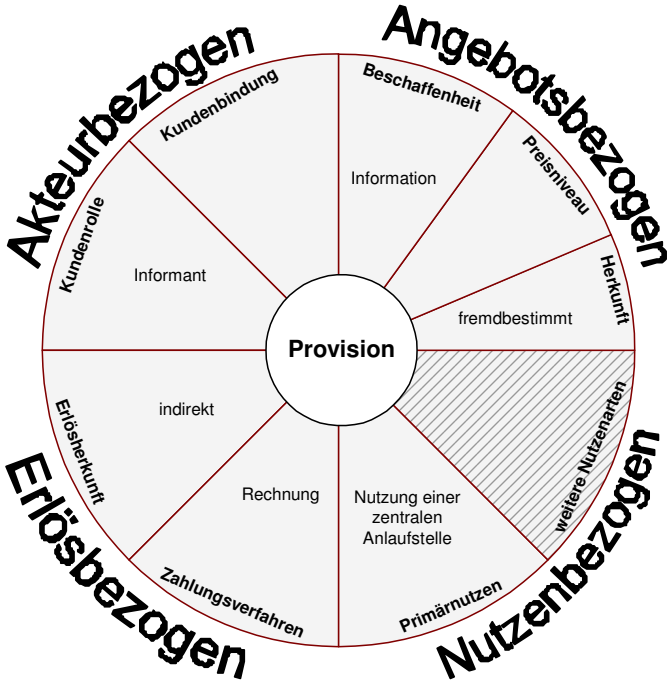


Abbildung 4-15: Erlösmodell Provision in der Basisvariante

Neben der indirekten Erlösherkunft kann festgehalten werden, dass mit den Auftraggebern vertragliche Bindungen bestehen, so dass die *Rechnung* als Zahlungsverfahren ausreicht. Weitere, über den Primärnutzen hinausgehende Nutzenaspekte, können hier angegeben werden, weshalb das Kriterium in der Darstellung schraffiert zu sehen ist. Wie auch schon beim vorherigen Erlösmodell kann die konkrete Gestaltung von Geschäftsmodellen, die das Erlösmodell PROVISION einsetzen, sehr unterschiedlich sein, so dass eine Aussage über die jeweiligen potenziellen Nutzenarten nicht möglich ist.

Grundsätzlich können Provisionen für die Vermittlung von Kunden von sämtlichen Geschäftsmodellen als optionales Erlösmodell mit einbezogen werden. Ein e-Shop kann in seinem Angebot Produkte aufnehmen, die nicht bei ihm direkt, sondern bei einem Partnerunternehmen zu erwerben sind. Ein Kunde würde dann an dieses Unternehmen weitergeleitet werden, sobald er auf die Darstellung klickt. Ein Nachrichtenmagazin kann in seinem redaktionellen Inhalt ebenfalls auf andere Unternehmen hinweisen und somit Kunden vermitteln, ebenso wie ein Marktplatzbetreiber den Verkäufern besondere Dienstleistungen offeriert, die nicht von ihm, sondern von einem Partnerunternehmen angeboten werden. Darüber hinaus existieren jedoch auch Geschäftsmodelle, die sich ausschließlich oder schwerpunktmäßig über dieses Erlösmodell finanzieren und ihr Angebot darauf ausgerichtet haben. Als Referenz für zwei Subtypen werden deshalb zum einen eine Preisvergleichseite sowie ein Online-Versicherungsvermittler gewählt. Bevor auf die Unterschiede beider Subtypen eingegangen wird, können zunächst die gemeinsamen Ausprägungen der Kriterien betrachtet werden.

Der primäre Nutzen für die Kunden kann in der *Nutzung einer zentralen Anlaufstelle* gesehen werden. Das Geschäftsmodell fungiert als Portal für die Kunden, indem es eine Vielzahl von unterschiedlichen Angeboten zentral bündelt und dem Kunden zugänglich macht. Somit erspart sich der Kunde eine umfangreiche und mühselige Recherche unter den einzelnen Anbietern. Er agiert in der Rolle des *Informanten*. Das Angebot der Geschäftsmodelle ist gleichzeitig *fremdbestimmt*. Es wird eben von den Anbietern bereitgestellt, die auf eine Vermittlung der interessierten Kunden hoffen. Ähnlich wie ein Marktplatzbetreiber bietet der Betreiber des Geschäftsmodells somit lediglich *Informationen* über die Produkte und Dienstleistungen an.

Das entscheidende Kriterium der Subtypenbildung stellt die Kundenbindung dar. Sie hat, wie in den folgenden Abschnitten gezeigt wird, auch Auswirkung auf das Preisniveau der Vermittlungen.

4.2.5.1 Provision Variante Vermittlung anonymer Kunde

Eine Preisvergleichseite wird als Referenz für diese Variante gewählt. Das Geschäftsmodell basiert darauf, die Produkt- und Dienstleistungsangebote zahlreicher e-Shops auf der eigenen Website zu vereinen und dem Kunden die passendsten Angebote aufzulisten. Standardisierte Produkte wie elektronische Artikel werden in der Regel anhand des Preises verglichen, Dienstleistungen wie Flüge oder Reisen müssen zusätzlich eine Reihe von Kriterien erfüllen, um dem Kunden angeboten zu werden. Beispiele sind Datum, Dauer, Personenanzahl sowie gewünschter Komfort. Abbildung 4-16 zeigt die Kriterienausprägungen.

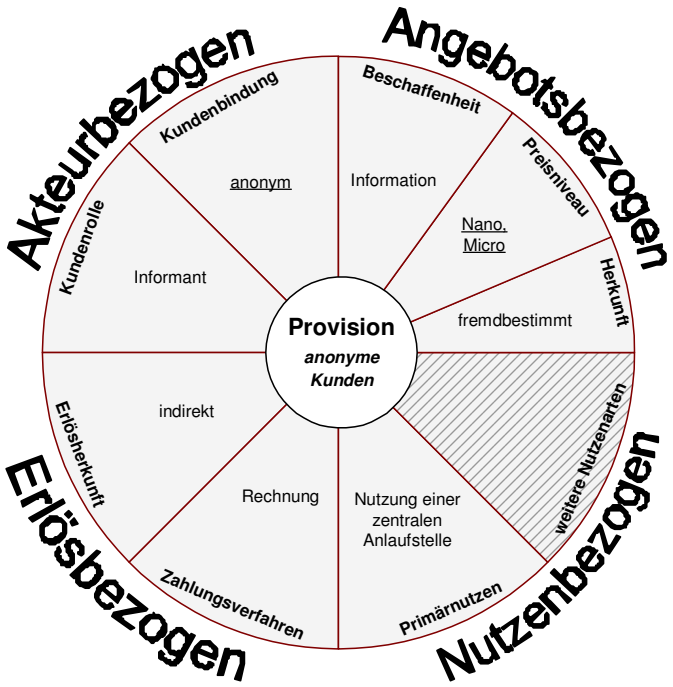


Abbildung 4-16: Erlösmodellklassifikation Provision Variante anonymer Kunde

Der Kunde agiert in dieser Variante *anonym*. Er gibt lediglich seine Kriterien an, die die Auswahl und Aufbereitung der Produkte beeinflussen und betrachtet das Ergebnis. Entscheidet er sich für ein Angebot, so wählt er dieses aus und stellt auf diese Weise die Vermittlung zum Anbieter (Betreiber des e-Shops) her. Dabei gelangt der Kunde in der Regel auf der Website des Anbieters direkt zu dem von ihm gewählten Produkt. Sofern bereits zusätzliche Wünsche bezüglich einer Konfiguration angegeben wurden, werden auch diese auf der Seite des Anbieters berücksichtigt.

Die Vermittlung des Kunden an den Anbieter führt zu einer Buchung der vertraglich vereinbarten Gebühr auf dem Konto des Anbieters. Periodisch werden die angefallenen Vermittlungsbeträge vom Geschäftsmodellbetreiber kumuliert und dem Anbieter in Rechnung gestellt. Da der vermittelte Kunde zunächst nur ein Interessent ist, dessen Kaufentscheidung nocht nicht feststeht, und der darüber hinaus noch anonym agiert, wird der Anbieber bei dieser Variante nur geringe Beträge pro Vermittlung bereit sein zu zahlen. Das Preisniveau bewegt sich somit im Bereich der *Nano*- und *Micropayments*.

Eine Betrachtung des Erlösmodells unter dem Gesichtspunkt einer Softwarearchitekturgestaltung lässt folgende Rückschlüsse zu:

- das fremdbestimmte Angebot muss durch die Anbieter zu verwalten sein
- der Betreiber beschränkt sich auf die Pflege des Produktkatalogs
- die Anbieter, mit denen vertragliche Vereinbarungen bestehen, müssen verwaltet werden
- jede Vermittlung muss protokolliert und der entsprechende Betrag dem Anbieter belastet werden
- periodisch ist eine Rechnungsstellung durchzuführen

Der nächste Abschnitt betrachtet die Vermittlung identifizierter Kunden.

4.2.5.2 Provision Variante Vermittlung identifizierter Kunde

Hier bildet ein Online-Versicherungsvermittler das Referenzgeschäftsmodell. Anders als in der vorherigen Variante wird sich der Kunde hier identifizieren müssen, um das Ergebnis zu erhalten. Der Hauptnutzen für den Kunden besteht auch hier in der zentralen Anlaufstelle und somit in einer Reduzierung der eigenen Recherchetätigkeiten. Aus seiner Sicht sind die Produkte (hier Versicherungspolicen) sehr komplex, so dass er bereit ist, Informationen über sich preiszugeben, die für eine Ermittlung des spezifischen Angebotes notwendig sind. Die Berechnung der Beitragskosten sowie die Ermittlung der optimalen Parameter wie beispielsweise die Höhe der Versicherungssumme erfordern diese Angaben. Erst nachdem der Kunde die erforderlichen Angaben geleistet hat, die auch persönliche Daten wie Name und Anschrift erhalten, erhält er das Ergebnis dargestellt.

Der Kunde wird auf diese Weise eine Reihe persönlicher Angaben leisten, die sein Profil anreichern und später vom Geschäftsmodellbetreiber an die Anbieter weitergereicht werden kann. Das Erlösmodell weist in dieser Variante somit einige Parallelen zu dem Erlösmodell PROFILHANDEL auf. Die folgende Abbildung zeigt die Kriterienausprägungen.

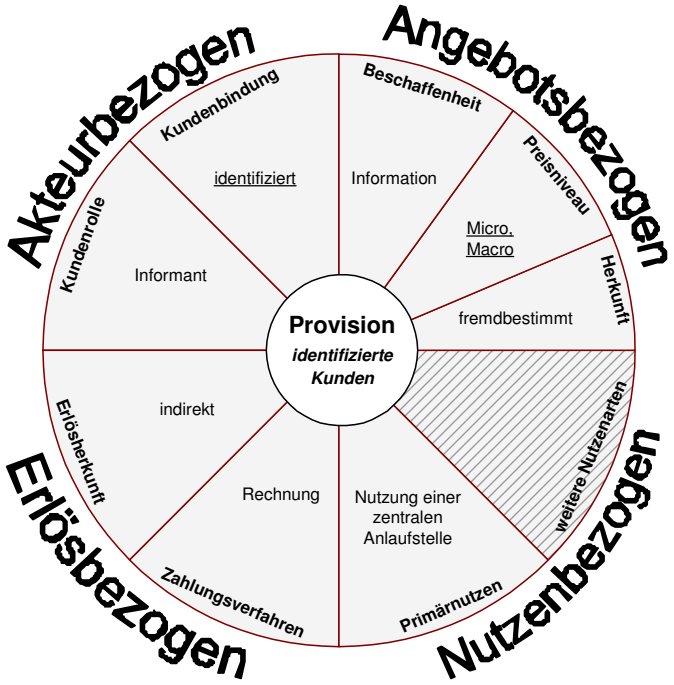


Abbildung 4-17: Erlösmodellklassifikation Provision Variante identifizierter Kunde

Der Kunde ist also zumindest vor Darstellung des Ergebnisses identifiziert. Diese Informationen können anschließend entweder an einen oder aber an alle Anbieter, die in der Ergebnisliste auftauchen, weitergereicht werden. Diese auch als *Lead* benannten Vermittlungen weisen somit eine hohe Qualität für den Empfänger auf, da sie bereits umfangreiche Informationen enthalten. Somit wird der Empfänger bereit sein, einen höheren Betrag für eine Vermittlung zu bezahlen, als dies bei anonymen Vermittlungen der Fall ist. Das Preisniveau kann von *Micro-* bis *Macropayments* reichen.

Die abweichenden Anforderungen müssen auch von einer Softwarearchitektur berücksichtigt werden:

- das fremdbestimmte Angebot muss durch die Anbieter zu verwalten sein; die Produkte sind dabei sehr komplex und werden nicht sehr häufig aktualisiert; sie müssen in die Ablauflogik der Anwendung mit angebunden werden[15]
- die Anbieter, mit denen vertragliche Vereinbarungen bestehen, müssen verwaltet werden
- für die Kunden müssen Profile verwaltet werden, die die erhobenen Daten enthalten
- die erhobenen Profile müssen ausgewertet und aufbereitet werden, um sie an die Anbieter weiterreichen zu können

15. Die konkrete Berechnung der Versicherungspolice hinsichtlich des Leistungsumfangs und der Kosten erfolgt durch einen Rechenkern, den der Anbieter bereitsstellt und der Eingabeparameter benötigt, so dass das Produkt (die Police) über Schnittstellen an die Anwendung angebunden werden muss.

- jede Vermittlung muss protokolliert und der entsprechende Betrag dem Anbieter belastet werden
- periodisch ist eine Rechnungsstellung durchzuführen

Diese Variante erfordert somit als wesentlichen Unterschied eine veränderte Verwaltung des Angebotes sowie die zusätzliche Profilverwaltung der identifizierten Kunden.

4.2.5.3 Provision Variante Abschlussprovision

Diese Variante des Erlösmodells PROVISION stellt einen Sonderfall dar. Er berücksichtigt, dass ein Anbieter nicht für die bloße Vermittlung (anonymer oder identifizierter) Nutzer eine Gebühr bezahlt, sondern nur, wenn diese Vermittlung auch einen tatsächlichen Kauf des Produktes nach sich zieht.

Aus Sicht der Architektur ändert sich jedoch für die Umsetzung dieser Variante nichts. Es wird lediglich vertraglich vereinbart, dass die Gebühr nur in diesem Fall zu entrichten ist. Technisch kann der Betreiber des Geschäftsmodells lediglich die Vermittlung protokollieren, nicht jedoch den Abschluss des Kunden, der für ihn *systemextern* erfolgt. Somit gelten auch hier die bereits in den vorherigen Varianten aufgestellten Architekturen; je nachdem, ob anonyme oder identifizierte Vermittlungen unterstützt werden sollen. Der Sonderfall wird somit im weiteren Verlauf der Arbeit nicht mehr berücksichtigt.

4.2.6 Werbung

Das letzte betrachtete Erlösmodell ist WERBUNG. Allein die Charakterisierung des Internets als ein *neues Medium* verdeutlicht, dass sich diese Form der Erlösgenerierung anbietet und auch umfangreich genutzt wird. Grundsätzlich kann auch dieses Erlösmodell von zahlreichen unterschiedlichen Geschäfsmodellen eingesetzt werden. Dabei kann es zusätzlich zu einem anderen Erlösmodell verwendet werden (z.B. bei e-Shops), oder auch das einzige Modell bilden. Beispiele hierfür sind vor allem die Geschäftsmodelle, die darauf abzielen, über frei zugängliche Angebote eine breite Masse von Kunden anzusprechen und so hohe Zugriffszahlen zu erreichen. Diese starke Frequentierung des Webauftritts ist wiederum interessant für andere Firmen, dort Werbungen zu platzieren. Gleichzeitig sind die hohen Zugriffszahlen auch die Voraussetzung für ein Geschäftsmodell, über Werbung die erforderlichen Erlöse zu erwirtschaften. Das Erlösmodell richtet sich somit wiederum nicht an den eigentlichen Kunden des Angebotes, sondern an andere Unternehmen. Somit wird es als *indirektes* Erlösmodell klassifiziert.

Aus diesem Grund wird hier als Referenzgeschäftsmodell grundsätzlich ein Anbieter von frei zugänglichen Informationen betrachtet. Als Beispiele dienen ein Online-Nachrichtenmagazin, eine Community oder ein Unternehmen, dass kostenlose Routenberechnungen anbietet. Die Beschaffenheit ihres Angebotes ist als *Information* klassifiziert. Dementsprechend agieren die Kunden als *Informanten* oder im Falle einer Community als *Value-Integrator*, nicht jedoch als Käufer oder Verkäufer. Es wird hier nicht festgelegt, inwiefern die Kunden identifiziert oder anonym auftreten; jedoch versuchen die meisten Anbieter, die

Zugriffsschranken des Angebotes so gering wie möglich zu gestalten, um so viele Kunden wie möglich anzusprechen. Aus diesem Grund sind die meisten Angebote für die Kunden nicht nur kostenlos, sondern auch anonym abzurufen.

Wie in Abbildung 4-18 zu sehen ist, sind neben diesem Kriterium auch die Angebotsherkunft sowie die weiteren Nutzenarten schraffiert dargestellt worden, da sie nicht allgemeingültig für Geschäftsmodelle mit diesen Erlösmodellen angegeben werden können und auch nicht relevant sind.

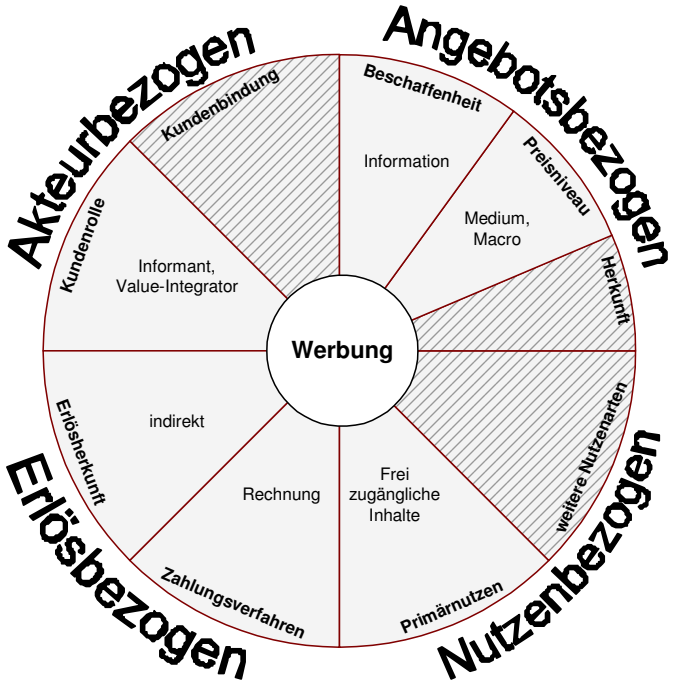


Abbildung 4-18: Erlösmodell Werbung

Der Primärnutzen für einen Kunden besteht in den *frei zugänglichen Inhalten*. Dafür nimmt er auch in Kauf, als Gegenleistung mit Werbung *belästigt* zu werden. Da mit den Auftraggebern, den Schaltern der Werbung, vertragliche Vereinbarungen über die zu schaltenden Werbebanner in Form einzelner Kampagnen bestehen, ist die *Rechnung* als Zahlungsmittel ausreichend. Das Preisniveau reicht von *Medium-* bis *Macropayments*.

Es sind grundsätzlich zwei Arten dieses Erlösmodells zu unterscheiden: die Bezahlung von *AdImpressions* sowie die Bezahlung von *AdClicks*. AdImpressions umfassen lediglich die Anzeige der Werbebanner[16] innerhalb einer Website. Als Einheit für die Abrechnung von Werbebannern hat sich *TPK* (Tausend-Kontakt-Preis) als der Preis je eintausend Werbeeinblendungen etabliert. Eine grobe Analyse im März 2005 hat gezeigt, dass die Preise je nach Werbeform, Seite (Homepage, Nebenseiten oder spezielle Rubrik) und Unternehmen

16. Hier wird der Einfachheit halber nur auf die Form der Werbebanner als Werbemittel eingegangen. Andere Formen wie Pop-ups etc. werden vernachlässigt.

stark schwanken und bei etwa 4€ bis 110€ reichen können[17]. AdClicks dagegen besagen, dass der Kunde einen angezeigten Werbebanner auch tatsächlich durch einen Klick aktiviert hat und somit die zugehörige Aktion (zumeist die Weiterleitung auf eine Homepage des werbenden Unternehmens oder des beworbenen Produktes) auslöst. Da ein AdClick verständlicherweise eine höhere Qualität für das beauftragende Unternehmen darstellt als ein AdImpression, variieren die Preise entsprechend[18].

Je nach Art wird in den Kampagnen, die ein Unternehmen bezahlt, eine bestimmte Anzahl der Anzeigen oder der Aktivierungen vereinbart, die demnach zu bezahlen sind.

Um die Qualität zu steigern und damit höhere Preise zu erzielen, ist es darüber hinaus von großem Interesse für ein Geschäftsmodell mit diesem Erlösmodell, die Kunden so spezifisch wie möglich zu klassifizieren und dementsprechend die Werbungen denjenigen Gruppen anzuzeigen, die das größte Interesse an dem Thema haben. Auf diese Weise sollen Streuverluste verhindert werden, die dann auftreten, wenn eine Werbung einem Kunden angezeigt wird, dem diese aufgrund seines mangelnden Interesses nicht interessiert. Aus diesem Grund stellt das Profiling der Nutzer und die Schaltung geeigneter Werbebanner eine zentrale Aufgabe im Umgang mit diesem Erlösmodell dar.

Die breite Anwendbarkeit des Erlösmodells WERBUNG einerseits und die hohen Anforderungen an die zielgruppengerechte Schaltung der Banner andererseits hat dazu geführt, dass sich Softwareanbieter bereits darauf spezialisiert haben, eigene Produkte zur Umsetzung dieses Erlösmodells zu fertigen. Ein *AdServer* bietet als Softwareanwendung umfangreiche Dienste an, die diese Anforderungen abdecken. Deshalb wird im Rahmen dieser Arbeit eine Referenzarchitektur eines solchen AdServers wiedergegeben, jedoch wird in der Praxis eher ein bestehendes System gekauft als ein neues System entwickelt werden.

Dennoch sollen hier die grundsätzlichen Anforderungen an eine Softwarearchitektur eines AdServers kurz wiedergegeben werden, wie sie sich als Resultat der Klassifikation ergeben:

- die Werbebanner müssen verwaltet werden können
- die Einbindung der Banner in die Website muss gewährleistet werden
- das Profiling des Kunden muss während einer Sitzung erfolgen und ausgewertet werden
- es müssen Regeln definiert und angewandt werden können, um einem Kunden eine Werbung entsprechend seiner Zielgruppe anzeigen zu können
- die AdImpressions sowie die AdClicks müssen protokolliert und gegen die Kampagnen abgerechnet werden können
- die Auftraggeber müssen Kampagnen buchen können

17. betrachtete Qellen am 10.03.05: Tomorrow Focus AG (http://sales.tomorrow-focus.de) , Frankfurter Allgemeine Zeitung Online (www.faz.net), Westdeutsche Allgemeine (www.waz.de)
18. Die Website www.leo.org verlangte beispielsweise im März 2005 je AdClick 0,45€ (Quelle:http://www.leo.org/werbung/werbung_preise_de.html, abgerufen am 01.03.05).

Im Rahmen der Herleitung der Softwarearchitektur wird später in dieser Arbeit in dem zugehörigen Abschnitt noch einmal detaillierter auf diese Anforderung und ihre Umsetzung eingegangen.

4.3 Schlussfolgerung

In diesem Kapitel wurden zunächst Klassifikationskriterien identifiziert, anhand derer Erlösmodelle beschrieben werden können. Diese Kriterien wurden in vier Kategorien unterteilt, die den Bestandteilen eines Geschäftsmodells gemäß der Definition von Timmers entsprechen. Auf diese Weise wird gezeigt, inwiefern ein Erlösmodell ausgehend von einem Geschäftsmodell charakterisiert werden kann. Es wird also berücksichtigt, dass ein Erlösmodell in der Praxis immer ein *umgebendes* Geschäftsmodell aufweist. Diese Berücksichtigung der Abhängigkeit ist eines der Teilziele dieser Arbeit, wie einleitend in Abschnitt 1.1 erläutert wurde.

In einem weiteren Schritt wurden diese Kriterien dazu verwendet, sechs Erlösmodelltypen mit insgesamt 17 Erlösmodellsubtypen zu klassifizieren und detailliert zu beschreiben. Die Identifikation dieser Modelltypen baute dabei auf der Liste der zusammengetragenen Erlösmodelle als Ergebnis der Literaturrecherche des vorangegangenen Kapitels auf.

Die folgende Tabelle 4-1 zeigt eine Gegenüberstellung der hier behandelten Erlösmodelle mit den Ergebnissen der Literaturrecherche.

Erlös-modelle	Timmers	Mahadevan	Buchholz	Durbosson, Osterw., Pigneur	Amit / Zott	Zerdick et al. (Wirtz)	Skiera / Lambrecht (Bartelt / Lamersdorf)
Einzeltransaktion	Verkauf von Produkten und Dienstleistungen			Verkauf von Produkten		Einzeltransaktion	Produkte
Subskription	Mitglieds-beiträge	Mitglieds-gebühren	Subskriptions-gebühr	Subskriptions-gebühr	Subskriptions-gebühr	Abonnement, sonstige Grund-gebühren	Produkte
Transaktions-gebühr	Transaktions-gebühren	Transaktions-gebühr von Anbietern und Käufern	Transaktions-gebühr	Transaktions-anteile	Transaktions-gebühr		Dienstleistung (Teil von Produkten)
Profilhandel			Informations-verkauf			Datamining	Informationen
Provision				Provision	Vermittlungsgebühren, Provision	Kommission	Kontakte
Werbung	Werbung	Werbung	Werbung	Werbung, Sponsoring	Werbung	Werbung	Kontakte
Nicht behandelt			Gain sharing, Lizensierung der Technologie, Gebühren für Consulting	Einnahmeaufteilung (revenue sharing)		Lizenz-, Anschluss- und Rundfunk-gebühren, spezielle Empfangs-geräte, sonstige Formen, Subventionen	

Tabelle 4-1: Gegenüberstellung der Erlösmodelle

Die Tabelle stellt die hier identifizierten Erlösmodelle den jeweiligen zugehörigen Modellen der Autoren gegenüber. Sie dient dazu, eine Betrachtung der Vollständigkeit der in dieser Arbeit verwendeten Erlösmodelle durchzuführen.

Zunächst ist ersichtlich, dass die von Timmers und Mahadevan genannten MITGLIEDSBEITRÄGE dem hier identifizierten Erlösmodell der SUBSKRIPTION zugeordnet wurden. Beide Autoren geben Communities als Geschäftsmodelle an, die sich über Mitgliedsbeiträge finanzieren können. Letztendlich verbirgt sich also hinter einer Mitgliedsgebühr eine wiederkehrende Zugangsgebühr. Der wiederkehrende Charakter entspricht einer Subskription, weshalb hier diese Zuteilung vorgenommen wurde. Die in Abschnitt 4.2.2 behandelten Varianten des Erlösmodells SUBSKRIPTION können sehr einfach an dieses Modell angepasst werden, indem der Bezug der Leistungen oder Produkte nicht protokolliert werden muss.

Eine weitere Auffälligkeit ist die Abdeckung der Erlösmodelle nach Skiera und Lambrecht. Ihre drei Quellen der Erlöse sind Produkte, Informationen und Kontakte, wobei Produkte sowohl Güter als auch Dienstleistungen umfassen. Da der in dieser Arbeit gewählte Ansatz zur Klassifikation der Erlösmodelle eher auf die *Erlösherkunft* ausgerichtet wurde, werden die drei Erlösmodelle beider Autoren entsprechend mehrfach zugeteilt. EINZELTRANSAKTION und SUBSKRIPTION beziehen sich sowohl auf Güter als auch auf Dienstleistungen, TRANSAKTIONSGEBÜHREN werden für Dienstleistungen (z.B. Bereitstellung der Marktplattform) entrichtet, Informationen entsprechen dem PROFILHANDEL und die Kontakte können entweder in Form von Werbebannern (WERBUNG) oder Vermittlungsprovisionen (PROVISION) auftreten.

Als ein erstes Resultat kann demnach festgehalten werden, dass die in dieser Arbeit klassifizierten Erlösmodelle auch in den Arbeiten anderer Autoren in gleicher oder vergleichbarer Form aufgelistet wurden. Darüber hinaus lieferten diese Autoren jedoch auch solche Modelle, die hier keine Berücksichtigung fanden. Diese Modelle werden nun noch einmal detaillierter betrachtet.

Buchholz definiert GAIN SHARING (Betreiber einer Plattform erhält Anteil der Einsparungen von Käufern oder Verkäufern), LIZENSIERUNG DER TECHNOLOGIE (durch Bereitstellung der Plattform für andere Betreiber) sowie BERATUNGSGEBÜHREN als Erlösmodelle von Business-to-Business Marktplätzen. Das GAIN SHARING kann als eine sehr spezielle Sonderform des hier identifizierten Erlösmodells TRANSAKTIONSGEBÜHREN angesehen werden, dass sich nicht direkt am Transaktionswert, sondern an den Einsparungen orientiert. Die Einsparungen ergeben sich jedoch aus dem Transaktionswert, so dass hier diese besondere Form nicht explizit mit berücksichtigt werden muss. Die LIZENSIERUNG DER TECHNOLOGIE ergibt sich durch eine gebührenpflichtige Bereitstellung und Wartung eines Softwaresystems. Es ist somit ein sehr allgemeines und weniger ein e-Business-spezifisches Erlösmodell und ist beispielsweise vergleichbar mit der Lizensierung einer Buchhaltungssoftware. Aufgrund dieses sehr allgemeingültigen Charakters wird dieses Erlösmodell hier nicht berücksichtigt. Ebenfalls werden BERATUNGSGEBÜHREN im Rahmen dieser Arbeit nicht als Erlösmodell des e-Business interpretiert und deshalb auch nicht weiter verfolgt.

Dubosson-Torbay, Osterwalder und Pigneur liefern die EINNAHMEAUFTEILUNG (revenue sharing) als zusätzliches Erlösmodell, das hier nicht berücksichtigt wird. Der Grund dafür liegt darin, dass die Aufteilung von Einnahmen lediglich einer Verteilungsregel für die erzielten Erlöse entspricht und kein eigenständiges Erlösmodell darstellt.

Zerdick et al. liefern eine ganze Reihe von Erlösmodellen, die im Rahmen dieser Arbeit unberücksichtigt bleiben: LIZENZ-, ABSCHLUSS- UND RUNDFUNKGEBÜHREN, SPEZIELLE EMPFANGSGERÄTE, SONSTIGE FORMEN und SUBVENTIONEN. Begründung für das Ignorieren dieser Erlösmodelle ist ihre explizite Ausrichtung auf den Mediensektor, den Zerdick et al. als Grundlage ihrer Untersuchungen gewählt haben. Lediglich die SUBVENTIONEN können als branchenübergreifend angesehen werden, allerdings wird hier der Charakter von Subventionen nicht als Erlösmodell, sondern als eine (zumeist staatliche, zeitlich begrenzte und *zusätzliche*) Finanzierungshilfe für ein Unternehmen angesehen.

Es kann zusammenfassend festgehalten werden, dass die hier nicht berücksichtigten Erlösmodelle der übrigen Autoren aus Sicht dieser Arbeit als Spezialfälle einzustufen sind. Aus diesem Grund wurden sie nicht in die Klassifikation der Erlösmodelle mit aufgenommen.

Grundsätzlich ist es darüber hinaus nicht möglich, sämtliche Formen potenzieller Erlösmodelle zu berücksichtigen, da sowohl die fortschreitende technische Entwicklung als auch andere Einflussfaktoren wie beispielsweise die Akzeptanz der Akteure, für besondere Angebote zu bezahlen, eine ständige Weiterentwicklung möglicher Erlösmodelle mit sich bringen wird. Der Gegenüberstellung der hier entwickelten Klassifikation mit den Auflistungen anderer Autoren zeigt jedoch, dass ein überwiegender Großteil der bisher identifizierten Erlösmodelle des e-Business berücksichtigt worden ist. Im weiteren Verlauf wird nun für diese Erlösmodelle jeweils eine geeignete Software-Referenzarchitektur hergeleitet.

5 Softwarearchitekturen im e-Business

In diesem Kapitel werden Grundlagen von Softwarearchitekturen behandelt. Dazu werden Softwarearchitekturen anhand existierender Literatur analysiert und diskutiert. Wesentliche Aspekte wie Definitionen, Sichten, Stile und Beschreibungssprachen von Softwarearchitekturen werden dabei untersucht. Darüber hinaus ist es Ziel des Kapitels, die für diese Arbeit geeignetsten Ansätze auszuwählen, um Softwarearchitekturen entsprechend des Vorhabens der Arbeit zu beschreiben und darzustellen. Aus der Vielzahl der Gesichtspunkte sind diejenigen zu identifizieren, die sich für den Einsatz im e-Business am besten eignen.

Softwarearchitekturen werden unter anderem dazu verwendet, im Rahmen eines Softwareentwicklungsprozesses bereits frühzeitig als Kommunikationsinstrument zu dienen und Designentscheidungen festzulegen (in Abschnitt 5.3 wird ausführlicher auf die Gründe für den Einsatz von Softwarearchitekturen eingegangen). Sie stellen somit nur ein Mittel dar, den Weg bis zum eigentlichen Ziel zu begleiten: Die Realisierung einer Softwareanwendung.

Um die Entscheidung treffen zu können, welche Beschreibungs- und Darstellungsmöglichkeiten einer Softwarearchitektur optimal zu verwenden sind, sind aus diesem Grund zunächst die Charakteristiken der aus ihr abzuleitenden Softwareanwendungen zu untersuchen. Da sich diese Arbeit mit Softwarearchitekturen speziell im e-Business befasst, wird zunächst auf die Anforderungen an Softwareanwendungen in diesem Umfeld eingegangen.

5.1 Anforderungen an Softwareanwendungen im e-Business

Einige Charakteristiken des e-Business beeinflussen die Gestaltung der Softwareanwendungen. Ein wesentliches Merkmal des e-Business ist die Dynamik der technologischen Veränderungen. Wie Österle [Öst99] bemerkt, sind Veränderungen an sich nicht neu, allerdings ist die Geschwindikgeit und die Intensität der stattfindenden Veränderungen und ihrer Auswirkungen im e-Business außergewöhnlich. Speziell im Bereich des e-Business spielen Aspekte wie der flexible Einsatz von Systemen und der Anspruch auf immer kürzere Produktzyklen eine sehr wichtige Rolle (vgl. [JH98], [Wes99]).

Eine Kombination dieser Charakteristiken führt zu der Forderung nach einer hohen Flexibilität der Unternehmen. Dieses ist sowohl organisatorisch als auch technologisch zu erzielen. Während die organisatorische Flexibilität auf Ebene der Geschäftsmodelle zu realisieren ist, muss die technologische Flexibilität mit der Softwareanwendung umzusetzen sein. Die technologische Felxibilität bezieht sich dabei sowohl auf die Anpassbarkeit an ein sich veränderndes Umfeld, wie sie z.B. Shankaranarayan, Balasubramanian und Chen [SBC00] fordern, als auch auf die Möglichkeit, die Softwareanwendung auf verschiedene Standorte und Rechner zu verteilen [BW98].

Während Flexibilität die offensichtlichste Anforderung an Softwareanwendungen im e-Business ist, werden im Folgenden weitere Anforderungen aufgezeigt. Die folgende Liste basiert auf eigenen Überlegungen sowie auf Arbeiten von Hoque [Hoq00], der sich ebenfalls mit Softwareanwendungen und Architekturen im e-Business beschäftigt hat.

Die Anforderungen an Softwareanwendungen im e-Business sind zusammengefasst:

- Flexibilität

  Softwareanwendungen müssen in der Lage sein, auf die rasant wechselnden Anforderungen schnell anpassbar zu sein. Neue Standards müssen berücksichtigt und integriert werden, sobald sie sich etablieren. Einzelne Elemente sind auszutauschen oder zu verändern, ohne dass das Gesamtsystem anzufassen ist. Die Anwendung muss also dafür Sorge tragen, dass die einzelnen Komponenten nicht starr fixiert, sondern lose miteinander verbunden sind und miteinander kommunizieren, so dass einzelne Komponenten mit geringem Aufwand ausgetauscht oder modifiziert werden können. Hoque [Hoq00] bezeichnet diese Anforderung mit *Agility*.

- Erweiterbarkeit

  Diese Anforderung lässt sich ebenfalls auf den beschleunigten Wandel der Rahmenbedingungen im e-Business zurückführen. In einem Umfeld, in dem relativ kurzfristig neue technologische oder fachliche Veränderungen auftreten, müssen Softwareanwendungen derart gestaltet sein, dass sie erweiterbar und anpassbar sind. Hoque [Hoq00] greift mit seinem Begriff der *Cycle Time* in etwa diesen Gedanken auf, indem er die Applikationsentwicklung als einen kontinuierlichen Prozess versteht.

- Skalierbarkeit

  Ein wesentliches Problem von Softwareanwendungen im e-Business ist die in der Planungsphase nicht exakt zu definierende Anzahl der Nutzer sowie die Verteilung der Zugriffe. Die Anwendungen müssen aus diesem Grund in der Lage sein, zügig auf steigende Lasten zu reagieren. Während ein Aufrüsten in der Regel durch den Einsatz zusätzlicher Hardware-Komponenten wie Applikations- oder Datenbankserver erfolgt, müssen die Softwareanwendungen in der Lage sein, dass diese Aufrüstung zu einer tatsächlichen Erhöhung der Kapazitäten durch eine Lastverteilung führt.

- Wiederverwendbarkeit

  Die Forderung nach Wiederverwendbarkeit umfasst zwei Aspekte. Zum einen sollen einzelne Komponenten nur einmalig innerhalb der Anwendung auftreten, so dass keine redundante Haltung der Funktionalitäten vorliegt, die nur aufwändig zu warten ist. Zum anderen sollen bei der Entwicklung neuer Funktionalitäten bestehende Elemente wiederverwendet, abgewandelt oder erweitert werden.

- Verteilbarkeit

  e-Business zeichnet sich unter anderem durch eine ortsübergreifende Zusammenarbeit aus. Einzelne Systeme können somit nicht mehr nur zentral an einem Ort lokalisiert sein, sondern werden dezentral auf mehreren Rechnern verteilt gehalten.

- Schnittstellenunterstützung

  Die Anwendung muss in der Lage sein, mit anderen externen Applikationen in einem Gesamtsystem zu interagieren. Dies beinhaltet zwangsläufig die Forderung nach einer Unterstützung von standardisierten Schnittstellen, über die die einzelnen Elemente miteinander kommunizieren. Hoque [Hoq00] bezeichnet diese Forderung mit *Interoperability*.

- Wartbarkeit

  Die ersten e-Business Anwendungen wurden in einem sehr pragmatischen Ansatz durch eine Verflechtung der Darstellung und Ablauflogik realisiert. Häufig wurde über das Common Gateway Interface (CGI, vgl.[CR99]) Ablauflogik umgesetzt, die je nach Anfrage durch den Client eine aufbereitete Präsentation in Form von HTML-Seiten zurücklieferte. Diese enge Verflechtung der unterschiedlichen Ebenenen führte zu einem schwer durchschaubaren System, dass nur aufwendig zu pflegen und warten war [Wei02].
  Um dies zu vermeiden, müssen Softwareanwendungen derart gestaltet sein, dass durch eine saubere Trennung der Teilaufgaben ihre Wartbarkeit gewährleistet ist. Die Anwendung muss eine Verteilung der einzelnen Komponenten in mehrere Schichten zulassen und unterstützen.

- Performanz

  Der Aspekt der Performanz wurde bereits implizit bei der Skalierbarkeit von e-Business Anwendungen angesprochen. Da das Zugriffsverhalten auf eine Anwendung im e-Business sehr stark schwanken und nur unzureichend vorherbestimmt werden kann, muss sie darauf ausgelegt sein, auch bei hoher Last ausreichend performant zu operieren.

- Zuständigkeit

  Hoque [Hoq00] führt den Aspekt der *Ownership* auf, mit dem er die Forderung verbindet, dass klar defniert sein muss, wer für welche Subsysteme zuständig ist. Dies ist umso wichtiger, als dass Softwareanwendungen im e-Business auch unternehmensübergreifende Teilsysteme integrieren können. Obwohl diese Anforderung im Rahmen der Arbeit vernachlässigt wird, zeigt auch sie die Notwendigkeit von Anwendungen, die einzelne, klar zuzuordnende Elemente aufweisen.

Diese Zusammenfassung der Anforderungen an Softwareanwendungen im e-Business zeigt, dass vor allem eine Modularität der Anwendungen eine wesentliche Rolle spielt. Diese Modularität wird durch den Begriff der Komponente veranschaulicht, der für nahezu alle Anforderungen eine Bedeutung darstellt.

Im nächsten Abschnitt wird durch das Aufzeigen der Eigenschaften von *komponentenbasierten* Softwareanwendungen deutlich, wie sehr die hier aufgelisteten Anforderungen mit diesem Ansatz erfüllt werden. Auf diese Weise wird belegt, dass komponentenbasierte Softwareanwendungen geeignet sind, um Geschäftsmodelle im e-Business technologisch zu realisieren. In den nächsten Abschnitten wird anschließend untersucht, welche Softwarearchitekturen verwendet werden sollen, um den Entwicklungsprozess solcher komponentenbasierter Anwendungen bestmöglich zu unterstützen.

5.2 Komponentenbasierte Softwareanwendungen

Komponentenbasierte Softwareanwendungen basieren auf einer konzeptionellen Weiterentwicklung der Objektorientierung. Die objektorientierte Softwareentwicklung verfolgt das Ziel, durch die Wiederverwendung von Software die Entwicklung von Anwendungen zu erleichtern und zu beschleunigen, was vor allem vor dem Hintergrund der kürzer werdenden Entwicklungszeiten notwendig erscheint. In der Praxis haben sich jedoch die Anforderungen an eine hohe Wiederverwendbarkeit der Objekte nicht erfüllen können, so dass weiterführende Ansätze verfolgt wurden [PS96]. Um den wachsenden Anforderungen an Zeit, Kosten und Qualität der Softwareentwicklung gerecht zu werden, wurde das Thema der *komponentenbasierten* Softwareentwicklung als Weiterführung der Objekttechnologie aufgebracht und diskutiert.

Die komponentenbasierte Softwareentwicklung[1] (Component-Based Software Development, kurz CBSD oder nur CBD) ist ein Softwareengineering Ansatz, der die Wiederverwendung von Softwarekomponenten als oberstes Ziel hat, wodurch die gewünschte hohe Flexibilität und Qualität bei gleichzeitiger Kostensenkung in der Entwicklung erwartet wird [Gri98]. Den Entwicklern stehen mit diesem Ansatz Methoden zur Verfügung, um Anwendungen in sinnvolle Bausteine zu zerlegen. Neue Systeme können nach dem Baukastenprinzip aus bereits vorhandenen, neu entwickelten oder zugekauften Komponenten zusammengesetzt werden [GT00].

Die Vorteile durch den Ansatz einer komponentenbasierten Entwicklung werden durch Clements folgendermaßen zusammengefasst [Cle96b]:

1. Häufig wird in der Literatur der Begriff der Softwareentwicklung und nicht der Softwareanwendung verwendet. Eine klare Abgrenzung wird in diesem Abschnitt vernachlässigt, da sich auch die Softwareentwicklung mit dem Ziel einer Softwareanwendung befasst und lediglich der Fokus mehr auf dem Prozess als auf dem Produkt liegt. Die grundsätzlichen Konzepte und Vorteile von Komponenten treten jedoch in beiden Ansätzen auf.

- Verkürzte Entwicklungszeiten. Durch die Möglichkeit existierende Komponenten in neue Anwendungen zu integrieren, wird die Entwicklungszeit erheblich reduziert, was letztendlich eine Senkung der Entwicklungskosten zur Folge hat. Oft benötigte Basiskomponenten müssen nur einmal entwickelt oder beschafft werden und es kann eine Fokussierung auf die zu realisierende Zielanwendung erfolgen.
- Verbesserte Qualität und Zuverlässigkeit. Eine bereits eingesetzte und getestete Komponente besitzt in der Regel weniger Fehler und Unzulänglichkeiten. Dies führt zu einer Steigerung der Qualität und Zuverlässigkeit des Gesamtsystems.
- Höhere Flexibilität. Durch die Eigenschaft einer Komponente, ihre eigentliche Implementierung transparent zu gestalten, besteht die Möglichkeit, diese gegen eine andere Komponente, welche die gleichen Anforderungen erfüllt, auszutauschen. Dies erhöht die Flexibilität und Wartungsfähigkeit eines Systems zur Entwicklungs- und Betriebszeit.

Die Vorteile decken sich mit den zuvor identifizierten Anforderungen an Softwareanwendungen im e-Business. Neben der Flexibilität werden direkt die Wiederverwendbarkeit und Wartbarkeit angesprochen.

Nachfolgend wird der Begriff der Komponente exakter definiert.

5.2.1 Der Komponentenbegriff

Der Begriff der Komponente wird in der Literatur variantenreich interpretiert. Eine Gegenüberstellung verschiedener Definitionen ist in [Szy98] und [Gri98] zu finden. Rautenstrauch [Rau99] stellt in seiner Analyse zahlreicher Definitionen fest, dass in nahezu allen Auffassungen die Elemente

- Unabhängigkeit (von bestimmten Anwendungen),
- Abgeschlossenheit (abgeschlossene Funktionalität) und
- Offenheit (Fähigkeit zum Interagieren, über Schnittstellen)

vorkommen. Eine häufig zitierte Definition findet sich in [OHE96], die im Rahmen dieser Arbeit als zugrundeliegende Definition dienen soll.

Definition 8: Komponente

Eine Komponente ist ein Stück Software, das klein genug ist, um es in einem Stück zu erzeugen und pflegen zu können, groß genug ist, um eine sinnvolle Funktionalität zu bieten und eine individuelle Unterstützung zu rechtfertigen und das mit standardisierten Schnittstellen ausgestattet ist, um mit anderen Komponenten zusammenzuarbeiten.

Eine Komponente realisiert in der Regel eine abgeschlossene Funktionalität, die sie in Form von Services anderen Komponenten in beliebigen Umgebungen über Schnittstellen zur Verfügung stellt. Nach Denninger und Peters [DP00] ist vor allem das Konzept der Schnittstelle von zentraler Bedeutung für das Komponentenparadigma. Sie stellt demnach den Vertrag dar, den sich die Komponente verpflichtet zu erfüllen. Da unter einer Komponente ein zumeist statisches Strukturierungskonstrukt für Softwaremodule verstanden wird

[Gri98], umfasst die Komponentenbildung die Gliederung eines Softwaresystems in einzelne, funktional zusammengehörende Module und die Festlegung der Schnittstellen zwischen diesen Komponenten bzw. Softwaremodulen.

Eine zusammenfassende Übersicht über die Charakteristiken von Komponenten sind in [GS02] und [GT00] aufgeführt. Auf ihrer Basis werden hier diese Merkmale vorgestellt:

- Trennung von Schnittstelle und Implementierung

  Diese Forderung wurde bereits angesprochen. Komponenten stellen ihre Services ausschließlich über Schnittstellen der Außenwelt zur Verfügung. Die Realisierung der Services bleibt dagegen die interne Aufgabe der Komponente und ist nach außen hin nicht sichtbar und manipulierbar. Auf diese Weise ist sichergestellt, dass eine Komponente modifiziert werden kann, ohne dass die Schnittstelle angepasst wird. Dies setzt natürlich voraus, dass der Service weiterhin das zuvor fest definierte Ergebnis liefert.

- Integrations- und Kompositionsfähigkeit

  Es wurde bereits angesprochen, dass sich Systeme modular aus Komponenten zusammensetzen sollen. Darüber hinaus wird auch gefordert, dass Komponenten selber ebenfalls aus mehreren (anderen) Komponenten zusammengesetzt, also komponiert werden können. Diese Komposition soll nicht nur statisch fix erfolgen, sondern auch zur Laufzeit dynamisch erfolgen.

- Wohldefinierter Zweck

  Dieser Aspekt lässt sich aus dem Definitionsbestandteil der abgeschlossenen Funktionalität ableiten. Komponentenbasierte Softwareentwicklung als Weiterentwicklung der Objektorientierung wurde unter anderem deshalb motiviert, da das Abstraktionsniveau zu steigern war. Eine Komponente soll somit ein höheres Abstraktionsniveau aufweisen als ein einzelnes Objekt. Auf diese Weise wird die Definition und Berücksichtigung der fachlichen Anforderungen erleichtert. Aus diesem Grund ist es notwendig, dass eine Komponente einen eindeutigen, wohldefinierten Zweck erfüllt. Entsprechend der Definition ist zu gewährleisten, dass dieser Zweck nicht zu grobgranular bestimmt wird, so dass eine Komponente keine eigenständige Anwendung sein kann. Eine Komponente ist allein nicht „lebensfähig“.

- Kontextunabhängigkeit

  Die Kontextunabhängigkeit als Forderung an eine Komponente zeigt den Unterschied zwischen Komponente und Komponentenmodell auf. Die Beschreibung von Komponenten erfolgt losgelöst von technologischen Rahmenbedingungen, also unabhängig vom Kontext der Ablaufumgebung. Die Schnittstellen geben Auskunft darüber, wozu ein Dienst einer Komponente in Anspruch genommen werden kann. Die tatsächliche (=technische) Umsetzung innerhalb eines Komponentenmodells wird somit an dieser Stelle nicht berücksichtigt.

- Portabilität und Programmiersprachenunabhängigkeit

  Komponenten sollen grundsätzlich in beliebigen Programmiersprachen implementiert werden können. Entscheidend ist nur, dass eine realisierte Komponente plattformübergreifend ohne Portierung einsetzbar sein soll. Dieser Aspekt der Unabhängigkeit in Bezug auf Programmiersprachen ist für Orfali, Harkey und Edwards [OHE96] eine der Besonderheiten von Komponenten im Gegensatz zu Objekten.

- Ortstransparenz

  Dieser Aspekt ist eng verbunden mit der bereits geforderten Kontextunabhängigkeit von Komponenten. Die Nutzung der Services einer Komponente soll unabhängig davon möglich sein, wo sich diese befindet. Der Benutzer muss sich also nicht darum kümmern, ob die Komponente innerhalb der Prozessumgebung, auf dem lokalen Rechner oder im Netzwerk zur Verfügung steht.

- Selbstbeschreibungsfähigkeit

  Um die Forderung der Wiederverwendbarkeit zu unterstützen, ist eine Selbstbeschreibungsfähigkeit der Komponenten erforderlich. Das heißt, das die Komponente zur Laufzeit nach außen hin Informationen über ihre Schnittstellen und ihr Verhalten geben kann. Die Mindestforderung ist die Auskunftbereitschaft über die Attribute und Methoden einer Komponente.

- Sofortige Einsatzbereitschaft

  Im Sinne des plug&play sollen Komponenten unmittelbar nach ihrer Verfügbarkeit auch installiert und einsatzfähig sein. Um dies zu verwirklichen, ist eine Selbstregistrierung der Komponente innerhalb der Ablaufumgebung erforderlich.

- Wiederverwendbarkeit

  Dieser Aspekt wurde bereits bei der Entwicklung der Objektorientierung verfolgt und gilt somit auch für Komponenten. Unterschiedlich ist allerdings, dass eine Wiederverwendbarkeit der anwendungsnahen Bausteine aufgrund des höheren Abstraktionsniveaus eher auf Anwendungsebene erfolgt. Es werden also eher fachliche als technische Elemente wiederverwendet, was einen wesentlichen Fortschritt in der Beschleunigung und Vereinfachung der Anwendungsentwicklung darstellt.

- Konfigurierbarkeit, Anpassbarkeit

  Um die Langlebigkeit der Komponenten zu erhöhen und die Wartung von Systemen zu vereinfachen, sollten Komponenten über Parameter konfigurierbar sein. Eine Komponente zur Validierung von Adressen sollte in der Lage sein, über eine Parametrisierung auf eine neues PLZ-System angepasst zu werden, um eine schnelle und einfache Aktualisierung zu gewährleisten.

- Bewährtheit

  Die Zuverlässigkeit von Komponenten sollte vor ihrer Verwendung ausgiebig untersucht und getestet werden. Sie sollten auf diese Weise bereits ihre Bewährtheit unter Beweis gestellt haben.

- Binärcode-Verfügbarkeit

  Diese letzte Anforderung an Komponenten besagt, dass sie in ausführbarer, auf verschiedenen Plattformen lauffähiger Form bereitstehen sollten. Durch das Binärformat soll ihre Portabilität und Programmiersprachenunabhängigkeit gewährleistet werden (vgl. auch [Gri98]).

Um gemäß dieser Anforderungen reale Komponenten entwerfen und entwickeln zu können, haben sich verschiedene Standards etabliert. Diese Standards werden als Komponentenmodelle bezeichnet. Beispiele sind *DCOM*, *CORBA*, *JavaBeans*, *Enterprise JavaBeans* oder *.NET*. Ein Vergleich der praktischen Umsetzungen dieser Komponentenmodelle zeigt, dass keines von ihnen sämtliche Anforderungen an Komponenten gemäß der eben vorgestellten Liste vollständig unterstützt (vgl. [GS02], [Ost04]). Insbesondere in Bezug auf die Programmiersprachenunabhängigkeit und Portabilität sind technologische Restriktionen einzugestehen. Unabhängig davon sind aber die aufgeführten Anforderungen ein Maßstab für die grundsätzliche Gestaltung von Komponenten.

5.2.2 Fazit

Zusammenfassend lässt sich sagen, dass eine weitgehende Übereinstimmung zwischen den Anforderungen an Softwareanwendungen im e-Business (vgl. Abschnitt 5.1) einerseits und den Charakteristiken von Komponenten andererseits zu erkennen ist. Die Forderung nach Flexibilität der Anwendungen wird durch die Integrations- und Kompositionsfähigkeit der Komponenten gewährleistet. Die selbe Eigenschaft der Komponenten gewährleistet darüber hinaus die Erweiterbarkeit von Softwareanwendungen. Die Ortstransparenz und die Kontextunabhängigkeit der Komponenten erfüllen die Anforderung einer verteilbaren Anwendung. Die Wiederverwendbarkeit von Komponenten erfüllt direkt die gleiche Anforderung an die Softwareanwendung, die sich aus Komponenten zusammensetzt. Diese Beispiele zeigen, dass der komponentenbasierte Ansatz für Softwareanwendungen im e-Business optimal geeignet ist.

Wie einleitend erwähnt, soll in diesem Kapitel untersucht werden, welche Beschreibungs- und Darstellungsmöglichkeiten einer Softwarearchitektur optimal zu verwenden sind. Nachdem nun gezeigt wurde, dass komponentenbasierte Softwareanwendungen ideal geeignet sind, den Anforderungen an Softwareanwendungen im e-Business gerecht zu werden, wird nun vor diesem Hintergrund untersucht, welche der zahlreichen Ansätze von Softwarearchitekturen zu deren Entwicklung eingesetzt werden können. Dazu werden zunächst die unterschiedlichen Aspekte von Softwarearchitekturen vorgestellt und anhand existierender Literatur behandelt.

5.3 Softwarearchitekturen

In den folgenden Abschnitten werden Aspekte von Softwarearchitekturen behandelt. Abbildung 5.3.1 definiert zunächst den Begriff und erläutert die Gründe für den Einsatz von Softwarearchitekturen.

Abschnitt 5.3.2 behandelt *Sichten* von Softwarearchitekturen. Kruchten [Kru95] weist darauf hin, dass Aspekte wie die Darstellung von dynamischen Strukturen und Interaktionen eine Differenzierung in verschiedene Sichten notwendig machen. Abhängig von der betrachteten Perspektive werden verschiedene Merkmale dargestellt, aufgrund derer es notwendig ist, unterschiedliche Sichtweisen zu identifizieren und zu beschreiben.

Abschnitt 5.3.3 befasst sich mit verschiedenen *Architekturstilen.* Ein Architekturstil legt fest, wie die enthaltenen Komponenten und ihre Interaktionen mitsamt dem Nachrichtenaustausch angeordnet und dargestellt werden sollen.

Ein weiterer wichtiger Aspekt ist die *Beschreibungssprache* von Softwarearchitekturen, womit sich Abschnitt 5.3.4 befasst. Üblicherweise ähneln viele Architekturbeschreibungen einem einfachen Diagramm mit einer Anordnung von Rechtecken und verbindenden Linien (engl. Box-and-Lines Diagrams). Um die Semantik der verwendeten Elemente festzulegen, wurden eine Reihe von Architekturbeschreibungssprachen (engl. architecture definition language, Abk. ADL) entwickelt. Diese unterstützen die verschiedenen Aspekte der architektonischen Entwurfsphase unterschiedlich gut, da den einzelnen Sprachen oft eine voneinander abweichende Sichtweise auf die Softwarearchitektur zugrunde liegt [HR97]. In dem Abschnitt wird untersucht, welche Beschreibungssprache für die vorliegende Arbeit verwendet werden soll.

5.3.1 Definition und Zweck von Softwarearchitekturen

Der Entwurf von Softwarearchitekturen kann allgemein als die Lehre von der Konstruktion großer Softwaresysteme verstanden werden. Diese Konstruktion beinhaltet eine Beschreibung des zu realisierenden Systems aus interagierenden Komponenten [HR97]. Für den Begriff der Softwarearchitektur existiert keine einheitlich verwendete Definition und es herrscht in der Literatur kein Mangel an vorgeschlagenen Begriffsbildungen, die aufgrund verschiedener Sichtweisen vorliegen [CN96].

Garlan und Shaw [GS93] betrachten in ihrer viel beachteten Arbeit die Architektur eines Softwaresystems als: "*...a collection of computational components - or simply components - together with a description of the interactions between these components - the connectors.*"

Die wesentlichen Bestandteile einer Architektur sind demnach also Komponenten und eine Beschreibung ihrer Interaktionen. Diese Definition deutet somit bereits an, dass Softwarearchitekturen nicht nur statische, sondern auch dynamische Aspekte in Form von Interaktionen berücksichtigen. Noch deutlicher wird diese Sichtweise in einem ebenfalls viel beachteten Buch von Bass, Clements und Kazman herausgestellt. Dort findet man für den Begriff der Softwarearchitektur die folgende Definition [BCK97]:

Definition 9: Softwarearchitektur

The software architecture of a program or computing system is the structure or structures of the system, which comprise software components, the externally visible properties of those components, and the relationships among them.

Diese Definition beeinhaltet einen wesentlichen Aspekt: Es wird explizit darauf hingewiesen, dass eine Architektur nicht aus einer, sondern aus mehreren Strukturen bestehen kann. Es gibt folglich nicht notwendigerweise nur eine Beschreibung des Systems, sondern zahlreiche. Jede dieser Beschreibungen (Strukturen) kann dabei je nach Sichtweise auf bestimmte Schwerpunkte focussieren.

Die Autoren leiten direkt aus ihrer Definition die Anforderung ab, dass eine Beschreibung des *Verhaltens* jeder Komponente Bestandteil der Architektur ist[2]. Somit fordern sie neben einer Beschreibung der statischen Elemente (Komponenten, Schnittstellen) zusätzlich auch eine Darstellung des dynamischen Verhaltens, wodurch die in der Definition erwähnten Struktur*en* verständlich werden.

Auf diese Tatsache, dass die Architektur komplexer Systeme[3] nicht nur eine, sondern mehrere Beschreibungen umfasst, geht auch Zachman bereits in einer sehr frühen Arbeit zu diesen Thema ein: *There is not an information systems architecture, but a set of them!* (aus [Zac87]). Er fügt hinzu, dass es verschiedene Perspektiven von verschiedenen beteiligten Personen auf das System gibt und jeweils geeignete Beschreibungen (*representations*) einzusetzten sind. Es bleibt also an dieser Stelle festzuhalten, dass sich eine Softwarearchitektur nicht aus einer, sondern aus mehreren Beschreibungen zusammensetzt.

Nachdem der Begriff der Softwarearchitektur für diese Arbeit definiert wurde, wird nun der Begriff der (Software-)Referenzarchitektur genauer untersucht, da diese das Ziel der vorliegenden Arbeit sind. Auch hierzu liefern Bass, Clements und Kazman den entscheidenden Ansatz. Sie definieren zunächst den Begriff des Referenzmodells (aus [BCK97]):

Definition 10: Referenzmodell

A reference model is a division of functionality together with data flow between the pieces. A reference model is a standard decomposition of a known problem into parts that cooperatively solve the problem. (...) [They] are often obtained by domain analysis (...).

Darauf aufbauend liefern die Autoren eine Definition von Referenzarchitekturen (aus [BCK97]):

Definition 11: (Software-)Referenzarchitektur

A reference architecture is a reference model mapped onto software components (...) and the data flows between these components.

2. [BCK97], Seite 24: „(...) the behavior of each component is part of the architecture (...)“
3. Zachman verwendet nicht explizit den Begriff der *Software*architektur, sondern behandelt 'information system architecture'; von dieser Unterscheidung kann hier jedoch abstrahiert werden

Es wird ersichtlich, inwiefern Referenzarchitekturen im Rahmen dieser Arbeit verwendet werden. Bezogen auf die *Domäne* der Erlösmodelle im e-Business werden die jeweiligen Charakteristika analysiert und die zur Realisierung eines Softwaresystems notwendigen Funktionalitäten ermittelt. Für diese Funktionalitäten werden Komponenten definiert und sowohl in ihrer statischen Struktur als auch in ihrem dynamischen Verhalten beschrieben. Diese Referenzarchitekturen stellen somit eine *Musterlösung* für das gegebene Problem in Form einer softwaretechnischen Umsetzung eines Erlösmodells im e-Business dar.

Warum werden jedoch Softwarearchitekturen grundsätzlich entwickelt und angewandt? Die Gründe für eine Softwareentwicklung auf Basis einer Architektur liegen nach Clements und Nothrop in den folgenden drei fundamentalen Vorteilen [CN96]:

- Basis für die Kommunikation

  Eine Softwarearchitektur stellt eine allgemeine und hohe Abstraktion eines Systems dar und kann somit für alle involvierten Interessensgruppen als Basis für eine gemeinsame Kommunikation wie beispielsweise das Festlegen von Konsensen benutzt werden.

- Frühzeitige Designentscheidungen

  Softwarearchitekturen erfordern eine Anzahl von frühzeitigen Designentscheidungen für ein System. Diese Entscheidungen zu treffen, ist ein äußerst kritischer Schritt, da sie sich nachhaltig und bedeutend auf die Weiterentwicklung und Wartung des Systems auswirken. Aus diesem Grund sind die einmal getroffenen Entschlüsse zu einem späteren Zeitpunkt der Systementwicklung nur schwer zu ändern. Beispielsweise legt eine Architektur für die Entwickler zahlreiche Richtlinien für die Implementierung fest oder erlaubt Aussagen über bestimmte Eigenschaften des zukünftigen Systems noch vor dessen endgültiger Implementierung.

- Wiederverwendbare Abstraktion eines Systems

  Eine Softwarearchitektur verkörpert ein relativ kleines und leicht zu erfassendes Modell der Struktur eines Systems und der Zusammenarbeit der enthaltenen Komponenten. Ein solches Modell kann auf andere Systeme mit ähnlichen Anforderungen übertragen werden und somit eine Wiederverwendung auf einer sehr hohen Ebene ermöglichen.

Hierbei ist vor allen Dingen der im dritten Punkt betrachtete Aspekt von hoher Bedeutung für diese Arbeit. Für jedes der im bisherigen Verlauf identifizierte Erlösmodell wird im Folgenden eine geeignete Softwarearchitektur erstellt, die jeweils als Referenz für diesen Anwendungsfall dient. Auf dieser Basis lassen sich Softwareysteme mit gleichen oder sehr ähnlichen Anforderungen einfacher und schneller realisieren, als wenn sie von Grund auf neu entworfen werden müssten. Für die Entwicklung von Softwarearchitekturen für Geschäftsmodelle des e-Business mit übereinstimmenden Erlösmodellen kann also auf die vorhandenen Referenzarchitekturen dieser Arbeit zurückgegriffen werden. Diese Vorgehensweise erhöht die Wirtschaftlichkeit und Effizienz von individuell entwickelten Software-

systemen in erheblichem Maße. Solche Architekturen werden in diversen wissenschaftlichen Arbeiten auch als Domänen-spezifische Softwarearchitekturen (engl. domain-specific software architectures, Abk. DSSA) bezeichnet.

In den folgenden Abschnitten wird nun zusätzlich untersucht, wie die zu entwerfenden Software-Referenzarchitekturen am geeignetsten darzustellen und zu beschreiben sind.

5.3.2 Sichten auf Softwarearchitekturen

Softwarearchitekturen dienen, wie bereits erwähnt, wesentlich als Kommunikationsinstrument. Da während eines Entwicklungsprozesses einer Softwarearchitektur über die gesamte Laufzeit unterschiedliche Personengruppen miteinander kommunizieren, müssen die Archtitekturen derart beschrieben werden können, dass sie allen Zielgruppen gerecht werden. Deshalb erscheint es sinnvoll, ein Modell zu benutzen, das verschiedene Sichten berücksichtigt. Die Sichten repräsentieren jeweils Teilaspekte einer Softwarearchitektur, die spezifische Eigenschaften des gesamten Systems darstellen. Dabei sind die einzelnen Perspektiven in der Regel für die beteiligten Interessensgruppen (z.B. Anwender, Entwickler, Projektmanager) unterschiedlich relevant [Kru95]. Die jeweiligen Sichten sind nicht vollkommen unabhängig und isoliert voneinander, sondern es können Elemente aus einer Sicht mit denen aus anderen Sichten korrelieren [CN96].

5.3.2.1 Sichten nach Kruchten

Eine sehr häufig zitierte Arbeit liefert Kruchten mit seiner Definition von fünf Hauptsichten [Kru95]. Dabei beschäftigen sich vier dieser Sichten direkt mit einer Beschreibung der Architektur und werden durch eine fünfte Darstellung von ausgesuchten Anwendungsszenarien (engl. Use Cases) ergänzt. Abschnitt 5-1 veranschaulicht diese Sichten zusammen mit ihren Verknüpfungen. Für jede dieser Sichten können unabhängig voneinander die benötigten Elemente (z.B. Komponenten, Verbindungen), die logischen Grundlagen und die Einschränkungen definiert werden, um eine eigene Notation für ihre Darstellung zu erhalten. Zusätzlich kann für jede Sichtweise ein eigener Architekturstil ausgewählt werden und in Folge dessen können in einem System mehrere solcher Stile koexistieren (siehe auch Abschnitt 5.3.3).

Die *Logische Sicht* unterstützt primär die Darstellung der funktionalen Anforderungen eines Systems in Form von Diensten, die zur Verfügung gestellt werden. Das System wird in Module (in der Regel Objekte oder Klassen), die eng an die spezifische Anwendungsdomäne gekoppelt sind, abstrahiert und auf Vererbung und Kapselung hin untersucht. Dieses Vorgehen dient nicht nur der funktionalen Analyse, sondern auch der Identifikation von gemeinsamen Mechanismen und Elementen in den einzelnen Teilen des Systems. Da der Sicht ein objektorientierter Architekturstil (siehe dazu Abschnitt 5.3.3) zugrunde liegt, werden für die grafische Repräsentation Klassendiagramme eingesetzt. Alternativ können beispielsweise bei vorwiegend datengesteuerten Anwendungen auch andere Notationen wie Entity-Relationship Diagramme verwendet werden.

Die *Prozess-Sicht* berücksichtigt nichtfunktionale Anforderungen wie Performanz und Verfügbarkeit. Der Fokus liegt also auf dem Laufzeitverhalten eines Systems und den damit verbundenen dynamischen Aspekten wie Kommunikationsmechanismen, Nebenläufigkeit oder Synchronisation. Die strukturellen Elemente sind dementsprechend einzelne Prozesse. Diese bestehen aus einer Sequenz von Instruktionen mit eigenem Kontrollfluss und können während der Systemausführung beispielsweise gestartet, gestoppt oder neu konfiguriert werden.

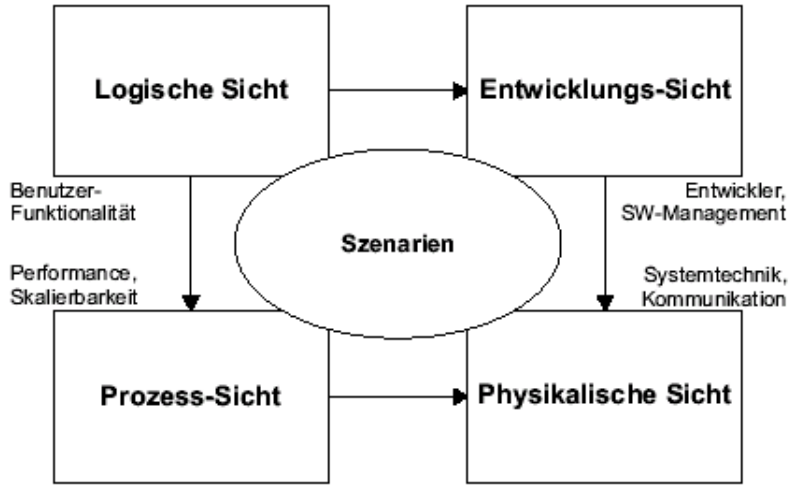


Abbildung 5-1: Sichten auf Softwarearchitekturen nach [Kru95]

Die *Entwicklungs-Sicht* legt den Schwerpunkt auf die Organisation der einzelnen Softwaremodule innerhalb der Entwicklungsumgebung und ist somit eng an die konkrete Implementierung gebunden. Die Software wird dabei in überschaubare Einheiten wie Programmbibliotheken oder Teilsysteme zerlegt und in der Regel in hierarchischen Schichten organisiert (vgl. Abschnitt 5.3.3). Die Entwicklungs-Architektur dient somit als Basis für die Verteilung von Anforderungen, Entwicklung, Wiederverwendung von Software und Aspekten der Portabilität und Sicherheit.

Die *Physikalische Sicht* stellt eine Abbildung der Software auf die entsprechende Hardware dar. Hierbei werden nichtfunktionale Anforderungen wie Fehlertoleranz, Skalierbarkeit oder Verfügbarkeit betrachtet. Software, die in einem Netzwerk von Computern ausgeführt wird, muss in verschiedene Prozesse partitioniert werden, um diese auf die verfügbaren Systeme zu verteilen. Die Abbildung auf die Hardware sollte möglichst flexibel sein und einen minimalen Einfluss auf den Programmcode haben, da die einzelnen physikalischen Systemkonfigurationen in hohem Maße voneinander abweichen können (z.B. in Entwicklungs- oder Produktionssystemen).

Alle vier Sichten repräsentieren unterschiedliche strukturelle Aspekte eines Systems. Die einzelnen Sichten sind dabei nicht unabhängig voneinander, sondern können durch verschiedene Elemente miteinander verbunden sein. *Szenarien*, die als fünfte Sicht bezeichnet werden, sind für die Darstellung dieser Zusammenhänge geeignet. Nicht jede

Anwendung benötigt eine Darstellung aller Sichten. Erfahrungen haben gezeigt, dass die Unterschiede zwischen den verschiedenen Sichten mit der Komplexität des Gesamtsystems zunehmen (vgl. [CN96]), so dass bei kleineren Systemen beispielsweise die logische und die Entwicklungs-Architektur kombiniert beschrieben werden können.

Für das Vorhaben dieser Arbeit erscheint vor allem die Logische Sicht relevant zu sein. Die darin verwendeten Klassendiagramme geben jedoch nur statische Aspekte der enthaltenen Komponenten wieder. Da sich die Prozess-Sicht mit nichtfunktionalen Anforderungen und nicht mit dem dynamischen Verhalten der Komponenten befasst und darüber hinaus auch keine andere explizite Sicht diese Aufgabe übernimmt, erscheint der Ansatz jedoch nicht ausreichend geeignet zu sein. Im folgenden wird deshalb eine weitere Arbeit bezüglich Sichten auf Architekturen untersucht.

5.3.2.2 Sichten nach Gruhn und Thiel

Bei Gruhn und Thiel liegt den einzelnen Sichtweisen auf eine Architektur die komponentenbasierte Softwareentwicklung von Geschäftsanwendungen zugrunde [GT00]. Die Softwarearchitektur besteht dabei aus den drei Teilen *fachliche Architektur*, *softwaretechnische Architektur* und *systemtechnische Architektur*.

Die *fachliche Architektur* beschreibt die fachliche Sicht auf die Softwarearchitektur sowie der in ihr enthaltenen Komponenten und legt somit die fachliche Basis des Systems fest. Da die Beschreibung des Systems primär aus der Sicht des Anwenders geschieht, werden Entwurfsentscheidungen oder Realisierungsdetails auf dieser Ebene nicht berücksichtigt. Die fachlichen Komponenten, oft als Geschäftsobjekte oder Geschäftskomponenten bezeichnet, stellen dabei logisch zusammenhängende Teile der zukünftigen Anwendung dar.

Die *softwaretechnische Architektur* legt durch den so genannten Komponentenschnitt die einzelnen technischen Softwarekomponenten fest, die den Funktionsumfang des Systems realisieren. Diese technischen Komponenten sind die lauffähigen Softwarebestandteile eines Systems und werden auf Basis eines konkreten Komponentenmodells implementiert. In der Regel realisieren mehrere Softwarekomponenten eine fachliche Komponente.

Die Beschreibung der Ablaufumgebung für die einzelnen Komponenten geschieht in der *systemtechnischen Architektur*. Zusätzlich müssen die relevanten Rahmenbedingungen (z.B. Betriebssystem, Datenbanken, Applikationsserver, etc.) für den Einsatz der jeweiligen Technologien und Produkte identifiziert und festgelegt werden.

Die Unterteilung in die drei Sichten fachliche, softwaretechnische und systemtechnische Architektur richtet sich sehr stark an die jeweils beteiligten Adressaten innerhalb eines Softwareentwicklungsprozesses. Somit erfüllt dieser Ansatz den Anspruch an Architekturen, ein Kommunikationsinstrument für die Beteiligten zu sein. Die Unterteilung geht nicht darauf ein, inwiefern jeweils statische oder dynamische Aspekte zu beschreiben sind. Somit ist es möglich, innerhalb jeder der vorgeschlagenen Sichten beide Aspekte zu berücksichtigen.

5.3.2.3 Fazit

In dieser Arbeit werden Software-Referenzarchitekturen hergeleitet. Um den Referenzcharakter einzuhalten, muss von Details auf Implementierungsebene abstrahiert werden. Ebensowenig befassen sich die Architekturen mit Fragen der Verteilung von Soft- auf Hardware oder mit Fragen zu konkreten Ablaufumgebungen. Aus diesem Grund werden diese Sichtweisen nicht näher berücksichtigt, wodurch also die *systemtechnische Architektur* nach Gruhn und Thiel sowie die *Physikalische* und die *Entwicklungs-Sicht* von Kruchten außen vor bleiben.

Stattdessen wird im Rahmen dieser Arbeit die *softwaretechnische* Architektur im Vordergrund stehen. Sie wird dabei sowohl statische als auch dynamische Aspekte der Elemente beschreiben. Da die softwaretechnische Architektur eine Weiterentwicklung der *fachlichen* Architektur darstellt, wird diese implizit mit berücksichtigt. Diese ist weitgehend identisch mit der *Logischen Sicht* nach Kruchten. Auch die von Kruchten eingeführten *Szenarien* werden angewendet, wie später zu sehen sein wird.

5.3.3 Architekturstile

Während die Sichten auf Softwarearchitekturen dazu dienen, unterschiedliche Teilaspekte je nach Zielgruppe und zeitlichem Entwicklungszeitpunkt darzustellen, befassen sich die Architekturstile damit, wie die grundlegenden Elemente in der Darstellung zueinander in Beziehung zu setzen sind. Architekturstile finden somit in jeder Architektursicht Anwendung.

Wie bereits erwähnt, besteht eine Softwarearchitektur vereinfacht ausgedrückt aus Komponenten und ihren Verbindungen. Diese Repräsentation führt zu einer abstrakten Architekturbeschreibung in Form eines Graphen, in dem die Knoten die Komponenten und die Kanten die Verbindungen darstellen. Dabei können die Elemente in Abhängigkeit der zugrunde liegenden Sichtweise eine unterschiedliche Bedeutung besitzen. Die Verbindungen können beispielsweise Interaktionen, Prozeduraufrufe oder Datenflüsse sein. Aus diesem Grund definiert ein Architekturstil die Semantik der Notation und die mit dem jeweiligen Stil verbundenen Restriktionen und Kontrollstrukturen [GS93].

Einige bekannte Architekturstile werden nachfolgend kurz beschrieben. In der Praxis übernimmt der Entwickler einen oder mehrere dieser Stile, um die Architektur des Softwaresystems zu konstruieren. Dabei können Architekturstile verwendet werden, um verschiedene Sichtweisen zu repräsentieren oder um die einzelnen Elemente einer Darstellung sukzessive zu verfeinern, was solange wiederholt werden kann, bis alle architektonischen Anforderungen abgedeckt sind [Sha96] .

- Pipes and Filters

  In einem Pipes and Filters System liegt der Fokus auf dem Datenfluss. Es existiert eine Anzahl von Komponenten, deren Ausgaben jeweils Eingaben für andere Komponenten darstellen. Die einzelnen Komponenten sind dabei völlig voneinander isoliert und teilen sich weder Daten noch Zustände. Dieser Stil ist gut für die Programmanalyse oder Stapelverarbeitung aber weniger für interaktive Applikationen geeignet.

- Objektorientierte Architektur

  Bei diesem sehr verbreiteten Architekturstil werden die Datenstrukturen und die mit ihnen assoziierten Operationen durch abstrakte Datentypen oder Objekte gekapselt. Objekte interagieren durch Funktions- und Prozeduraufrufe miteinander. Zwei wichtige Aspekte müssen hierbei berücksichtigt werden: als erstes sind die Objekte für die Integrität ihrer Repräsentation verantwortlich und zweitens muss diese Repräsentation anderen Objekten verborgen bleiben.

- Schichtenmodell

  Bei dem Schichtenmodell liegt der Schwerpunkt auf der Identifikation hierarchischer Abstraktionsebenen eines Systems. In der Regel werden bei der Interaktion der einzelnen Schichten Restriktionen festgelegt. So dürfen nur Elemente aus benachbarten oder der eigenen Schicht miteinander kommunizieren. Die Verwendung von Schichtmodellen unterstützt die Wiederverwendbarkeit von Software, da die Möglichkeit besteht, Standardschichten zu implementieren, die als Grundlage für verschiedene Anwendungen dienen können. Ein bekanntes Beispiel ist das ISO/OSI Modell. Abbildung 5-2 illustriert diesen Stil.

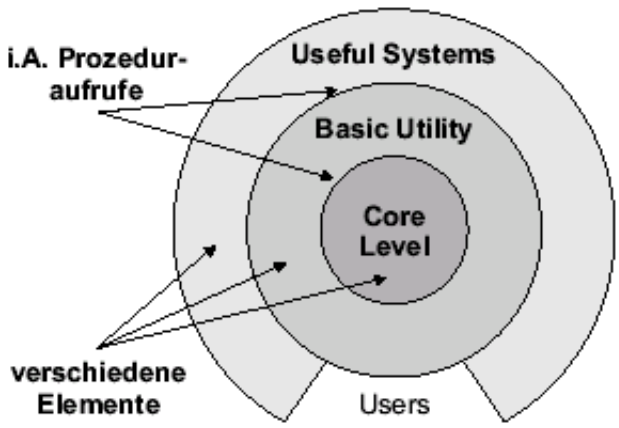


Abbildung 5-2: Schichtenmodell nach [GS93]

- Client/Server Architektur

  Bei diesem System stellt ein Server einen Prozess dar, der bestimmte Dienste anderen Prozessen, den Clients, zur Verfügung stellt. Normalerweise ist dem Server die Identität oder Anzahl der Clients, die zur Laufzeit auf ihn zugreifen, unbekannt. Hierdurch wird die Verteilung der einzelnen Prozesse eines Systems unterstützt.

- Domänen-spezifische Softwarearchitektur (DSSA)

  Bei der Entwicklung einer solchen Architektur ist die Organisationsstruktur eng an einen spezifischen Problembereich einer Anwendungsdomäne gekoppelt. Kernelemente sind dabei die Spezifikation eines Domänenmodells, einer Referenzanforderung und einer Referenzarchitektur. Die wesentliche Motivation ist auch hier wieder die Verbesserung der Grundlagen für eine systematische Wiederverwendung [Hay94].

Neben den aufgelisteten gibt es eine Vielzahl weiterer Architekturstile und spezielle Ausprägungen, auf die hier jedoch nicht weiter eingegangen wird (vgl. z.B. das Model-View-Controller Modell von [Sha96]).

5.3.3.1 Fazit

Wie bereits erwähnt, werden häufig mehrere Architekturstile miteinander kombiniert verwendet. Da in dieser Arbeit Komponenten mitsamt ihren Diensten und Beziehungen zueinander identifiziert und beschrieben werden, wird eine Mischung aus einem objektorientierten und einem Client-Server-Architekturstil verfolgt. Auch die Ansätze der Domänen-spezifischen Architektur finden Verwendung, da mit den Erlösmodellen ein sehr spezifisches Anwendungsgebiet behandelt wird.

5.3.4 Beschreibungssprachen von Softwarearchitekturen

Die soeben behandelten Architekturstile definieren, wie die Bestandteile von Architekturen (Komponenten und Verbindungen zwischen ihnen) zueinander angeordnet werden sollen. Sobald die konkrete Modellierung der Bestandteile erfolgen soll, kommen Beschreibungssprachen zum Einsatz, die die Elemente grafisch und textlich beschreiben.

Eine Reihe von Sprachen ist entwickelt worden, um eine formale und eindeutige Beschreibung von Softwarearchitekturen zu ermöglichen. Solche Beschreibungssprachen (engl. Architecture Description Language, Abk. ADL) werden bei Clements und Kogut als eine Sammlung von Notationen, Sprachen, Standards und Konventionen für die Modellierung einer Softwarearchitektur bezeichnet[4] [CK94].

4. "Informally, an ADL is a set of notations, languages, standards and conventions for an architecture model. An ADL defines a set of notations (e.g. diagrams, formal languages, natural language test fields) for each view that the ADL includes." (aus [CK94])

Durch die Verwendung einer formalen ADL ergeben sich eine Reihe von Vorteilen für die Durchführung von Softwareprojekten. Mit Hilfe einer Beschreibungssprache können formale Analysen durchgeführt werden, um beispielsweise zu verifizieren, ob eine Architekturbeschreibung konsistent und vollständig ist [All97]. Zudem soll durch den Einsatz einer ADL die Softwarearchitektur zwischen den Beteiligten eines Projekts eindeutig verstanden und kommuniziert werden. Ein weiterer Aspekt ist die Verwendung einer ADL, um mittels einer Umformung der formalen Beschreibung in eine Programmiersprache den Schritt vom Design zur Implementierung zu unterstützen [EM01].

Im universitären Forschungsumfeld sind zu diesem Zweck eine Reihe von Architektursprachen entwickelt worden. Eine Schwierigkeit bei der Evaluierung solcher Sprachen ist, dass diesen oftmals sehr spezifische Sichtweisen auf die Architektur eines Systems zugrunde liegen und daher gänzlich verschiedene Aspekte unterstützt werden. In verschiedenen Arbeiten wurde bereits versucht, eine Klassifikation der wichtigsten Beschreibungssprachen vorzunehmen und die unterschiedlichen Eigenschaften zu vergleichen. Medvidovic und Taylor betrachten dabei die Sprachen *ACME*, *Aesop*, *C2*, *Darwin*, *MetaH*, *Rapide*, *SADL*, *UniCon*, *Weaves* und *Wright* und liefern mit ihrer Arbeit einen entsprechenden Klassifikationsrahmen [MT00]. Fuxmann bewertet eine ähnliche Auswahl von Beschreibungssprachen hinsichtlich ihrer Eignung für die Modellierung der zentralen Elemente einer Softwarearchitektur [Fux00]. Ein wichtiger Aspekt hierfür ist die Fähigkeit einer Beschreibungssprache, verschiedene Architekturstile zu unterstützen [HR97]. Darüber hinaus sollen neben einer leichten Verständlichkeit auch Merkmale wie eine hierarchische Gliederung und Komposition, Offenheit und Skalierbarkeit, Abbildung von dynamischen Verhalten, sukzessive Verfeinerung oder nichtfunktionale Eigenschaften unterstützt werden [MT00].

Zur Auswahl einer oder mehrerer Architekturbeschreibungssprachen für die Modellierung einer Softwarearchitektur können dementsprechend verschiedene Kriterien berücksichtigt werden. Zum einen ist festzustellen, welchem Zweck (z.B. Verifikation oder Simulation) der Einsatz einer Architektursprache dienen soll und wie gut dieser unterstützt wird. Zum anderen sind die verschiedenen Sichten auf ein System zu berücksichtigen sowie die Eigenschaft, inwieweit diese durch eine oder mehrere ADLs beschrieben werden können.

Neben den Vorteilen formaler ADLs existieren jedoch auch gravierende Nachteile. Die Tatsache, dass die einzelnen Architekturbeschreibungssprachen jeweils für einen speziellen Einsatzschwerpunkt entwickelt wurden, lassen diese für eine Verwendung in dieser Arbeit ungeeignet erscheinen. Zum Beispiel wird *Darwin* für dynamische Architekturen benutzt, *Unicon* fokussiert auf die Erzeugung eines ausführbaren Codes, *ACME* ist eher ein Austauschformat zwischen verschiedenen Architektursprachen und *Rapide* ist auf die Simulation und Konformität von Architekturen spezialisiert [Fux00].

Der im Rahmen dieser Arbeit noch größere Nachteil ist jedoch, wie Clements erkennt, dass die beabsichtigte Zielgruppe der meisten hier vorgestellten Sprachen der Anwendungsentwickler und weniger der Manager oder Projektleiter ist [Cle96a]. Hierdurch ist die Anforderung an eine leichte Verständlichkeit der Architekturbeschreibung im Kontext dieser Arbeit nicht gegeben. Wenn ausgehend von Geschäfts- und Erlösmodellen des e-Business geeignete Softwarearchitekturen beschrieben werden sollen, müssen diese in hohem Maße allgemein verständlich sein.

In den nächsten beiden Abschnitten werden aus diesen Gründen zwei weitere, sich ergänzende Möglichkeiten vorgestellt, um die in dieser Arbeit zu entwickelnden Komponenten, ihre Beziehungen und die Kommunikation zu beschreiben.

5.3.4.1 UML als Beschreibungssprache

Die Unified Modeling Language (Abk. UML)[5] „dient zur Modellierung, Dokumentation, Spezifikation und Visualisierung komplexer Softwaresysteme unabhängig von deren Fach- und Realisationsgebiet. Sie liefert die Notationselemente gleichermaßen für die statischen und dynamischen Modelle von Analyse, Design und Architektur und unterstützt insbesondere objektorientierte Vorgehensweisen“ [JRH04].

Aufgrund ihrer Fähigkeiten erscheint es nicht verwunderlich, dass die UML inzwischen als ein allgemein akzeptierter und häufig benutzter Standard für eine grafische Modellierungssprache gilt (vgl. [FS97], [RRS99]), die als praxisorientierte Methode für die Beschreibung von Softwarearchitekturen empfohlen wird (vgl. [Hil99], [GT00]).

Die UML eignet sich aufgrund ihrer Elemente sehr gut dazu, die unterschiedlichen Strukturen darzustellen, wie es in der Definition von Softwarearchitekturen gefordert wurde. Kernstück der UML sind insgesamt 13 Diagrammtypen, die sich in Struktur- und in Verhaltensdiagramme unterteilen lassen. Die mit am häufigsten verwendeten Diagrammtypen, die auch in dieser Arbeit zum Einsatz kommen, sind Klassendiagramme, Sequenzdiagramme und Use Case Diagramme.

Ein Klassendiagramm bildet statisch die Komponenten mitsamt ihren Eigenschaften, Methoden und gegenseitigen Abhängigkeiten ab. Um das dynamische Verhalten der Komponenten zueinander aufzuzeigen, wird das Sequenzdiagramm eingesetzt. Mit seiner Hilfe werden die Interaktionen zwischen den Kompenten in Form von Methodenaufrufen und Rückmeldungen dargestellt. Zusammen mit dem Klassendiagramm bildet es das Kernstück einer softwaretechnischen Architektur. Um den geforderten Kompentenschnitt im Rahmen der softwaretechnischen Architektur zu liefern, werden die Klassendiagramme zusätzlich um grafische Elemente ergänzt, die die einzelnen Klassen ihren jeweiligen Komponenten zuordnen. Diese Erweiterung beeinträchtigt jedoch nicht die Vorgaben der UML. Details werden im nächsten Kapitel vorgestellt.

Im Rahmen der Anforderungsanalyse werden Use Case Diagramme eingesetzt, die die Interaktion von Akteuren mit dem System darstellen. Als Ergebnis werden Funktionalitäten ersichtlich, die das System beziehungsweise die Komponenten der Außenwelt zur Verfügung stellen müssen. Mit diesem Diagrammtyp werden die Szenarien realisiert, wie sie in Abschnitt 5.3.2 als eine der Sichten auf Architekturen vorgestellt wurden.

Die UML stellt mit diesen Diagrammen eine Beschreibungssprache von Softwarearchitekturen dar, die ideal für die Anforderungen dieser Arbeit erscheinen. Zusätzlich wird im nächsten Abschnitt jedoch noch eine weitere Beschreibungsmethode vorgestellt, um Komponenten und vor allem deren angebotene Dienste praxistauglich zu beschreiben.

5. In dieser Arbeit wird die UML in der Version 2.0 zugrunde gelegt.

5.3.4.2 ACDL als Beschreibungssprache

Speziell für den Bereich der komponentenbasierten Softwareentwicklung können für die Spezifikation von Komponenten Sprachen wie *ACDL*, *PI* oder *Resolve* eingesetzt werden. Gute praktische Erfahrungen wurden dabei insbesondere mit der strukturierten und auf natürlicher Sprache basierenden ACDL (Abk. für A Componenten Definition Language) gemacht, da sie gegenüber anderen Sprachen nur einen minimalen Einarbeitungsaufwand erfordert [GT00]. Hierdurch eignet sich diese Sprache besonders gut als Kommunikationsbasis für die unterschiedlichen Projektbeteiligten. Abbildung 5-3 zeigt ausschnittsweise ein Beispiel für eine ACDL Spezifikation.

Komponente	Ort
Identifikator	fachlichK-02
Verantwortlicher	Otto Mustermann
Komponenten-Objektmodell	Komponenten der fachlichen Klasse, Version 1.7
Komponenten Rolle	Server
Verwendete Komponenten	
Aufrufende Komponenten	(fachlichK-03, Objekt), (fachlichK-04, SchadenLeistung), (fachlich-08, Vertrag)
Service	LokationsVerwaltung
Service-Identifikator	fachlichS-02
Kurzbeschreibung	Beschreibt alle für eine Versicherung sinnvollen Orte und Streckenangaben.
Langbeschreibung	Es wird ein Ort, eine Strecke oder eine Route und deren Beziehung zueinander beschrieben.
Service-Funktion	PostalischeAdresseErzeugen
Kurzbeschreibung	Eine postalische Adresse anlegen.
Langbeschreibung	Die Stammdaten einer postalischen Adresse werden angelegt, indem über die...
Service-Funktion	PostalischeAdresseLiefern
Kurzbeschreibung	Eine postalische Adresse...

Abbildung 5-3: Beispiel einer ACDL Spezifikation nach [BGH00]

Die Beschreibung einer Komponente mit der ACDL umfasst zunächst ihren Namen, einen abstrakten Identifikator und den verantwortlichen Entwickler. Die Angabe des Objektmodells ist eine Referenz auf ein entsprechend spezifiziertes Modell. Die Komponentenrolle legt fest, ob es sich um eine client- oder serverseitige Komponente handelt. Um Abhängigkeiten von anderen Komponenten darzustellen, werden alle aufrufenden und verwendeten Komponenten mit ihrem Namen und ihrem Identifikator aufgelistet. Jeder Dienst, den eine Komponente anderen Komponenten anbietet, wird mit einer Kurz- und Langbeschreibung, seinem Namen und einem eindeutigen Identifikator versehen. Die einzelnen Servicefunktionen, die für die Realisierung eines solchen Dienstes verantwortlich sind, werden ebenfalls durch ihren Namen und eine Kurz- und Langbeschreibung aufgezählt. Danach werden die einzelnen Schnittstellen, die für eine Implementierung der Komponente nötig sind, durch ihren Namen, einen Identifikator und eine Beschreibung in Kurz- und Langform angegeben. Als letztes werden die einzelnen Methoden der Schnittstellen beschrieben, wobei auch hierbei neben dem Namen und einer Signatur eine Beschreibung angegeben wird [BGHR00].

5.3.4.3 Fazit

Aufgrund der diskutierten Nachteile formaler ADLs wird hier die UML in der Version 2.0 zur Beschreibung der Architekturbestandteile eingesetzt. Kernelemente der Architekturbeschreibung sind dabei die Klassendiagramme für die statische Beschreibung und die Sequenzdiagramme für die dynamische Beschreibung. Da die angepassten Klassendiagramme gleichzeitig auch den Komponentenschnitt mit darstellen, ist eine zusätzliche Beschreibung der Komponenten durch die ACDL nicht mehr notwendig.

5.4 Schlussfolgerung

In diesem Kapitel wurde gezeigt, dass komponentenbasierte Softwareanwendungen für den Einsatz im e-Business optimal geeignet sind. Die Anforderungen an Softwaresysteme des e-Business werden weitereichend durch die Eigenschaften der komponentenbasierten Anwendungen erfüllt. Im Rahmen der Entwicklung einer konkreten Softwareanwendung muss zunächst eine geeignete Softwarearchitektur hergeleitet werden. Dies hat den Vorteil, dass anhand der Architekturen die Kommunikation unter allen Projektbeteiligten vereinfacht wird und dass wichtige Designentscheidungen des Systems frühzeitig festgelegt werden können. Sofern bereits Softwarearchitekturen für Systeme mit vergleichbaren Anforderungen existieren, können diese als Referenzen verwendet werden und so die Entwicklungszeiten verkürzen und die Wirtschaftlichkeit des Projektes erhöhen. Dieser Gedanke der Wiederverwendung liegt dieser Arbeit zugrunde.

Für die Erlösmodelle des e-Business, die in den vorangegangenen Kapiteln klassifiziert wurden, sollen im weiteren Verlauf geeignete Softwarearchitekturen hergeleitet werden. Sie stellen somit Referenzen für die betrachtete Domäne der Geschäftsmodelle des e-Business dar, auf die in konkreten Anwendungsfällen zurückgegriffen werden kann. Es handelt sich folglich um Software-Referenzarchitekturen.

Eine Betrachtung der unterschiedlichen Möglichkeiten, solche Softwarearchitekturen geeignet zu beschreiben, hat ergeben, dass eine softwaretechnische Sichtweise zum Einsatz kommen wird. Diese Sicht baut auf einer fachlichen Architekturbeschreibung auf, so dass diese implizit ebenfalls mitgeliefert wird. Dabei wird darauf geachtet, Funktionalitäten in Komponenten zusammenzufassen, so dass das Konzept der komponentenbasierten Softwareentwicklung unterstützt wird. Als Notation wird UML in der Version 2.0 verwendet. Einzelheiten werden im folgenden Kapitel explizit behandelt.

6 Katalog der Referenzarchitekturen

In diesem Kapitel werden die Referenzarchitekturen der Erlösmodelle hergeleitet und beschrieben. Da Erlösmodelle immer nur im Gesamtkontext eines Geschäftsmodells betrachtet und angewendet werden können, müssen die Systemgrenzen der abgeleiteten Referenzarchitekturen den jeweiligen Kontext entsprechend gezogen werden. Eine Architektur, die ausschließlich nur die Komponenten enthält, die für die Anwendung eines bestimmten Erlösmodells benötigt werden, würden bezugs- und interaktionslos aufzulisten sein, wodurch vor allem dynamische Aspekte der Referenzarchitekturen unberücksichtigt blieben. Das Zusammenspiel mit Komponenten der restlichen[1] Architektur ist zum Verständnis unbedingt notwendig und wird mit betrachtet. Dadurch bedingt kommt es vor, dass die hier vorgestellten Referenzarchitekturen jeweils Teilbereiche mit aufweisen, die strenggenommen nicht zum Erlösmodell, sondern zum einbettenden Geschäftsmodell gehören. Andererseits werden dort auch Grenzen in der Betrachtung gezogen, in der der Kontext zum Erlösmodells fehlt.

Die Abgrenzung der betrachteten Architekturbestandteile erfolgt jeweils in einem separaten Abschnitt (*Anforderungsanalyse und Systemabgrenzung*) mit Hilfe von UML-Anwendungsfällen (UML-Use-Cases). In diesem Abschnitt werden die Anforderungen aufgezeigt, denen die Anwendung gerecht werden muss, um das Erlösmodell zu realisieren. Die Anforderungsbeschreibung erfolgt dabei weitgehend aus Sicht der Anwender des Systems und wird mit Hilfe der UML Anforderungsfälle beschrieben. Die unterschiedenen Akteure[2] des Systems sind:

- Betreiber

  Die Rolle des Betreibers repräsentiert die Mitarbeiter des Unternehmens, die das System betreiben. Sie umfasst somit alle (unternehmens-internen) Anwender der verschiedenen Fachabteilungen sowie Administratoren. Zur besseren Übersicht wird diese Rolle nicht weiter unterteilt. Die Aktivitäten des Betreibers beschränken sich weitgehend auf Verwaltungstätigkeiten, um die Grundlagen dafür zu schaffen, dass der Nutzer das Angebot in Anspruch nehmen kann. Aus diesem Grund werden die Anwendungsfälle des Betreibers weniger detailliert betrachtet und in die Architekturgestaltung mit einbezogen.

1. „restlich“ im Sinne der übrigen Architekturbestandteile des gesamten Geschäftsmodells, von dem das Erlösmodell nur einen Teil darstellt
2. die Akteure sind dabei als Rollen zu verstehen

- Nutzer

  Der Nutzer stellt den Endanwender des Systems dar und grenzt sich damit vom Betreiber ab. In den folgenden Abschnitten wird der Begriff des Kunden synonym verwendet, sofern es sich anbietet. Dies ist beispielsweise für Nutzer eines e-Shops der Fall, wenn sie Produkte kaufen. Die Interaktionen des Nutzers mit dem System stehen im Vordergrund der Betrachtungen der Referenzarchitekturen, da sie weitgehend die Prozesse aufzeigen, die je nach Erlösmodell durchgeführt werden. Die Anwendung und somit auch die Architektur müssen sicherstellen, dass die Funktionalitäten zur Durchführung dieser Prozesse bereitgestellt werden.

- Teilnehmer

  Die Rolle des Teilnehmers tritt im Zusammenhang mit dem Geschäftsmodell eines Marktplatzes auf. Der Marktplatzbetreiber stellt die Infrastruktur bereit, auf der die (Markt-)Teilnehmer ihren Handel betreiben können. Die Teilnehmerrolle kann weiter unterteilt werden in Teilnehmer/Verkäufer und Teilnehmer/Käufer, wodurch ihre Funktion innerhalb der Handelstransaktion zum Ausdruck kommt. Beide Rollen werden für die Anforderungsanalyse des Erlösmodells TRANSAKTIONSGEBÜHR verwendet.

- Geschäftskunde

  Die Geschäftskunden sind ebenfalls Unternehmen, die mit dem Betreiber des Systems vertragliche Vereinbarungen über zu liefernde oder in Anspruch genommene Produkte oder Dienstleistungen getroffen haben. Diese Rolle wird für das Erlösmodell WERBUNG verwendet.

Ein Nachrichtenmagazin finanziert sich ausschließlich über Werbung. Der Betreiber stellt sein Angebot den Endanwendern kostenfrei zur Verfügung, um eine möglichst hohe Nutzergemeinde aufzubauen und seine Werbeflächen bestmöglich zu verkaufen. Die Geschäftskunden des Nachrichtenmagazins können diese Werbeflächen innerhalb von Kampagnen buchen und schließen dazu einen Vertrag mit dem Betreiber ab. Sie beziehen als Dienstleistung das Schalten der Werbebanner, die sie ihrerseits als Datei bereitstellen. Darüber hinaus erhalten die Geschäftskunden regelmäßig Auswertungen über die Kampagne in Form von Reports. Als Geschäftskunden können sowohl die Werbeträger selbst als auch Agenturen auftreten, die wiederum mehrere Werbeträger vertreten.

- Anbieter

  Die Rolle des Anbieters wird für das Erlösmodell PROVISION benötigt. Der Betreiber des Systems vermittelt oder vertreibt Produkte eines Anbieters und erhält als Gegenleistung von ihm Vermittlungs- oder Abschlussprovisionen.

  Ein Versicherungsmakler geht eine vertragliche Bindung mit den von ihm vertretenen Versicherungsunternehmen über sein angebotenes Portfolio der Versicherungen ein. Die Versicherungsunternehmen sind die Anbieter (der Policen), die bei Abschluss einer Versicherungspolice dem Makler eine Provision zahlen.

- <<System>> Controller

  Als zusätzliche Rolle wird die des Controllers eingeführt, die keine natürliche Person oder Organisation darstellt, sondern ein Teil-System der Anwendung. Gemäß der UML repräsentieren Anwendungsfälle die Interaktion der Außenwelt mit einem System. Die ausführenden Akteure können dabei Personen oder (Sub-)Systeme sein. Das interne Systemverhalten wird entsprechend der UML jedoch nicht mit Anwendungsfällen modelliert.

  In dieser Arbeit stellt sich jedoch das Problem, dass die Erlösmodelle jeweils nur ein Teilsystem des gesamten Geschäftmodells bilden und somit auch die restlichen Systembestandteile mit diesen Teilsystemen interagieren. Sofern ein wichtiger Aspekt oder eine wichtige Funktionalität des Systems mit aufgezeigt werden soll, wird aus diesem Grund die Rolle des Controllers eingeführt. Es wird dabei nicht weiter spezifiziert, welches Teil-System diesen Akteur jeweils repräsentiert.

Zur besseren Strukturierung werden die UML-Anwendungsfälle in Szenarien zusammengefasst, die unterschiedliche Tätigkeitsbereiche abgrenzen. Beispiele sind die Produkt- sowie die Nutzerverwaltung.

Die Anwendungsfälle geben nicht nur Aufschluss über die Funktionalitäten, die das System bereitstellen muss, sondern sie zeigen auch die Grenzen des betrachteten Systems auf. In dieser Arbeit liegt der Fokus der Betrachtung auf den Erlösmodellen. Somit werden Anwendungsfälle, die im Grenzbereich liegen, erkenntlich hervorgehoben. Auf diese Weise wird aufgezeigt, welche Bereiche die Referenzarchitektur mit abdeckt und welche sie nur rudimentär berücksichtigt oder gänzlich außen vorlässt. Darüber hinaus werden einige Anwendungfälle in den zugehörigen Diagrammen redundant aufgeführt, da sie fachlich zu mehreren Szenarien gehören. Der Anwendungsfall $\mathrm{Anmelden}$ eines Nutzers gehört beispielsweise sowohl zum Szenario der Nutzerverwaltung als auch zur Angebotsnutzung. Um jeden Anwendungsfall dennoch eindeutig zuzuweisen, wird die redundante Verwendung ebenfalls grafisch kenntlich gemacht. In Abbildung 6-1 werden die unterschiedlichen Darstellungsformen vorgestellt.

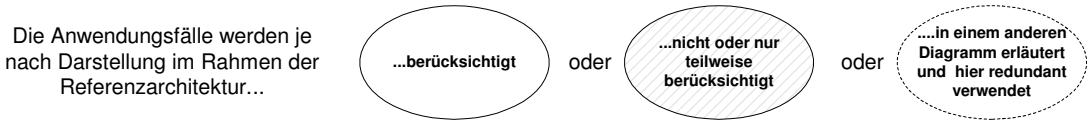


Abbildung 6-1: Systemabgrenzung durch Darstellung der Anwendungsfälle

Um die Anzahl der Anwendungsfälle möglichst gering zu halten, werden sie im Rahmen dieser Arbeit relativ grobgranular identifiziert. Häufig wird der Begriff des *Verwaltens* eines Objektes verwendet, wodurch grundsätzlich das Zugreifen, Hinzufügen, Ändern und Löschen (des Objektes) mit eingeschlossen wird.

> Als Beispiel dient der Anwendungsfall Produktattribute verwalten. Er beinhaltet den vollständigen Umgang des Akteurs mit dem System zur Pflege der Produktattribute. Wird ein Produkt neu in das Sortiment aufgenommen, editiert der Akteur initial die Attributwerte wie Produktbezeichnung, Produktbeschreibung und Preis. Darüber hinaus kennzeichnet er das Produkt als neu, womit eine gesonderte Darstellung auf den Webseiten verbunden sein wird. Sobald sich der Preis des Artikels ändert, muss das entsprechende Attribut vom Akteur angepasst werden. Darüber hinaus wird das Flag, welches das Produkt als neu kennzeichnet, nach einer bestimmten Zeitspanne automatisch vom Akteur gelöscht. Zur Kontrolle der aktuellen Werte kann über eine GUI jederzeit lesend auf die Attribute zugegriffen werden.

Die Anwendungsfallnamen werden innerhalb der Erläuterungen optisch durch die folgende Formatierung hervorgehoben: Anwendungsfall durchführen.

Der Abschnitt *Architekturbeschreibung* enthält das jeweilige Klassendiagramm als zentralen Bestandteil der (statischen[3]) Referenzarchitektur. Es wird versucht, die für die Architektur notwendigen Klassen auf einer fachlichen Ebene so detailliert wie möglich zu beschreiben, ohne jedoch den Referenzcharakter zu verlieren. Es fließen dabei sowohl die Erkenntnisse der Erlösmodelltypologie aus Kapitel 4 bezüglich der grundlegenden Architekturanforderungen als auch die Erkenntnisse aus der Anforderungsanalyse anhand der Anwendungsfälle mit ein.

Ziel ist es, die wesentlichen Komponenten der Architektur zu identifizieren, wobei gleichzeitig deren enthaltenen Klassen mitsamt ihren Abhängigkeiten aufgezeigt werden sollen. Die Diagramme zeigen somit vordergründig nicht die Komponenten, sondern die Klassen und deren Beziehungen zueinander. Der Komponentenschnitt wird zusätzlich in den Diagrammen durch Gruppierungen der Klassen grafisch hervorgehoben. Aus Platzgründen werden

3. Da die UML-Klassendiagramme zu den Strukturdiagrammen zählen, werden sie hier als *statisch* bezeichnet. Im Gegensatz dazu gehören die Sequenzdiagramme zu den Verhaltensdiagrammen, die *dynamische* Aspekte darstellen.

dabei die Komponenten nicht mit ihren typischen UML-Symbolen dargestellt, sondern der Anordung auf dem Zeichenblatt angepasst. Die folgende Abbildung 6-2 zeigt exemplarisch ein Klassendiagramm, wobei der Komponentenschnitt die enthaltenen Klassen zu drei Komponenten gruppiert hat.

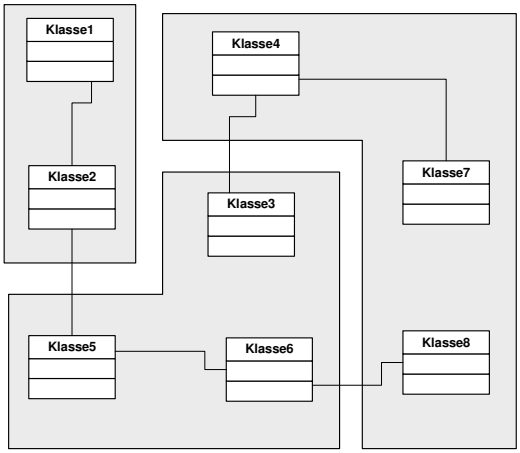


Abbildung 6-2: Beispiel eines Klassendiagramms mitsamt Komponentenschnitt

Wie man sieht, werden die Komponenten den Anordnungen der Klassen entsprechend gezeichnet.

Darüber hinaus werden in den Architekturen auch Komponenten verwendet, deren Dienste und Komplexität nicht vollständig durch die Darstellung der enthaltenen Klassen abgedeckt werden können.

> Ein Beispiel ist die Komponente `Logging`, die im Zusammenhang mit dem Erlösmodell PROFIHANDEL ihre Dienste anbietet (vgl. Abschnitt 6.4 auf Seite 209ff). Zu ihren Diensten gehört das Protokollieren der Aktivitäten eines Nutzers, indem das Logfile des Web-Servers entsprechend vorhandener Regeln ausgewertet wird und die gefilterten Daten in einem Sitzungsprotokoll des Nutzers persistent gespeichert werden. Die Komponente muss also über einen Dienst zur Konfiguration verfügen, einen Dienst zum Aktivieren sowie Deaktivieren der Protokollierungen anbieten sowie den Dienst der Protokollierung des Nutzerverhaltens entsprechend der Konfiguration enthalten.

Diese Komponenten werden nicht anhand ihres vollständigen Klassendiagramms, sondern nur mit ihren wesentlichen Klassen und deren wichtigsten Methoden beschrieben. Der Fokus der Beschreibung liegt somit auf den bereitgestellten Diensten der Komponenten und weniger auf den enthaltenen Klassenstrukturen. Zur grafischen Darstellung wird in den Diagrammen direkt das UML-Komponentensymbol verwendet, wobei die relevanten Klas-

sen innerhalb dieses Komponentensymbols aufgeführt werden. Der folgende Abschnitt 6-3 zeigt beispielhaft die Repräsentation der Komponente `Logging` (das vollständige Diagramm ist in Abbildung 6-79 auf Seite 215 zu sehen).

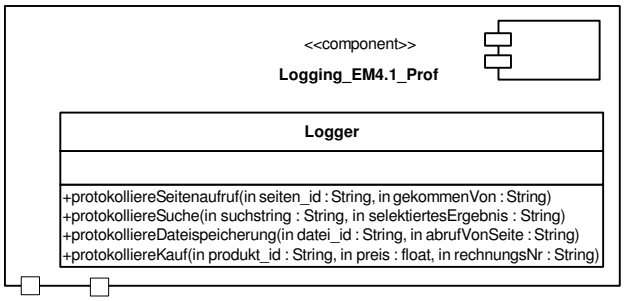


Abbildung 6-3: Gesonderte Darstellung einer Komponente innerhalb des Klassendiagramms

Die Komponente enthält als wesentliches Element die Klasse `Logger` mit den wichtigsten Methoden und darüber hinaus zwei *Ports* (dargestellt durch die beiden Quadrate am unteren Rand), über die die Anbindung mit der Restarchitektur erfolgt.

Zur besseren Identifizierung werden die Elemente der statischen Architektur jeweils mit eigenen Formatierungen versehen. Die folgende Liste zeigt die verwendeten optischen Hervorhebungen:

- Formatierung von `Komponentennamen`
- Formatierung von `Klassennamen`
- Formatierung von `Methodennamen`
- Formatierung von `Attributnamen`

Detailaspekte der Implementierung bleiben bei der Architekturbeschreibung in der Regel unberücksichtigt und werden nur aufgeführt, wenn auf besondere Aspekte hingewiesen werden soll. Beispielsweise werden die Konstruktoren der Klasse nur in einigen Fällen explizit aufgeführt und beschrieben.

Zusätzlich zu den statischen Aspekten werden in den Architekturbeschreibungen Sequenzdiagramme eingesetzt, die dynamische Aspekte veranschaulichen, indem sie die Interaktionen der Klassen oder Komponenten innerhalb bestimmter Prozeduren darstellen.

Die UML sieht vor, dass zu jedem synchronen Methodenaufruf eine zugehörige Antwortnachricht in den Diagrammen eingezeichnet wird. Um die Übersichtlichkeit der Diagramme in dieser Arbeit zu erhöhen, wird jedoch in bestimmten Fällen auf diese Antwortnachrichten verzichtet. Dies ist dann der Fall, wenn mittels einfacher set- oder get-Methoden auf selbsterklärende Attribute schreibend oder lesend zugegriffen wird. Das folgende Beispiel verdeutlicht dies.

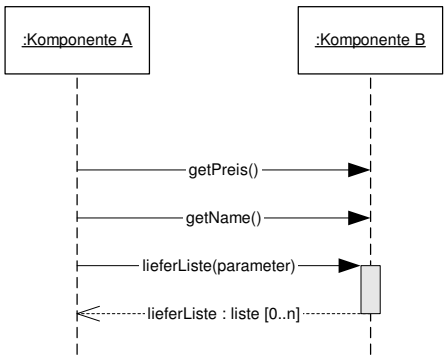


Abbildung 6-4: Beispiel von synchronen Nachrichten mit und ohne Antwort

Das Beispiel in Abbildung 6-4 zeigt, dass die *normalen* get-Methoden ohne Antwortnachrichten dargestellt werden. Falls dagegen davon abweichende Methoden aufgerufen werden, wird auch die Antwortnachricht mit dargestellt.

6.1 Referenzarchitektur für das Erlösmodell Einzeltransaktion

Das Erlösmodell der Einzeltransaktion zeichnet sich durch einen abgeschlossenen Kauf eines Produktes oder einer Dienstleistung aus. Unterschieden werden die Varianten *physikalisches Gut* und *digitales Gut* (Abschnitt 6.1.2 auf Seite 148).

6.1.1 Variante physikalisches Gut

Dieses Erlösmodell wurde ausführlich in Abschnitt 4.2.1.1 auf Seite 65 beschrieben. Die folgende Abbildung zeigt noch einmal die Klassifikationskriterien in ihren Ausprägungen.

Abbildung 6-5: Erlösmodellklassifikation Einzeltransaktion Variante physikalisches Gut

Aufbauend auf dieser Beschreibung werden im nächsten Abschnitt zunächst die Anforderungen mit Hilfe der Anwendungsfälle umfassender betrachtet.

6.1.1.1 Anforderungsanalyse und Systemabgrenzung

Zunächst wird analysiert, welche Anwendungsfälle für ein System existieren, mit dem das Erlösmodell EINZELTRANSAKTION von physikalischen Gütern abgebildet werden kann. Zur besseren Strukturierung werden die Anwendungsfälle in die Szenarien Produktverwaltung, Nutzerverwaltung, Angebotsnutzung, Bestellung und Bezahlung gruppiert.

Die Produktverwaltung physikalischer Güter erfordert vorrangig einen Produktkatalog, in dem die angebotenen Produkte aufgenommen und mit ihren Attributen verwaltet werden können, und eine Lagerverwaltung, mit der die Bestände aktualisiert und kontrolliert werden können. Die Akteure zur Durchführung der Arbeiten sind zum einen der Betreiber des Systems und zum anderen ein Controller zur automatisierten Bestandsverwaltung. Abbildung 6-6 zeigt die Anwendungsfälle beider Akteure.

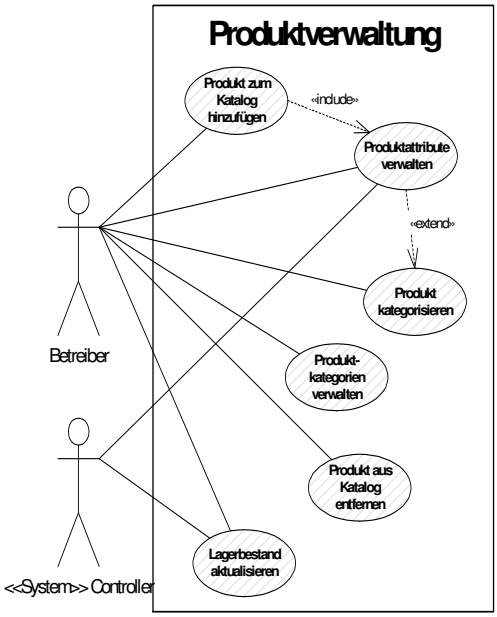


Abbildung 6-6: Anwendungsfälle der Produktverwaltung

Der Betreiber des Systems ist dafür zuständig, neue Produkte zum Katalog hinzuzufügen, was immer auch ein initiales Eingeben, also ein Verwalten der Produktattribute mit einschließt[4]. Das Verwalten der Attribute stellt eine Erweiterung des Anwendungsfalls Produkt kategorisieren dar. Die Kategorisierung der Produkte ermöglicht das schnelle und gezielte Auffinden durch die Nutzer und ist somit von großem Interesse für den Betreiber. Als Voraussetzung dazu muss er die angebotenen Produktkategorien verwalten. Der Umgang mit der Bestandsverwaltung der Produkte wird grob durch den Anwendungsfall Lagerbestand aktualisieren repräsentiert. Dieser Anwendungsfall ist sowohl dem Betreiber als auch dem Controller als Teil-System zugeordnet. Während der Betreiber zum Beispiel den Eingang neuer Lieferungen manuell vermerken wird, registriert der Controller den Abfluss der Produkte durch die Kundenbestellungen und Warenauslieferungen automatisch.

Die schraffierte Darstellung der Anwendungsfälle zeigt an, dass die Anwendungsfälle dieses Szenarios nur eingeschränkt in der Architektur berücksichtigt werden. Der Grund dafür liegt in der Tatsache, dass es sich hier eher um ein Produktmanagement- und Bestandsführungssystem handelt und somit nur begrenzt Einfluss auf das Erlösmodell EINZELTRANSAKTION hat. Da die hier betrachtete Variante jedoch auf dem Handel mit physikalischen Gütern basiert, wird dieser Bereich nicht gänzlich ausgeblendet, sondern in einem wenig detaillierten Ausmaß mit berücksichtigt.

4. Die Verwendung und Bedeutung des Begriffs *Verwaltung* im Zusammenhang mit den Anwendungsfällen wurde in der Einleitung dieses Kapitels erläutert.

Da diese Variante des Erlösmodells den Verkauf physikalischer Güter unterstellt, liegt eine identifizierte Kundenbeziehung zu Grunde, die eine Nutzerverwaltung mitsich zieht. Während die Produkte grundsätzlich auch durch ein geeignetes Zahlungsverfahren anonym bezahlt werden könnten, erfordert spätestens die Zustellung der Güter eine Identifizierung des Käufers.

Akteure der Nutzerverwaltung sind sowohl der Betreiber als auch der Nutzer selber. Automatisierte Abläufe übernimmt der Controller. Die folgende Abbildung 6-7 zeigt die Anwendungsfälle des Szenarios.

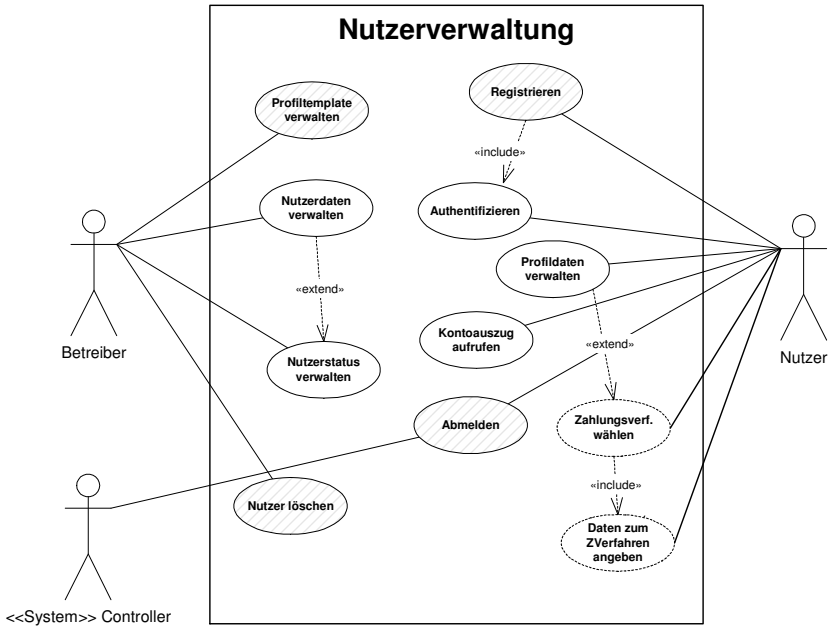


Abbildung 6-7: Anwendungsfälle der Nutzerverwaltung

Die Grundlage zur Verwaltung der Nutzerdaten bilden ihre Profile. Diese müssen zunächst vom Betreiber als Templates im System eingerichtet werden, was der Anwendungsfall Profiltemplate verwalten repräsentiert. Realen Nutzern zugewiesene Profile, die mit konkreten Attributwerten gefüllt sind, stellen die Instanzen der Profiltemplates dar. Auch auf diese Instanzen hat der Betreiber Zugriff und kann die Nutzerdaten verwalten. Als eine Besonderheit im Umgang mit den Nutzern hat er die Möglichkeit, den Status eines Nutzers zu verwalten, um ihn beispielsweise von Kaufaktivitäten oder auch vom Systemzugang auszuschließen. Inaktive Nutzer kann der Betreiber löschen.

Der Nutzer muss sich initial am System registrieren, um sich ein Profil zuzuweisen. Verfügt er bereits über ein Profil, muss er sich lediglich dem System gegenüber authentifizieren, was in der Regel über ein Login und ein Passwort erfolgt. Die Authentifizierung ist die Voraussetzung des Nutzers für den Zugriff und die weitere Verwaltung seiner Profildaten. Einen Sonderfall stellt die Wahl des Zahlungsverfahrens dar, die die Angabe der verfahrensabhängigen Daten erfordert. Beispielsweise muss eine Bankverbindung für die Zah-

lungsverfahren Rechnung und Lastschrift hinterlegt werden, während eine Kreditkarte die Angabe des Anbieters, der Nummer und der Gültigkeitsdauer erfordert. Die Hinterlegund des Zahlungsverfahrens in den Nutzerprofilen stellt die Standardbelegung einer Bestellung dar, die jedoch bei jedem Bestellvorgang manuell verändert werden kann (siehe unten Szenario Bestellung). Der Anwendungsfall Kontoauszug aufrufen wird hier mit aufgeführt, da das (Debitoren-)Konto ein Bestandteil des Nutzerprofils ist. Der Kunde kann über den Kontoauszug die Abrechnung seiner Bestellungen und den Zahlungseingang seiner Überweisungen oder Lastschrifteinzüge nachvollziehen.

Eine identifizierte Kundenbeziehung mitsamt einer Authentifikation erfordert auch die Funktionalität, sich vom System wieder abzumelden. Das Abmelden erfolgt entweder manuell durch den Nutzer oder durch den Controller, der sie nach einer definierten Zeitspanne ohne Interaktion des Nutzers mit dem System automatisch durchführt.

Anhand der Schraffierung einiger Anwendungsfälle ist zu erkennen, dass die Verwaltung der Nutzerdaten sowohl durch den Betreiber als auch durch den Nutzer im Vordergrund stehen. Die erweiterten Bereiche wie die Templateverwaltung oder das Löschen eines Nutzers bilden die Systemgrenzen und werden in der Architekturgestaltung nicht mit berücksichtigt.

Die Inanspruchnahme des Angebotes durch den Nutzer umfasst grundsätzlich sowohl die Betrachtung und Selektion der Produkte als auch deren Bestellung. Da die Bestellung das zentrale Szenario des Erlösmodells EINZELTRANSAKTION darstellt, wird es separat betrachtet. Abbildung 6-8 zeigt zunächst die Anwendungsfälle der Angebotsnutzung.

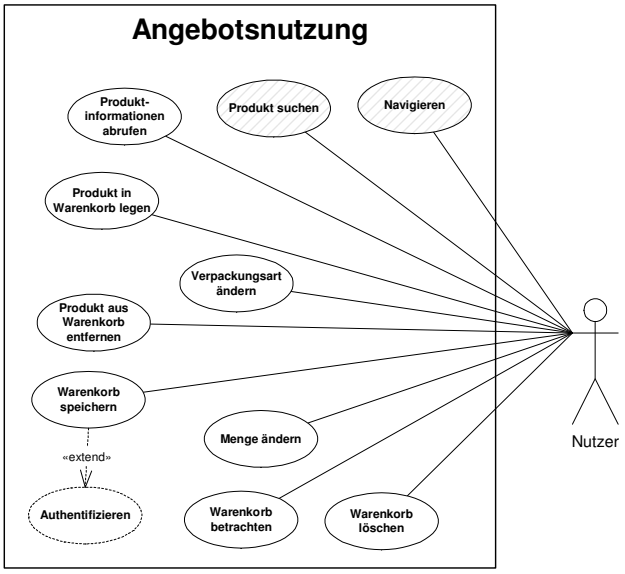


Abbildung 6-8: Anwendungsfälle der Angebotsnutzung

Der Nutzer kann durch das strukturierte Angebot des Anbieters navigieren oder gezielt nach einzelnen Produkten suchen. Bevor er ein Produkt in seinen Warenkorb legt, wird er die Produktinformationen wie Beschreibung und Preis betrachten. Er kann einzelne Produkte komplett wieder aus dem Warenkorb entfernen oder aber die Menge der Positionen im Warenkorb ändern. Durch den Anwendungsfall Verpackungsart angeben kann der Nutzer bestimmen, ob ein Produkt als Geschenk verpackt werden soll oder nicht. Um den Warenkorb über die Dauer einer Sitzung hinweg persistent zu speichern und zu einem späteren Zeitpunkt erneut aufzurufen, ist eine vorherige Anmeldung des Nutzers am System erforderlich. Neben der Betrachtung des Warenkorbinhaltes steht dem Anwender eine Funktion zur Löschung sämtlicher Positionen zur Verfügung.

Wie bereits erwähnt stellt die Bestellung als Szenario den Kern der Anwendungsfälle diese Erlösmodells dar. Die folgende Abbildung 6-9 zeigt die zugehörigen Anwendungsfälle.

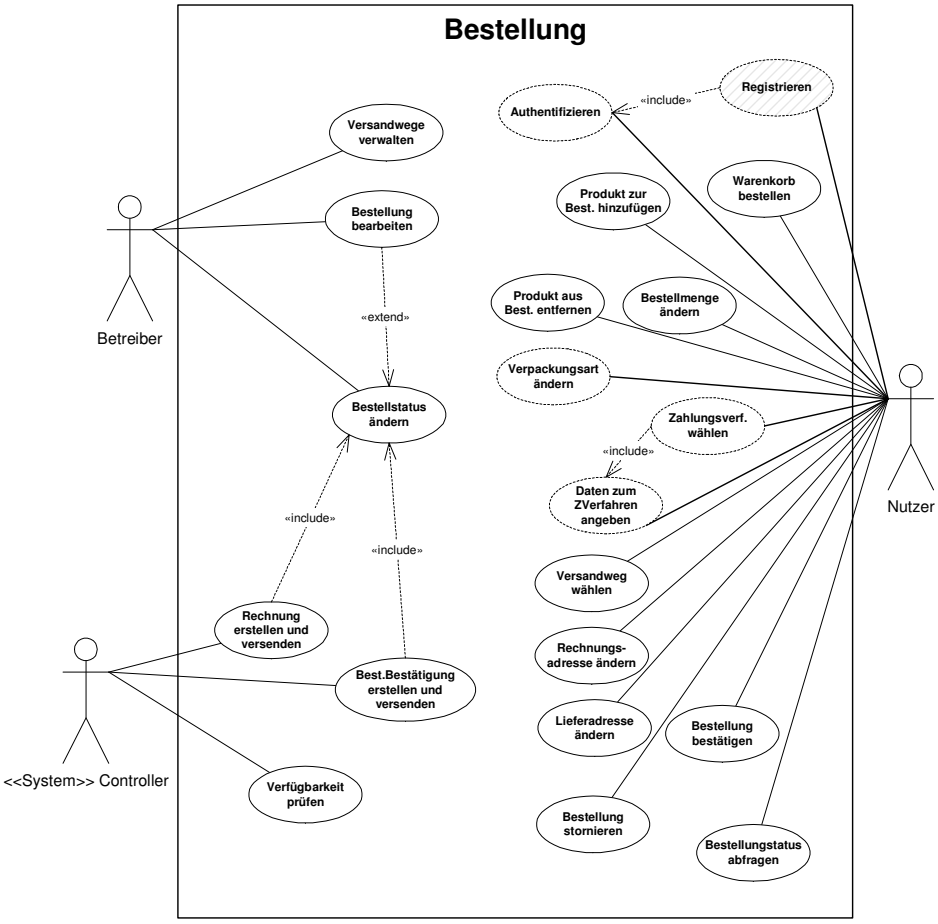


Abbildung 6-9: Anwendungsfälle der Bestellung

Der Betreiber des Systems muss zunächst die zur Verfügung stehenden Versandwege verwalten. Diese können beispielsweise neben einem Standard-Versand eine Express-Version und einen versicherten Versand beinhalten. Eine vom Kunden bestätigte Bestellung muss bearbeitet werden. Dazu gehört zum einen die Abwicklung der Logistik und zum anderen die Änderung des Bestellstatus, die damit einhergeht.

Die Anpassung des Bestellstatus erfolgt entweder manuell oder automatisiert durch einen Controller und zieht weitere Aktivitäten nach sich. Eine kommissionierte und versendete Bestellung erfordert eine Bestellbestätigung und eine Rechnung, die jeweils erstellt und versendet werden müssen. Diese Aktivitäten können mit jeweiligen Statuswechseln einhergehen und werden automatisiert abgearbeitet. Eine weitere Aufgabe des Controllers besteht in der Überprüfung der Lagerbestände während der Bestellung, da sich diese fortlaufend ändern kann und die Verfügbarkeit von Produkten durch die Option der persistenten Speicherung der Warenkörbe von der ursprünglichen Aussage gegenüber dem Kunden zum Zeitpunkt der Bestellung abweichen kann.

Der Kunde muss sich zunächst *Anmelden*, was eine vorherige *Registrierung* voraussetzt, bevor er eine Bestellung aufgeben kann. Zu einer Bestellung kann er darüber hinaus seine bestehende Profilvorgabe ändern und ein neues *Zahlungsverfahren wählen* und die notwendigen *Daten angeben*. Zu einzelnen Positionen der Bestellung kann er die zuvor gewählte *Verpackungsart ändern*. Diese Anwendungsfälle wurden bereits vorgestellt.

Eine Bestellung bezieht sich auf den Inhalt des Warenkorbs. Während der Warenkorb bestellt wird, kann der Nutzer noch einzelne Produkte neu hinzufügen oder entfernen sowie die Bestellmenge bestehender Positionen ändern. Er kann die Vorgaben seines Profils ändern und eine bestellungsspezifische Rechnungs- und Lieferadresse angeben sowie den Versandweg wählen. Diese Änderungen können solange durchgeführt werden, bis der Nutzer die Bestellung bestätigt. Nach dieser Bestätigung steht ihm dagegen nur noch die Option zur Verfügung, diese vollständig zu stornieren. Die Stornierung hängt darüber hinaus noch vom Status der Bestellung ab. Sobald die Bestellung bereits zur Kommissionierung bearbeitet wird und der Status entsprechend angepasst wird, ist eine solche Stornierung nicht mehr möglich. Den aktuellen Status der Bestellung kann der Nutzer abfragen.

Das letzte Szenario der Anwendungsfälle behandelt die Bezahlung.

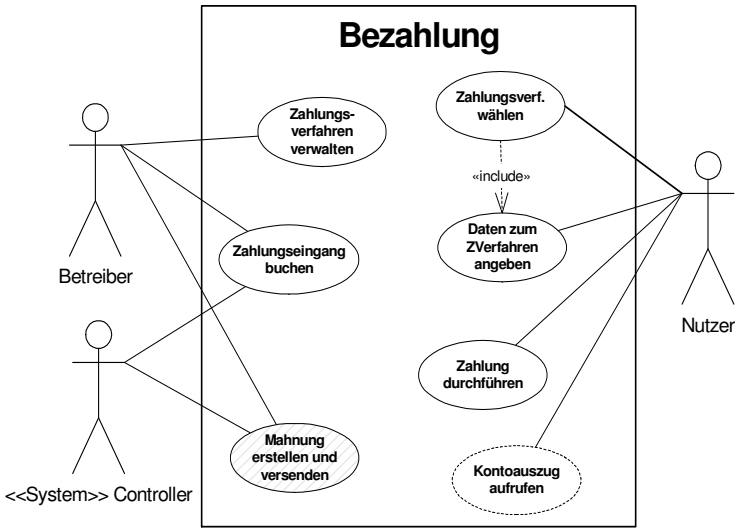


Abbildung 6-10: Anwendungsfälle zur Bezahlung

Abbildung 6-10 zeigt die Anwendungsfälle im Zusammenhang mit der Bezahlung.

Zur Durchführung der Bezahlung muss der Betreiber zunächst die unterstützten Zahlungsverfahren im System verwalten. Dazu gehört die grundsätzliche Auswahl der Verfahren sowie deren Konfiguration. Ein Zahlungseingang eines Kunden wird auf dessen Debitoren-Konto gebucht, was entweder manuell oder automatisch erfolgt. Sofern ein Kunde nicht bezahlt, sind Mahnungen zu erstellen und zu versenden. Die Schraffierung zeigt, dass dieser Bereich hier jedoch nicht näher betrachtet wird.

Der Nutzer interagiert nur begrenzt direkt mit dem System zur Durchführung der Bezahlung. Die Wahl des Zahlungsverfahren und die Angabe der erforderlichen Daten wurde bereits im Kontext der Nutzerverwaltung erwähnt. Die Durchführung der Zahlung mit dem System erfolgt jedoch nicht mit allen Verfahren. Rechnungen und Lastschriften werden beispielsweise über die Bank und nicht mit dem System abgewickelt. Die Bezahlung mittels Kreditkarte kann dagegen aus Sicht des Nutzers direkt an dem System durchgeführt werden, wobei die erforderlichen Daten eingegeben werden. Zur Kontrolle der Zahlungsflüsse dient dem Nutzer die Funktion des Kontoauszugs, über den er die Buchungen nachvollziehen kann.

Die betrachteten Anwendungsfälle zeigen im Zusammenhang mit ihrer Kennzeichnung als relevant oder nicht relevant für die Referenzarchitektur die hier betrachteten Systemgrenzen auf. Weitere Aspekte eines e-Shops, in dem das Erlösmodell EINZELTRANSAKTION vorrangig zum Einsatz kommt, wurden bewusst ausgeklammert. Beispiele sind Funktionalitäten zur Personalisierung oder die Abwicklung der Logistik und Reklamation. Diese Bereiche interagieren nicht oder nur minimal mit den Erlösmodellen und werden deshalb aus der Betrachtung ausgeblendet.

Vor dem Hintergrund der Systemabgrenzung und den relevanten Anwendungsfällen wird in den nächsten Abschnitten die Referenzarchitektur aufgezeigt und vorgestellt.

6.1.1.2 Architekturbeschreibung

Die statische Beschreibung der Referenzarchitektur umfasst als Kernstück ein Klassendiagramm, das die folgende Abbildung 6-11 zeigt. Neben der Auflistung der Klassen mitsamt ihren Beziehungen zueinander sind die Komponenten ersichtlich, denen die Klassen zugeordnet sind. Die einzelnen Komponenten werden im weiteren Verlauf nun beschrieben.

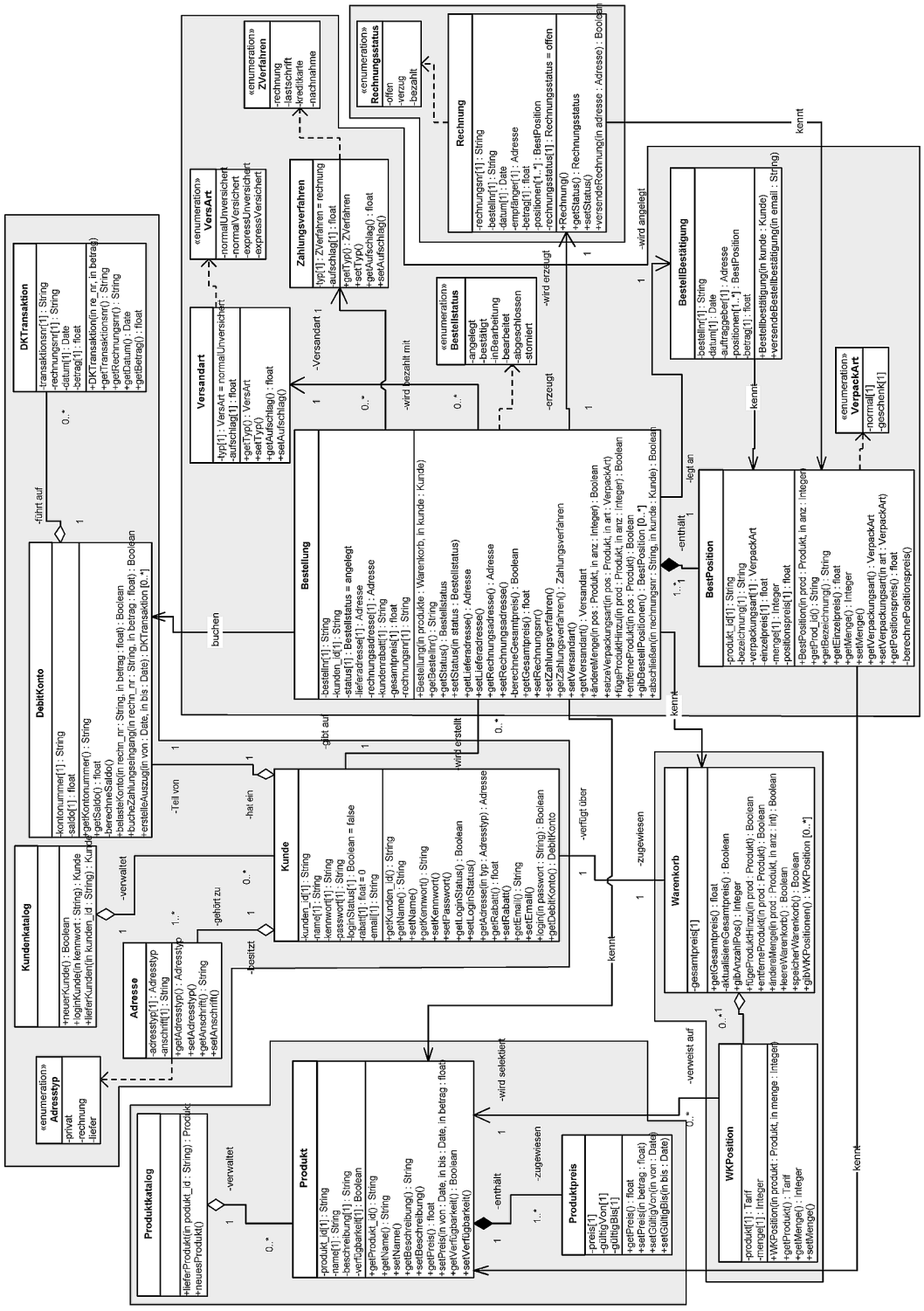


Abbildung 6-11: Statische Architektur des Erlösmodells Einzeltransaktion in der Variante physikalisches Gut

Wie aus der Anforderunsanalyse des vorherigen Abschnitts hervorgeht, wird eine Nutzerverwaltung benötigt. Die Komponente Kunde deckt diese Anforderungen ab[5]. Für eine spätere Wiederverwendung in anderen Erlösmodellen beziehungsweise zur Abgrenzung gegenüber weiteren Varianten der Komponente erhält sie die konkrete Bezeichnung `Kunde_EM1.1_ET`. Die folgende Abbildung 6-12 zeigt die Klassen der Komponente.

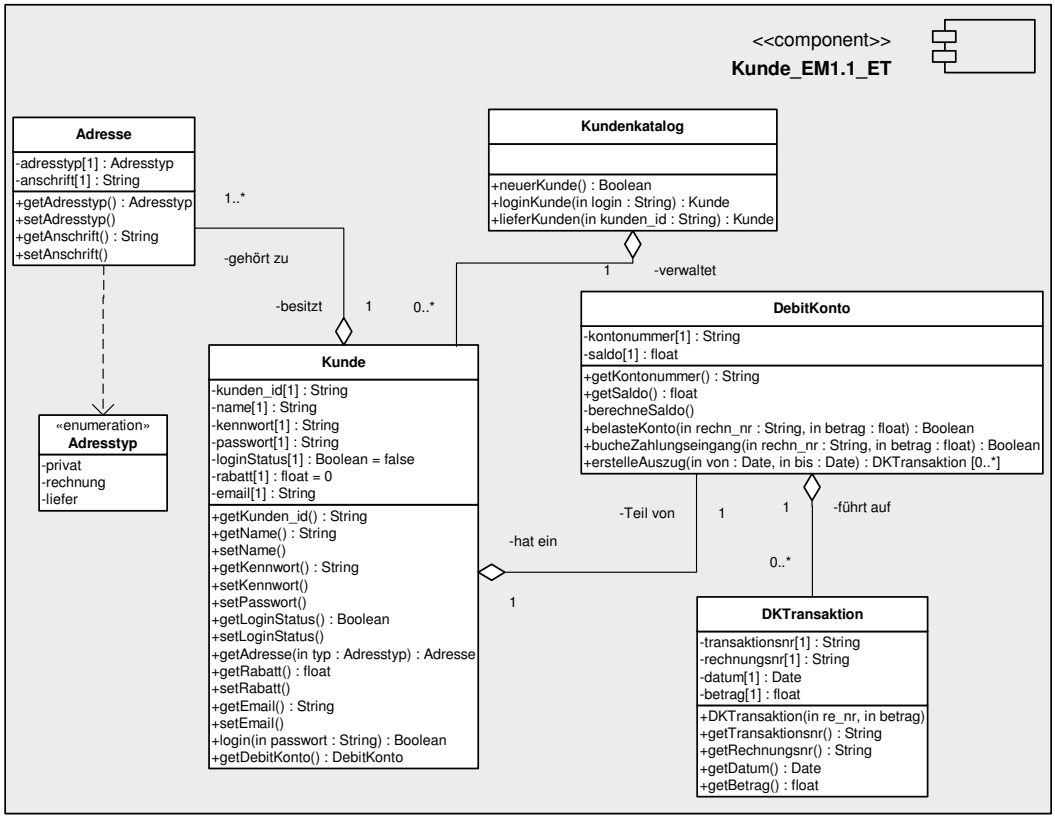


Abbildung 6-12: Komponente Kunde_EM1.1_ET mitsamt ihren Klassen

Die Komponente weist die Klassen `Kundenkatalog`, `Kunde`, `Adresse`, `DebitKonto` und `DKTransaktionen` auf. Somit werden in ihr sämtliche relevante Merkmale gekapselt.

Der `Kundenkatalog` verwaltet die registrierten Nutzer oder Kunden des Unternehmens. Diese Klasse ist hier nur ansatzweise beschrieben worden, da der Schwerpunkt der Betrachtungen auf den einzelnen Nutzerprofilen liegt, sofern sie eine Aufgabe zur Realisirung des Erlösmodells übernehmen. Es wird jedoch keine vollständige Kundenverwaltung geliefert, in der ein Kundenkatalog beispielsweise auch das Einrichten und Manipulieren der Nutzertemplates unterstützen würde. Die Methode `loginKunde` erhält als Parameter ein Kunden-Kennwort als String übergeben und liefert das Kundenobjekt zurück, dessen

5. Der Nutzer dieses Erlösmodells wird auch als Kunde bezeichnet, da er bei Einzeltransaktionen in dieser Rolle agiert.

Attribut `kennwort` mit diesem String übereinstimmt. Die Methode `lieferKunden` ermittelt das Kundenobjekt, dessen Kunden-ID (Attribut `kunden_id`) mit dem übergebenen Parameter übereinstimmt. Diese Methoden stellen somit Schnittstellen dar, über die andere Komponenten auf einzelne Kundenobjekte zugreifen können.

Die Klasse `Kunde` repräsentiert das Nutzer- beziehungsweise Kundenprofil. Nachdem der Kundenkatalog ein Kundenobjekt an die aufrufende Instanz übergeben hat, kann diese die Methode `login` aufrufen und dabei das Passwort des Kunden übergeben, so dass die Methode überprüft, ob die Authentifizierung in Form einer Übereinstimmung des Kennwort/Passwort-Wertepaares erfolgreich oder fehlerhaft ist. Die Authentifizierung der Kunden ist in diesem Erlösmodell essenziell, damit die Kaufaktivitäten eindeutig zugeordnet werden können und ein Missbrauch durch nicht legitimierte Nutzer ausgeschlossen wird.

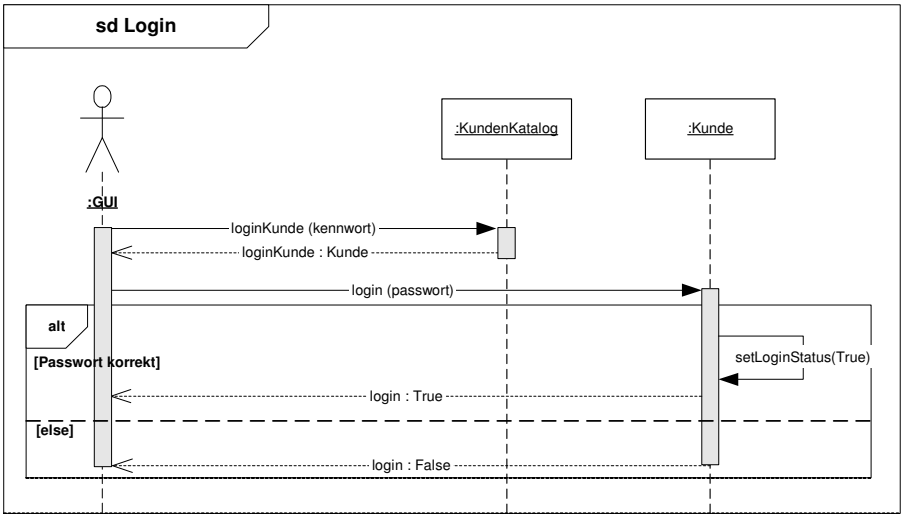


Abbildung 6-13: Sequenzdiagramm Login

Das Sequenzdiagramm in Abbildung 6-13 zeigt diesen Sachverhalt und das Zusammenspiel zwischen den Klassen `Kundenkatalog` und `Kunde`. Es wird auch ersichtlich, dass die Klasse `Kunde` bei erfolgreicher Authentifizierung den Wert des Attributes `loginStatus` entsprechend anpasst. Aus Sicherheitsgründen verfügt ein `Kunde` nur über eine `set-Passwort`, nicht aber über eine `getPasswort`-Methode, die ein Auslesen dieses kritischen Attributes ermöglichen würde. Der Abgleich des Kennwort/Passwort-Wertepaares erfolgt gekapselt in der Methode `login`. Da das Anmelden eines Nutzers gegenüber dem System im weiteren Verlauf dieser Arbeit sehr häufig durchgeführt wird, wird an diesen Stellen auf das hier vorliegende Diagramm als Interaktionsreferenz verwiesen.

Da ein Kunde verschiedene Adressen wie beispielsweise Privat-, Rechnungs- und Lieferadresse in seinem Profil hinterlegen kann, muss der Methode `getAdresse` zunächst eine genaue Typbezeichnung übergeben werden, um die entsprechenden Werte als Rückgabewert zu erhalten. Die vorgesehenen Adresstypen sind im Klassendiagramm im Enumeration-Datentyp `Adresstyp` aufgelistet.

Die Klasse `Adresse` ist in dem Diagramm ebenfalls sehr allgemein beschrieben und kann je nach konkretem Anwendungsfall an die spezifischen Anforderungen angepasst werden. Hervorzuheben ist an dieser Stelle, dass aufgrund der dargestellten Aggregation (*1:n* Multiplizität) eine Vielzahl von Adressen eines Kunden verwaltet werden können.

Die Klassen `DebitKonto` und `DKTransaktion` bilden zusammen das Debitorenkonto des Kunden, auf dem die Rechnungsbeträge der Bestellungen einerseits und die Zahlungseingänge andererseits verbucht werden. Jeder Kunde verfügt über genau ein Debitorenkonto, was durch eine *1:1* Aggregation ausgedrückt wurde.

Aus Revisionsgründen muss jede einzelne Transaktion mit ihren relevanten Details persistent gespeichert werden. Diese Aufgabe übernimmt die Klasse `DKTransation`. Da für jede Transaktion eine eigene Instanz der Klasse `DKTransaktionen` existiert, besteht eine 1:n Assoziation mit der Klasse `DebitKonto`. Die Methode `belasteKonto` der Klasse `DebitKonto` erhält als Parameter eine korrespondierende Rechnungsnummer und den zu buchenden Betrag übergeben. Sie ruft den Konstruktor `DKTransaktion` auf und übergibt die Paramter, wodurch dieser einen neuen Datensatz anlegt. Abschließend ruft die Methode die private Methode `berechneSaldo` auf, die den aktuellen Kontostand nach der zusätzlichen Transaktion ermittelt. Analog wird die Methode `bucheZahlungseingang` zur Gegenbuchung verwendet. Um einen Kontoauszug zu erhalten, stellt das Debitorenkonto die Methode `erstelleAuszug` bereit, die als Eingabeparameter den Zeitraum benötigt. Die Verwendung beider Klassen und ihr Zusammenspiel mit anderen Klassen erfolgt unter anderem in Zusammenhang mit der Bestellabwicklung und wird im weiteren Verlauf noch detaillierter in Form von Sequenzdiagrammen vorgestellt.

Die Anforderungen an eine Produktverwaltung werden mit der Komponente `Produkt_EM1.1_ET` abgedeckt.

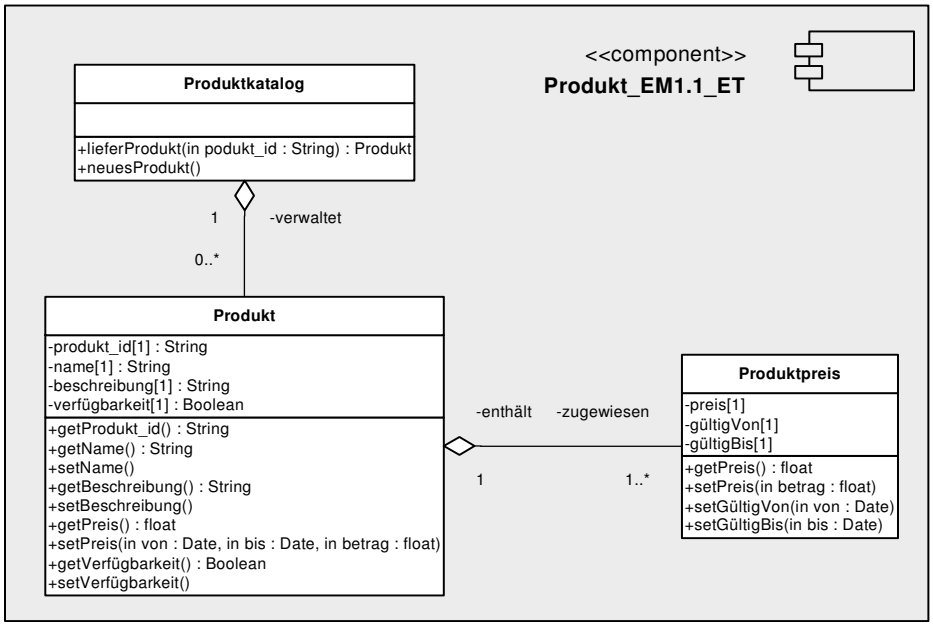


Abbildung 6-14: Komponente Produkt_EM1.1_ET mitsamt ihren Klassen

Die Abbildung 6-14 weist die Klassen `Produktkatalog`, `Produkt` und `Produktpreis` auf, die gemeinsam die Komponente repräsentieren.

Wie der Kundenkatalog für die Objektverwaltung der Kunden zuständig ist, übernimmt die Klasse `Produktkatalog` diese Aufgabe für die `Produkte`. Die Methode `liefer-Produkt` bekommt als Parameter eine Produkt-ID übergeben und liefert das zugehörige Produktobjekt zurück.

Damit bei einem `Produkt` nicht nur ein Preis fest zugewiesenen werden kann, sondern zur zeitlichen Planung von Preiserhöhungen oder Sonderaktionen mehrere Preise hinterlegt werden können, wird der Preis über die Klasse `Produktpreis` verwaltet. Jedem Preisobjekt können mehrere Produktpreise zugewiesen werden, wobei sie über die Attribute `gültigVon` und `gültigBis` terminiert sind und durch die Programmlogik sichergestellt werden muss, dass die Gültigkeitsdauern der Preise überschneidungsfrei sind. Die Ein- und Ausgabe der Preise erfolgt über die Klasse `Produkt` mittels der passenden `get-Preis` und `setPreis`-Methoden. `getVerfügbarkeit` gibt Auskunft darüber, ob ein Produkt zum Zeitpunkt der Bestellung im Lager vorrätig ist oder nicht. Diese Methode stellt die sehr vereinfachte Form einer Schnittstelle zu einem Warenwirtschaftssystem dar, das hier jedoch nicht in die Betrachtung der Architektur mit einbezogen wird.

Eine weitere Komponente ist der Warenkorb. Über den Warenkorb hat der Nutzer die Möglichkeit, ihn interessierende Produkte abzulegen. Bevor er sich endgültig für den Kauf des Warenkorbinhaltes entscheidet, können beliebig viele Produkte hinzugefügt, wieder ent-

fernt und persistent gespeichert werden. Die Komponente wird in Abbildung 6-15 dargestellt und enthält die beiden Klassen `Warenkorb` und `WKPosition`. Ihre vollständige Bezeichnung ist `Warenkorb_EM1.1_ET`.

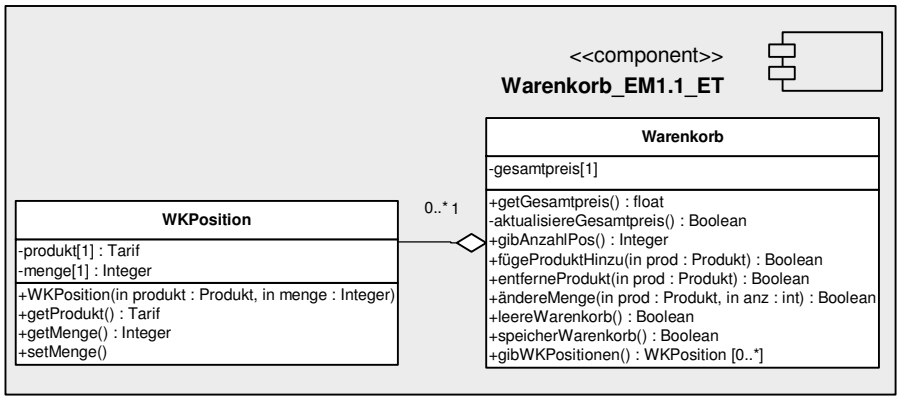


Abbildung 6-15: Komponente Warenkorb_EM1.1_ET mitsamt ihren Klassen.

Einem Warenkorb sind *n* Positionen zuzuordnen. Eine neue Instanz von WKPositionen wird erzeugt, indem die Methode `fügeProduktHinzu` der Klasse `Warenkorb` mitsamt einer Referenz auf das entsprechende Produkt als Parameter aufgerufen wird und diese Methode ihrerseits den Konstruktor `WKPositionen` aufruft. Die Klasse Warenkorb verwaltet als einziges Attribut den `gesamtpreis`, der Auskunft gibt über den Preis aller enthaltenen Positionen in der gewählten Menge. Dieses Attribut wird nach jedem Hinzufügen (`füge-ProduktHinzu`), Entfernen (`entferneProdukt`) und Ändern der Menge einer Position (`ändereMenge`) über die private Methode `aktualisiereGesamtpreis` angepasst. Während die Methode `entferneProdukt` nur das als Parameter überlieferte Produkt aus dem Warenkorb entfernt, können mit der Methode `leereWarenkob` alle Positionen entfernt werden.

Ein Speichern des Warenkorbes (`speicherWarenkorb`) setzt eine Authentifizierung des Nutzers voraus. Das folgende Sequenzdiagramm in Abbildung 6-16 zeigt diese Abhängigkeit der Warenkorbspeicherung.

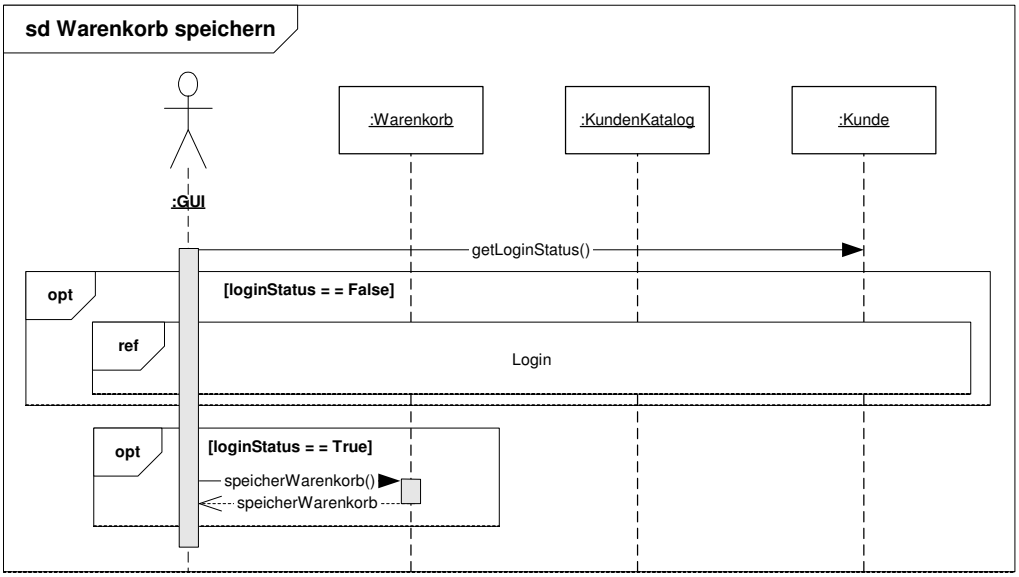


Abbildung 6-16: Die Speicherung des Warenkorbs erfordert einen authentifizierten Kunden

Die verwendeten optionalen Fragmente werden nur durchlaufen, wenn die zugehörige Bedingung erfüllt ist. Da es sein kann, dass der Login-Status eines Kunden auch nach dem Durchlauf der Login-Prozedur noch nicht *True* ist, wird die Speicherung nicht notwendigerweise durchgeführt. Stattdessen würde eine entsprechende Rückmeldung an die GUI erfolgen, was in dem Sequenzdiagramm jedoch zur besseren Übersichtlichkeit nicht abgebildet wurde.

Die Methode `gibWKPositionen` der Klasse `Warenkorb` übergibt der aufrufenden Instanz als Rückgabewert ein Listobjekt über alle Positionen des Warenkorbs, während `gibAnzahlPos` nur die Menge als Wert zurückliefert. Die Methoden werden zur Abwicklung der Bestellung benötigt und ihr Zusammenspiel wird im weiteren Verlauf noch dargestellt.

Die zentrale Komponente des Erlösmodells EINZELTRANSAKTION ist die Bestellung (`Bestellung_EM1.1_ET`). Sie ist dafür zuständig, dass die im Warenkorb des Kunden selektierten Produkte in eine Bestellung überführt und bearbeitet werden. Die Prozesse ausgehend von der initialen Bestellung über die Statusanpassungen während der Abarbeitung bis hin zur Erstellung und dem Versand der Bestellbestätigung sowie der Rechnung mitsamt der Belastung des Kundenkontos werden mit ihrer Hilfe durchgeführt. Abbildung 6-17 zeigt die Komponente mit ihren Klassen `Bestellung`, `Bestellpositionen`, `Versandart`, `Zahlungsverfahren` und `Bestellbestätigung`.

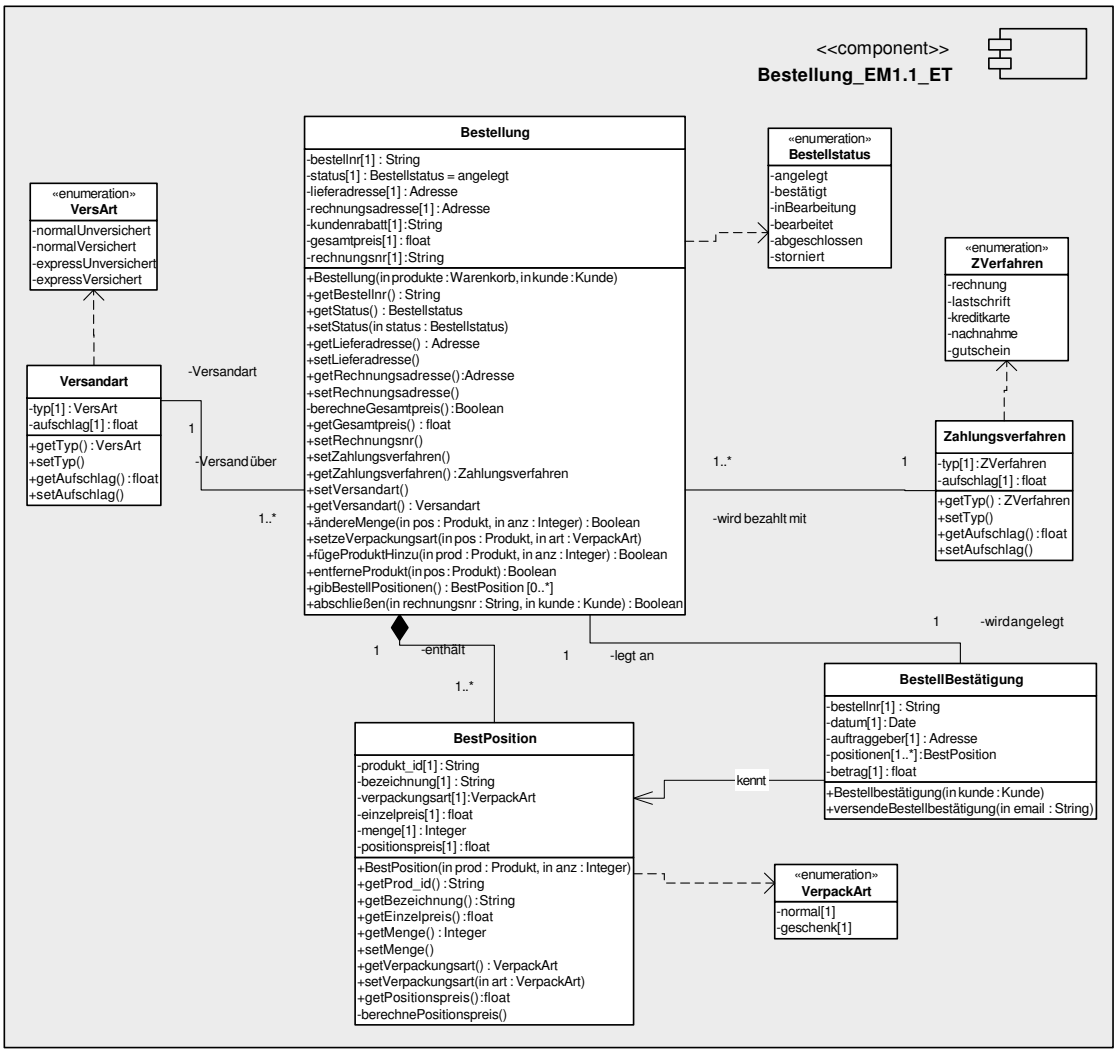


Abbildung 6-17: Komponente Bestellung_EM1.1_ET mitsamt ihren Klassen

Eine Bestellung durchläuft mehrere Status. In der Architektur dient dazu das Attribut `status` der Klasse `Bestellung` vom Enumeration-Datentyp Bestellstatus. Zunächst wird eine Bestellung *angelegt*. Dies ist die initiale Beabsichtigung des Kunden, die Positionen seines Warenkorbs zu bestellen. Zu diesem Zeitpunkt ist es dem Kunden jedoch noch möglich, einzelne Produkte zu der Bestellung hinzuzufügen oder zu entfernen oder grundsätzliche Angaben wie die Versandart oder das Zahlungsverfahren zu bestimmen. Erst, wenn die Bestellung *bestätigt* wurde, sind alle Angaben fix und es können keine Veränderungen mehr vorgenommen werden. Der Kunde hat lediglich die Option, eine Bestellung zu *stornieren*, solange sie noch nicht zur Kommissionierung durch den Betreiber in den Status *in Bearbeitung* überführt wurde. Nach erfolgreicher Bearbeitung wird die Lieferung veranlasst und die Bestellung als *bearbeitet* markiert. Dieser Status initiiert die Erstellung einer Bestellbestätigung sowie einer Rechnung. Sobald die Lieferung das Lager verlassen hat und der Rechnungsbetrag auf dem Debitoren-Konto des Kunden verbucht wurde,

erhält die Bestellung den Status *abgeschlossen*. Wie noch im weiteren Verlauf dieses Abschnittes gezeigt werden wird, übernimmt die Methode `setStatus` der Klasse `Bestellung` eine zentrale Rolle in der Abarbeitung einer Bestellung mitsamt den durchzuführenden Aktionen und Methodenaufrufen in anderen Klassen ein.

Die Bestellung stellt ein wesentliches Element bei der Realisierung dieses Erlösmodells dar, indem sie die eigentliche Erlösgenerierung repräsentiert. Der Kunde bestellt Produkte und wird als Bezahlung einen entsprechenden Betrag an das Unternehmen zahlen. Aus diesem Grund werden der Ablauf und die Interaktionen der involvierten Klassen nun sehr detailliert vorgestellt. Zunächst zeigt das folgende Sequenzdiagramm in Abbildung 6-18 die Interaktion der Klassen während der initialen Erzeugung der Bestellung.

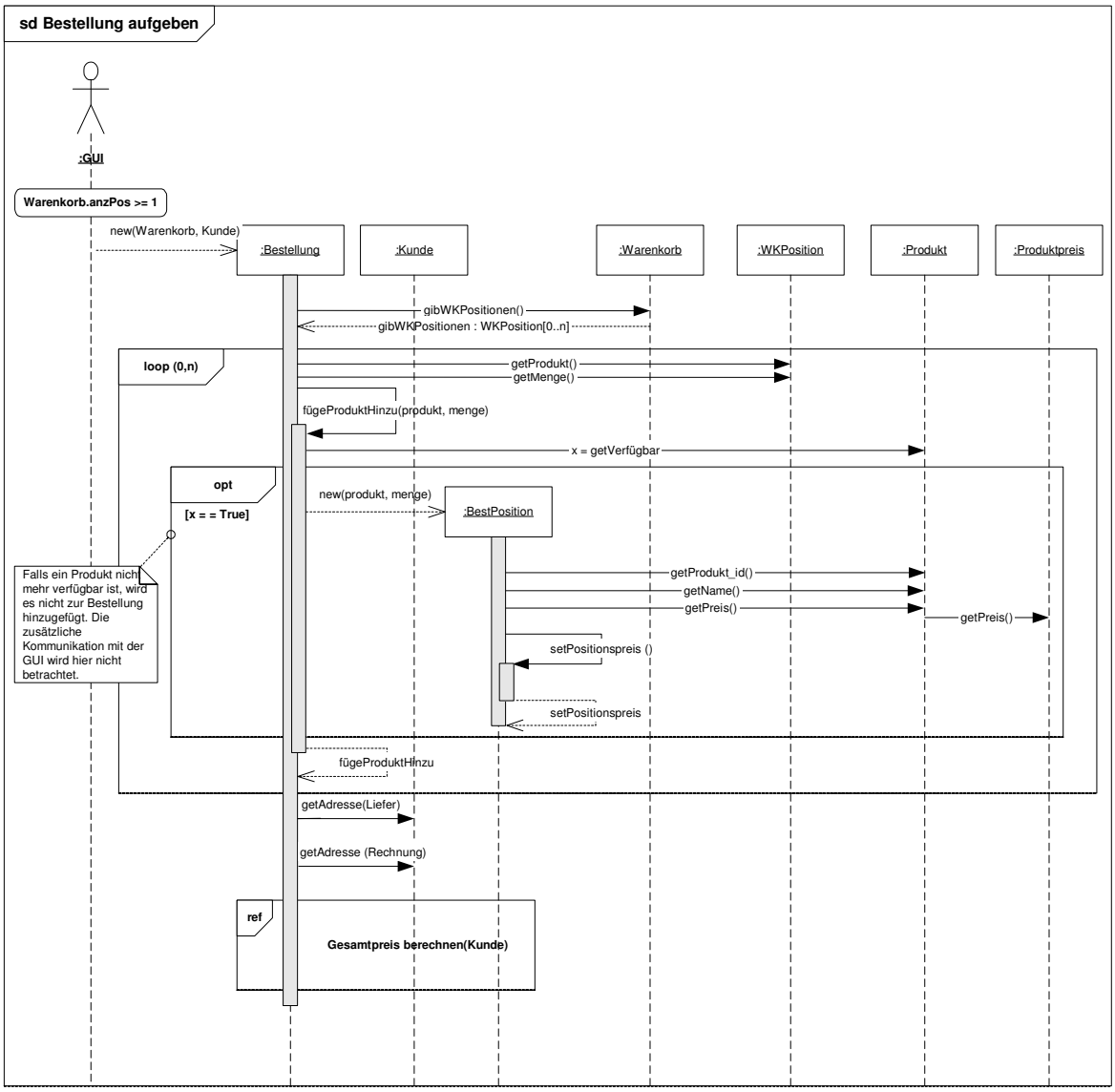


Abbildung 6-18: Interaktion der Klassen während der initialen Aufgabe einer Bestellung

Der Konstruktor der `Bestellung` wird aufgerufen und erhält als Parameter den Kunden und seinen Warenkorb als Referenzen überliefert. Als erstes ermittelt die Bestellung die Warenkorbpositionen über die entsprechende Methode (`gibWKPositionen`) der Klasse `Warenkorb`. Als Rückgabewert erhält sie eine Liste der Positionen. In einer Schleife wird nun jede einzelne Position des Warenkorbs durchlaufen. Zunächst wird die Produktreferenz und die zu bestellende Menge ermittelt (`getProdukt` und `getMenge`) und der Methode `fügeProduktHinzu` als Parameter übergeben.

Diese Methode überprüft zu Beginn die Verfügbarkeit des Produktes (`getVerfügbarkeit`) und legt bei positiver Rückmeldung eine neue Instanz der Klasse `BestPosition` an. Diese Überprüfung ist notwendig, da ein Kunde seinen Warenkorb langfristig speichern kann und somit erst zum Zeitpunkt der Bestellung eine endgültige Verfügbarkeit ermittelt werden kann. Im Falle einer negativen Rückantwort müsste eine Routine einen geeigneten Hinweis an den Nutzer leiten und ein hier nicht definiertes Verfahren angewandt werden. Vorstellbar ist beispielsweise, dass die Bestellung normal angelegt wird und in einen sofort und in einen später lieferbaren Teil aufgesplittet wird. Ein konkretes Verfahen ist dabei aber abhängig vom konkreten Anwendungsfall und wird hier mit Rücksicht auf den Referenzcharakter der Architektur sowie den Umfang möglicher Alternativen nicht weiter betrachtet.

Die Berechnung der Preise erweist sich als komplex. Der Grund dafür liegt darin, dass durch die Architektur sowohl eine Realisierung temporär gültiger Produktpreise als auch verschiedene Verpackungsarten unterstützt werden, die zusätzliche Mehrkosten verursachen können. Diese Preisberechnung wird von der Klasse `BestellPosition` durchgeführt. Der Konstruktor `BestPosition` ermittelt anhand der ihm übergebenen Parameter die relevanten Attribute des Produktes und ruft die Methode `setPositionspreis` auf. Diese Methode ermittelt den Preis einer einzelnen Position unter Berücksichtigung des `Einzelpreises`, der `Menge` und der `Verpackungsart`. Die Verpackungsart ist bei der initialen Aufnahme der Bestellung noch nicht explizit verändert worden und enthält den Wert *normal*, kann jedoch später manuell durch den Kunden auf *geschenk* angepasst werden. Die Architektur sieht eine mögliche Belegung des Enumeration-Datentypen `Verpackungsart` mit diesen beiden Werten vor und kann bei Bedarf angepasst werden. Die Methode `setVerpackungsart` führt ebenfalls eine Aktualisierung des Positionspreises unter Verwendung der Methode `setPositionspreis` durch und in diesem Fall kann sich der Preis je nach Verpackungsart ändern.

Nachdem alle Positionen des Warenkorbs auf diese Weise in Bestellpositionen überführt worden sind, ermittelt die Klasse `Bestellung` die als Standard hinterlegten Liefer- und Rechnungsadressen des Kunden aus dessen Profil (`getAdresse`) und belegt die entsprechenden Attribute. Der Kunde hat die Möglichkeit, diese Vorbelegung anschließend für eine einzelne Bestellung zu ändern, so dass die Methoden `setLieferadresse` und `setRechnungsadresse` verfügbar sind. Wie oben bereits erwähnt, ist diese Änderungsoption abhängig vom Status der Bestellung.

Abschließend wird die Gesamtpreisberechnung durchgeführt. Die Referenzarchitektur ist derart gestaltet, dass hierbei sowohl Kundenrabatte, Zusatzgebühren je Zahlungsverfahren und auch Gebühren für unterschiedliche Versandarten berücksichtigt werden. Auf-

grund dieser Komplexität zeigt das folgende Sequenzdiagramm in Abbildung 6-19 diesen Ablauf im Detail. Auf das Diagramm wurde zur besseren Übersichtlichkeit in Abbildung 6-18 referenziert.

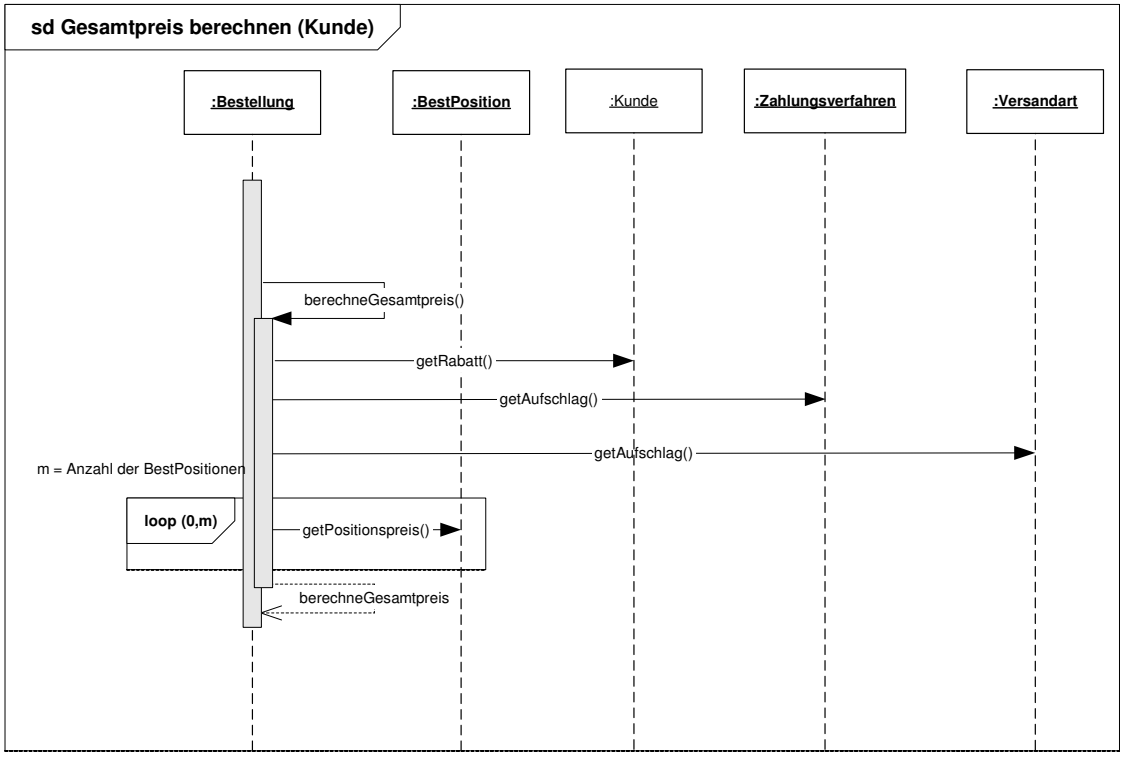


Abbildung 6-19: Sequenzdiagramm zur Berechnung des Gesamtpreises der Bestellung

Die Gesamtpreisberechnung wird von der Methode `berechneGesamtpreis` durchgeführt. Der Rabatt des Kunden ist in dessen Stammdaten hinterlegt, die Aufschläge in den Klassen `Zahlungsverfahren` und `Versandart`. Wie dem Klassendiagramm in Abbildung 6-17 zu entnehmen ist, werden zunächst Defaultwerte unterstellt, die der Kunde ebenso wie die Liefer- und Rechnungsadresse abhängig vom Status der Bestellung ändern kann. Die Preise der einzelnen Bestellpositionen, die abhängig sind vom Produkt, von der Menge und von der Verpackungsart werden abschließend durch die Methode `getPositionspreis` ermittelt, so dass der vollständige Gesamtpreis unter Berüscksichtigung aller Einflussfaktoren berechnet werden kann.

Nach der initialen Anlage der Bestellung sind somit alle Attibute der Klasse belegt und die Bestellpositionen mittels Instanzen der Klasse `BestPosition` zugeordnet. Der `status` der Bestellung ist auf *angelegt* gesetzt. In diesem Zustand hat der Kunde nun die Möglichkeit, Modifikationen durchzuführen. Die Methoden `setZahlungsverfahren` und `setVersandart` sind selbsterklärend und führen neben einer Aktualisierung der Werte eine Neuberechnung des Gesamtpreises durch. Dazu rufen sie erneut die Methode `berechneGesamtpreis` auf, deren Ablauf oben beschrieben wurde. Des Weiteren kann der Kunde Produkte zu der Bestellung hinzufügen (`fügeProduktHinzu`), gänzlich von ihr entfernen

(entferneProdukt), oder die Bestellmenge einzelner Positionen ändern (ändere-Menge). Diese Methoden interagieren jeweils mit der entsprechenden Instanz von Best-Position (wobei fügeProduktHinzu eine neue Instanz erzeugt) und rufen abschließend ebenfalls zur Aktualisierung des Preises berechneGesampreis auf. Diese wird auch von der Methode setzeVerpackungsart aufgerufen. setzeVerpackungs-art empfängt als Parameter das Produkt und die Verpackungsart und ruft die Methode setVerpackungsart der zuvor zu identifizierenden Instanz von BestPosition auf.

Die Methode setVerpackungsart der Klasse BestPosition aktualisiert das entsprechende Attribut und ruft anschließend die private Methode berechnePositionspreis auf, um den Preis abhängig von dieser Verpackung neu zu berechnen und ebenfalls das zugehörige Attribut positionspreis zu aktualisieren.

Sobald der Kunde die Bestellung bestätigt und der Status entsprechend geändert wird, wird eine solche Bestätigung erstellt und dem Kunden zugestellt. Die beteiligte Klasse ist vorrangig die BestellBestätigung. Die folgende Abbildung 6-20 zeigt das zugehörige Sequenzdiagramm, um das Zusammenspiel der beteiligten Klassen zu verdeutlichen.

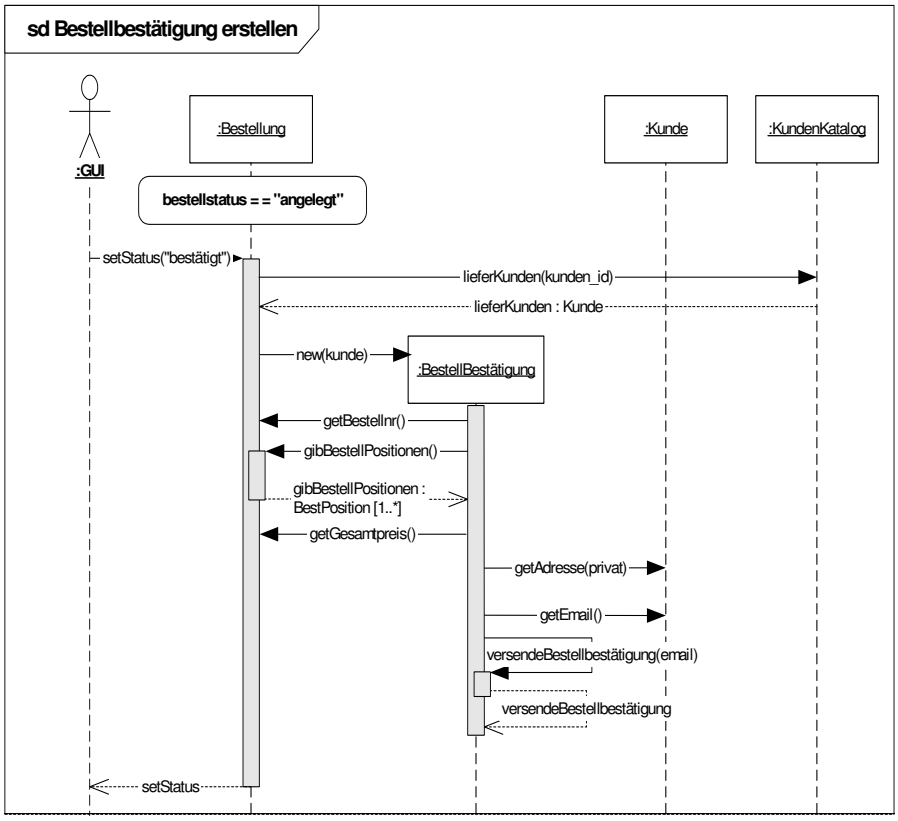


Abbildung 6-20: Sequenzdiagramm zum Erzeugen einer Bestellbestätigung

Dem Diagramm ist zu entnehmen, dass zunächst der Status der Bestellung aktualisiert wird und die Methode `setStatus` in Abhängigkeit des Wertes (hier: *bestätigt*) den Konstruktor von `BestellBestägitung` aufruft und eine Kundenreferenz übergibt. `BestellBestätigung` ruft die Methode `gibBestellPositionen` auf und erhält als Rückgabewert ein Listobjekt über alle Positionen, die sie zur Auflistung benötigt (dazu dient das mengenwertige Attribut `positionen[1..*]`). Vom Kunden wird die Privatadresse (Attribut `auftraggeber`) und die email ermittelt, an die die Bestätigung verschickt wird (`versendeBestellbestätigung`).

Die unternehmensinterne Verarbeitung der Bestellung in Form der Kommissionierung und des Versands wird hier nicht weiter untersucht. Sobald die Bestellung jedoch komplett bearbeitet wurde, muss eine Rechnung erstellt werden. Dazu steht die Komponente `Rechnung_EM1.1_ET` zur Verfügung, die die Klasse `Rechnung` enthält und in der nächsten Abbildung dargestellt ist. Die Funktionalität der Rechnungsstellung wurde hier bewusst in einer eigenständigen Komponente realisiert und nicht innerhalb der Komponente `Bestellung` umgesetzt. Grund dafür ist die Tatsache, dass über viele verschiedene Erlösmodelle hinweg Rechnungen zu erstellen sind und deshalb diese Komponente häufiger in dieser oder in leicht abgewandelter Form wiederverwendet werden kann.

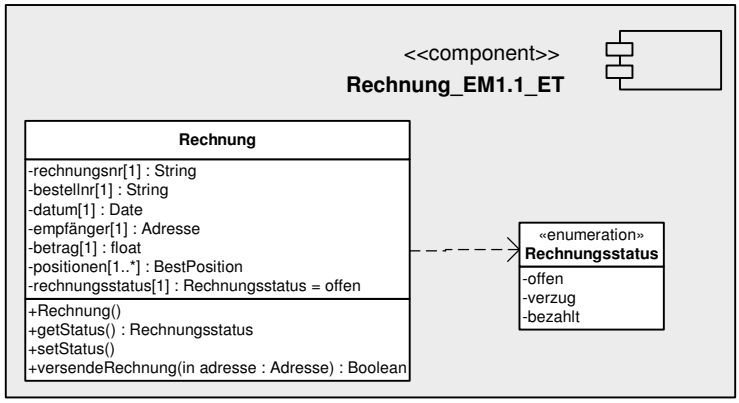


Abbildung 6-21: Komponente Rechnung_EM1.1_ET mit ihrer Klasse

Das Attribut `rechnungsstatus` der Klasse `Rechnung` kann die Ausprägungen *offen*, *verzug* und *bezahlt* annehmen. Initial erhält es vom Konstruktor den Wert *offen*. Wird die Rechnung nach einer bestimmten Frist nicht vom Kunden beglichen, wird sie in *verzug* gesetzt und initiiert damit ein Mahnverfahren. Dieser Prozess wird an dieser Stelle nicht weiter betrachtet und soll entsprechend der Systemabgrenzung des vorherigen Abschnitts (siehe Abbildung 6-10 auf Seite 130) das Mahnwesen nur andeuten. Der Status bezahlt wird bei Abschluss der Bestellung automatisch gesetzt, wie weiter unten (Abbildung 6-24 auf Seite 147) gezeigt wird.

Die wesentlichen Methoden der Klasse `Rechnung` sind ihr Konstruktor sowie `versendeRechnung`. Ihre Funktionsweise wird im Zusammenhang mit dem folgenden Sequenzdiagramm deutlich. Dieses Diagramm zeigt im Detail, wie die Rechnung das Bindeglied zwischen der Leistungserbringung (hier realisiert durch die Bestellungs-Komponente) und

der Kundenbelastung (realisiert durch Kunden-Komponente) bildet. Da der Prozess der Rechnungsstellung ein wesentlicher Vorgang im Rahmen der Realisierung von Erlösmodellen ist, wird sie auch im weiteren Verlauf detailliert betrachtet.

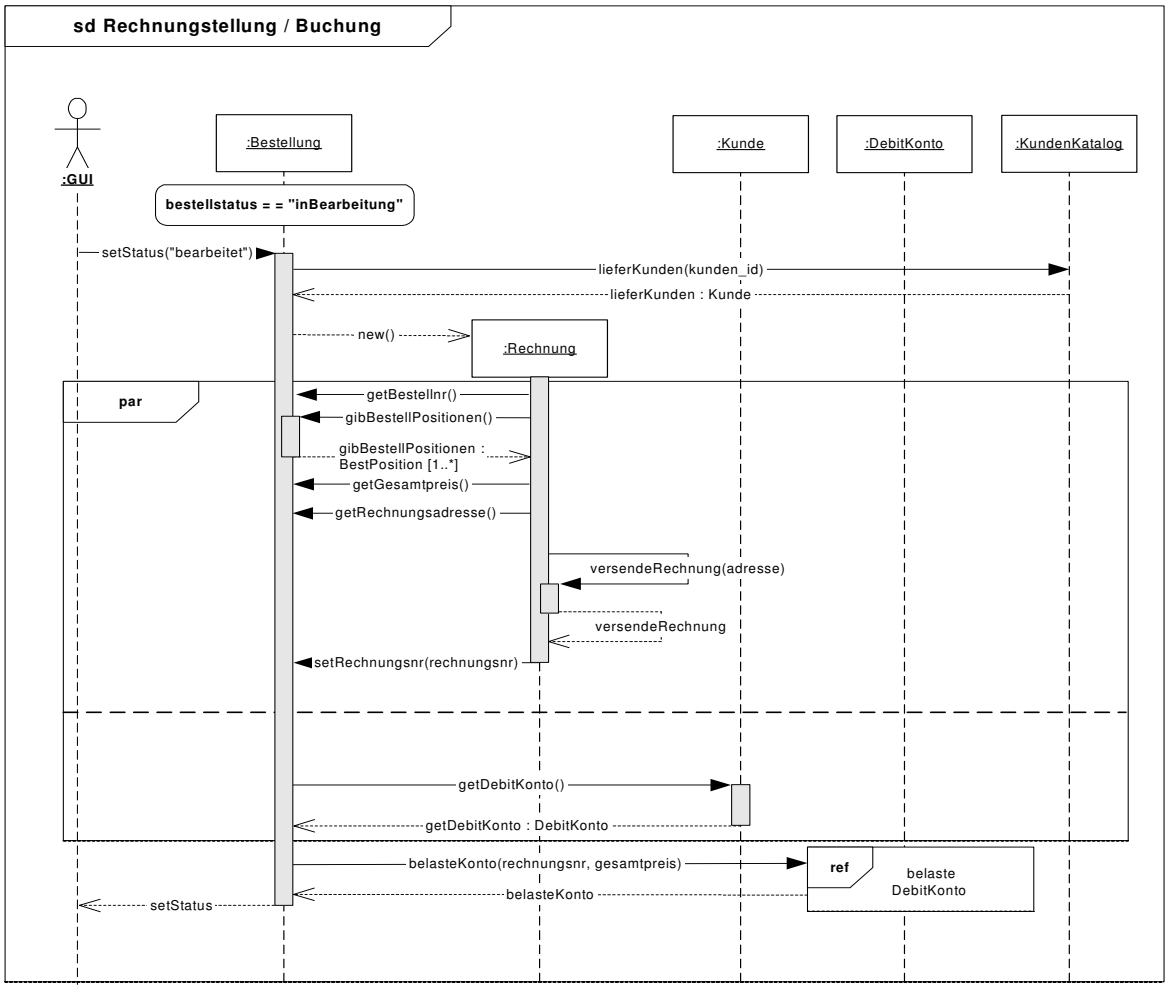


Abbildung 6-22: Sequenzdiagramm zur Rechnungsstellung und Buchung

Auch im Rahmen der Rechnungsstellung übernimmt die Methode `setStatus` eine wesentliche Steuerungsfunktion. Sobald der Rechnungsstatus auf *bearbeitet* geändert wird, erfolgt die Rechnungsstellung. Zunächst wird der Konstruktor `Rechnung` aufgerufen. Das Sequenzdiagramm zeigt durch die Verwendung des parallelen Fragments (zu erkennen am Kürzel *par*), dass die folgenden Interaktionen zwischen den Klassen `Rechnung` und `Bestellung` einerseits und zwischen `Bestellung und Kunde` andererseits nebenläufig sein können. Der Methodenaufruf `belasteKonto` kann jedoch erst erfolgen, nachdem die Rechnungsnummer in der `Bestellung` gesetzt wurde (`setRechnungsnr`). Nach ihrer Instanziierung ruft die Rechnung die für sie notwendigen Informationen über die get-Methoden von `Bestellung` ab und setzt ihre Attributwerte. Die Rechnungsadresse wird nicht vom Kunden, sondern ebenfalls von der Bestellung ermittelt,

da ein Kunde seine Standardvorgabe aus den Profildaten für einzelne Bestellungen ändern kann. Abschließend sorgt die `Rechnung` für den Versand einer physikalischen Rechnung an den Kunden mittels der `versendeRechnung` Methode.

Das Sequenzdiagramm in Abbildung 6-22 zeigt darüber hinaus, dass mit Erstellung der Rechnung auch die Belastung des Kundenkontos erfolgt. Dazu „holt sich“ die Methode `setStatus` zunächst vom `Kunden` dessen `DebitKonto` über dessen Methode `getDebitKonto` und ruft dann die Methode `belasteDebitKonto` vom `DebitKonto` auf. Als Parameter übergibt sie dabei die Rechnungsnummer und den zu buchenden Betrag. Da diese Funktionalität im Rahmen dieser Arbeit noch häufiger verwendet wird, verweist das Diagramm an dieser Stelle auf eine Interaktionsreferenz. Diese ist in der nächsten Abbildung dargestellt.

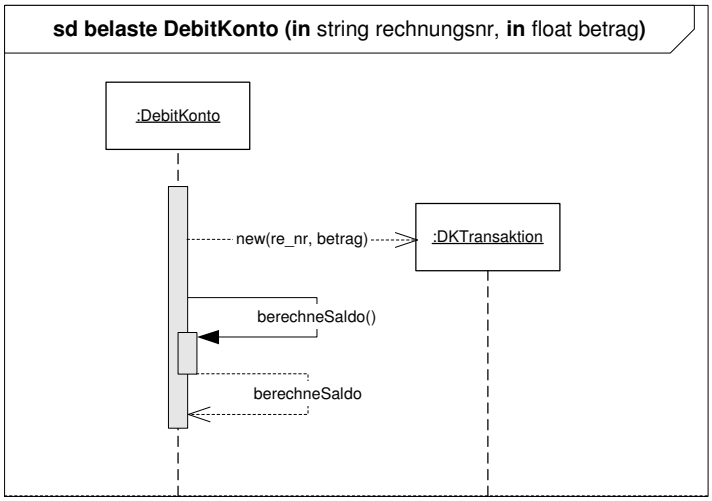


Abbildung 6-23: Sequenzdiagramm zur Kontobelastung als Interaktionsreferenz

Das Debitoren-Konto des Kunden ruft den Konstruktor der Klasse `DKTransaktion` auf und übergibt beide Parameter, die der Konstruktor den zugehörigen Attributen zuweist. Abschließend wird die Methode `berechneSaldo` aufgerufen, die den neuen Kontostand ermittelt.

Um den Geschäftsvorfall zu beenden, ist als letzer Schritt der tatsächliche Zahlungseingang des Kunden zu verbuchen und damit die Bestellung abzuschließen. Der Eingang des Geldes kann je nach verwendetem Zahlungsverfahren einige Zeit in Anspruch nehmen. Die in der Referenzarchitektur verwendeten Zahlungsverfahren Rechnung, Lastschrift, Kreditkarte und Nachnahme werden mit der Hilfe von Banken durchgeführt, so dass die Buchung der Zahlungseingänge zeitlich versetzt erfolgt und über eine Schnittstelle je nach Kontrolle voll- oder teilautomatisiert aktiviert wird. Sobald jedoch der Zahlungseingang erfolgt, wird der Status der `Bestellung` angepasst und die erforderlichen Prozesse angestoßen. Das Sequenzdiagramm in Abbildung 6-24 zeigt den Ablauf und das Zusammenspiel der Klassen und ihrer Methoden.

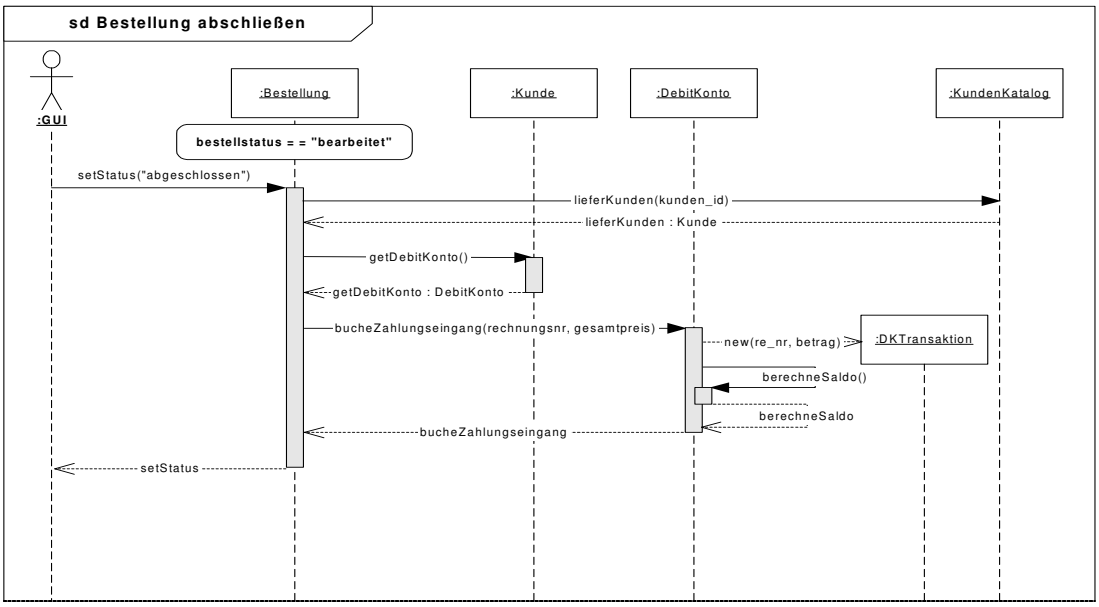


Abbildung 6-24: Sequenzdiagramm zum Abschluss einer Bestellung

Auch hier übernimmt die Methode `setStatus` die steuernde Funktion. Sie setzt zunächst den Status der Bestellung auf *abgeschlossen*. Anschließend ermittelt sie den zugehörigen Kunden und fordert von der überlieferten Instanz das zugehörige `DebitKonto` an. Dort wird der Zahlungseingang mit Hilfe der Methode `bucheZahlungseingang` verbucht. Analog zur Kontobelastung (vgl. Abbildung 6-23) wird auch hier eine neue Transaktion angelegt, indem `DebitKonto` eine neue Instanz von `DKTransaktion` generiert. Der Saldo des Debitoren-Kontos wird abschließend neu kalkuliert (`berechneSaldo`) und spiegelt den Stand nach Abschluss der Einzeltransaktion wider.

Die vorgestellten Komponenten mitsamt ihren Klassen bilden somit die Referenzarchitektur für das Erlösmodell EINZELTRANSAKTION in der Variante für physikalische Güter. Im nächsten Abschnitt wird die Architektur der Variante untersucht, die mit digitalen Gütern handelt und damit anonyme Kundenbindungen ermöglicht.

6.1.2 Variante Digitales Gut

Diese Variante, die auch als *Pay per Access* bezeichnet werden kann, wurde bereits in Abschnitt 4.2.1.2 auf Seite 67 detailliert vorgestellt. Sie weist folgende Ausprägungen der Beschreibungskriterien auf:

Abbildung 6-25: Erlösmodellklassifikation Einzeltransaktion Variante digitales Gut

Durch die anonyme Kundenbindung wird in dieser Variante unterstellt, dass ausschließlich digitale Zahlungsverfahren unter Verwendung eines entsprechenden externen Dienstleisters zur Verfügung stehen. Die Voraussetzung für eine anonyme Kundenbindung ist durch den Verkauf digitaler Güter gewährleistet und das Preisniveau ermöglicht den Einsatz des digitalen Zahlungsverfahrens. Digitale Güter können grundsätzlich auch über identifizierte Kundenbeziehungen veräußert werden, so dass auch die Zahlungsverfahren *Rechnung*, *Lastschrift* oder *Kreditkarte* eingesetzt werden könnten. In diesem Fall entspräche die Referenzarchitektur weitestgehend der Variante für physikalische Güter, die im vorherigen Abschnitt vorgestellt wurde. Um jedoch die technischen Möglichkeiten auszunutzen, die durch den Handel mit digitalen Gütern und die damit verbundene potenzielle anonyme Kundenbindung gegeben sind, wird hier eine Referenzarchitektur auf dieser Grundlage entwickelt. Es wird zunächst auf die Anforderungen und anschließend auf die Architektur eingegangen.

6.1.2.1 Anforderungsanalyse und Systemabgrenzungen

Durch die anonyme Kundenbindung entfällt die Kundenverwaltung. Darüber hinaus erfolgt die Angebotsnutzung und Bestellung sehr viel weniger komplex als in der vorherigen Variante, da unterstellt wird, dass jeweils nur ein Produkt direkt erworben wird und somit der Einsatz eines Warenkorbs und das Management einer Bestellung mitsamt ihren Status nicht erforderlich sind. Die beiden Szenarien, in die die Anwendungsfälle gruppiert werden, sind die Produktverwaltung und die Angebotsnutzung.

Die wesentlichste Änderung gegenüber der vorherigen Variante ist die Einbeziehung eines externen Dienstleisters, der ein System zur Abwicklung des Produktzugriffs und der Zahlungsdurchführung zur Verfügung stellt. Der Kunde muss bei diesem Dienstleister registriert sein und sich über eine Kennwort/Passwort-Kombination authentifizieren, bevor ihm ein Zugriff auf das Dokument gewährt wird. Er hat durch die Registrierung vertraglich gewährleistet, dass die Kosten für einen Produktabruf auf seinem Kundenkonto verbucht werden und er diese periodisch begleichen muss. In der Regel geschieht dies über eine Lastschriftvollmacht, die der Kunde dem Dienstleister im Rahmen der Registrierung erteilt.

Technisch betrachtet handelt es sich bei dem System des externen Dienstleisters um einen Proxy-Server, über den der Zugriff auf die Produkte des Anbieters erfolgt. Der Abruf des Dokumentes erfolgt also nicht direkt innerhalb des System des Anbieters, sondern nur über den Umweg über diesen Proxy-Server. Dazu muss der Anbieter zunächst die URLs der Dateien, die erst nach Bezahlung abrufbar sein sollen, kodieren. Diese kodierten URLs weisen nicht direkt auf die Dateien, sondern enthalten einen Link zum externen Proxy-Server mitsamt einiger Parameter, die diesem Server eine Zuordnung des Produktes ermöglichen.

> Der Proxy-Server des externen Dienstleisters hat die URL www.externerPS.de. Das Dokument, das kostenpflichtig bereitgestellt wird, enthält die Produktkennung doc4711.html. Die kodierte URL lautet dann www.externerPS.de?produkt_id=doc4711.html.

Diese externe URL des Produktes wird in den Produktinformationen hinterlegt und innerhalb der Webseite als Link angeboten.

Darüber hinaus muss der Anbieter (oder der externe Dienstleister) auf dem Proxy-Server die kostenpflichtigen Produkte verwalten, indem dort zu jedem Produkt zumindest eine Produkt-ID und der Preis des Dokuments hinterlegt werden. Die Angaben werden benötigt, damit das Kundenkonto mit dem Betrag der Produkte belastet wird und damit der Proxy-Server das Dokument vom Anbieter anfordern kann. Eine Übermittlung des Preises an das System des externen Dienstleisters in Form eines Parameters ist aus Sicherheitsgründen nicht in Erwägung zu ziehen, da dieser Parameter durch den Nutzer manipuliert werden könnte.

Durch dieses Szenario ist sichergestellt, dass sich ein Kunde zunächst am Proxy-Server des Dienstleisters authentifizieren und den Kauf bestätigen muss, bevor er das Produkt abrufen kann. Sein Konto wird mit dem Kaufpreis belastet. Die Abrechnung zwischen dem Anbieter und dem Dienstleister erfolgt ohne Nutzung des Systems gemäß einer vorherigen vertraglichen Vereinbarung.

Bevor der Ablauf im Rahmen der Architekturbeschreibung dargestellt wird, werden zunächst die Anforderungen an ein solches Szenario in Form von Anwendungsfällen untersucht. Die Produktverwaltung weist einige bereits aus der Variante der physikalsichen Güter bekannte Anwendungsfälle auf, weicht allerdings auch in einigen Bereichen ab. Die folgende Abbildung zeigt den Betreiber als Akteur mitsamt seinen Anwendungsfällen.

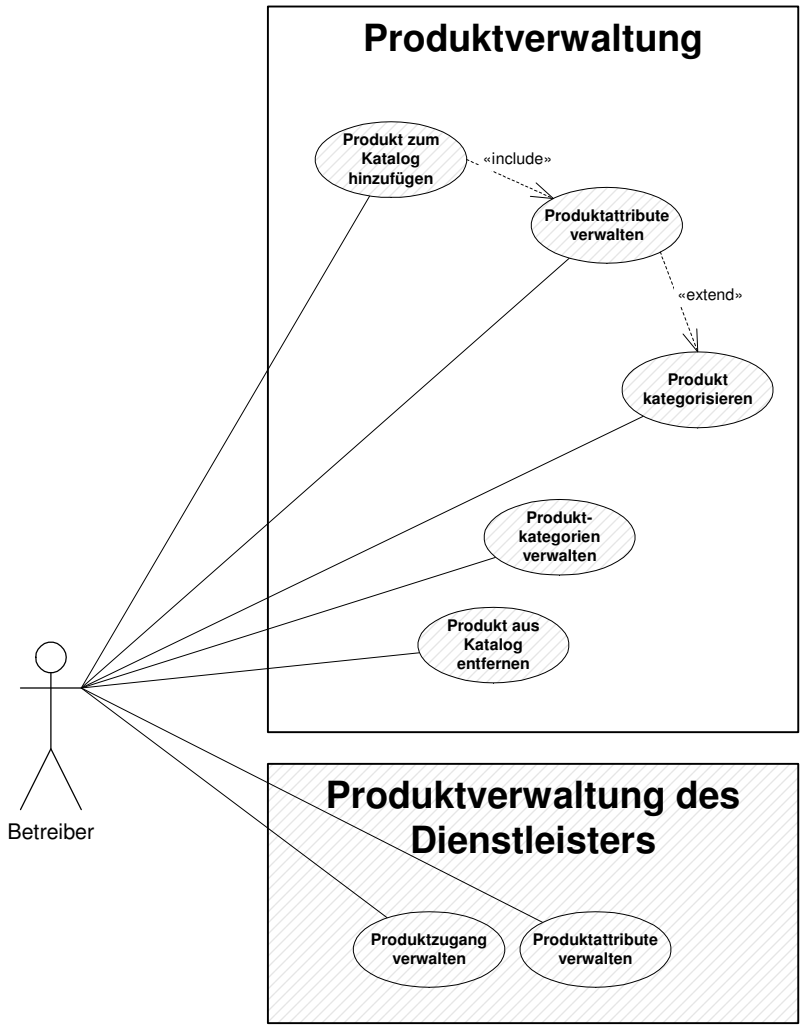


Abbildung 6-26: Die Anwendungsfälle der Produktverwaltung für digitale Güter

Obwohl das externe[6] System für den Produktabruf und zur Durchführung der Bezahlung aus der Architekturbetrachtung ausgegrenzt wird, werden grob einige Anwendungsfälle aufgeführt, damit der Gesamtablauf dieses Erlösmodells besser ersichtlich wird.

Die Anwendungsfälle Produkt zum Katalog hinzufügen, Produktattribute verwalten, Produkt kategorisieren, Produktkategorien verwalten und Produkt aus Katalog entfernen sind bereits aus der vorherigen Variante bekannt und wurden in Abschnitt 6.1.1.1 (siehe Abbildung 6-6 auf Seite 125) beschrieben. Der dort ebenfalls behandelte Anwendungsfall Lagerbestand aktualisieren entfällt hier, da digitale Güter zum einen keine kon-

6. „extern“, da es nicht das System des hier betrachteten Unternehmens ist, sondern das des externen Dienstleisters

ventionelle Lagerverwaltung benötigen und zum anderen aufgrund ihrer Möglichkeit zur beliebigen Vervielfältigung stets verfügbar sein können und somit auch der Einsatz eines Warenwirtschaftsystems überflüssig wird.

Der Betreiber muss jedoch dafür sorgen, dass das externe System hinreichend aktualisiert wird, damit der Nutzer die Produkte abrufen und bezahlen kann. Der Anwendungsfall $\mathrm{Produktattribute\ verwalten}$ deckt diese Anforderungen ab. Neben der eindeutigen Kennung der Produkte müssen vor allem die Preise hinterlegt und gepflegt werden, die dem Kunden in Rechnung zu stellen sind. Über die Kennung wird zum einen sichergestellt, dass das externe System mittels Parameterübergabe die notwendigen Informationen erhält, um das gewünschte Produkt zu identifizieren, und zum anderen, dass der Proxy-Server das digitale Dokument anfordern und dem Kunden zur Verfügung stellen kann. Diese technische Anforderung wird in der Architekturgestaltung berücksichtigt, wirkt sich aber nicht explizit auf die Anwendungsfälle aus.

Neben der Produktverwaltung existiert noch das Szenario der Angebotsnutzung, dessen Anwendungsfälle die folgende Abbildung 6-27 zeigt.

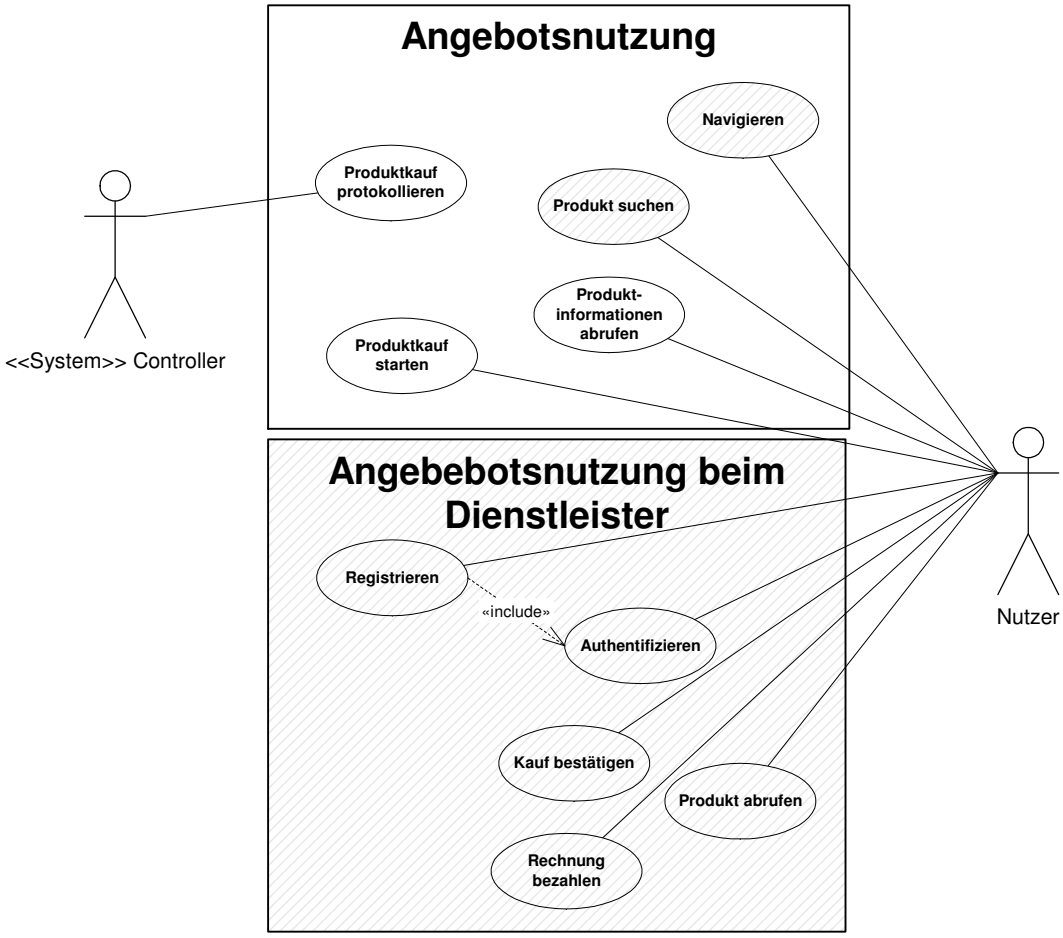


Abbildung 6-27: Die Angebotsnutzung unter Einbeziehung des externen Dienstleisters

Die Möglichkeiten des Nutzers zur Produktsuche mittels Navigation und gezielter Suche sind ebenfalls bereits bekannt und werden auch in dieser Variante nicht näher betrachtet. Eine Besonderheit im Umgang mit digitalen Produkten stellt der Anwendungsfall $\mathrm{Produkt\text{-}informationen\ abrufen}$ dar. Das Produkt ist vollständig digital zugänglich, jedoch soll der Kunde zunächst den Preis für den Zugriff bezahlen. Deshalb muss das Produkt mit umfassenden Informationen beschrieben werden, ohne es jedoch vollständig frei zur Verfügung zu stellen. Das folgende Beispiel verdeutlicht dies.

> Ein Anbieter, der Dissertationen und Diplomarbeiten kostenpflichtig anbietet, kann etwa das Inhaltsverzeichnis, das Abstract sowie einen Auszug aus einem Kapitel kostenlos zur Verfügung stellen und diese Beispiele um zusätzliche Informationen über das Dokument und den Autor anreichern. Ein Anbieter, der Musik- und Video-Files zum Download anbietet, stellt dagegen Auszüge in verminderter Qualität kostenlos bereit.
>
> Die frei zugänglichen Informationen müssen also einerseits das Kaufinteresse des Nutzers so weit wie möglich stärken, andererseits aber nicht zu viel vom vollständigen Produkt preisgeben.

Der Anwendungsfall Produktkauf starten wird vom Nutzer über eine entsprechende Funktionalität aktiviert. Er sorgt lediglich für die Weiterleitung des Nutzers an das System des externen Dienstleisters.

Die Authentifizierung und vorherige Registrierung des Kunden erfolgt nur auf dem Server des Dienstleisters, wodurch die Anonymität dem Verkäufer gegenüber gewährleistet bleibt. Sobald der Nutzer angemeldet ist, wird er vom externen System aufgefordert, den Kauf des Produktes zu dem angegebenen Preis noch einmal explizit zu bestätigen. Sofern er dies tut, kann er das Produkt abrufen. Es wird entweder innerhalb einer zugehörigen Applikation direkt geöffnet oder aber zum Download angeboten. Es wird auch der Anwendungsfall Rechnung bezahlen hier mit aufgeführt, um den gesamten Rahmen der Angebotsnutzung zu betrachten.

Der Controller als Teil-System sorgt in diesem Szenario mit dem Anwendungsfall Produktkauf protokollieren dafür, dass ein Abruf des Produktes registriert wird, da dies die Grundlage für die Abrechnung mit dem Dienstleister bildet. Die Protokollierung erfolgt dabei anonym; das heißt, der Betreiber erfährt nicht, wer der Käufer des Produktes ist, sondern weiß lediglich, dass eine Kopie des Produktes verkauft wurde.

Die Anwendungsfälle der Akteure mit dem externen System werden nicht näher bei der Architekturbeschreibung berücksichtigtsind und wurden deshalb schraffiert dargestellt.

Zusammenfassend lässt sich sagen, dass durch die Kooperation mit einem externen Dienstleister und den damit verbundenen Einsatz eines externen Systems die Anforderungen an die Architektur des Systems des Betreibers sehr gering sind. Sobald die organisatorische und die technische Anbindung des externen Dienstleisters mitsamt seinem System durch ein Unternehmen abgeschlossen worden ist, muss der Betreiber lediglich die Verwaltung seines Produktangebots auf das externe System ausweiten und die Verkäufe zur Kontrolle der Abrechnung protokollieren.

Die technische Anbindung eines Proxy-Servers über eine API an die Anwendung des hier betrachteten Unternehmens wird nicht näher untersucht und beschrieben. Ebenso wird die konkrete Abrechnung der Produktverkäufe zwischen den beiden Unternehmen nicht betrachtet. Für diese Abrechnung sind zahlreiche Modelle ausgehend von einfachen prozentualen Abschlägen für den Dienstleister bis hin zu komplexen Verfahren vorstellbar. So

ist es zum Beispiel theoretisch möglich, dass ein Dienstleister für unterschiedliche Produkte unterschiedliche Gebühren erhebt. In diesem Fall müsste für jedes Produkt nicht nur ein Verkaufspreis, sondern auch eine Gebühr (fix, sprungfix, prozentual etc.) hinterlegt und gepflegt werden. Die Architektur müsste diese Optionen berücksichtigen. Da die Vielfalt möglicher Modelle jedoch hier nicht abgedeckt werden kann, bleibt dieser Aspekt unberücksichtigt und es wird von einem einfachen Abrechnungsmodell (prozentual oder per Transaktion) zwischen Diensleister und Verkäufer ausgegangen. Aus Sicht des Verkäufers ist somit vor allem die Protokollierung der Kauftransaktionen wichtig.

Vor dem Hintergrund der Auslagerung wesentlicher Funktionalitäten auf einen externen Dienstleister umfasst die Architektur des Erlösmodells nur sehr wenige Komponenten, wie der nächste Abschnitt zeigen wird.

6.1.2.2 Architekturbeschreibung

Die folgende Abbildung 6-28 zeigt grob die Architekturen der beiden Systeme mitsamt den wesentlichen Komponenten und den hier betrachteten Klassen.

Da in dieser Variante neben dem Softwaresystem des Geschäftsmodellbetreibers auch das externe System des Finanzdienstleisters an der Abwicklung der Kaufaktivitäten beteiligt ist, werden in der Architektur beide Systeme aufgeführt, die während der Kaufabwicklung miteinander kommunizieren.

Die auffälligsten Komponenten sind der `LinkController` und das `VerkaufsProtokollKonto`. Beide Komponenten dienen dazu, einen Abruf eines digitalen Produktes zu registrieren und zu protokollieren. Dies ist zur generellen Nachvollziehbarkeit der geleisteten Transaktionen und zur Kontrolle der Abrechnung mit dem externen Dienstleister für das Unternehmen von großem Interesse. Sämtliche Komponenten werden nun im einzelnen vorgestellt.

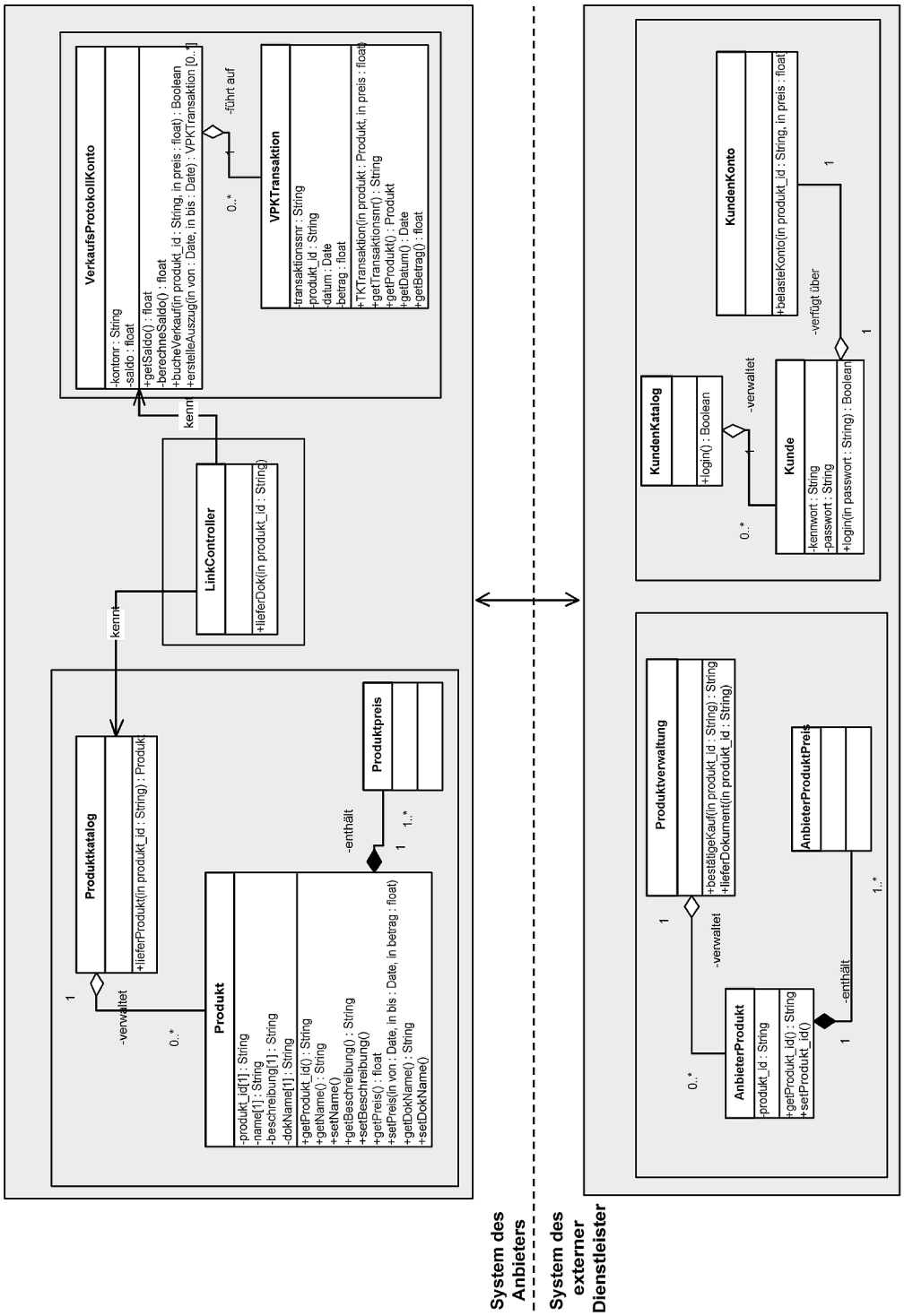


Abbildung 6-28: Komponentendiagramm der Variante für digitale Güter

Die Komponente `Produkt_EM1.2_ET` wurde weitgehend unverändert aus der vorherigen Variante übernommen. Die Klasse `Produktpreis` ist identisch übernommen worden, so dass ihre Attribute und Mehtoden hier nicht erneut beschrieben werden (vgl. Abbildung 6-29).

Die Klasse `Produkt` weist einige Veränderungen auf. Sie enthält nun nicht mehr das Attribut `verfügbarkeit`, welches zum Zeitpunkt einer Bestellung bei der Variante der physikalischen Güter zur Überprüfung des aktuellen Warenlagerbestand des Artikels benötigt wurde. Dies ist bei physikalischen, nicht jedoch bei digitalen Gütern von Bedeutung. Stattdessen weist die Klasse nun das Attribut `dokName` mitsamt den zugehörigen `get-` und `set-`Methoden auf. Das Attribut enthält den Namen des digitalen Dokuments, unter dem es abgespeichert ist und zur Übertragung angefordert werden kann. In der GUI wird für jedes Produkt als Link die URL des externen Dienstleisters ergänzt um das Attribut `produkt_id` eingebunden. Diese Produkt-ID wird somit als Parameter an das externe System übergeben und dient dort dazu, das relevante Produkt zu identifizieren und in Rechnung zu stellen.

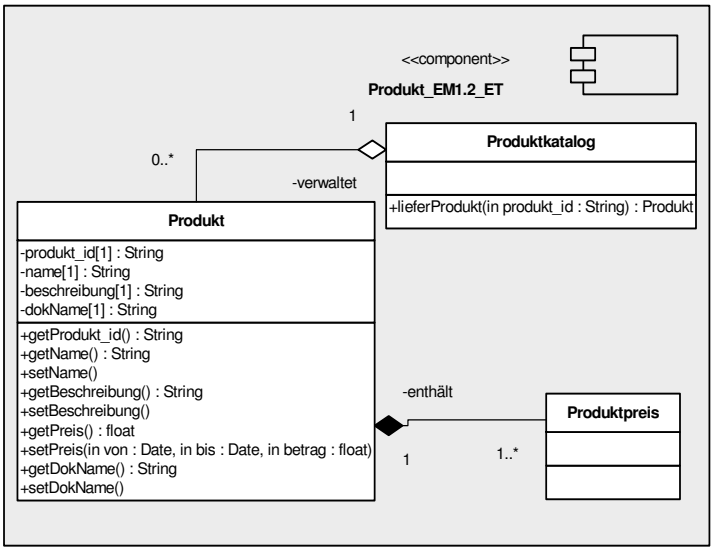


Abbildung 6-29: Komponente Produkt_EM1.2_ET

Neben der Produktverwaltung besteht die Anforderung der Protokollierung der verkauften Produkte. Dazu dient die Komponente `TransaktionsProtokollKonto` (die vollständige Bezeichnung ist `TransaktionsProtokollKonto_EM1.2_ET`) mitsamt ihren Klassen `TransaktionsProtokollKonto` und `TPKTransatkion`, wie sie in Abbildung 6-30 zu sehen ist. Zunächst ist auffallend, dass dieses Protokollierungskonto keinem Nutzer zugeordnet ist, sondern als eigenständige Komponente existiert. Die Protokollierung erfolgt also nicht personalisiert, sondern losgelöst und anonym in Bezug auf die Käufer. Da jeder einzelne Produktabruf einschließlich der relevanten Details protokolliert werden soll, weist die Klasse `VerkaufsProtokollKonto` eine *1:n* Aggregation zur Klasse `VPKTransaktion` auf, in deren Instanzen die Datensätze hinterlegt werden.

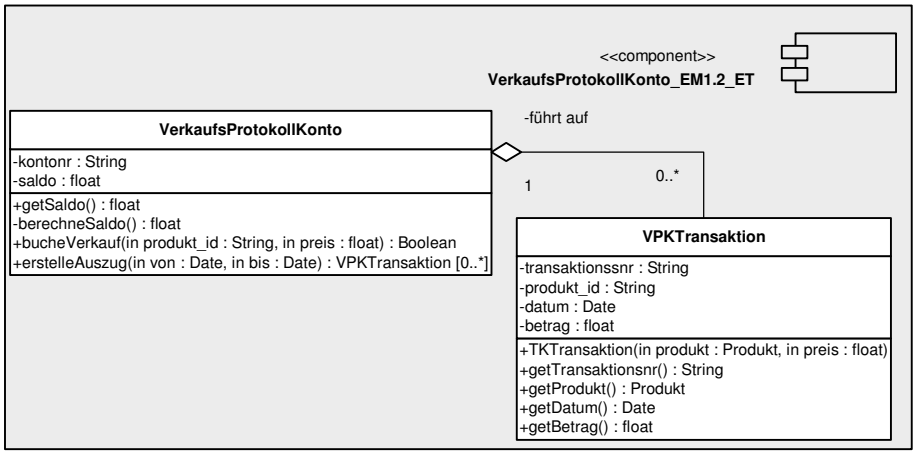


Abbildung 6-30: Komponente VerkaufsProtokollKonto_EM1.2_ET mitsamt ihren Klassen

Die Methode `bucheVerkauf` der Klasse `VerkaufsProtokollKonto` wird bei jeder Produktanforderung durch den Proxy-Server des externen Dienstleister aufgerufen und erhält als Parameter die Produkt-ID und den Verkaufspreis des Produktes. Sie erzeugt eine neue Instanz von `VKPTransaktion`, indem sie den Konstruktor aufruft und die Parameter übergibt. Der Konstruktor fügt zusätzlich noch einen Zeitstempel hinzu. Abschließend ruft `bucheVerkauf` die Methode zur Aktualisierung des Konto-Saldos auf. Die Methode `erstelleAuszug` liefert als Rückgabewert ein Listobjekt über alle Transaktionen, die innerhalb des übergebenen Zeitraums liegen. Es kann beispielsweise die Basis einer Monatsabrechnung oder eines Reports sein, auf die hier jedoch nicht eingegangen wird.

Aus Sicht des Anbieters ist es von großem Interesse zu registrieren, welche Produkte zu welcher Zeit durch den Proxy-Server des Dienstleisters angefordert und somit an den Kunden verkauft werden. Aus diesem Grund wird die URL des Dokumentes nicht direkt in der Produktverwaltung des externen Systems hinterlegt. Dies wäre technisch ebenfalls möglich, jedoch würde dann der Abruf durch den Proxy-Server auf Seiten des Anbieters nicht zu protokollieren sein, da der Zugriff direkt erfolgen würde. Stattdessen wird dem Proxy-Server nur die Produkt-ID mitegeteilt. Anhand dieses Parameters muss das externe System die Datei (als digitales Dokument) zunächst vom System des Anbieters anfordern. Diese Anfrage wird erst gestartet, wenn sich der Kunde authentifiziert hat und den Kaufpreis noch einmal explizit bestätigt hat. Damit wird sichergestellt, dass der Kunde das Produkt auch tatsächlich erwirbt und dass dieser Verkauf somit zu protokollieren ist. Für die Annahme dieser Anfrage durch den Proxy-Server des externen Dienstleisters steht im System des Anbieters eine Controller-Komponente bereit, die neben der Auslieferung des Files auch die interne Protokollierung der Anfrage übernimmt.

Die Komponente `LinkController`, die als Controller agiert, wird in Abbildung 6-31 mit ihrer gleichnamigen Klasse aufgezeigt.

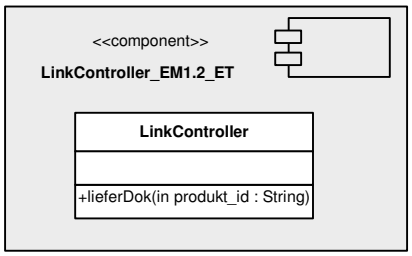


Abbildung 6-31: Komponente LinkController_EM1.2_ET

Die Klasse `LinkController` enthält die Methode `lieferDok`. Sie erhält als Parameter die Produkt-ID überliefert. Der Aufruf der Methode erfolgt durch das System des Dienstleisters. Die Methode `lieferDok` nutzt den übermittelten Parameter `produkt_id`, um zunächst über den `ProduktKatalog` eine Referenz auf das `Produkt` zu ermitteln. Von dieser Instanz werden nun die erforderlichen Daten für die interne Protokollierung des Abrufs sowie der Speicherort des Dokuments ermittelt. Die relevanten Werte sind der Produktpreis sowie das Attribut `dokName`. Zur Protokollieurng ruft die Methode `lieferDok` die Methode `bucheVerkauf` der Klasse `VerkaufsProtokollKonto` auf und übergibt ihr die Parameter `produkt_id` und `preis`. Anschließend liefert die Methode das elektronische Dokument in Form eines Bytestream als Response an den Proxy-Server des Dienstleisters zurück. Der Ablauf ist im folgenden Sequenzdiagramm in Abbildung 6-32 dargestellt.

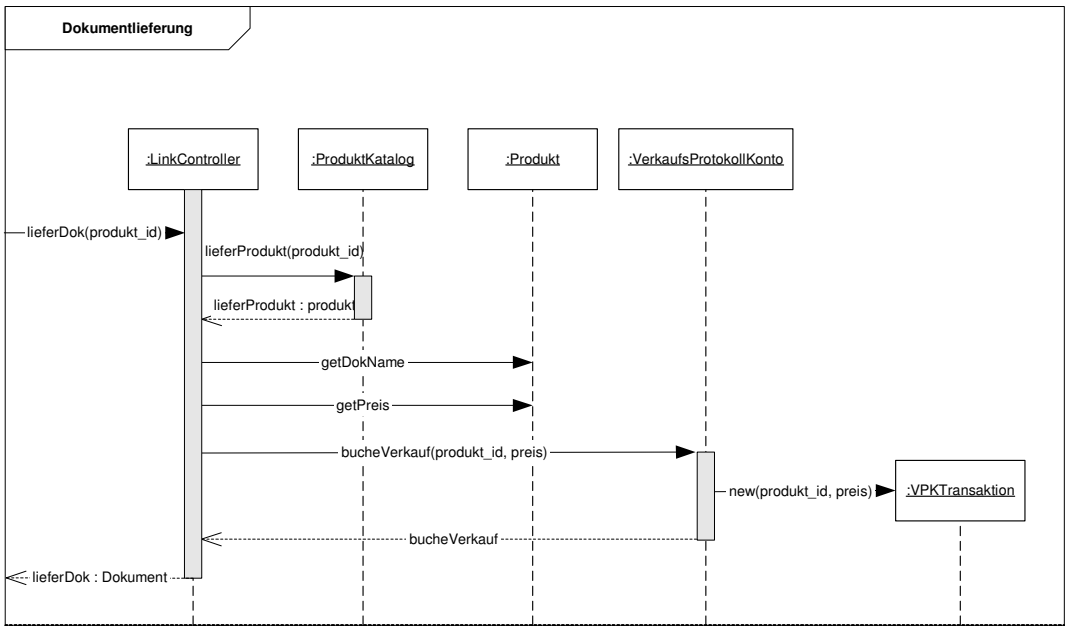


Abbildung 6-32: Sequenzdiagramm zur Rücklieferung des angeforderten digitalen Dokuments und Protokollierung des Abrufs

Der initiale Methodenaufruf erfolgt von außerhalb des Sequenzdiagramms innerhalb des Gesamtablaufs des Produktkaufs. Das Zusammenspiel wird im folgenden Diagramm ersichtlich, in dem eine Referenz auf den hier dargestellten Aspekt integriert ist. Das vollständige Diagramm ist in Abbildung 6-36 zu sehen und wird weiter unten vorgestellt.

Zunächst müssen jedoch die wichtigsten Komponenten des externen Dienstleistersystems vorgestellt werden. Das Gesamtsystem wird nur stark vereinfacht wiedergegeben, da sich nur auf diejenigen Klassen konzentriert wird, die im Zusammenspiel mit dem System des Anbieters eine Rolle spielen. Im Wesentlichen wird eine Produktverwaltung und eine Kundenverwaltung benötigt.

Die Produktverwaltung dient dazu, dass zunächst zu jedem Produkt eine zugehörige Kennung (`produkt_id`) sowie der Preis hinterlegt werden. Die Preishinterlegung auf dem System des externen Dienstleisters ist aus Sicherheitsgründen notwendig, da eine Übergabe als Parameter durch die Käufer manipuliert werden könnte. Die Pflege dieser Informationen auf dem externen System wird wahrscheinlich dem Betreiber des Geschäftsmodells überlassen, so dass eine geeignete Schnittstelle zur Verfügung gestellt werden muss. Aus funktionaler Sicht stellt die Produktverwaltung einen Dienst bereit, der das Einholen einer Kaufbestätigung unterstützt, die der Nutzer nach seiner Authentifizierung explizit geben muss und die der Auslöser für den Dokumentenabruf darstellt. Der Abruf des Dokuments von dem System des Anbieters sowie die Weiterleitung an den Kunden wird durch den zweiten wesentlichen Dienst der Komponente übernommen. Beide Funktionen werden hier als Methoden der Klasse `Produktverwaltung` dargestellt, die Bestandteil der Komponente `DL-Produktverwaltung` sind. Die folgende Abbildung 6-33 zeigt diese Komponente des externen Systems.

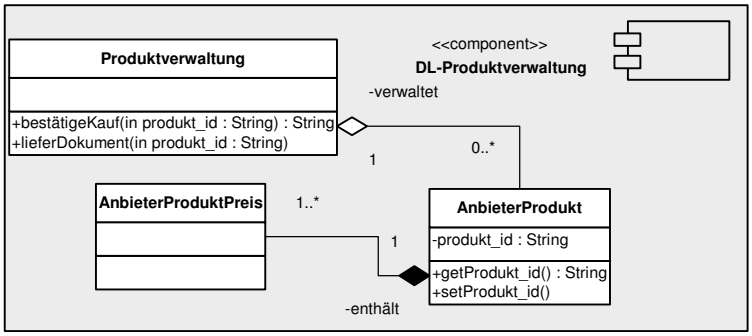


Abbildung 6-33: Komponente DL-Produktverwaltung

Die Methode `bestätigeKauf` wird aufgerufen und ermittelt zunächst den aktuell gültigen Produktpreis des Produktes, dessen ID sie als Parameter erhalten hat. Um auch hier temporär gültige Preise verwalten zu können, ist die selbe Produktstruktur übernommen worden, wie sie bereits aus der Komponente `Produkt_EM1.2_ET` bekannt ist.

Mit dieser Information kann der Kunde über ein Formular aufgefordert werden, die explizite Bestätigung zu geben. Die Methode `lieferDokument` erhält ebenfalls die Produkt-ID als Parameter übergeben und wendet sich an das System des Anbieters (konkreter: an den

`LinkController`), um das digitale File anzufordern. Als Rückgabewert liefert sie das elektronische Dokument als Bytestream an den Kunden beziehungsweise zunächst an den eigenen Web-Server, der die Weiterleitung an die GUI des Kunden übernimmt. Eine detaillierte Ablaufdarstellung ist im Sequenzdiagramm in Abbildung 6-36 zu sehen.

Die Kundenverwaltung als zweite wesentliche Komponente des externen Systems sorgt dafür, dass jeder Kunde zunächst registriert ist und somit über ein Profil mitsamt (Debitoren-)Konto verfügt, auf dem die Preise der gekauften digitalen Produkte verbucht werden. Darüber hinaus führt sie die Authentifizierung des Nutzers zu Beginn der Kauftransaktion durch. Die Klassen und Methoden der Komponente `DL-Kundenverwaltung` sind in Abbildung 6-34 dargestellt. Da die Komponente eine vereinfachte Version der Kundenverwaltung aus der Variante für physikalische Güter ist (vgl. Abbildung 6-12), wird sie hier nicht näher vorgestellt.

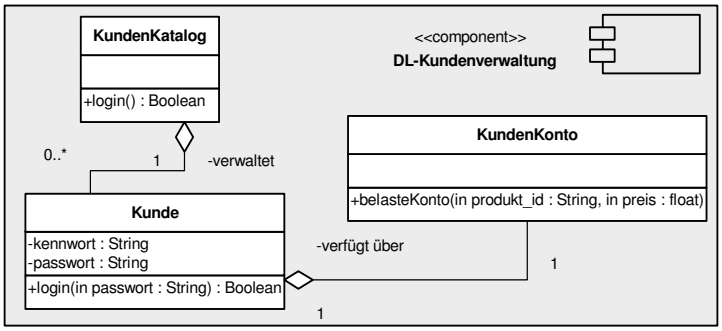


Abbildung 6-34: Komponente DL-Kundenverwaltung

Die Interaktionen der Systeme sowie die internen Abläufe innerhalb beider Systeme ist im folgenden Sequenzdiagramm dargestellt worden. Aufgrund des besonderen Charakters dieses Interaktionsdiagramms (systemübergreifende sowie systeminterne Kommunikation wird in einem Diagramm zusammengefasst) wird ein eigener Nachrichtentyp verwendet, der keinen Methodenaufruf repräsentiert, sondern die Kommunikation zwischen GUI und Web-Server wiedergibt.

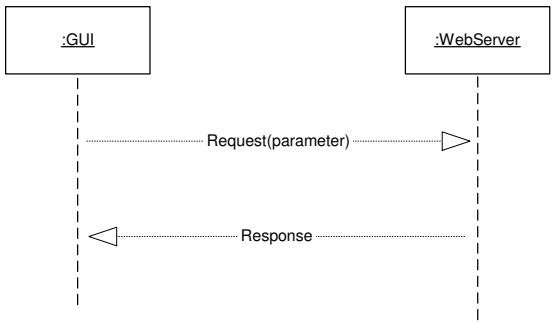


Abbildung 6-35: Nachrichtentyp zur Darstellung der Dialogsteuerung

Abbildung 6-35 zeigt die verwendeten Nachrichtentypen für einen Request und den Response, der als gepunktete Linie mit offener Pfeilspitze dargestellt wird. Der Request enthält relevante Parameter als Anhang in Klammern aufgeführt. Der Typ wird verwendet, um die Dialoge zu verdeutlichen, die sich während des Ablaufs ergeben. Der Web-Server wird als Rechteck mit dem gleichen Symbol wie auch die GUI und die beteiligten Klassen dargestellt.

Das folgende Sequenzdiagramm in Abbildung 6-36 dient folglich dazu, den Gesamtablauf und die Interaktion der beteiligten Komponenten darzustellen, wobei aufgrund der unterschiedlichen Ebenen der beteiligten Elemente auf eine vollständige Konformität mit der UML verzichtet wird.

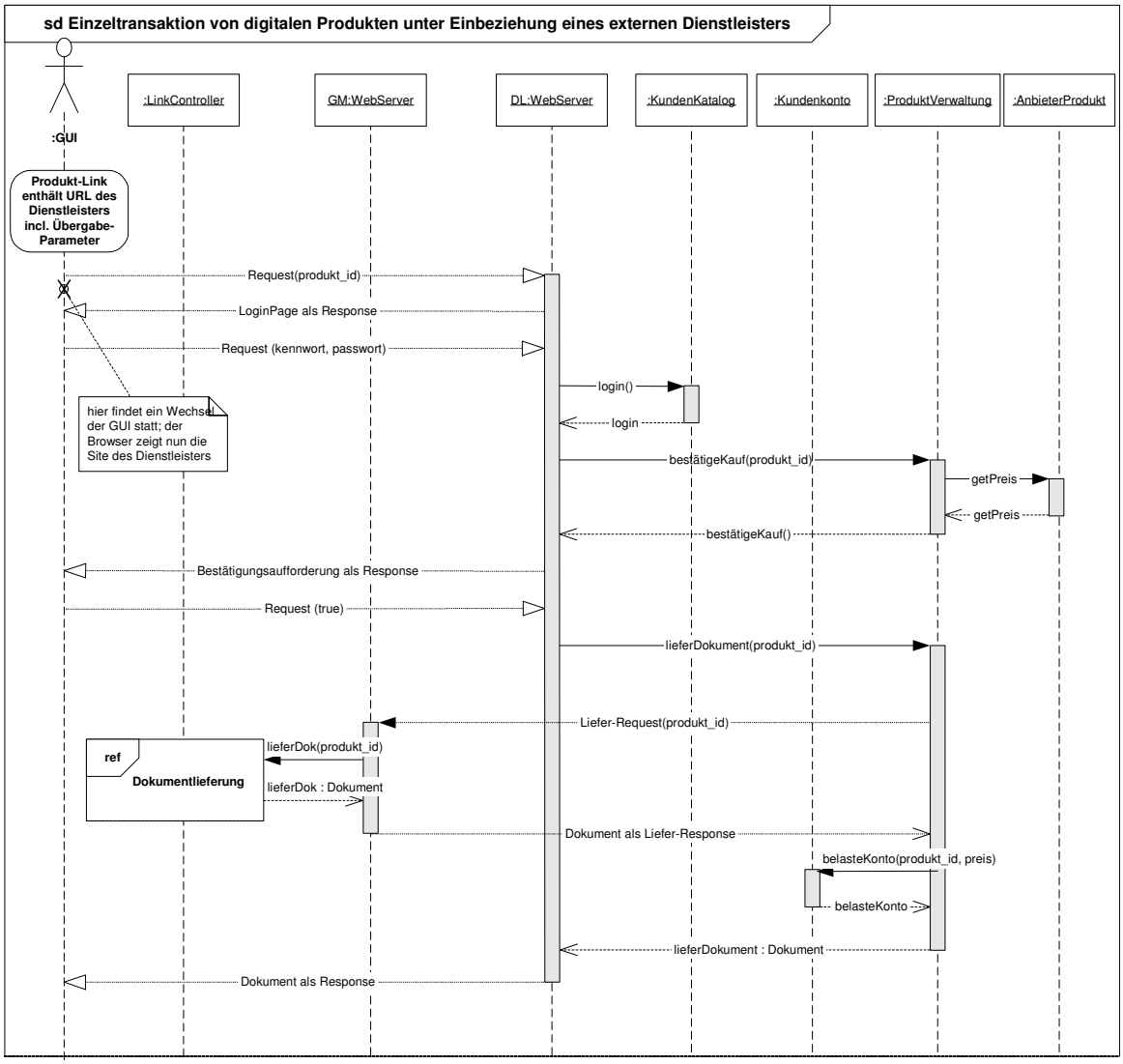


Abbildung 6-36: Kaufdurchführung von digitalen Produkten unter Einsatz eines externen Dienstleisters

Ausgangspunkt des Diagramms ist die Darstellung des Produktlinks innerhalb der GUI des Anbieters. Sobald der Kunde diesen Link klickt, wird ein Request mitsamt der Produkt-ID als Parameter an den Web-Server des Dienstleisters (DL:WebServer) gesendet. In diesem Moment wechselt die GUI, indem nicht mehr die Website des Anbieters, sondern die des Dienstleisters im Browser darsgestellt wird.

Der Web-Server veranlasst zunächst die Authentifizierung des Kunden, den er über einen entsprechenden Dialog zur Abgabe des Kennworts und Passworts auffordert. Die erhaltenen Daten werden an die zugehörige Methode der Klasse `KundenKatalog` weitergereicht.

Anschließend (eine erfolgreiche Authentifizierung wird hier unterstellt) wird die explizite Kaufbestätigung des Kunden eingefordert. Dazu muss zunächst der Preis des Produktes ermittelt werden, wozu die Methode `bestätigeKauf` der `ProduktVerwaltung` aufgerufen wird. Sie liefert als Rückgabewert den Preis des Produktes, der innerhalb der Komponente vorliegt (die Klasse `AnbieterProduktPreis` wurde im Diagramm aus Platzgründen nicht mit aufgeführt). Mit dieser Information kann dem Nutzer ein Formular angezeigt werden, das die Bestätigung erfordert (Bestätigungsaufforderung als Response).

Nachdem der Kunde diese Bestätigung erteilt hat, fordert der Web-Server über die ProduktVerwaltung (Methode `lieferDokument`) an. Diese Methode wiederum wendet sich an den Web-Server des Anbieters (GM:WebServer, GM für Geschäftsmodell), um von dort das Dokument zu erhalten. Der Liefer-Request wird innerhalb des Anbieter-Systems an die Komponente `LinkController` weitergeleitet. Der dortige Ablauf ist als Interaktionsreferenz dargestellt und wurde bereits in Abbildung 6-32 auf Seite 158 diskutiert. Nachdem das Dokument als Bytestream-Response an die `ProduktVerwaltung` zurückgeliefert wurde, kann dort die Belastung des Kunden-Kontos erfolgen (Methode `belasteKonto`[7]). Abschließend liefert die Methode das digitale Dokument als Rückgabewert an den Web-Server, der die Weiterleitung an die GUI übernimmt.

Die Darstellung der Referenzarchitektur hat eine Rechnung nicht explizit berücksichtigt. Hintergrund ist die Unterstellung, dass zwischen dem Betreiber des Geschäftsmodells und dem externen Dienstleister eine vertragliche Vereinbarung besteht, die die Abrechnung der bezogenen Produkte unter Berücksichtigung der Gebühren für den Dienstleister regelt. Anderenfalls kann eine Rechnungsstellung unter Verwendung des `VerkaufsProtokollKontos` sehr einfach realisiert werden. Eine geeignete Rechnungskomponente, die ein solches Protokollierungskonto verwendet, wird im folgenden Erlösmodell vorgestellt.

7. Es wird an dieser Stelle nicht darauf eingegangen, ob der Preis zunächst erneut über das Produkt ermittelt werden muss oder zuvor als Paramter hinterlegt wurde und somit verfügbar ist.

6.2 Referenzarchitektur für das Erlösmodell Subskription

Die Varianten des Erlösmodells SUBSKRIPTION richten sich nach der Beschaffenheit des Angebotes. Es werden in den folgenden Abschnitten die Varianten für Dienstleistungen *(*Abschnitt 6.2.1) sowie für digitale und physikalische Güter (Abschnitt 6.2.2 auf Seite 180*)* unterschieden.

6.2.1 Variante Dienstleistung

Das Erlösmodell wurde in Abschnitt 4.2.2.1 auf Seite 70 umfassend beschrieben. Die folgende Grafik zeigt noch einmal die Klassifikationskriterien.

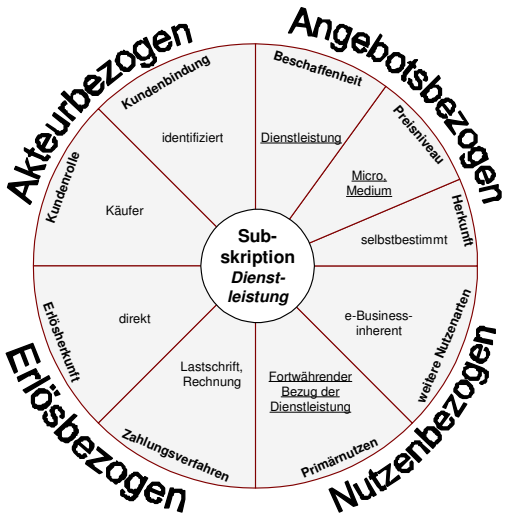


Abbildung 6-37: Erlösmodellklassifikation Subskription Variante Dienstleistung

Ein wesentliches Merkmal dieses Erlösmodells ist die vertragliche Bindung des Kunden mit dem Unternehmen. Die Zahlungsverfahren können somit auf Rechnung und Lastschrift begrenzt werden, da sie für beide Parteien die bequemste und optimale Lösung bilden. Die Gefahr auf Seiten des Verkäufers wird dadurch begrenzt, dass zahlungsunwillige Kunden nach maximal einer Abrechnungsperiode gesperrt werden können. Durch die eher geringen Beiträge im Rahmen einer Subskription ist dieses Risiko für ihn kalkulierbar.

6.2.1.1 Anforderungen und Systemabgrenzung

Die Nutzer interagieren identifiziert mit dem System des Anbieters, wodurch eine Nutzerverwaltung benötigt wird. Die Anforderungen an eine Nutzerverwaltung sind identisch mit den Anforderungen des Erlösmodells EINZELTRANSAKTION in der Variante für physikalische Güter (siehe Abschnitt 6.1.1). Die folgende Abbildung zeigt noch einmal das Anwendungsfalldiagramm (vgl. Abbildung 6-7 auf Seite 126), die Beschreibung der einzelnen Anwendungsfälle wird hier jedoch nicht wiederholt.

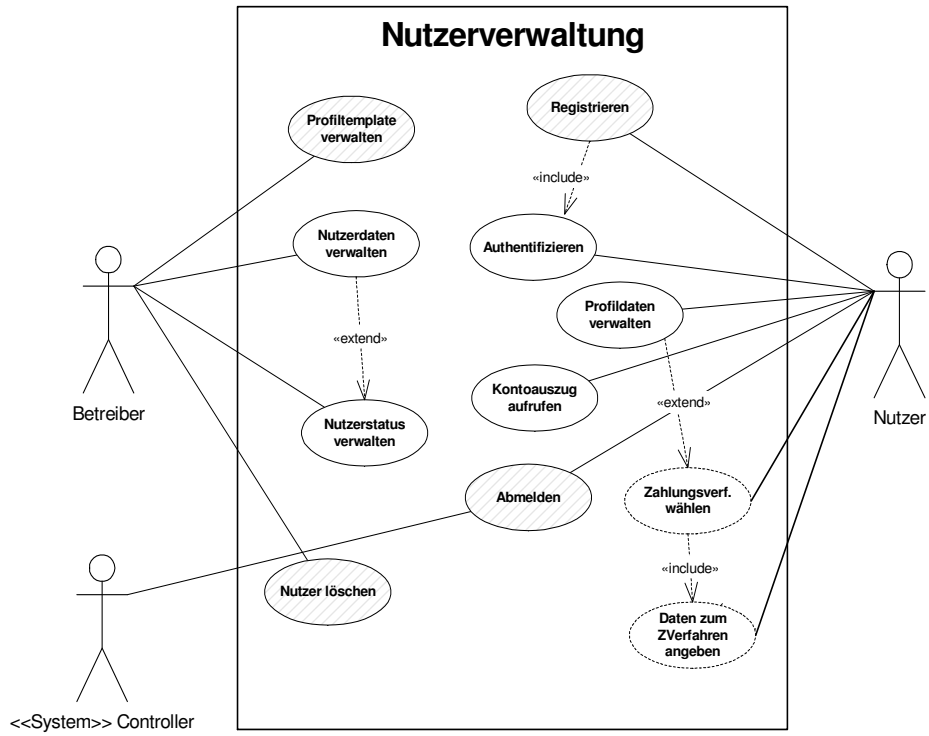


Abbildung 6-38: af Subskription Nutzerverwaltung

In dieser Variante wird das Erlösmodell SUBSKRIPTION von Dienstleistungen untersucht und beschrieben. Es entfallen somit die Anforderungen an eine (konventionelle) Produktverwaltung, wie sie aus dem vorherigen Erlösmodell bekannt ist. Stattdessen werden Verträge zwischen dem Kunden und Anbieter abgeschlossen, die den Umfang und die Art des Abonnements sowie den Basiskosten und die Kosten für den Bezug weiterer, über den Umfang hinausgehender Dienstleistungen regeln. Da hier zahlreiche Kombinationsmöglichkeiten existieren, stehen den Kunden mehrere Varianten der Abonnements in Form von Tarifen zur Verfügung[8]. Die Anforderungen im Umgang mit den Verträgen und Tarifen aus Sicht der beteiligten Akteure sind im Szenario Vertrags-und Tarifverwaltung zusammengefasst und in Abbildung 6-39 aufgelistet.

8. Man kann die einzelnen Verträge mitsamt ihren Tarifen auch als die Produkte des Dienstleisters betrachten, jedoch soll hier bewusst eine sprachliche Trennung eingeführt werden, um den Dienstleistungscharakter dieser Variante zu stärken und sich von anderen Varianten abzugrenzen.

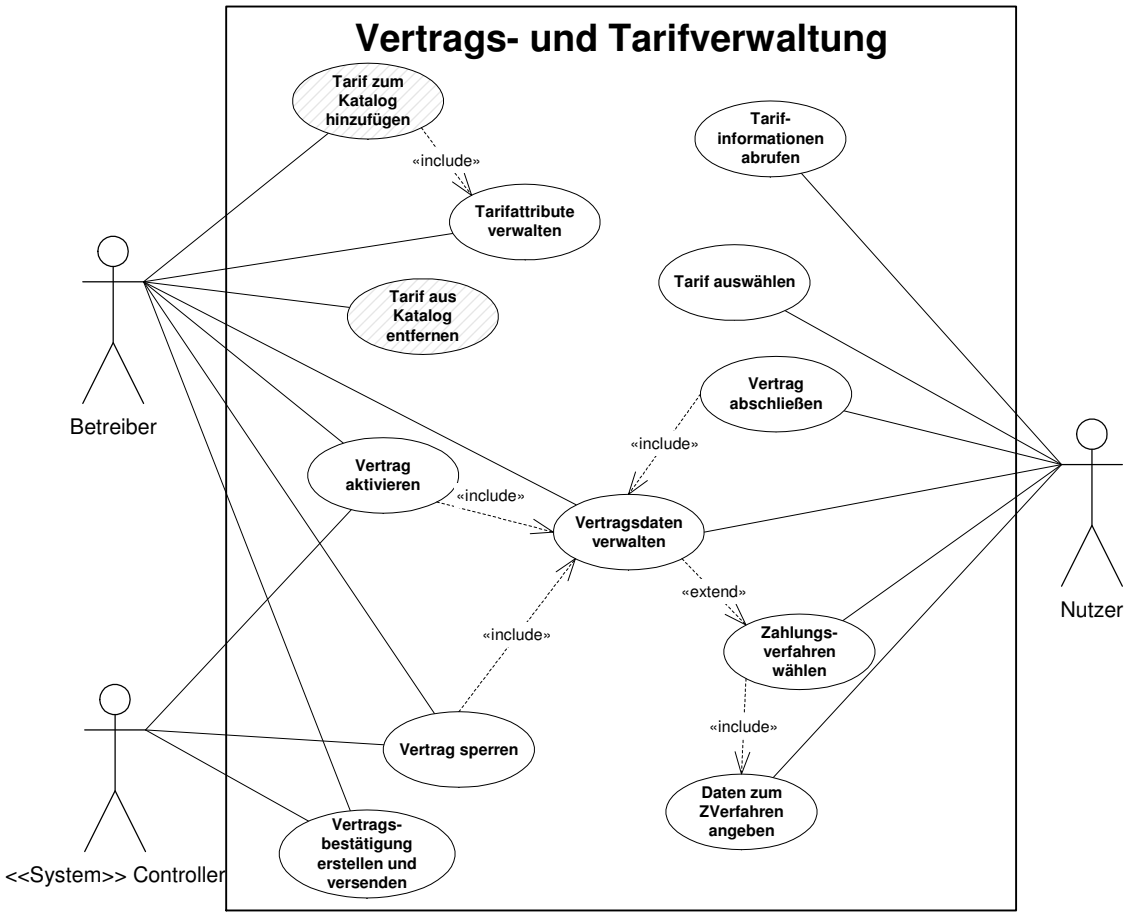


Abbildung 6-39: Szenario der Vertrags- und Tarifverwaltung

Zunächst muss der Betreiber sein Angebot in Form der Tarife einrichten. Er muss neue Tarife zum Katalog hinzufügen, was auch immer ein Verwalten der Tarifattribute beinhaltet. Es muss festgelegt werden, ob der Tarif auf eine zeitliche oder auf eine mengenmäßige Bezugsgröße der Dienstleistung ausgerichtet ist, welche Bezugsperioden für die Inanspruchnahme des Abonnements gelten sollen, was der Basisbezug kostet und wieviel für eine zusätzliche Einheit der Dienstleistung zu bezahlen ist. Nicht mehr angebotene Tarife sind aus dem Katalog zu entfernen.

Bevor ein Vertrag eines Kunden wirksam wird, und der Kunde die Dienstleistungen in Anspruch nehmen kann, ist der Vertrag zu aktivieren. Es können beispielsweise zunächst Plausibilitätsprüfungen der vom Nutzer angegebenen Adress- und Bankverbindungen durchgeführt werden, um die Absicherung gegenüber einem Missbrauch zu erhöhen. Diese Überprüfungen können je nach Art vom Betreiber oder auch automatisiert durch das System erfolgen. Über einen freigeschalteten Vertrag wird eine Vertragsbestätigung erstellt und an den Kunden versendet. Sollte ein Kunde den Zahlungsaufforderungen nicht nachkommen, wird sein Vertrag gesperrt, so dass ein weiterer Bezug der Dienste unterbunden wird. Der Vertragsstatus wird somit auch während der Angebotsnutzung überprüft und ent-

scheidet über die Bereitstellung der Dienste für den Kunden oder deren Blockierung. Grundsätzlich hat der Betreiber die Möglichkeit, auf die Vertragsdaten zuzugreifen und sie im Rahmen seines Einflussbereiches zu ändern, was durch den Anwendungsfall Vertragsdaten verwalten zum Ausdruck kommt.

Der Nutzer wird zunächst einen sich am besten geeigneten Tarif auswählen. Dazu ruft er die Tarifinformationen ab und schließt einen Vertrag ab, der diesen Tarif enthält. In diesem Zusammenhang muss er die von ihm benötigten Informationen angeben und so die Vertragsdaten verwalten. Einer der Angaben in diesem Zusammenhang ist die Wahl eines Zahlungsverfahrens und die Angabe der zugehörigen Daten. Da der Nutzer identifiziert ist und eine dauerhafte vertragliche Bindung mit dem Unternehmen eingeht, bieten sich die Zahlungsverfahren Rechnung und Lastschrift an. Sie sind für beide Parteien kostengünstig, sicher und bequem durchzuführen.

Die Inanspruchnahme der Leistungen, die dem Kunde im Rahmen des Vertrages zustehen, wird im folgenden Diagramm (Abbildung 6-40) dargestellt.

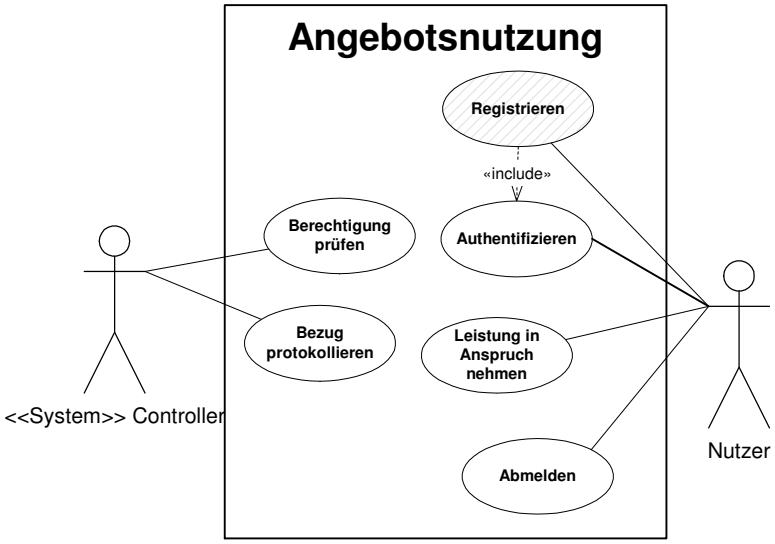


Abbildung 6-40: Anwendungsfälle des Szenarios der Angebotsnutzung

Der Betreiber muss in dieser Phase keine Aufgaben durchführen. Stattdessen übernimmt der Controller automatisiert die bereits angesprochene Prüfung der Berechtigung. Anhand des Vertragsstatus kann bei jeder Authentifizierung ermittelt werden, ob ein Nutzer Zugriff hat auf die Dienste oder nicht. Die wesentliche Aufgabe während der Angebotsnutzung ist es jedoch, den Bezug zu protokollieren. Dabei ist die im Tarif hinterlegte Bezugsgröße (Zeit oder Menge) zu berücksichtigen.

Der Kunde ist der wesentliche Akteur dieses Szenarios. Nach einer Registrierung (die auch hier nicht näher im Rahmen der Architekturgestaltung berücksichtigt wird) und Authentifizierung nimmt er die Leistung in Anspruch. Dieser Anwendungsfall wird hier

nur sehr allgemeingültig beschrieben, um dem Referenzcharakter des Modells gerecht zu werden. Unabhängig von der Art und Weise der Dienstleistung ist für die Architektur auch eher interessant, wie der Bezug protokolliert und die Leistungen abgerechnet werden. Beendet ein Kunde eine Sitzung, meldet er sich vom System ab.

Als letzter Bereich werden die Anwendungsfälle im Zusammenhang der Abrechnung dargestellt. Die folgende Abbildung 6-41 zeigt das zugehörige Anwendungsfalldiagramm.

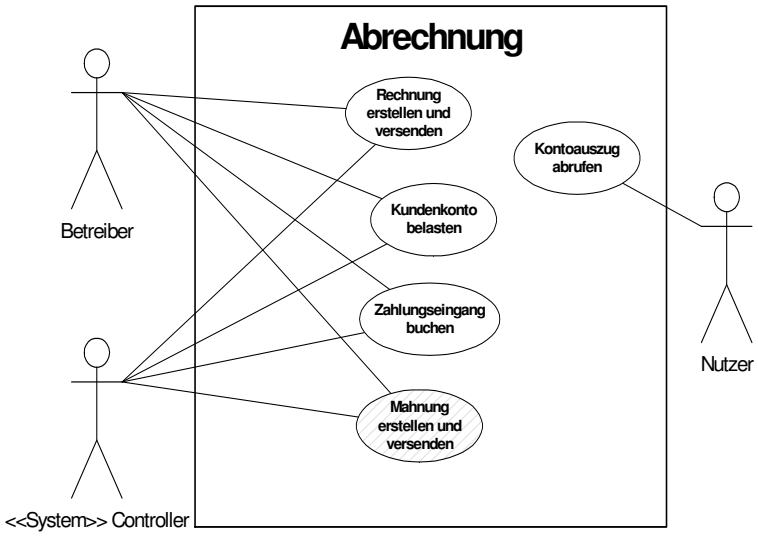


Abbildung 6-41: Anwendungsfälle der Abrechnung des Erlösmodells Subskription

Wie auf dem Diagramm zu sehen ist, sind die beiden hauptverantwortlichen Akteure dieses Szenarios der Betreiber mit manuellen Interaktionen sowie das System selber in Form des automatisierten Controllers. Der Kunde führt die Zahlungsverfahren nicht mit Hilfe des Systems durch, so dass lediglich der Anwendungsfall $\mathrm{Kontoauszug\ abrufen}$ seine Interaktion mit dem System ausmacht. Über den Kontoauszug kann er seine aktuellen Finanztransaktionen einsehen und somit Zahlungseingänge und Belastungen überprüfen.

Die Anwendungsfälle des Anbieters sind bereits aus vorherigen Abschnitten bekannt und weitgehend selbsterklärend. Das Mahnwesen wird auch hier nur angedeutet, bei der weiteren Betrachtung der Architekturen jedoch nicht berücksichtigt.

Zusammenfassend lässt sich festhalten, dass die wesentlichen Anforderungen in der Verwaltung der Nutzer und der Verträge mitsamt ihren Tarifen sowie in der Protokollierung und Abrechnung der Angebotsnutzung liegen. Diese Anforderungen sind die Grundlage für die Architekturgestaltung, die im nächsten Abschnitt beschrieben wird.

6.2.1.2 Architekturbeschreibung

Die folgende Abbildung zeigt das Klassendiagramm des Erlösmodells SUBSKRIPTION für Dienstleistungen.

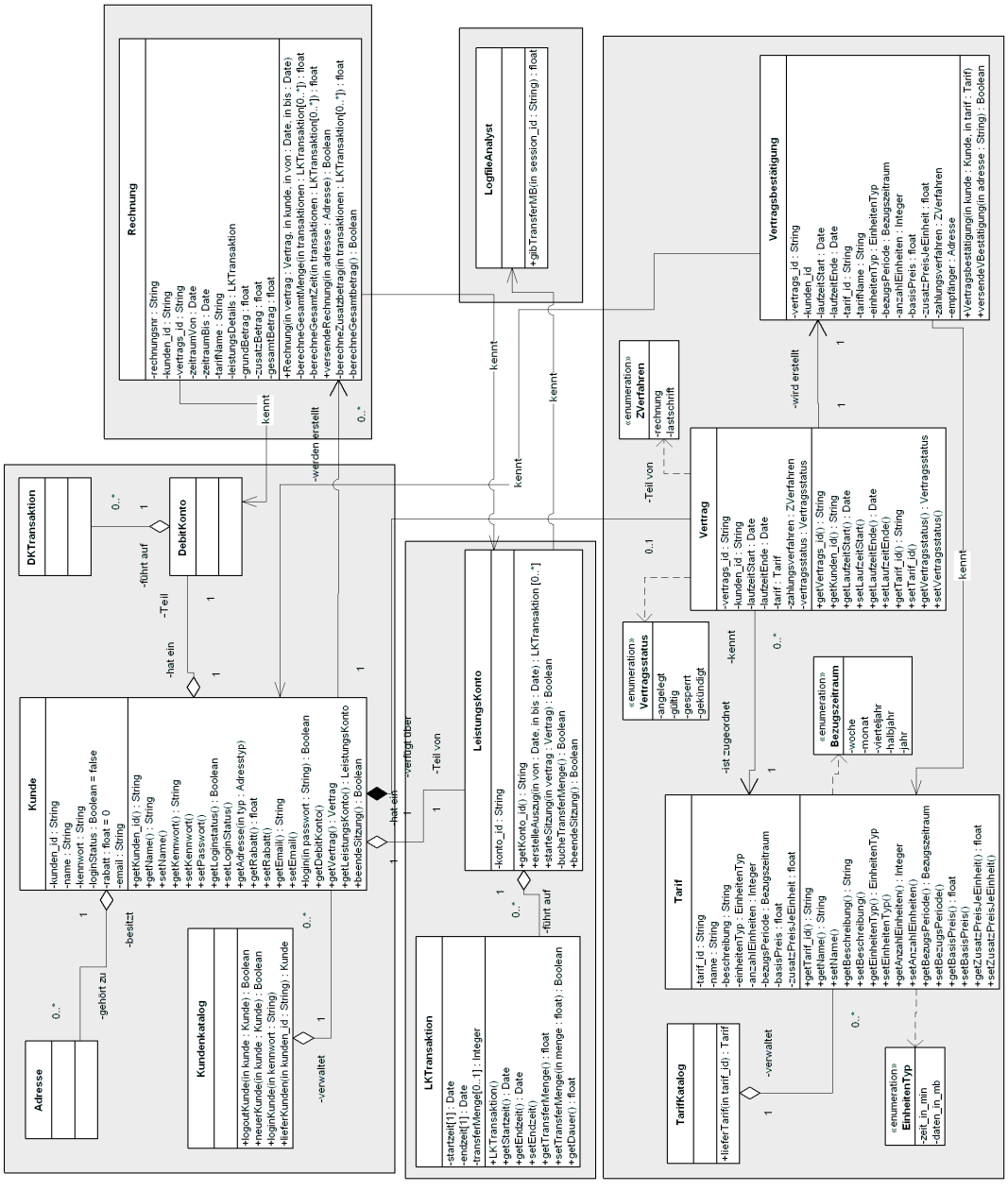


Abbildung 6-42: Statische Architektur des Erlösmodells Subskription in der Variante für Dienstleistungen

Die hervorzuhebenden Anforderungen an die Gestaltung der Software-Referenzarchitektur sind zum einen die Realisierung des Vertrags und der Tarife und zum anderen die Umsetzung der Protokollierung des Leistungsbezugs. Einige neue, bisher nicht bekannte Komponenten befassen sich direkt mit diesen Aufgaben, andere müssen teilweise dazu angepasst werden. Innerhalb der Komponentenbeschreibung wird auf diese Besonderheiten im Folgenden eingegangen.

Die Komponenten der Architektur sind `Kunde`, `LeistungsKonto`, `Vertrag`, `Rechnung` und `LogfileAnalyse`. Sie werden mit ihren Klassen und ihrer Funktionsweise im weiteren Verlauf beschrieben.

Zunächst wird die Komponente zur Kundenverwaltung betrachtet. Abbildung 6-43 zeigt sie mitsamt ihren Klassen.

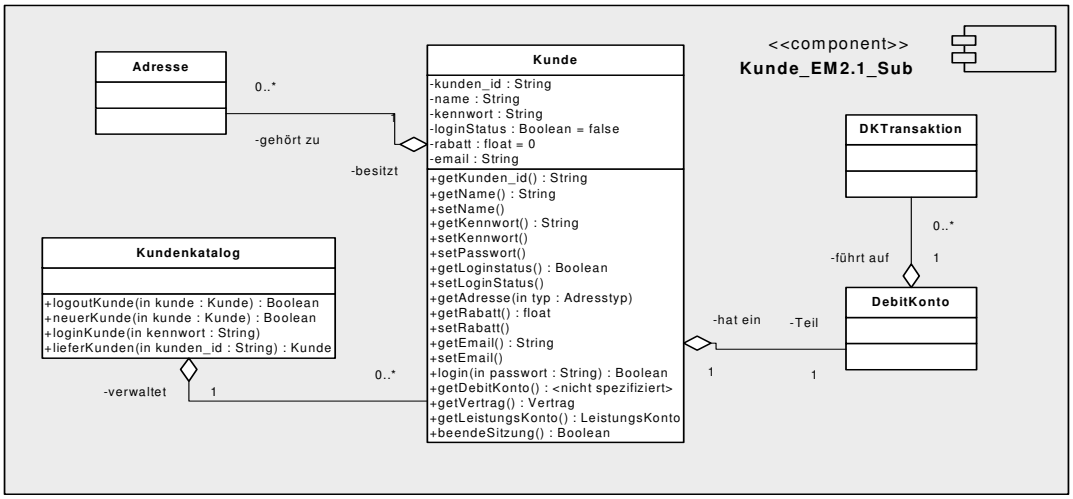


Abbildung 6-43: Komponente Kunde_EM2.1_Sub

Die Komponente ist weitgehend identisch mit der Version des Erlösmodells EINZELTRANSAKTION, weshalb die Klassen `Adresse`, `DebitKonto` und `DKTransaktion` nicht erneut wiedergegeben wurden (zur detaillierten Beschreibung vgl. Abbildung 6-12 auf Seite 133 und die zugehörigen Erläuterungen). Abweichend von den Ähnlichkeiten enthalten die Klassen `KundenKatalog` und `Kunde` zusätzliche Methoden, die in in der Komponente `Kunde_EM1.1_ET` nicht enthalten waren und deshalb hier beschrieben werden.

Der `KundenKatalog` verfügt in der Komponente `Kunde_EM2.1_Sub` zusätzlich über die Methode `logoutKunde`, die als Parameter den abzumeldenden Kunden überliefert bekommt. Diese Methode ruft die Methode `beendeSitzung` des entsprechenden Kunden auf. Der Vorgang des Abmeldens eines Nutzers ist in diesem Erlösmodell aufgrund der Protokollierung seiner Sitzung notwendig und wird deshalb mit diesen Methoden berücksichtigt. Hier zeigt sich somit bereits eine erste Anpassung der bekannten Komponente, die durch die besonderen Anforderungen einer Subskription notwendig ist. Da je nach Tarif die Sitzungsdauer von Relevanz ist, muss diese zu ermitteln sein.

Auch die Klasse `Kunde` ist weitgehend identisch aus der bekannten Komponente übernommen worden, jedoch enthält sie zwei zusätzliche Methoden. Die eine Methode liefert als Rückgabewert das Leistungskonto eines Kunden (Methode `getLeistungsKonto`). Die andere Methode `beendeSitzung` beendet die Sitzung des Kunden und ruft die Methode `beendeSitzung` der Klasse `LeistungsKonto` auf, die die Protokolle persistent speichert und weiter unten detailliert beschrieben wird (die Aufrufbeziehungen werden im Sequenzdiagramm in Abbildung 6-47 auf Seite 175 noch einmal ersichtlich).

Bevor auf die Interaktion der Klassen dieser Komponente mit der restlichen Architektur eingegangen wird, sind zuvor noch die weiteren beteiligten Komponenten zu beschreiben. Die Komponente `LeistungsKonto_EM2.1_Sub` dient der Protokollierung der in Anspruch genommenen Dienste. Sie umfasst die zwei Klassen `LeistungsKonto` und `LKTransaktion` wie die Darstellung in Abbildung 6-44 zeigt.

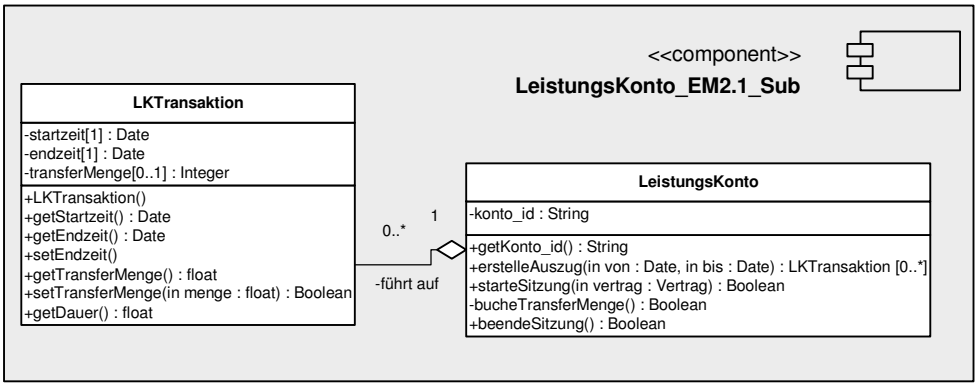


Abbildung 6-44: Komponente LeistungsKonto_EM2.1_Sub

Wie bereits erwähnt stellt die Protokollierung des Leistungsbezugs ein wesentliches Merkmal dieser Architektur dar. Es muss dabei grundsätzlich möglich sein, je nach vertraglich vereinbarter Bezugsgröße entweder die in Anspruch genommene Zeit oder Datenmenge zu ermitteln und festzuhalten. Es ist somit offensichtlich, dass die Protokollierung personalisiert, also für jeden Kunden separat erfolgen muss. Die Aggregation zwischen `Kunde` und `LeistungsKonto` im Klassendiagramm (Abbildung 6-42) kommt dieser Aufforderung nach.

Die Klasse `LeistungsKonto` stellt die Schnittstelle zur Restarchitektur dar und bietet Methoden an, um Transaktionen zu buchen, den Start- und Endzeitpunkt von Sitzungen zu protokollieren und um Auszüge innerhalb eines Zeitraums zu erstellen. Die Methode `starteSitzung` wird mit einer Referenz auf den Vertrag des Kunden als Parameter aufgerufen und führt dazu, dass eine neue Instanz der Klasse `LKTransaktionen` angelegt wird. Da aus Revisionsgründen eine detaillierte und vollständige Auflistung der Buchungen notwendig ist, wird für jede Buchung ein eigener Datensatz erzeugt. Der Konstruktor der Klasse setzt automatisch den Startzeitpunkt als Attributwert fest. Analog führt die Methode `beendeSitzung` dazu, den Endzeitpunkt in der Instanz von `LKTransaktion` festzuhalten, indem die dortige Methode `setEndzeit` aufgerufen wird. Darüber hinaus überprüft

die Methode beendeSitzung anhand des Tarifs des Kunden, ob auch die abgerufene Datenmenge während der Sitzung für eine Abrechnung relevant ist und ruft in Abhängigkeit davon die private Methode `bucheTransfer` auf. Diese ermittelt zunächst die transferierte Menge und sorgt anschließend mittels der Methode `setTransferMenge` der Klasse `LKTransaktionen` für die persistente Speicherung des Wertes. Das Sequenzdiagramm in Abbildung 6-47 auf Seite 175 zeigt diesen Ablauf.

Während die Protokollierung der Dauer einer Stizung unter Verwendung der bereits vorgestellten Methode der `Kunden`-Komponente erfolgen kann, muss für die Erhebung der Datenmenge eine weitere Komponente ihre Dienste bereitstellen. Diese Aufgabe übernimmt die Komponente `LogfileAnalyst`.

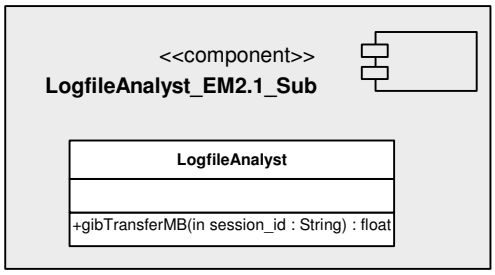


Abbildung 6-45: Komponente LogfileAnalyst_EM2.1_Sub

Abbildung 6-45 zeigt die Komponente in einer vereinfachten Form. Die enthaltene Klasse `LogfileAnalyst` bietet lediglich eine Methode an, die als Rückgabewert die transferierte Datenmenge in MB liefert. `getTransferMB` erhält dazu als Parameter die aktuelle Session-ID der Kundensitzung überliefert und kann anhand dieser Information das Logfile des Web-Servers parsen, um die erforderliche Information zu erhalten.

Die Durchführung der Protokollierung ist mit den bisher vorgestellten Komponenten somit weitgehend möglich. Die zweite Besonderheit des Erlösmodells SUBSKRIPTION stellt die Realisierung des Vertrags und der unterschiedlichen Tarife dar. Diese Geschäftsobjekte werden von der `Komponente Vertrag_EM2.1_Sub` realisiert. Sie ist zusätzlich auch für die Protokollierung relevant, da sie Auskunft über die zu speichernde Bezugsgröße erteilen kann.

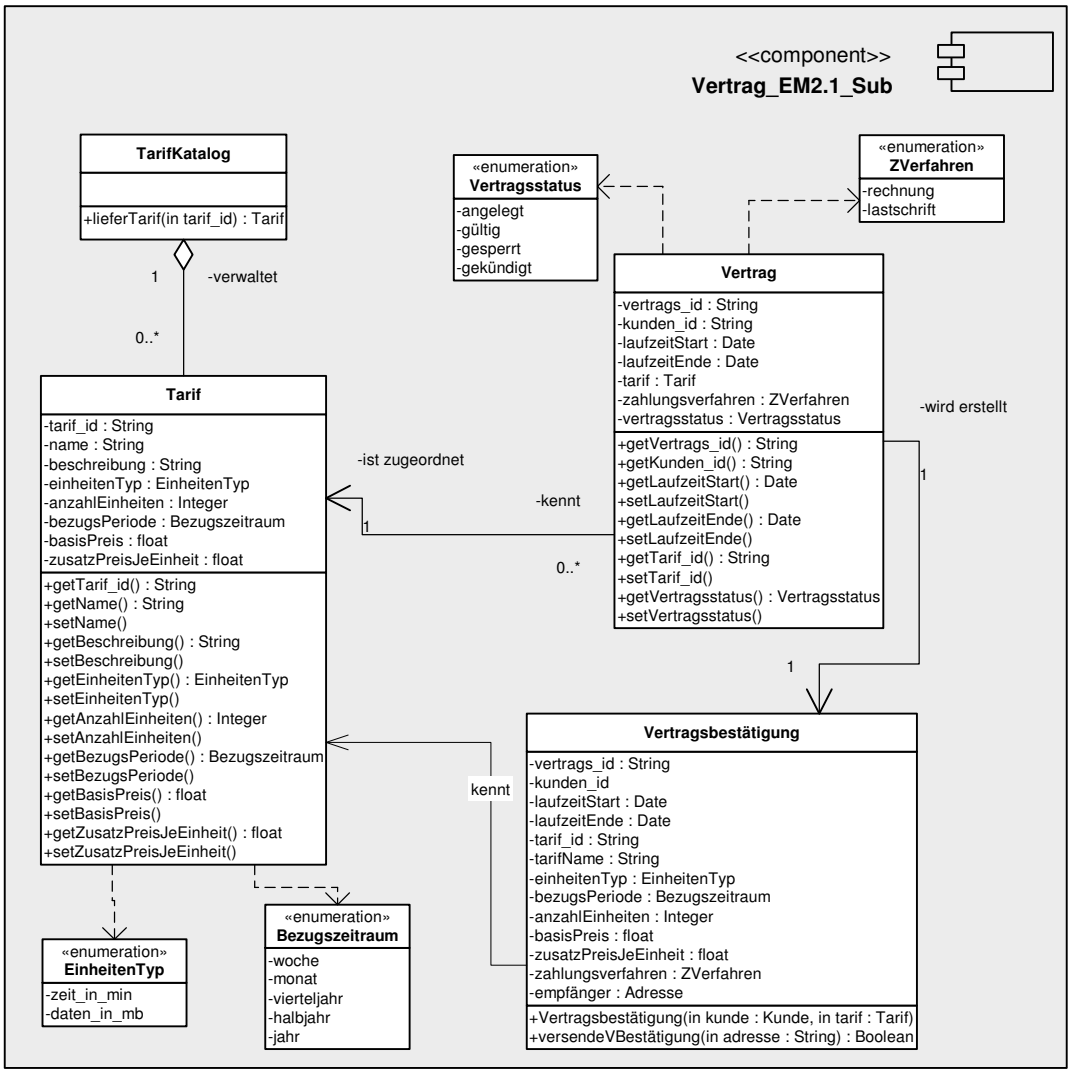


Abbildung 6-46: Komponente Vertrag_EM2.1_Sub

Wie das Diagramm in Abbildung 6-46 zeigt, enthält die Komponente die Klassen `Vertrag` `Vertragsbestätigung`, `TarifKatalog` und `Tarif`. Die wichtigsten Klassen sind hierbei `Vertrag` und `Tarif`. Da beide Geschäftsobjekte eng miteinander verbunden sind, werden sie auch in einer Komponente realisiert, jedoch in eigenständigen Klassen. Der Vertrag dient dabei sozusagen als *Vertragskopf*, der grundsätzliche Informationen enthält, während eine Ausprägung des Tarifs die *Vertragsdetails* repräsentiert. Tarife müssen änderbar sein, ohne dass alle Verträge direkt davon betroffen sind. Einem Vertrag ist immer genau ein Tarif zugeordnet, während ein Tarif von vielen Kunden gewählt werden kann und somit vielen Verträgen zugewiesen sein kann.

Ein Vertrag zwischen einem Kunden und dem Anbieter ist in diesem Erlösmodell vorrangig durch seine Laufzeit (Attribute `laufzeitStart` und `laufzeitEnde`), seinen Status (`vertragsstatus`), den gewählten Tarif (`tarif`) sowie das vereinbarte Zahlungsverfahen (`zahlungsverfahren`) charakterisiert. Entsprechend der Anforderungsanalyse im vorherigen Abschnitt werden in dieser Referenzarchitektur die Status *angelegt*, *gültig*, *gesperrt* und *gekündigt* unterschieden. Initial wird der Vertrag durch die Absicht des Kunden angelegt, bevor er im Anschluss an internen Prüfungen manuell oder automatisch gültig wird. Um den Zugriff der Kunden blockieren zu können, besteht die Möglichkeit, einen Vertag zu sperren. Die Funktionsweise wird in den Sequenzdiagrammen ersichtlich, die weiter unten zu sehen sind.

Als mögliche Zahlungsverfahren sind Rechnung und Lastschrift vorgesehen. Beide Verfahren eignen sich aufgrund der Identifizierung und der vertraglichen Bindung des Kunden und bieten beiden Parteien ein hohes Maß an Sicherheit und Komfort, ohne zusätzliche Kosten zu verursachen.

Das Attribut `tarif` enthält einen Verweis auf eine Instanz der gleichnamigen Klasse. In einem Tarif sind die wesentlichen Parameter des Abonnements hinterlegt. Das Attribut `einheitenTyp` kann entweder die Ausprägung Zeit (zeit_in_min) oder Menge (daten_in_mb) enthalten und legt damit fest, welche Einheit für die Abrechnung relevant ist. Somit wird unterschieden, ob sich die Dienstleistung auf die Dauer der Nutzung oder aber auf das Volumen der Nutzung bezieht. Der Umfang der abgedeckten Nutzung wird in dem Attribut `anzahlEinheiten` festgelegt. Diese Anzahl kann maximal innerhalb eines bestimmten Zeitraums vom Kunden in Anspruch genommen werden, ohne den Festpreis für die Subskription zu überschreiten. Dieser Zeitraum wird als Bezugszeitraum im gleichnamigen Attribut hinterlegt und kann als gängige Werte eine *Woche*, einen *Monat*, ein *Vierteljahr*, ein *Halbjahr* oder ein *Jahr* umfassen. Diese Vorgaben sind je nach Anwendungsfall beliebig anpassbar und sind hier nur beispielhaft vorgegeben. Der Preis, der maximal den Bezug der `anzahlEinheiten` abdeckt, wird als `basisPreis` bezeichnet. Falls ein Kunde innerhalb des Bezugszeitraums mehr als diese vertraglich vereinbarte Menge an Einheiten abruft, muss er für jede zusätzliche Einheit einen gesonderten Betrag bezahlen, der als `zusatzPreisJeEinheit` in der Klasse hinterlegt wird. Die Methoden der Klasse beziehen sich lediglich auf das Setzen und Lesen der Attributwerte und werden deshalb nicht näher betrachtet.

Die Klasse `TarifKatalog` übernimmt ähnlich wie die bisher vorgestellten Katalog-Klassen die Verwaltung der zugewiesenen Instanzen; hier der zugewiesenen Instanzen der Klasse `Tarif`. Die einzige relevante Methode `lieferTarif` erhält als Parameter die Tarif-ID übergeben und liefert als Rückgabewert einen Verweis auf die zugehörige Instanz.

Als letzte Klasse der Komponente ist die `Vertragsbestätigung` vorzustellen. Sie verfügt neben dem Konstruktor nur noch über die Methode `versendeVBestätigung`, die ein digitales Exemplar der Bestätigung an die überlieferte email-Adresse des Kunden verschickt. Der Konstruktor wird aufgerufen, sobald dem Vertrag der Status *gültig* zugewiesen wird. Anhand der übergebenen Referenzen auf `Kunde` und `Tarif` kann er die enthaltenen Attributwerte ermitteln und abspeichern.

Nachdem nun die wesentlichen Komponenten der Architektur vorgestellt wurden, soll das folgende Sequenzdiagramm ihr Zusammenspiel während der Protokollierung der Angebotsnutzung durch einen Kunden veranschaulichen.

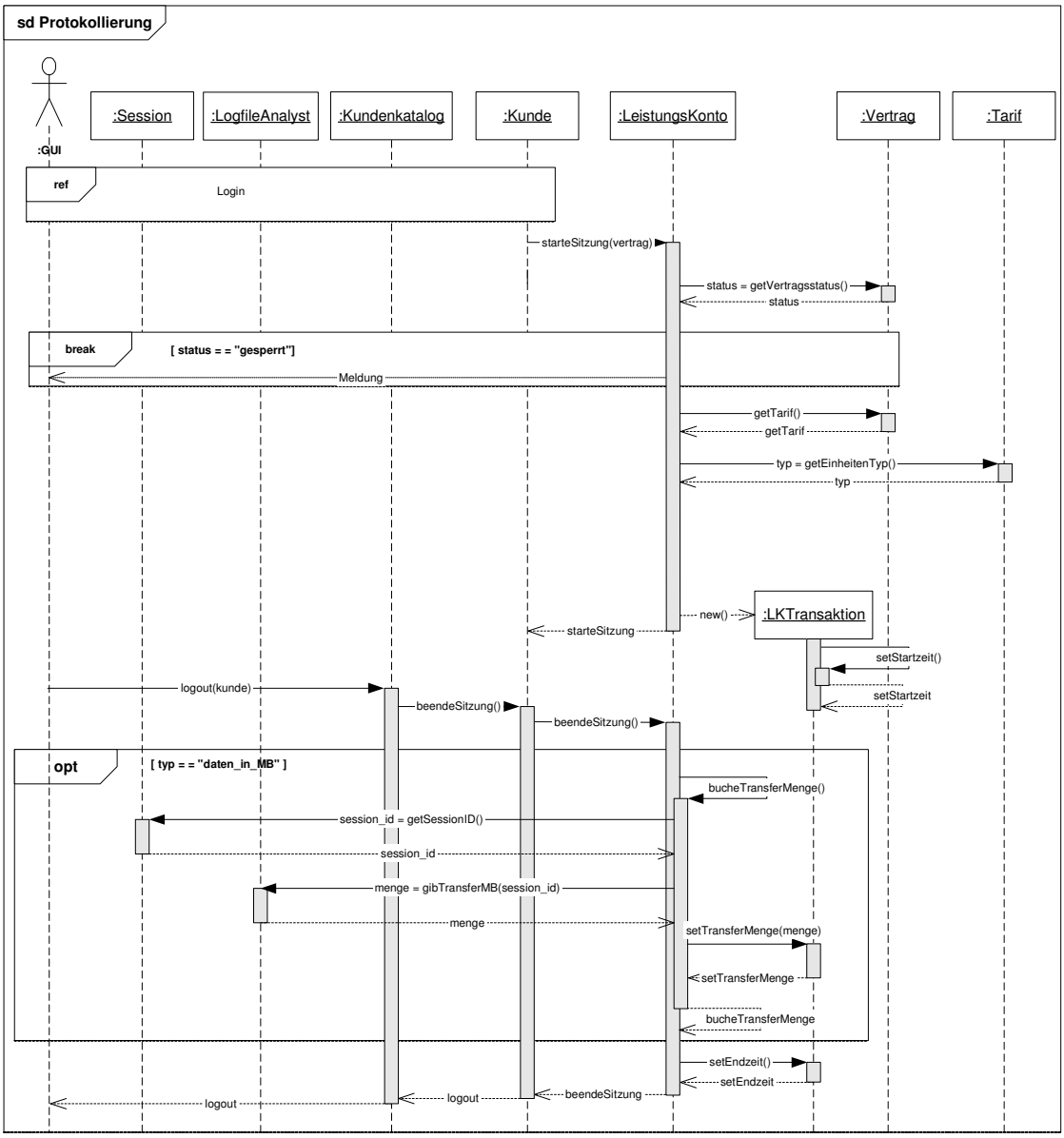


Abbildung 6-47: Sequenzdiagramm zur Protokollierung der Angebotsnutzung

Sobald sich der Nutzer über die GUI durch sein Kennwort/Passwort-Kombination am System legitimiert hat und der Kundenkatalog die passende Instanz zugewiesen hat (siehe die Interaktionsreferenz *login* in Abbildung 6-13 auf Seite 134), ruft die Klasse `Kunde` die Methode `starteSitzung` der Klasse `LeistungsProtokoll` auf und übergibt eine Referenz auf den zugehörigen Vertrag. Nun wird überprüft, ob der Kunde für einen Bezug

der Dienstleistungen berechtigt ist oder nicht. Dazu wird der Vertragsstatus ausgelesen (`getVertragsstatus`). Falls dieser den Wert gesperrt hat, wird der weitere Verlauf unterbrochen und es erscheint eine qualifizierte Rückmeldung an die GUI. Das Abbruchfragment *break* zeigt an, dass in diesem Fall der weitere Ablauf des Diagramms nicht mehr fortgeführt wird. Eine konkrete Behandlung des Abbruchs über die Meldung hinaus wird hier nicht vorgenommen, sie ist in einem konkreten Anwendungsfall geeignet zu implementieren.

Sofern der Kunde jedoch berechtigt ist, die Nutzung in Anspruch zu nehmen, ermittelt die Methode die zu protokollierenden Einheit. Dazu wird die Methode `getEinheitenTyp` der Klasse `Tarif` aufgerufen, deren Instanz zuvor über die Methode `getTarif` von `Vertrag` ermittelt wurde. Als letzte Aktion ruft die Methode den Konstruktor der Klasse `LKTransaktion` auf, wodurch eine neue Instanz erzeugt wird, die die aktuelle Sitzung protokolliert und sofort die Startzeit speichert (`setStartzeit`).

Der Ablauf der eigentlichen Nutzung der Dienstleistung ist für die Protokollierung nicht relevant. Die nächste wesentliche Aktion des Nutzers ist das Beenden der Sitzung, was er durch das Abmelden signalisiert. Das Sequenzdiagramm zeigt den Methodenaufruf `logout`, der von der GUI an den KundenKatalog übermittelt wird. Es ist jedoch auch möglich, dass eine Sitzung nicht explizit durch den Nutzer beendet wird und erst durch das Erreichen der timeout-Grenze automatisch vom Server veranlasst wird. Dieser Sonderfall wird nicht betrachtet, da bei dem vorliegenden Erlösmodell SUBSKRIPTION davon ausgegangen wird, dass der Kunde selber ein großes Interesse daran hat, seine Sitzungen ordnungsgemäß zu beenden und sich somit (vor allem, wenn er ein zeitliches Abrechnungsmodell gewählt hat) explizit abmelden wird.

Nachdem die Klasse `KundenKatalog` über das Ende der Sitzung informiert worden ist, ruft sie die Methode `beendeSitzung` des zugehörigen Kundenobjekts auf, die wiederum die Methode `beendeSitzung` der Klasse `LeistungsKonto` aktiviert. Diese Methode sorgt unter Verwendung der Methode `setEndzeit` dafür, dass der Zeitpunkt des Abmeldens von der Klasse `LKTransaktion` festgehalten wird und somit die Sitzungsdauer ermittelt werden kann. Um statistische Auswertungen zu ermöglichen, sieht die Architektur diese Protokollierung auch dann vor, wenn kein Tarif mit zeitlicher Bezugseinheit gewählt worden ist.

Sofern der Kunde ein mengenbezogenes Abonnement vereinbart hat, werden von der Methode weitere Aktionen durchgeführt, was das optionale Fragment *opt* verdeutlicht, dessen Interaktionen nur dann durchlaufen werden, wenn diese Bedingung zutrifft. Die Methode `beendeSitzung` ruft in diesem Fall die private Methode `bucheTransferMenge` auf. Die wiederum ermittelt die übertragene Datenmenge der soeben beendeten Sitzung des Kunden und verbucht den Wert in der zugehörigen Transaktion. Zunächst ermittelt die Methode dazu die aktuelle Session-ID, die sie von der Session erhält (`getSessionID`). Der Rückgabewert wird als Parameter an die Klasse `LogfileAnalyst` beziehungsweise an deren Methode `gibTransferMB` übergeben. Diese Methode analysiert das Logfile des Webservers und liefert die in dieser Session vom Kunden abgeforderte Datenmenge zurück. Der Rückgabewert wird von `bucheTransferMenge` wiederum an die Methode `bucheWert` der Klasse `LKTransaktion` als Parameter übergeben, wo er im entsprechenden Attribut persistent hinterlegt wird.

Neben der Protokollierung des Leistungsbezugs stellt die Abrechnung eine weitere Anforderung dar. Dazu steht die Komponente `Rechnung_EM2.1_Sub` zur Verfügung, die in Abbildung 6-48 dargestellt ist.

Anders als im vorherigen Erlösmodell EINZELTRANSAKTION wird eine Rechnung hier nicht ereignisgesteuert beim Kauf eines Produktes, sondern zeitgesteuert erstellt. Das Intervall wird durch die Bezugsperiode bestimmt - ein Attribut des Tarifs des Kunden. Mit Ablauf der Bezugsperiode muss die Rechnung zunächst ermitteln, inwiefern die in Anspruch genommenen Leistungen den Tarif überschreiten oder nicht: Falls mehr als die maximal zulässigen Einheiten bezogen wurden, sind diese zusätzlich zu berechnen; weniger konsumierte Einheiten verfallen dagegen und werden nicht gutgeschrieben.

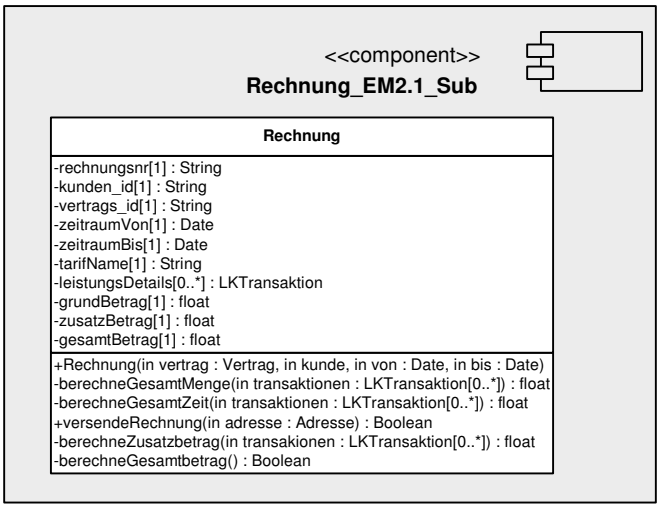


Abbildung 6-48: Komponente Rechnung_EM2.1_Sub

Die Komponente umfasst die Klasse `Rechnung`. Ihr Konstruktor wird mit Ablauf einer Rechnungsperiode automatisch aufgerufen. Dieser Aufruf erfolgt durch eine Systemkomponente, die fortlaufend kontrolliert, ob Bezugsperioden der Subskriptionen abgelaufen sind und somit eine Rechnung zu erzeugen ist. Diese Systemkomponente wird im Sequenzdiagramm (Abbildung 6-49) als Controller bezeichnet, jedoch wird nicht näher auf sie eingegangen. Beim Aufruf des Konstruktors werden ihm eine Referenz auf den Kunden, dessen Vertrag und der Abrechnungszeitraum als von-bis-Wertepaar übergeben. Die Attribute `rechnungsnr`, `kunden_id`, `vertrags_id`, `zeitraum_von`, `zeitraum_bis`, `tarifName` sind selbsterklärend und werden entweder vom Konstruktor eigenständig gesetzt oder zunächst von den beteiligten Klassen ermittelt. Das Sequenzdiagramm in Abbildung 6-49 zeigt den Ablauf der Methodenaufrufe, die der Konstruktor durchführt und die Interaktion der involvierten Klassen untereinander.

Das mengenwertige Attribut `leistungsDetails` enthält eine Liste der getätigten Sitzungen mitsamt der relevanten Informationen. Diese Informationen werden somit auf der Rechnung in Form eines Einzelverbindungsnachweises für den Kunden ausgewiesen. Anhand

der soeben beschriebenen Protokollierung können die im abzurechnenden Zeitraum erzeugten Instanzen der Klasse `LKTransaktionen` herangezogen werden, um diese Informationen auszuweisen. Die Klasse `LeistungsKonto` stellt dazu die Methode `erstelleAuszug` bereit und liefert ein entsprechendes Listobjekt zurück, wie dem Interaktionsdiagramm zu entnehmen ist.

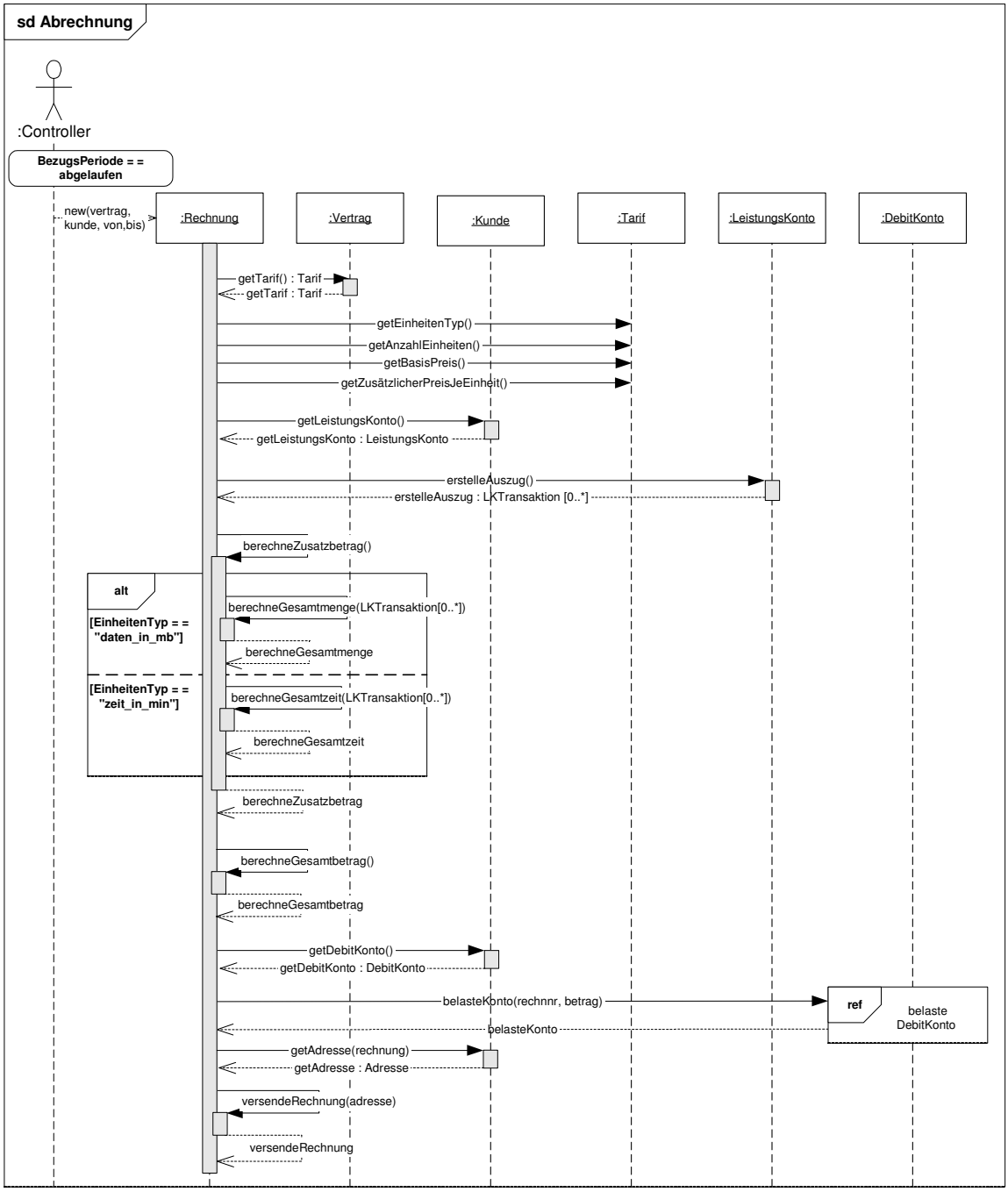


Abbildung 6-49: Sequenzdiagramm zur Abrechnung der Subskription

Um die Werte der Attribute `zusatzBetrag` und `gesamtBetrag` zu ermitteln, werden die Methoden `berechneZusatzbetrag` und `berechneGesamtbetrag` verwendet. Das Attribut `grundBetrag` dagegen wurde bereits vorher aus der Klasse `Tarif` ausgelesen (`getBasisPreis`).

Die Methode `berechneZusatzbetrag` muss in Abhängigkeit der Bezugseinheit Zeit oder Datenmenge die Zusatzkosten ermitteln, wozu sie eine der beiden privaten Methoden `berechneGesamtzeit` oder `berechneGesamtmenge` aufruft. Beide Methoden verwenden das überlieferte Listobjekt der Transaktionen und ermitteln anhand dessen die insgesamt verbrauchten Einheiten, die sie als Rückgabewert zurückliefern. Die Methode `berechneZusatzbetrag` kann anhand dieser Menge und der zuvor ermittelten Anzahl der vertraglich vereinbarten maximalen Einheiten (`getAnzahlEinheiten` ermittelte diesen Wert) die darüber hinaus verbrauchten Einheiten ermitteln und mit dem entsprechenden Preis gewichten (`getZusatzPreisJeEinheit` hat diesen Wert bereits ermittelt). Das Ergebnis kann die Methode dem Attribut `zusatzBetrag` zuweisen.

Der Gesamtbetrag wird anschließend anhand der beiden Werte `grundBetrag` und `zusatzBetrag` einfach von der Methode `berechneGesamtbetrag` ermittelt.

In dieser Variante ist die Rechnungs-Komponente auch für das Belasten des Debitoren-Kontos des Kunden verantwortlich. Dazu ermittelt der Konstruktor im nächsten Schritt das `DebitKonto` über den `Kunden` und ruft dort die Methode `belasteKonto` auf, der er die Rechnungsnummer und den zu belastenden Betrag als Parameter übergibt. Die Kontobelastung wurde als Interaktionsreferenz bereits in Abbildung 6-23 auf Seite 146 vorgestellt.

Abschließend wird die Methode `versendeRechnung` aufgerufen, die die Rechnungsadresse des Kunden ermittelt und den Versand einer physikalische Rechnung veranlasst.

6.2.2 Variante digitales und physikalisches Gut

Die erste Variante des Erlösmodells bezog sich auf Dienstleistungen wie sie beispielsweise ein Internet Service Provider durch seinen Internetzugang anbietet. Eine weitere Variante dieses Modells ist die Subskription digitaler oder physikalischer Güter, deren Architektur hier hergeleitet und beschrieben wird. Die Merkmale sind in der folgenden Abbildung 6-50 als Übersicht zu sehen. Im Detail wurde die Variante in Abschnitt 4.2.2.2 auf Seite 72 behandelt.

Abbildung 6-50: Erlösmodellklassifikation Subskription Variante digitales und/oder physikalsiches Gut

Beispielsweise kann eine Subskription den Bezug einer bestimmten Anzahl von Musikfiles oder Bücher innerhalb eines definierten Zeitraums beinhalten. Im Folgenden wird untersucht, inwiefern sich diese Änderung gegenüber der vorherigen Variante auf die Architektur auswirkt.

6.2.2.1 Anforderungen und Systemabgrenzung

Die Verwaltung von Produkten (digital oder physikalisch) wurde bereits im Erlösmodell EINZELTRANSAKTION in der Variante für physikalische Güter betrachtet. Die dort aufgeführten Anwendungsfälle im Rahmen der Produktverwaltung gelten auch hier. Eine Beschreibung der Anwendungsfälle einer Produktverwaltung für physikalische Güter wurde bereits in Abschnitt 6.1.1.1 auf Seite 124 durchgeführt und wird hier deshalb nicht wiederholt. Die Abbildung 6-6 auf Seite 125 zeigt das Anwendungsfalldiagramm. Digitale Güter sind grundsätzlich genauso zu verwalten, wobei bei ihnen aufgrund ihrer Optionen zur digitalen Vervielfältigung keine Verfügbarkeitsverwaltung notwendig ist. Von dieser Feinheit wird hier jedoch abstrahiert.

Ebenfalls wird der Bestellvorgang der Güter außen vor gelassen. Sowohl der Umgang mit einem Warenkorb als auch die Initiierung, Änderung und Bestätigung einer Bestellung, wie sie im Erlösmodell der EINZELTRANSAKTION untersucht wurde, bleibt hier unbeachtet. Grund dafür ist die Tatsache, dass der Fokus des Erlösmodells SUBSKRIPTION auf der Protokollierung und Abrechnung der bezogenen Güter liegt und nicht auf der Bestellabwicklung. Darüber hinaus wurden die Achitekturbausteine bereits in Abschnitt 6.1 behandelt und können im Falle einer konkreten Anwendung bei Bedarf integriert werden.

Die Anforderungen und die Komponenten zur Lagerung und Zustellung von physikalischen Gütern sowie zur Zustellung von digitalen Gütern bleibt in der Beschreibung der Architekturen für das Erlösmodell außerhalb der Betrachtungen.

Eine Änderung ergibt sich hinsichtlich der Angebotsnutzung durch die Kunden. Die folgende Abbildung zeigt das angepasste zugehörige Szenario.

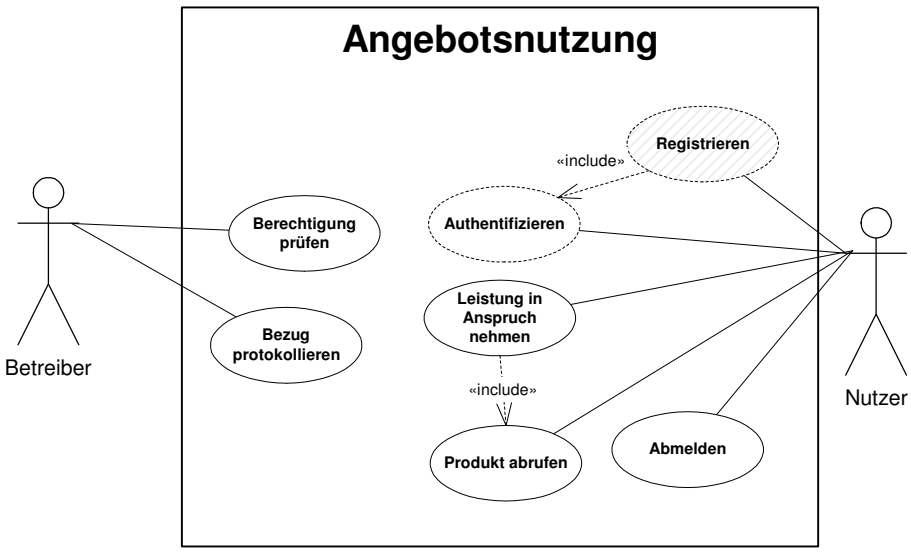


Abbildung 6-51: Angebotsnutzung bei digitalen und physikalischen Gütern

Die Abweichung zur vorherigen Variante ergibt sich im Anwendungsfall Leistung in Anspruch nehmen. Wie in Abbildung 6-51 zu sehen ist, beinhaltet der Anwendungsfall immer den Anwendungsfall Produkt abrufen. Da sich die Variante mit der Subskription von Produkten und nicht von Diensten befasst, muss das Produkt vom Kunden angefordert werden. Digitale Güter können direkt durch einen Download zum Kunden gelangen, physikalische Produkte müssen bestellt, kommissioniert und zugestellt werden. Beide Spezialisierungen werden mit dem Anwendungsfall Produkt abrufen behandelt.

Zusammenfassend lässt sich festhalten, dass zum einen die Anforderungen an eine Produktverwaltung hinzugekommen sind und sich zum anderen die Art der Angebotsnutzung verändert hat. Diese beiden Neuerungen müssen durch das Hinzufügen neuer und durch das Anpassen bekannter Architekturbausteine berücksichtigt werden. Der folgende Abschnitt befasst sich mit dieser Thematik.

6.2.2.2 Architekturbeschreibung

Die Architekturbeschreibung ist in weiten Teilen identisch mit der Architektur des Erlösmodells SUBSKRIPTION von Dienstleistungen (vgl. Abbildung 6-42 auf Seite 169). Unterschiede ergeben sich vor allem aus der hier notwendig gewordenen Produktverwaltung und der abweichenden Art der Protokollierung, da hier nicht mehr die Sitzungsdauer beziehungsweise die darin transferierte Datenmenge relevant sind, sondern die tatsächlich bezogenen Produkte. Die Referenzarchitektur ist an diese Veränderungen angepasst worden und wird im Folgenden vorgestellt.

Die folgende Abbildung 6-52 zeigt das angepasste Klassendiagramm für die Variante der digitalen und physikalischen Güter.

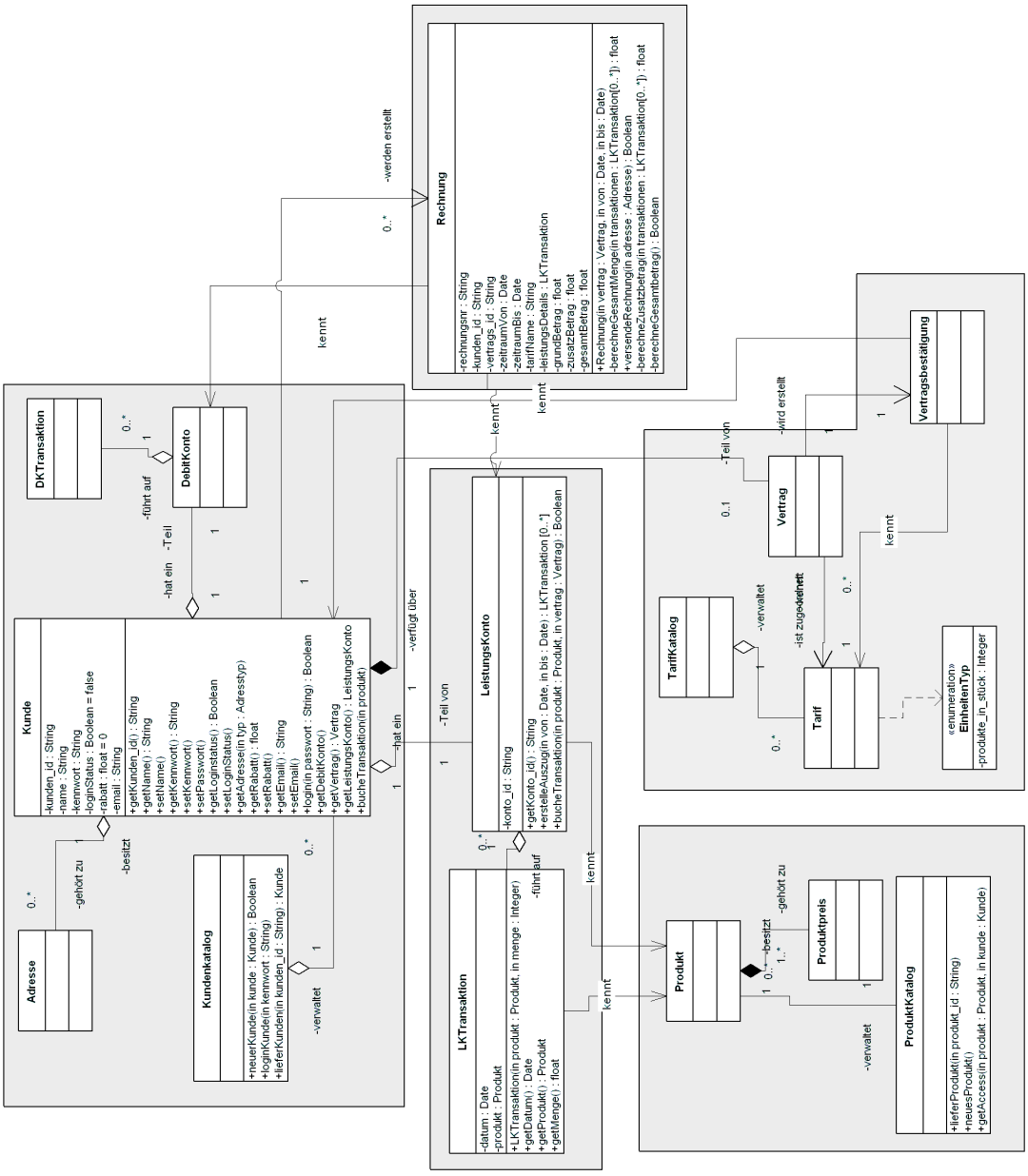


Abbildung 6-52: Statische Architektur des Erlösmodells Subskription in der Variante für digitale und physikalische Güter

Anhand der Klassen, die nicht mit ihren Eigenschaften und Methoden dargestellt sind, ist ersichtlich, dass ein Großteil der Komponenten aus den bisherigen Architekturen wiederverwendet werden kann und sich nur wenige Änderungen ergeben haben. Während die

Komponenten `Vertrag`, `Rechnung`, `Kunde` und `LeistungsKonto` zum Teil nur sehr gering angepasst wurden, entfiel die Komponente `LogfileAnalyst` komplett, und die Komponente `Produkt` wurde neu hinzugefügt.

Die in dieser Variante zusätzlich erforderliche Produktverwaltung wird durch die Komponente `Produkt_EM2.2_Sub` abgedeckt. Sie ähnelt der Version `Produkt_EM1.1_ET` aus dem Erlösmodell EINZELTRANSAKTION, und die Klassen `Produkt` und `Produktpreis` sind identisch übernommen worden, weshalb hier nur die Klassennamen wiedergegeben wurden. Details zu diesen Klassen sind auf Seite 136ff nachzulesen. Die Klasse `ProduktKatalog` weist jedoch in der vorliegenden Version eine zusätzliche Methode auf, die zur Abwicklung der Subskription notwendig ist.

Wie in Abbildung 6-53 zu erkennen ist, handelt es sich dabei um die Methode `getAccess`.

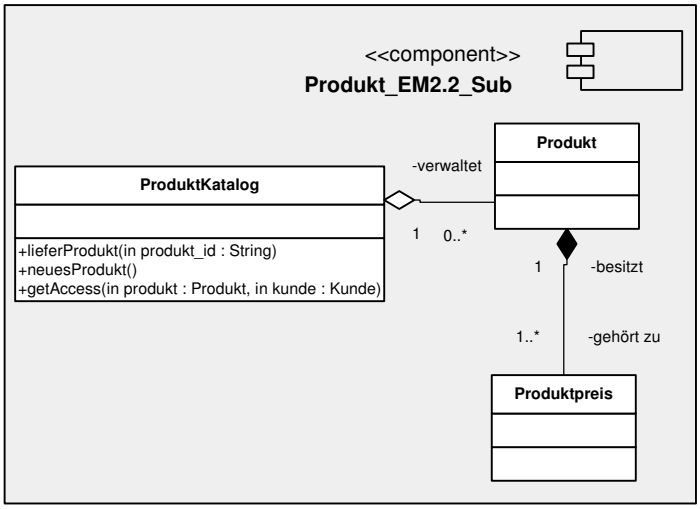


Abbildung 6-53: Komponente Produkt_EM2.2_Sub

Die Methode `getAccess` ist erforderlich, um den Bezug eines Produktes als Aktion zu registrieren und entsprechend die Protokollierung im Rahmen der Subskription zu veranlassen. Die Methode wird aufgerufen, sobald ein Kunde ein Produkt erwirbt. Wann genau dieser Aufruf erfolgt, hängt sehr stark vom konkreten Anwendungsfall ab. Da hier die Variante für digitale *und* physikalische Güter gemeinsam untersucht wird, können nur Beispiele angegeben werden. Im Umgang mit digitalen Gütern kann die Methode etwa dann aufgerufen werden, wenn sich ein Kunde für den Download eines Produktes entschieden (und ihn bestätigt) hat und der Zugriff auf das File erfolgen soll. Werden physikalische Güter vertrieben, wird die Methode eher im Zusammenhang mit der Abarbeitung einer Bestellung aufgerufen; beispielsweise wenn ein Statuswechsel der Bestellung erfolgt und die Ware das Lager verlässt.

Unabhängig von den konkreten Bedingungen des Methodenaufrufs werden eine Referenz auf das erworbene Produkt und auf den erwerbenden Kunden übermittelt. Die Methode selber ruft ihrerseits die Methode `bucheTransaktion` des Kundenobjekts auf. Da diese Methode bisher kein Bestandteil der Kundenklasse war, muss auch diese angepasst werden. Die folgende Abbildung zeigt die Komponente `Kunde_EM2.2_Sub`.

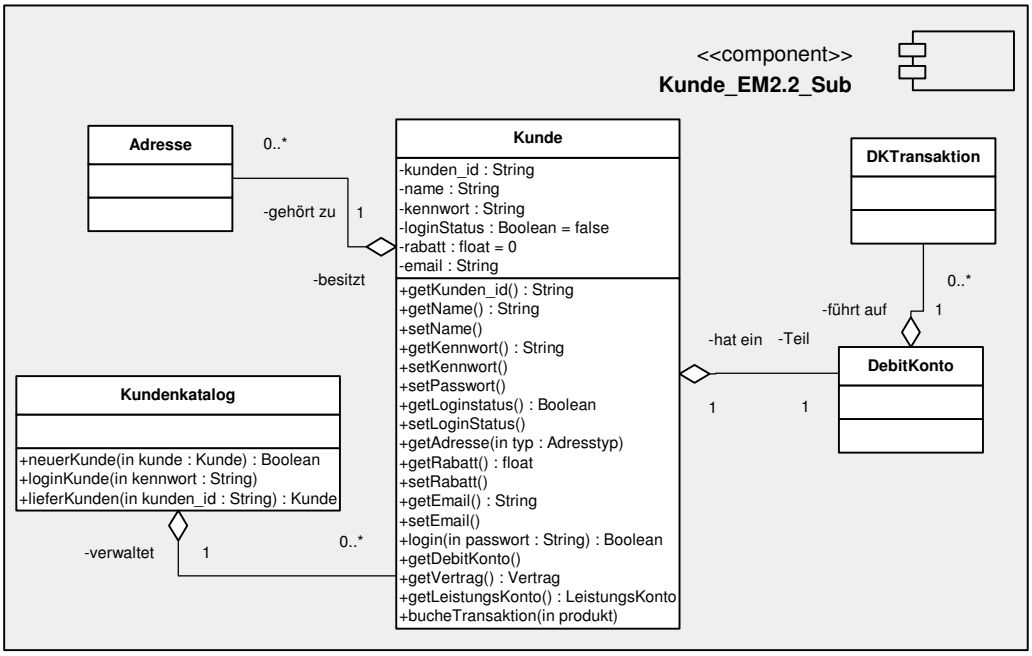


Abbildung 6-54: Komponente Kunde_EM2.2_Sub

Die Klassen `Adresse`, `DKTransaktion`, `DebitKonto` sind identisch mit der Version `Kunde_EM1.1_ET` und werden hier nur referenziert. `KundenKatalog` ist ebenfalls identisch mit dieser ersten betrachteten Version und weicht somit von der vorherigen Variante für Dienstleistungen ab, wo die Methode `logoutKunden` eine Protokollierungsaufgabe übernahm, die hier nicht benöttigt wird und in dieser Form auch nicht eingesetzt werden kann. Aus diesem Grund ist die Klasse hier noch einmal mit ihren bekannten Methoden aufgeführt worden (zur Beschreibung vgl. Seite 133ff).

Die Komponente zur Verwaltung der Verträge und Tarife kann wiederverwendet werden, indem die aus der vorherigen Variante bekannte Version `Vertrag_EM2.1_Sub` eingesetzt wird. Es findet lediglich eine Änderung in der Wertemenge des Enumeration Datentypen `EinheitenTyp` statt. Das Attribut `einheitenTyp` wird in der Klasse `Tarif` verwendet und konnte in der vorherigen Variante für die Subskription von Dienstleistungen sowohl die Ausprägung *zeit_in_min* als auch *daten_in_mb* annehmen. Da in dieser Variante digitale oder physikalische Güter abonniert werden, hanelt es sich um Stückgut, weshalb dem Attribut die Ausprägung *produkte_in_stück* fest zugewiesen wird. Der folgende Ausschnitt der Komponente zeigt diese feste Wertzuweisung.

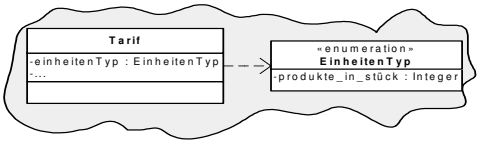


Abbildung 6-55: Ausschnitt aus der Komponente Vertrag mit angepasstem Datentyp

Anpassungen müssen auch für die Komponenten Leistungskonto und Rechnung durchgeführt werden. Diese beziehen sich auf die veränderte Art der Protokollierung und Abrechnung, da nun der Abruf einzelner Produkte (Stückgut) festgehalten werden muss und nicht mehr die einzelne Sitzung des Kunden relevant ist. Abbildung 6-56 zeigt die angepasste Komponente für das Leistungskonto in der Version `LeistungsKonto_EM2.2_Sub`.

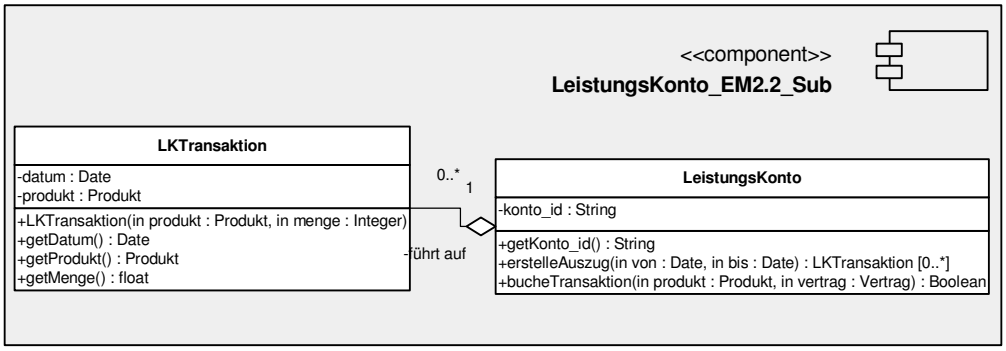


Abbildung 6-56: Komponente LeistungsKonto_EM2.2_Sub

Die Methoden der Klasse `LeistungsKonto` sind teilweise die selben wie in der Version aus der vorherigen Variante. Änderungen bestehen jedoch zum einen darin, dass es keine Methoden mehr zur Protokollierung des Startens (`starteSitzung`) und der Beendigung (`beendeSitzung`) einer Sitzung gibt, und zum anderen darin, dass die Methode `bucheTransaktion` verändert wurde. Die Klasse `LKTransaktion` bucht nicht mehr den Start- und den Endzeitpunkt der Sitzung, sondern lediglich den Abruf eines Produktes zu einem Zeitpunkt.

Die Funktionsweise der Methode `bucheTransaktion` wurde in dieser Version an die veränderten Anforderungen angepasst. Sie erhält bei ihrem Aufruf als Parameter eine Referenz auf das erworbene Produkt und auf den Vertrag des Kunden. Der Vertag ist notwendig, da vor einem Produktaufruf zunächst die Berechtigung des Kunden für diese Aktion überprüft werden muss. Diese Aufgabe wird in der Variante für Dienstleistungen bereits zu Beginn einer Sitzung übernommen (dort von der Methode `starteSitzung`), ist hier jedoch nicht Abhängig von der Sitzung, sondern nur von dem Bezug der Produkte und muss somit angepasst werden.

Bei erfolgreicher Überprüfung der Berechtigung sorgt die Methode dafür, dass eine neue Instanz von `LKTransaktion` angelegt und der Erwerb des Produktes dort persistent gespeichert wird. Den Ablauf der Protokollierung zeigt das folgende Sequenzdiagramm in Abbildung 6-57.

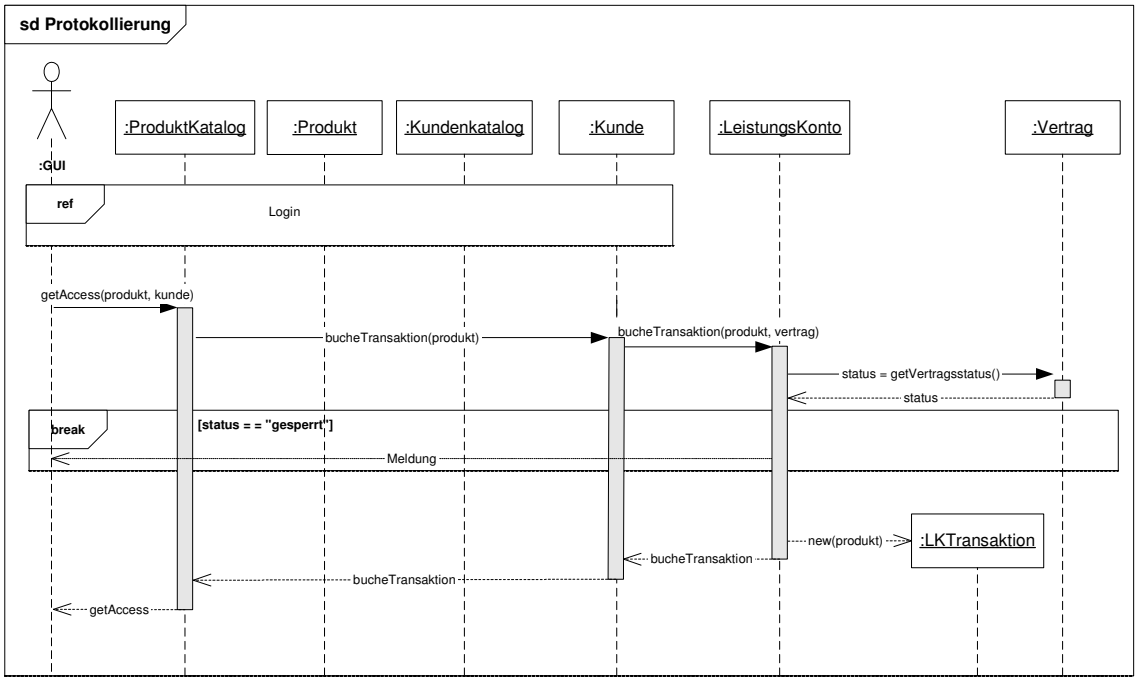


Abbildung 6-57: Sequenzdiagramm zur Protokollieung von Produktabrufen

Es wird deutlich, dass nach einem Login nicht direkt die Berechtigung überprüft und die Sitzung protokolliert wird. Stattdessen wird die Protokollierung erst mit dem Abruf eines Produktes über den Methodenaufruf `getAccess` aktiviert. Das als Parameter übermittelte Kundenobjekt sorgt für den Aufruf der zugehörigen Methode des Leistungskontos, wo zunächst die Überprüfung der Zugangsberechtigung eines Nutzers anhand seines Vertrages erfolgen und im negativen Fall eine Meldung an die Oberfläche erzeugt würde. Das Abbruch Fragment verdeutlicht, dass die weiteren Interaktionen in diesem Fall nicht durchlaufen würden. Sofern der Kunde für den Zugriff berechtigt ist, erfolgt die Protokollierung durch das Erzeugen einer neuen Instanz der Klasse `LKTransaktion`.

Auch die Rechnung, die zur Berechnung des Rechnungsbetrages auf die Protokollinformationen angewiesen ist, wird durch eine angepasste Version der Komponente realisiert. `Rechnung_EM2.2_Sub` wird in der folgenden Darstellung abgebildet.

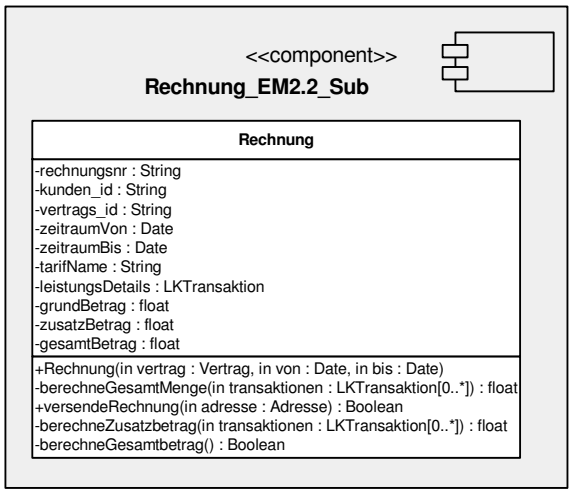


Abbildung 6-58: Komponente Rechnung_EM2.2_Sub

Die Attribute sowie die Methoden in dieser Version sind weitgehend identisch mit der Komponente `Rechnung_EM2.1_Sub`, jedoch weichen die Prozeduren des Konstruktors sowie der Methoden `berechneGesamtmenge` und `berechneZusatzbetrag` ein wenig ab. Die Berechnung der Gesamtmenge erfolgt auch hier auf Grundlage des Kontoauszugs der Klasse `LeistungsKonto`, so dass die dortige Methode `erstelleAuszug` verwendet wird und als Ergebnis das Listobjekt über `LKTransaktion` zurückgeliefert wird. Da hier jedoch nur die Anzahl der Produkte in Stück relevant ist, ist hier lediglich ein Zählen der Transaktionen notwendig. Die Berechnung des Zusatzbetrages erfolgt somit ebenfalls sehr viel einfacher, da die ermittelte Gesamtmenge nur gegen die vertraglich vereinbarte maximale Anzahl von Produkten verglichen werden muss und die überschreitende Menge mit dem vereinbarten Zusatzpreis je Einheit zu multiplizieren ist.

Auf eine detaillierte Darstellung des Ablaufs in Form eines Interaktionsdiagramms wird an dieser Stelle verzichtet, da das Zusammenspiel mit den anderen Klassen absolut identisch ist mit der vorherigen Variante. Das Sequenzdiagramm aus Abbildung 6-49 auf Seite 179 gilt somit auch hier weitestgehend und weicht nur innerhalb der Methode zur Berechnung des Zusatzbetrages ab.

6.3 Referenzarchitektur für das Erlösmodell Transaktionsgebühr

Dieses Erlösmodell TRANSAKTIONSGEBÜHR wird eingesetzt, wenn für die Bereitstellung einer Infrastruktur als Gegenleistung Gebühren erhoben werden und sich diese an einzelne Transaktionen richten. Die Transaktion kann dabei das Einstellen eines Angebotes oder die Durchführung eines Verkaufs sein (Verkäufer auf einem Markt) oder aber der Zugang zu einem Angebot oder die Durchführung eines Kaufs (Käufer auf einem Markt). Im folgenden werden die unterschiedlichen Varianten betrachtet.

6.3.1 Variante Verkäufer als Kostenträger

Es wird zunächst davon ausgegangen, dass der Verkäufer als Anbieter der Produkte sowohl für das Einstellen eines Angebotes als auch für den Verkauf seiner Waren Gebühren an den Betreiber zu entrichten hat.

Abbildung 6-59: Erlösmodellklassifikation Transaktionsgebühr Variante Verkäufer als Kostenträger

Der folgende Abschnitt befasst sich mit den Anforderungen an ein System zur Umsetzung des Erlösmodells und zeigt die betrachteten Systemgrenzen auf.

6.3.1.1 Anforderungsanalyse und Systemabgrenzung

Wie in der Klassifikation des Erlösmodells TRANSAKTIONSGEBÜHREN gezeigt wird, agieren die Nutzer als Teilnehmer in den Rollen Verkäufer und Käufer, indem sie ihre eigenen Produkte als Angebot bereitsstellen und auch erwerben. Somit sind eine Produktverwaltung, eine Nutzerverwaltung, eine Angebotsnutzung und Abrechnung die Szenarien, in die die Anwendungsfälle der Akteure unterteilt werden.

Das folgende Diagramm zeigt die Anwendungsfälle der Produktverwaltung.

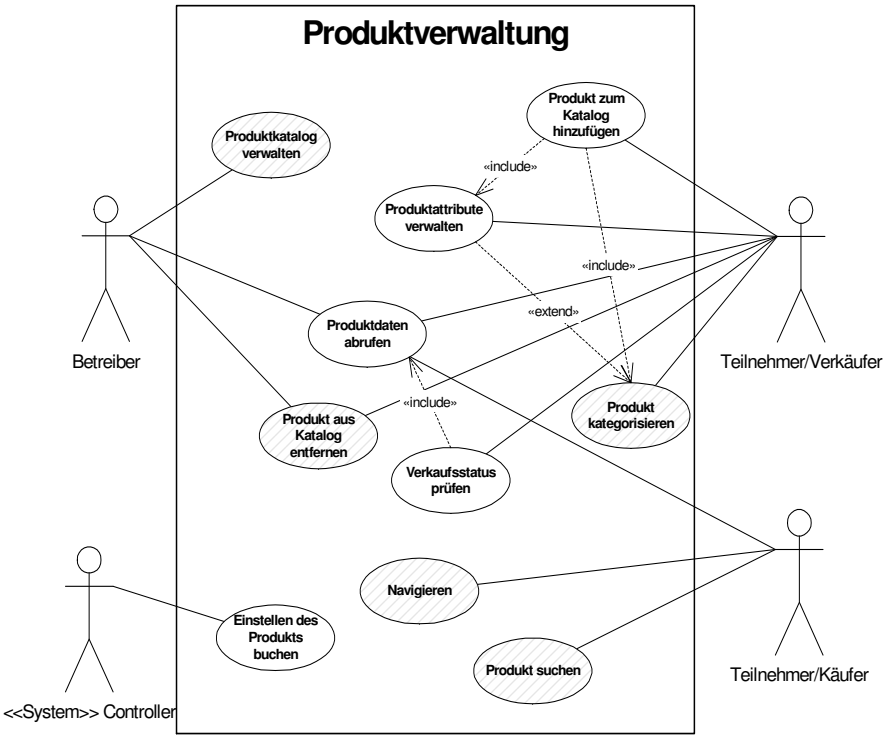


Abbildung 6-60: Anwendungsfälle der Produktverwaltung

Eine Besonderheit dieses Erlösmodells ist, dass die Produkte von den Teilnehmern (in der Rolle als Verkäufer) eingestellt und auch gekauft (in der Rolle als Käufer) werden. Somit beschränkt sich die Aufgabe des Betreibers zum einen auf die Verwaltung des Produktkatalogs, in dem er beispielsweise die Kategorien einrichten muss, in die die Teilnehmer ihre Produkte einordnen, und zum anderen auf die Kontrolle des Angebotes, wozu er die Produktdaten abrufen und überprüfen kann. Sofern ein Produkt nicht zum Verkauf angeboten werden soll, wird er es aus dem Katalog entfernen.

Inwiefern das System als automatisierter Controller bereits das Einstellen eines Produktes verbucht, hängt von der Art der Transaktionen ab, die für die Teilnehmer entgeltpflichtig sind. In dieser Variante übernimmt der Verkäufer die Kosten und es wird unterstellt, dass er sowohl für den Verkauf seiner Angebote als auch für das Einstellen der Produkte in den Produktkatalog eine Gebühr zu entrichten hat.

Der Kunde tritt wie bereits erwähnt in den Rollen als Verkäufer und als Käufer auf. Verkäufer fügen ihre Produkte zum Katalog hinzu, was immer eine Eingabe der erforderlichen Produktattribute (Produktattribute verwalten) und eine erste Kategorisierung des Produktes mit sich bringt. Um beispielsweise den aktuellen Verkaufsstatus seiner Produkte

zu prüfen, wird der Verkäufer über die entsprechenden Funktionalitäten die Produktdaten abrufen. Auch dem Verkäufer steht darüber hinaus die Option zur Verfügung Produkte aus dem Katalog zu entfernen, sofern sie von ihm eingestellt wurden.

Der Teilnehmer in der Rolle des Käufers wird sich über die bekannten Mechanismen der Produktsuche und Navigation einen Überblick über das Angebot verschaffen und bei Ineresse einzelne Produkte detaillierter betrachten, wozu er die Produktdaten abruft.

Die Anwendungsfälle zur Nutzerverwaltung sind in diesem Erlösmodell weitgehend identisch mit den bisher untersuchten Modellen. Eine Besonderheit stellt die Unterscheidung der Teilnehmer in Verkäufer und Käufer dar. Die folgende Abbildung 6-61 zeigt das Diagramm der Anwendungsfälle.

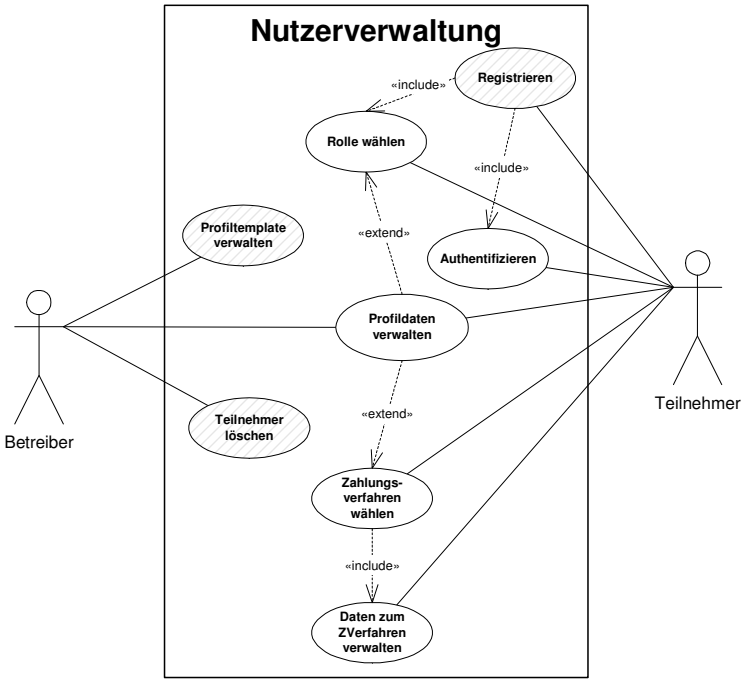


Abbildung 6-61: Anwendungsfälle der Nutzerverwaltung

Die Anwendungsfälle des Betreibers Profiltemplate verwalten, Profildaten verwalten und Teilnehmer löschen sind bereits bekannt. Ebenso die Interaktionen des Teilnehmers, der sich Registriert und Authentifiziert, seine Profildaten verwaltet und dabei ein Zahlungsverfahren wählt sowie die zugehörigen (Zahlungsverfahren-)Daten verwaltet. Lediglich der Anwendungsfall Rolle wählen zeigt an, dass sich ein Teilnehmer während der Registrierung entscheidet, in welcher Funktion er agieren wird. Dabei wird hier nicht festgelegt, dass er sich ausschließlich für eine Rolle entscheiden muss. Stattdessen soll es möglich sein, sowohl die Rolle des Verkäufers als auch die Rolle des Käufers einzunehmen.

Während der Angebotsnutzung ruft der Käufer die Produktdaten ab und entscheidet sich gegebenenfalls dafür, das Produkt zu kaufen. Der Verkäufer wird in diesem Zusammenhang lediglich eine Übersicht über den Status seiner Angebote anfordern, wozu er als Akteur des bereits aus der Produktverwaltung bekannten Anwendungsfall Verkaufsstatus prüfen auftritt. Der Anwendungsfall ist in Abbildung 6-62 entsprechend gekennzeichnet.

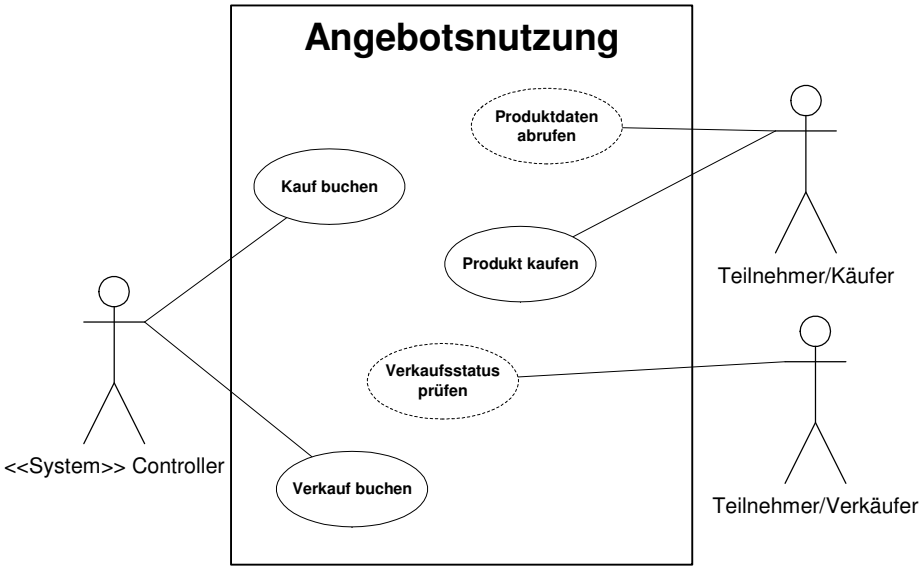


Abbildung 6-62: Anwendungsfälle der Angebotsnutzung

Der Controller bucht grundsätzlich sowohl den Kauf eines Produktes als auch den zugehörigen Verkauf, indem die Aktionen der jeweils betroffenen Teilnehmern protokolliert werden. Inwiefern diese Buchung in einer Abrechnung berücksichtigt wird, hängt wiederum von der gewählten Form der Transaktionsgebühren ab. Hier wird zunächst lediglich der Verkauf zu Gebühren für den Verkäufer führen.

Das abschließende Szenario der Abrechnung (Abbildung 6-63) verdeutlicht dies, indem nur der Verkäufer als Akteur auftritt und einen Kontoauszug abruft, um den aktuellen Stand zu überprüfen.

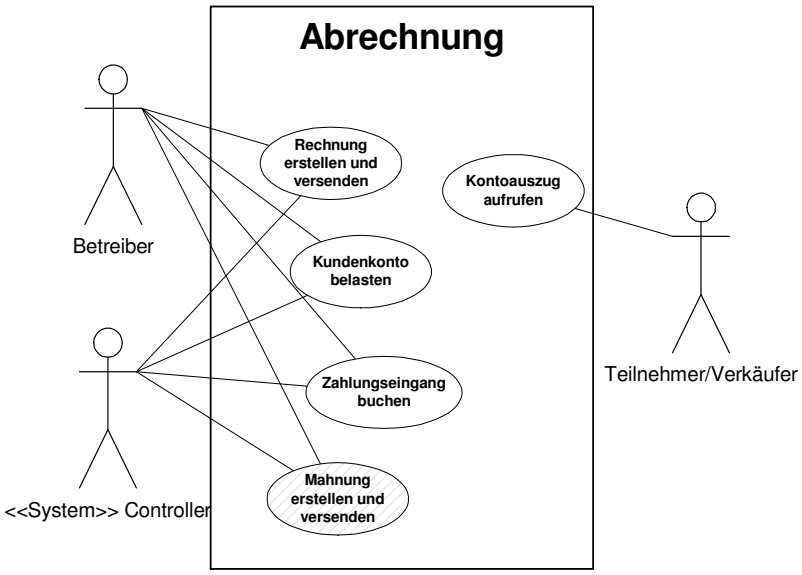


Abbildung 6-63: Anwendungsfälle der Abrechnung

Das Szenario ist bereits aus bisher betrachteten Erlösmodellen bekannt und wurde beispielsweise im Zusammenhang mit der SUBSKRIPTION in Abbildung 6-41 auf Seite 168 beschrieben[9].

Die Anforderungen dienen die Grundlage für die Architektur, die im folgenden Abschnitt beschrieben wird.

6.3.1.2 Architekturbeschreibung

Die folgende Abbildung stellt das Klassendiagramm für dieses Erlösmodell TRANSAKTIONSGEBÜHR dar, wobei zunächst unterstellt wird, dass der Verkäufer die Gebühren bezahlt. Inwiefern abweichende Varianten auch Änderungen der Architektur erfordern, wird anschließend in einem weiteren Abschnitt untersucht.

Die Besonderheiten dieses Erlösmodells sind vor allem die fremdbestimmte Herkunft des Angebotes sowie die Zahlung von Gebühren für unterschiedliche Leistungen. Die Komponente zur Produktverwaltung ist daraufhin angepasst und eine bisher nicht bekannte Komponente zur Buchung der Transaktionen ist hinzugefügt worden. Letztere bildet die Basis der Leistungsabrechnung. Alle Komponenten sind im Klassendiagramm in Abbildung 6-64 zu sehen.

9. Dort war der Verkäufer in der Rolle des Kunden, jedoch ist der Anwendungsfall identisch.

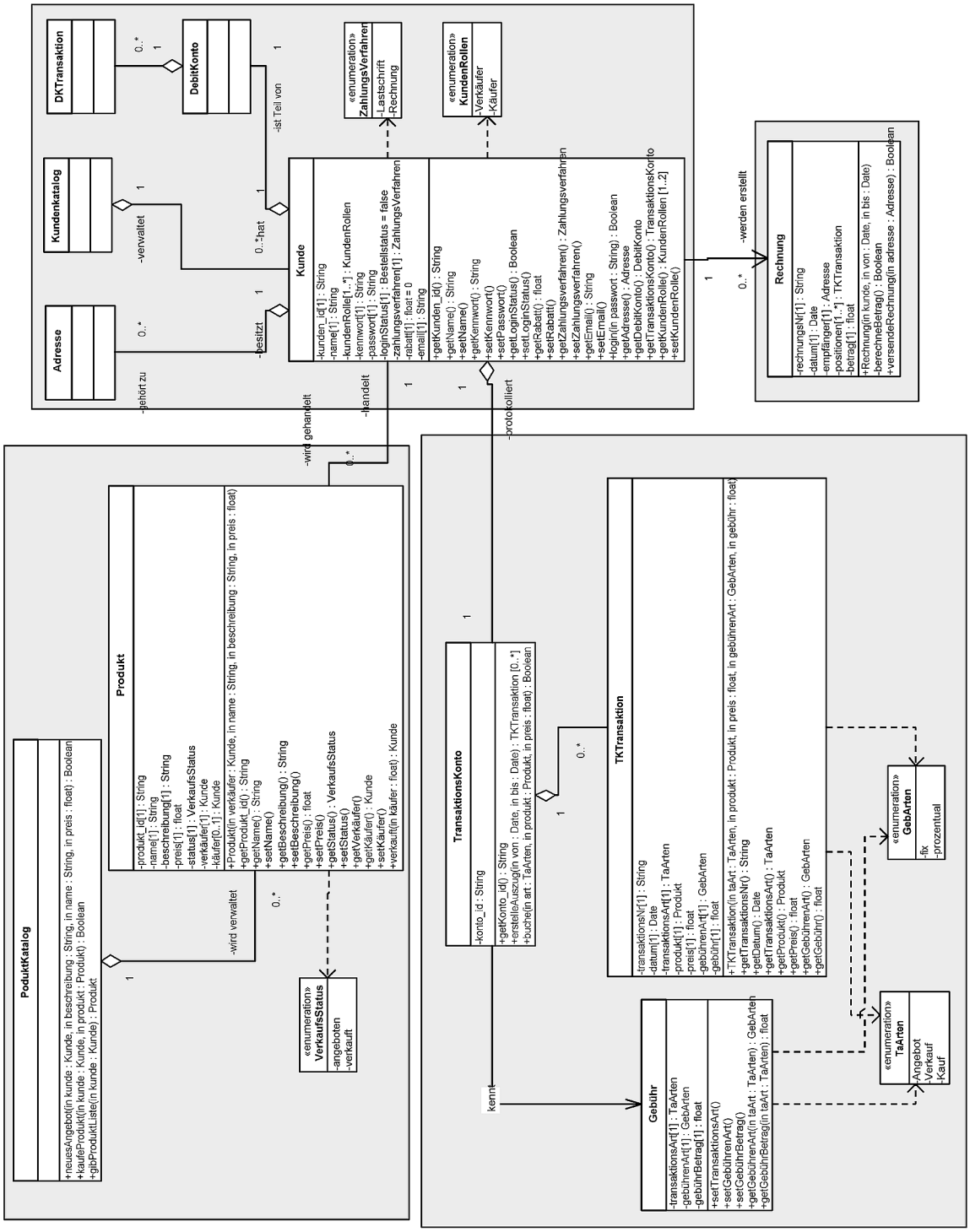


Abbildung 6-64: Statische Architektur des Erlösmodells Transaktionsgebühr

Das Diagramm enthält die Komponenten `Produkt`, `Kunde`, `Transaktionskonto` und `Rechnung` mitsamt ihren Klassen. Die dargestellten Assoziationen zwischen einigen Klassen verdeutlichen bereits die Besonderheiten des Erlösmodells: Zwischen `Kunde` und `Produkt` existiert eine *1:n* Beziehung, zwischen `Kunde` und `TransaktionsKonto` eine *1:1* Aggregation. Da die Kunden die Produkte selbstständig erzeugen und verwalten, sind diese ihnen auch personenbezogen zuzuweisen. Ebenso müssen die Transaktionen den jeweiligen Akteuren zugeordnet werden. Im Folgenden wird jede einzelne Komponente detailliert vorgestellt und das Zusammenspiel der Klassen mit Hilfe von Sequenzdiagrammen verdeutlicht.

Die Komponente zur Verwaltung der Teilnehmer[10] wird in der Version `Kunde_EM3.1_TG` verwendet. Sie enthält die Klassen `KundenKatalog`, `Kunde`, `Adresse`, `DebitKonto` und `DKTransaktion`. Die folgende Abbildung 6-65 zeigt die Komponente mit ihren Klassen.

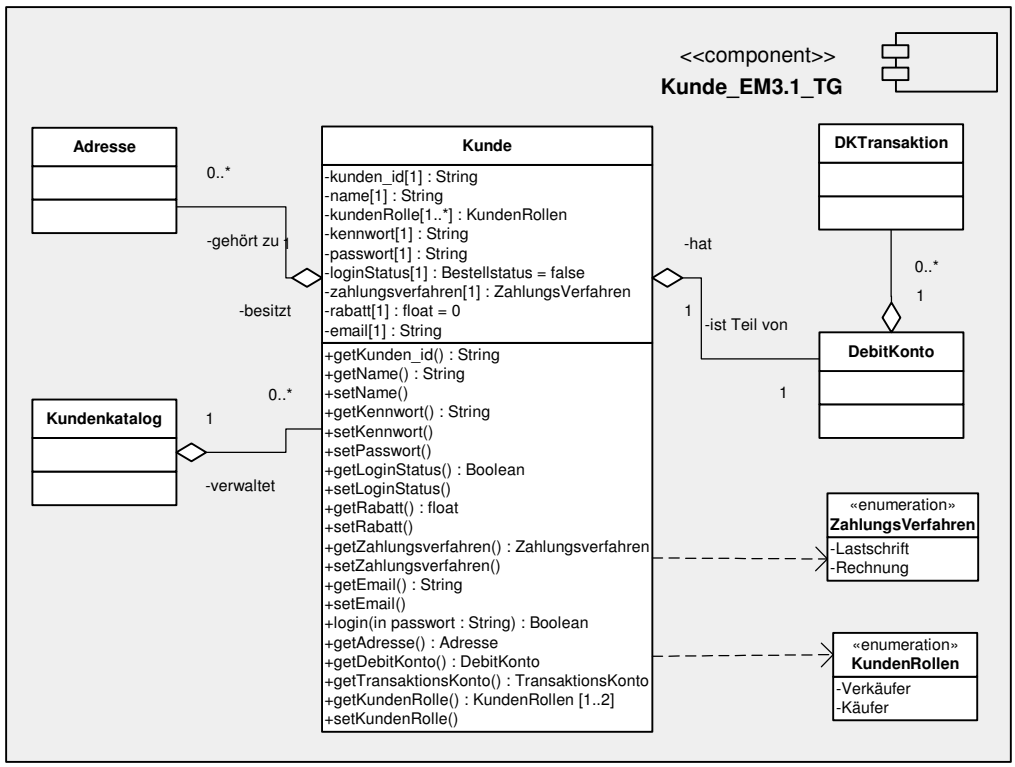


Abbildung 6-65: Komponente Kunde_EM3.1_TG

Bis auf `Kunde` sind die übrigen Klassen identisch mit den Klassen aus der bereits bekannten Version `Kunde_EM1.1_ET` und ihre Attribute und Methoden werden deshalb hier nicht erneut aufgeführt (zur Beschreibung siehe Abbildung 6-12 auf Seite 133). Auch der Kunde weist weitgehend die bekannten Attribute auf. Neu ist hier jedoch, dass direkt im

10. Die Begriffe *Teilnehmer* und *Kunde* werden hier synonym verwendet.

Kundenprofil das Zahlungsverfahren hinterlegt wird. Eine eigenständige Vertragskomponente ist nicht notwendig, da ein Teilnehmer bereits mit dem Anlegen seines Profils und dem Durchführen der Transaktionen die notwendigen Grundlagen für die Geschäftsbeziehung akzeptiert hat. Eine explizite Bestätigung der Geschäftsbedingungen kann über die Dialogsteuerung während der Registrierung erfolgen; jedoch wird hier darauf nicht näher eingegangen. Das Zahlungsverfahren, für das sich der Kunde entscheidet oder das ihm automatisch zugewiesen wird, ist also eine der wenigen rein *vertragsrelevanten* Informationen und wird hier direkt in die Kundenkomponente integriert. Als Zahlungsverfahren kann entweder Rechnung oder Lastschrift gewählt werden. Beide Verfahren erhalten jedoch keinerlei Aufschläge und sind somit für die Abwicklung der Rechnungsstellung und Bezahlung ohne Bedeutung für die Architektur, da sie zusätzlich auch außerhalb des Systems abgewickelt werden.

Eine bisher nicht behandelte Anforderung ist im Erlösmodell TRANSAKTIONSGEBÜHR die Unterstützung verschiedener Kundenrollen. Sowohl Käufer als auch Verkäufer müssen abbildbar sein. Die Umsetzung erfolgt mit Hilfe des Attributs `kundenRolle` (Klasse `Kunde`). Es ist vom Enumeration-Datentyp `KundenRollen` und kann die Ausprägungen *Verkäufer* und *Käufer* annehmen. Darüber hinaus handelt es sich um ein mengenwertiges Attribut, so dass einem Kundenobjekt beide Rollen gleichzeitig zugewiesen werden können[11]. Auf diese Weise kann ein Teilnehmer sowohl in der Rolle des Verkäufers als auch in der Rolle des Käufers agieren. Über die Unterscheidung anhand dieses Attributes ist es jedoch steuerbar, zusätzliche rollenspezifische Attribute oder Funktionalitäten anzubieten, auf die hier jedoch nicht eingegangen wird.

Die Methoden der Klasse `Kunde` sind die gewöhnlichen `get`- und `set`-Metdhoden sowie die Methoden zur Ermittlung des zugehörigen Debitoren- (`getDebitKonto`) und Transaktions-Kontos (`getTransaktionsKonto`).

Jedem Kundenobjekt ist genau ein Transaktions-Konto zugewiesen. Dieses protokolliert die Transaktionen des Kunden und stellt die Basis für die Abrechnung dar. Die Komponente `TransaktionsKonto_EM3.1_TG` stellt über die enthaltenen Klassen die notwendigen Dienste bereit. Abbildung 6-66 zeigt die Klassen mitsamt ihren Attributen und Methoden. Hervorzuheben ist hier, dass sowohl verschiedene Transaktionsarten unterstützt werden sollen als auch je nach Transaktionsart entweder fixe oder prozentuale Gebühren umzusetzen sein sollen.

11. Das mengenwertige Attribut `kundenRolle` sollte die Multiplizität *[1..2]* erhalten, jedoch ist diese Angabe der Obergrenze mit dem verwendeten Modellierungstool nicht möglich, so dass stattdessen *[1..*]* angegeben ist.

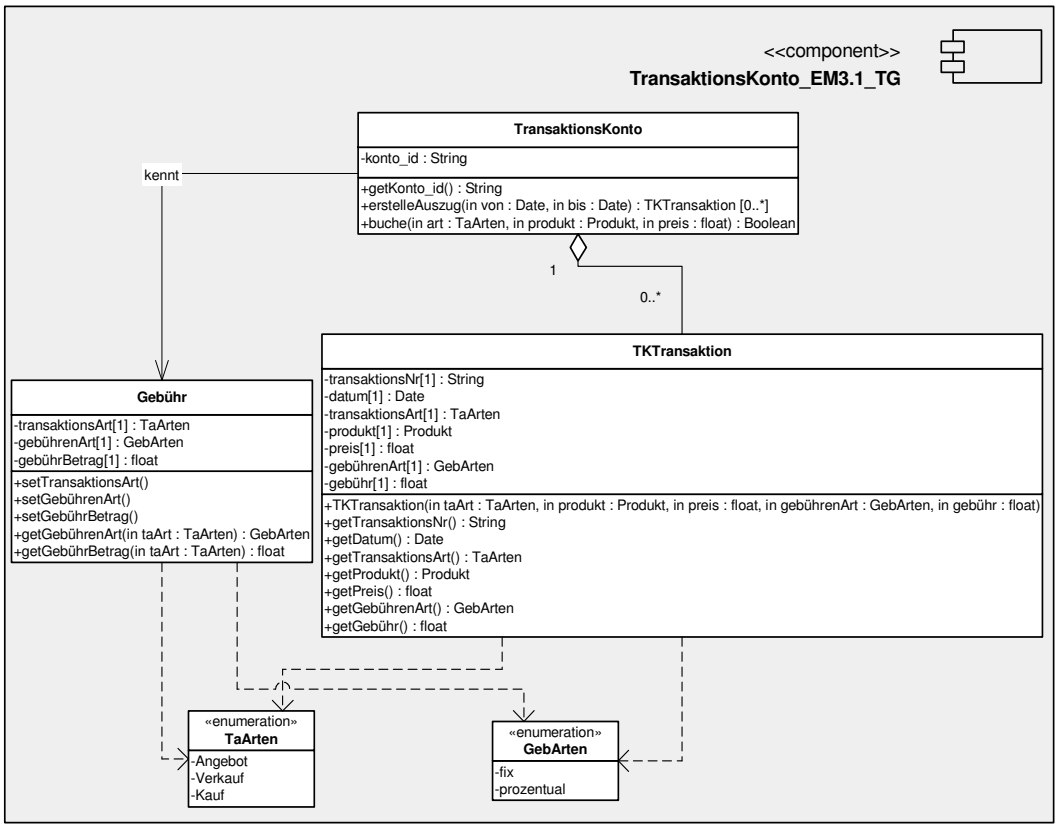


Abbildung 6-66: Komponente TransaktionsKonto_EM3.1_TG

Die Komponente enthält die bereits aus vorherigen Versionen bekannten Klassen `TransaktionsKonto` und `TKTransaktion` sowie zusätzlich die Klasse `Gebühr`. Die Komponente muss sicherstellen, dass die Anforderungen hinsichtlich der unterschiedlichen rechnungsrelevanten Buchungen der Transaktionsarten ermöglicht werden. In dieser Version sind sowohl die Einstellung eines neuen Angebotes als auch der Verkauf eines Produktes dem Verkäufer in Rechnung zu stellen. Darüber hinaus sollte die Architektur sowohl fixe als auch prozentuale Beträge unterstützen. So ist es vorstellbar, dass für die Aufnahme eines neuen Produktes eine fixe Gebühr erhoben wird, während sich die Transaktionsgebühr für einen erfolgreichen Verkauf prozentual vom erzielten Verkaufspreis berechnet.

Diese Anforderungen werden durch die Klasse `Gebühr` sowie ihr Zusammenspiel mit den beiden anderen Klassen realisiert. Die Klasse enthält die Attribute `transaktionsArt`, `gebührenArt` und `gebührBetrag`. Die `transaktionsArt` ist vom Enumeration-Datentyp `TaArten` und kann die Ausprägungen *Angebot*, *Verkauf* und *Kauf* annehmen. Somit sind für diese drei Transaktionen Gebühren zu hinterlegen. Zu jedem Typ kann zusätzlich festgelegt werden, ob es sich um eine *fixe* oder einen *prozentualen* Betrag handelt. Dies wird durch das Attribut `gebührenArt` des zugehörigen Enumeration-Datentypen realisiert. Der eigentliche (fixe oder prozentuale) Wert ist im letzten der drei Attribute hinterlegt.

Sofern eine Transaktion zu verbuchen ist, wird zunächst die Klasse `Transaktions-Konto` mit ihren Methoden aufgerufen. Die entscheidende Rolle übernimmt dabei die Methode `buche`. Sie wird aufgerufen mitsamt den Parametern *Transaktionsart*, *Produkt* und *Preis*. Zunächst muss die Methode ermitteln, welche Gebühren zu dieser Transaktionsart anfallen. Dazu bedient sie sich der Klasse `Gebühr` und ruft deren Methode `getGebührenArt` und `getGebührBetrag` auf, denen sie jeweils die Transaktionsart überliefert und die die zugehörigen Rückgabewerte liefern.

> Als Beispiel stellt ein Kunde (in der Rolle des Verkäufers) ein Buch zum Verkaufspreis von € 5,00 als ein neues Angebot ein. Die zu protokollierende Transaktion ist folglich vom Typ *Angebot*. In der Gebührenverwaltung (Klasse `Gebühr`) ist hinterlegt, dass für das Einstellen eines neuen Angebotes fix (Gebührenart) € 0,10 (Gebührbetrag) anfallen.
>
> Für den späteren Verkauf des Buches fallen dagegen 5 (Gebührbetrag) Prozent (Gebührenart) bezogen auf den Verkaufspreis an.
>
> Beide Transaktionen können mit der vorgestellten Komponente realisiert werden.

Anschließend kann die Methode eine neue Transaktion anlegen, indem sie eine neue Instanz der Klasse `TKTransaktion` erzeugt. Dem gleichnamigen Konstruktor werden nun die Parameter *Transaktionsart*, *Produkt*, *Preis*, *Gebührenart* und *Betrag* übermittelt. Dieser kann daraufhin dem Objekt mit einem Zeitstempel versehen und den übrigen Attributen die jeweiligen Werten zuweisen. Dabei berücksichtigt die Methode, ob es sich um eine prozentuale oder fixe Gebührenart handelt. sofern es sich um einen fixen Wert handelt, wird dem Attribut `gebühr` einfach der überlieferte Parameter zugewiesen. Falls es sich jedoch um eine prozentualen Wert handelt, berechnet die Methode die Gebühr mittels der Formel *Preis x Betrag/100*, wobei *Preis* und *Betrag* die überlieferten Paramter sind.

Der Ablauf der Buchung und das Zusammenspiel der beteiligten Klassen ist als Sequenzdiagramm in der folgenden Abbildung 6-67 zu sehen.

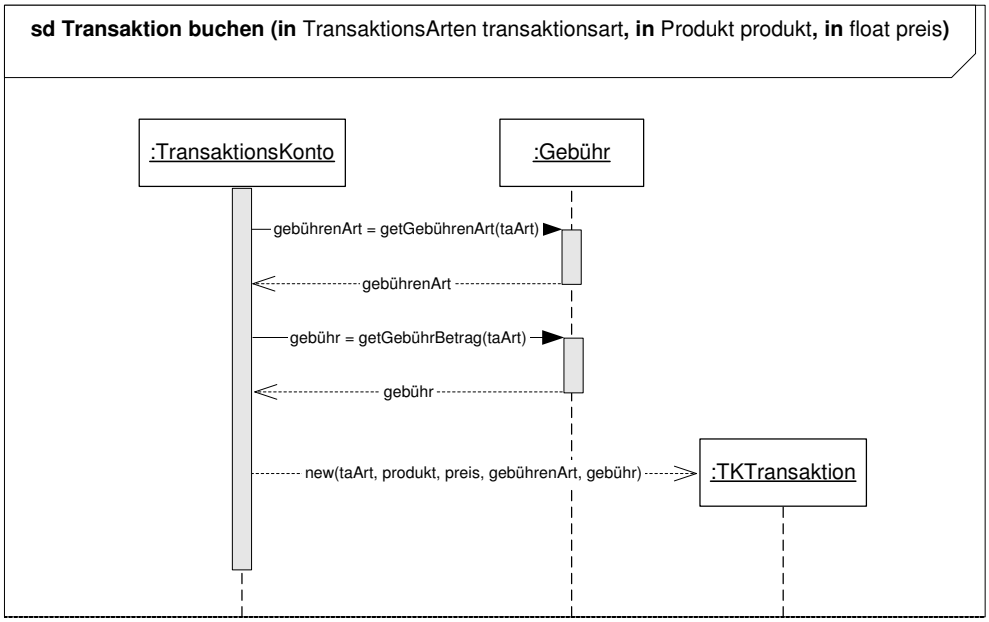


Abbildung 6-67: Sequenzdiagramm zur Verbuchung der Transaktionen

Die Methode `erstelleAuszug` der Klasse `TransaktionsKonto` ist bereits aus den bisher betrachteten Versionen bekannt und liefert als Rückgabewert ein Listobjekt, dass alle Transaktionen innerhalb des überlieferten Zeitraums (Parameter *von*, *bis*) aufführt. Es wird im Rahmen der Rechnungsstellung verwendet, auf die weiter unten in diesem Abschnitt eingegangen wird.

Die Komponente zur Verwaltung der Produkte wird in dieser Variante in der Version `Produkt_EM3.1_TG` eingesetzt. Sie enthält die beiden Klassen `ProduktKatalog` und `Produkt`, die jedoch einige Besonderheiten im Vergleich zu bisher betrachteten Versionen aufweisen. Diese ergeben sich, wie bereits erwähnt, aus der Anforderung, die Produkte den Kunden eindeutig zuweisen zu können. Daraus folgt auch, dass die Kunden über Möglichkeiten verfügen sollen, Informationen zu *ihren* Produkten zu erhalten. Für Verkäufer ist es beispielsweise wichtig zu erkennen, ob ein Angebot noch gültig ist oder ob das Produkt bereits verkauft wurde; Käufer sollten eine Übersicht über die von ihnen erworbenen Produkte erhalten können. Abbildung 6-68 zeigt die Methoden und Attribute der Klassen.

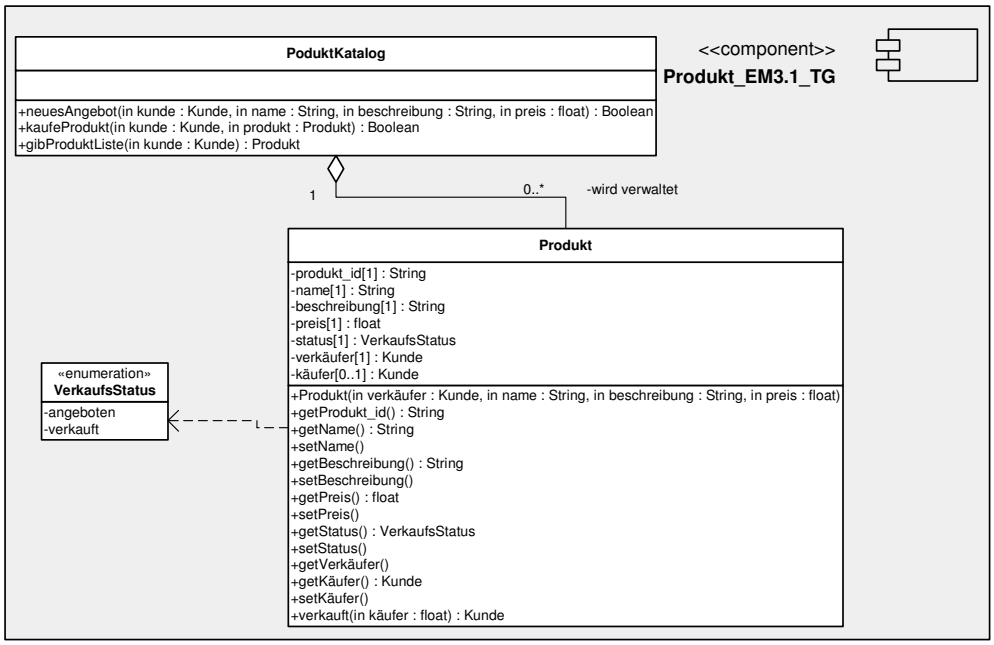


Abbildung 6-68: Komponente Produkt_EM3.1_TG mitsamt den Klassen

Das Produkt enthält zunächst die bereits bekannten Attribute `produkt_id`, `name` und `beschreibung`. Der `Preis` wird in dieser Version vereinfachend nicht als eigene Klasse realisiert (vgl. etwa Komponente `Produkt_EM1.1_ET` auf Seite 136), sondern lediglich als atomares Attribut. Der `Status` des Produktes gibt Auskunft darüber, ob ein Produkt (noch) *angeboten* wird oder bereits *verkauft* wurde. Dazu steht der Enumeration-Datentyp VerkaufsStatus zur Verfügung. Mit Hilfe dieses Attributes wird die Anforderung aus dem vorherigen Abschnitt umgesetzt, dass sich ein Verkäufer über den aktuellen Verkaufsstatus seiner Produkte informieren möchte. Die zugehörige Funktion stellt die Klasse `Produkt-Katalog` bereit und wird weiter unten vorgestellt. Die letzten beiden Attribute sind `Verkäufer` und `Käufer`. Die Multiplizität der Attribute zeigt, dass jedes Produkt immer genau einem Verkäufer (vom Typ `Kunde`) zugewiesen wird. Dies geschieht zum Zeitpunkt der Produktaufnahme; also genau dann, wenn ein Verkäufer ein neues Produkt zum Produktkatalog hinzufügt und es dadurch zum Verkauf anbietet. Zu diesem Zeitpunkt ist das Produkt jedoch noch nicht verkauft und dem Attribut `Käufer` kann kein Wert zugewiesen werden. Dieser wird somit erst gesetzt, wenn ein Teilnehmer das Produkt erworben hat. Gleichzeitig wird in diesem Fall der Status von *angeboten* auf *verkauft* gesetzt.

Neben den üblichen `get`- und `set`-Methoden sind der Konstruktor `Produkt` sowie die Methode `verkauft` relevant. Der Konstruktor erhält eine Liste von Parametern übergeben. Er wird von der Methode `neuesAngebot` der Klasse `ProduktKatalog` aufgerufen und erhält den Verkäufer als Referenz auf ein Kundenobjekt sowie den Namen, die Beschreibung und den Preis des Produktes überliefert. Den Verkäufer weist er somit initial dem entsprechenden Attribut zu, weshalb auf eine Methode `setVerkäufer` verzichtet wurde. Der `status` wird auf *angeboten* gesetzt, der `Käufer` wird nicht zugewiesen und die übrigen Attri-

bute entsprechend der überlieferten Parameter belegt. Des weiteren sorgt der Konstruktor dafür, dass die Buchung der Transaktion auf dem Transaktionskonto des Verkäufers erfolgt. Eine Beschreibung des Ablaufs ist im Interaktionsdiagramm in Abbildung 6-69 zu sehen.

Die Methode `verkauft` wird ebenfalls von der Klasse `ProduktKatalog` aufgerufen (Methode `verkaufeProdukt`) und erhält als Parameter den Käufer als Referenz auf das Kundenobjekt überliefert. Zunächst ändert die Methode den Status des Produktes auf *verkauft* und setzt den Attributwert des Käufers. Die weiteren Aktivitäten werden später im Zusammenhang mit dem Sequenzdiagramm in Abbildung 6-70 auf Seite 203 vorgestellt.

Durch die Attribute `verkäufer` und `käufer` sowie durch den `status` können somit Reports erstellt werden, die den Kunden je nach Rolle die jeweils relevanten Informationen über ihre Produkte ausweisen. Dazu stellt auch die letzte vorzustellende Klasse dieser Komponente, der `ProduktKatalog`, einen Dienst bereit.

Die Klasse `ProduktKatalog` verfügt über die drei Methoden `neuesAngebot`, `kaufeProdukt` und `gibProduktListe`. Die Methode `gibProduktListe` implementiert einen Filter, der über alle Produkte diejenigen heraussucht, die den als Parameter überlieferten Kunden entweder als Verkäufer oder als Käufer aufweisen. Es wird hier nicht näher betrachtet, inwiefern die Aufbereitung der Liste erfolgen kann, jedoch steht mit dieser Methode eine Funktion bereit, die die Anforderung realisiert, einem Verkäufer den Verkaufsstatus seiner aktuellen Produkte mitzuteilen.

`neuesAngebot` ist eine Methode, die von der GUI aufgerufen wird, nachdem dort über ein Formular die wesentlichen Produktinformationen Name, Beschreibung und Preis ermittelt wurden. Somit werden diese Daten als Parameter überliefert. Zusätzlich wird eine Referenz auf den aktuellen[12] Kunden mitgeliefert, der als Verkäufer das Angebot hinzufügt. Die Methode ruft ihrerseits den Konstruktor `Produkt` auf und reicht dabei die Parameter weiter, um einen neues Produktobjekt zu erzeugen.

Da mit der Neueinstellung eines Angebotes eine Transaktionsgebühr anfällt und diese zu verbuchen ist, wird dieser Ablauf und die Interaktionen der beteiligten Klassen im Sequenzdiagramm in Abbildung 6-69 detailliert dargestellt.

12. *aktuell* bedeutet hier, dass dieser Kunde derzeit authentifiziert ist und die Daten über die GUI eingegeben hat, bevor die Methode aufgerufen wurde.

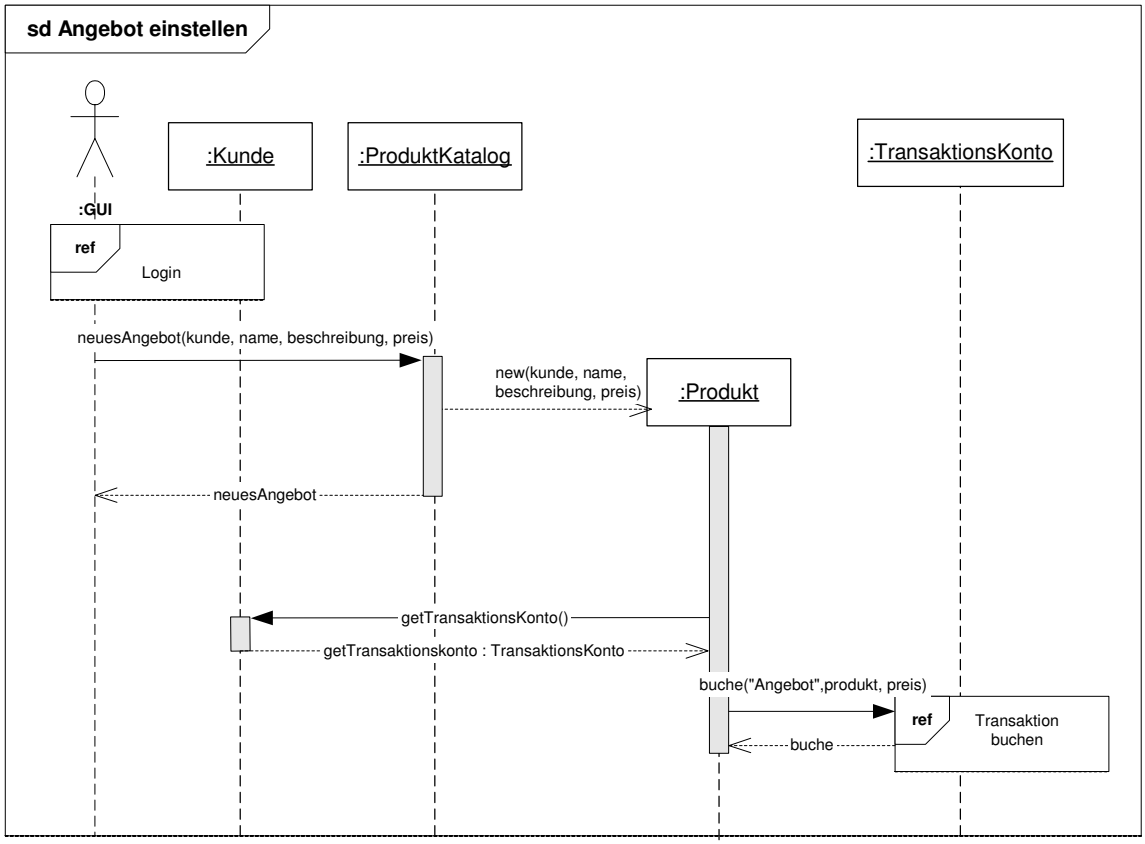


Abbildung 6-69: Sequenzdiagramm zum Einstellen eines neuen Produktes

Nachdem der authentifizierte Kunde (hier: Verkäufer) über die GUI die relevanten Daten eingegeben hat, wird die Methode `neuesAngebot` aufgerufen, die ihrerseits eine neue Instanz der Klasse `Produkt` erzeugt und die Parameter weiterreicht. Der Konstruktor `Produkt` ermittelt nach dem Setzen der Attributwerte (Beschreibung siehe oben) über die Klasse `Kunde` das `TransaktionsKonto` des Kunden (`getTransaktionsKonto` liefert eine Referenz auf die Klasse zurück) und sorgt für das Buchen der Angebotseinstellung. Dazu ruft er die Methode `buche` des Transaktionskontos auf und übergibt die relevanten Parameter, wobei in diesem Fall *Angebot* als fester Wert überliefert wird, da es sich um eine Neuaufnahme eines Produktes in den Katalog handelt. Das Buchen ist als Interaktionsreferenz (*Transaktion buchen*) modelliert worden und wird in Abbildung 6-67 auf Seite 199 dargestellt.

Die zweite erlösrelevante Transaktion in dieser Variante ist der Verkauf eines Produktes. Auch dieser Ablauf wird detailliert beschrieben: Die Methode `kaufeProdukt` des Produktkatalogs wird aufgerufen, wenn ein Käufer ein Produkt erworben hat. Als Parameter werden der Methode der Käufer als Referenz auf das Kundenobjekt sowie das erworbene Produkt (ebenfalls als Referenz) übergeben. Die Methode ruft daraufhin die Methode `verkauft` des Produktobjektes auf und übergibt ihr den Käufer. Das Interaktionsdiagramm in Abbildung 6-70 zeigt das Zusammenspiel.

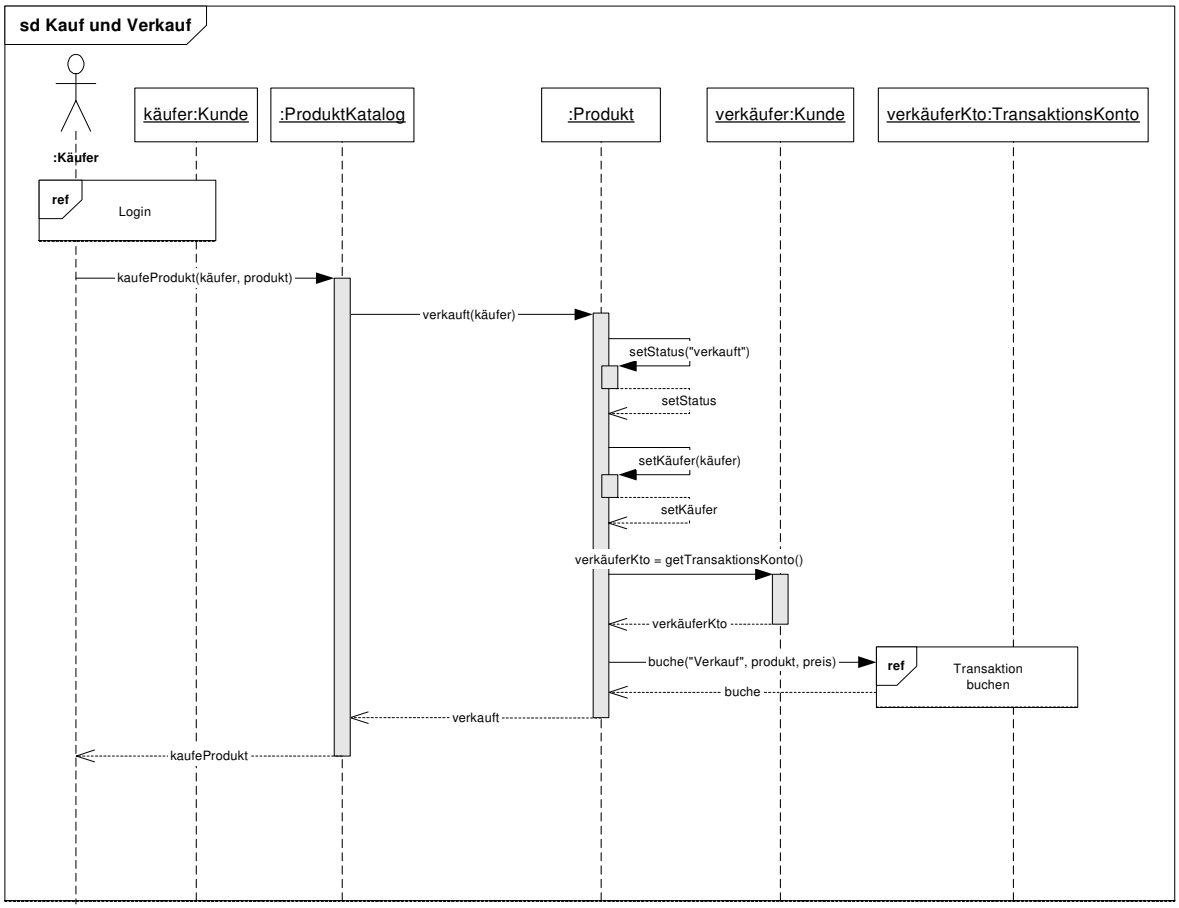


Abbildung 6-70: Sequenzdiagramm zur Durchführung eines Verkaufs/Kaufs eines Produktes

Nachdem der `ProduktKatalog` die Methode `verkauft` aufgerufen hat, werden dort zunächst der Verkaufsstatus angepasst und der Käufer dem gleichnamigen Attribut zugewiesen. Da der Verkäufer für den Verkauf eine Gebühr zu entrichten hat, muss die Aktion auf seinem Transaktions-Konto verbucht werden. Der Verkäufer ist im Produktobjekt hinterlegt (gleichnamige Attribut enthält Referenz auf den entsprechenden Kunden). Mit dieser Information ermittelt die Methode zunächst das Transaktions-Konto des Verkäufers (`getTransaktionsKonto`) und ruft anschließend die Methode `buche` des Kontos auf, wobei der feste Wert *Verkauf* als Transaktionsart überliefert wird[13].

Die Anforderungen an die Rechnungsstellung werden duch die Komponente `Rechnung_EM3.1_TG` abgedeckt. Die Rechnung muss auf Basis der protokollierten Transaktionen die Summe der einzelnen Gebühren ermitteln und diese zusammen mit

13. Die Interaktionsreferenz ist in Abbildung 6-67 auf Seite 199 zu sehen.

einem Einzelverbindungsnachweis aufbereitet ausweisen. Der Betrag muss auf dem Debitorenkonto des Kunden (in dieser Variante nur für die Verkäufer) verbucht werden. Die Komponente ist in der folgenden Abbildung dargestellt.

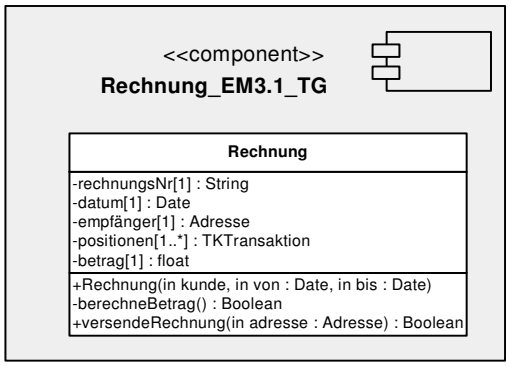


Abbildung 6-71: Komponente Rechnung_EM3.1_TG

Die Klasse `Rechnung` enthält die bereits bekannten Attribute und Methoden, jedoch muss deren Funktionsweise an die gegebenen Umstände angepasst werden. Das folgende Sequenzdiagramm (Abbildung 6-72) zeigt diesen Ablauf im Zusammenhang mit den beteiligten Klassen.

Der Controller repräsentiert eine Systemkomponente, die hier nicht näher betrachtet wird und deren Aufgabe es ist, fortlaufend zu überprüfen, ob für einen Kunden eine Rechnung zu erstellen ist. Sofern dies der Fall ist, ruft sie den Konstruktor der Klasse `Rechnung` auf und übergibt neben dem relevanten Rechnungszeitraum eine Referenz auf den betroffenen Kunden. In der hier betrachteten Variante handelt es sich dabei ausschließlich um Kunden in der Rolle als Verkäufer.

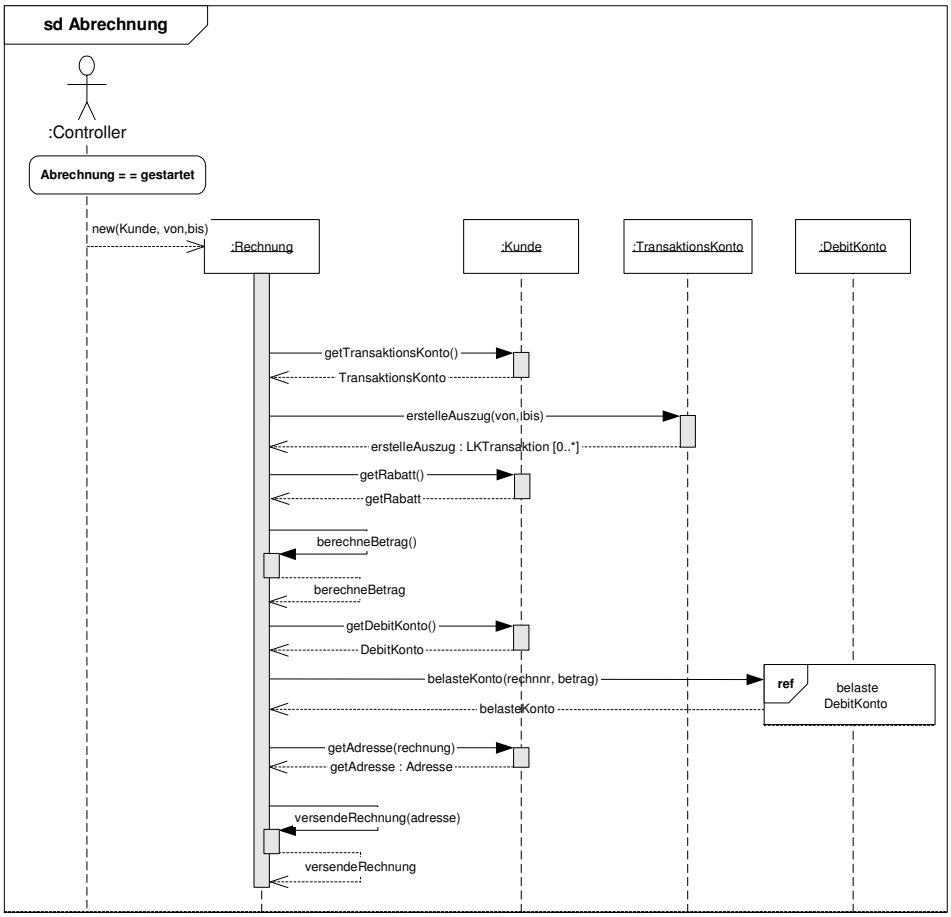


Abbildung 6-72: Die Abrechnung als Sequenzdiagramm

Der Kontsruktor ermittelt über den Kunden zunächst dessen Transaktions-Konto und fordert einen Auszug über den relevanten Zeitraum an (`erstelleAuszug`). Die Berechnung des Betrages (`berechneBetrag`) erfolgt durch Aufsummierung der einzelnen Transaktionsbeträge, wobei zusätzlich ein Kundenrabatt (`getRabatt`) berücksichtigt werden kann, der beispielsweise Großkunden eingeräumt wird. Abschließend wird der Rechnungsbetrag auf dem Debitoren-Konto des Kunden verbucht (`belasteKonto`) und die Rechnung in physikalischer Form versendet. Die Interaktionsreferenz zur Kontobelastung ist in Abbildung 6-23 auf Seite 146 dargestellt.

6.3.2 Variante Käufer und Verkäufer als Kostenträger

In dieser Variante bezahlt nicht nur der Verkäufer die Gebühren, sondern auch der Käufer. Eine Gebühr für den Käufer fällt an, sobald er ein Produkt erwirbt. Das Erlösmodell wurde in Abschnitt 4.2.3.2 auf Seite 76 beschrieben und wird hier noch einmal dargestellt:

Abbildung 6-73: Erlösmodellklassifikation Transaktionsgebühr Variante Verkäufer und Käufer als Kostenträger

6.3.2.1 Anforderungsanalyse und Systemabgrenzung

Die in der vorherigen Variante (siehe Abschnitt 6.3.1.1 auf Seite 189) beschriebenen Anforderungen gelten weitestgehend auch hier. Die Anwendungsfälle zur Produktverwaltung können uneingeschränkt übernommen werden. Im Bereich der Nutzerverwaltung ist die Verwaltung der Verkäufer nun auch auf die Käufer anzugleichen. Da in dieser Variante des Erlösmodells zusätzlich der Käufer Gebühren zu bezahlen hat, muss aus Sicht des Betreibers die Konfiguration des Systems dahingehend geändert werden, dass auch ein Käufer die notwendigen Angaben wie beispielsweise Kontoverbindungen in seinem Profil hinterlegen muss. Die Anwendungsfälle sind jedoch identisch mit denen für die Verwaltung der Verkäufer, so dass sie hier nicht erneut aufgeführt werden. Die Anwendungsfälle innerhalb der Angebotsnutzung können ebenso wie in der Produktverwaltung unverändert übernommen werden. Der Bereich der Abrechnung enthält aus Sicht des Betreibers ebenso die gleichen Anwendungsfälle, jedoch sind nun auch für den Käufer Rechnungen zu erstellen und zu bearbeiten. Die Interaktion des Käufers mit dem System erweitert sich in diesem Bereich lediglich um den Zugang zu seinem Konto und die Erstellung von Kontoauszügen. Der Anwendungsfall ist ebenso identisch mit dem des Verkäufers, der bereits beschrieben wurde.

6.3.2.2 Architekturbeschreibung

Die in der vorherigen Variante beschriebenen Klassen können auch für diese Form des Erlösmodells eingesetzt werden. Voraussetzung für diese Wiederverwendung ist in erster Linie die Berücksichtigung unterschiedlicher Transaktionsarten mit zugehörigen Transaktionsgebühren innerhalb der Komponente `TransaktionsKonto`. Neben Angebotseinstellungen und Produktverkäufen sind auch Produktkäufe als Transaktionsart vorgesehen und problemlos umzusetzen. Darüber hinaus wurde das Kundenprofil bereits derart angelegt, dass sowohl Verkäufer als auch Käufer unterschieden werden können und separat abzurechnen sind. Lediglich die Methode `verkauft` der Klasse `Produkt` ist auf die veränderten Anforderungen hin anzupassen. Das folgende Sequenzdiagramm zeigt deutlich, inwiefern diese Anpassungen vorzunehmen sind.

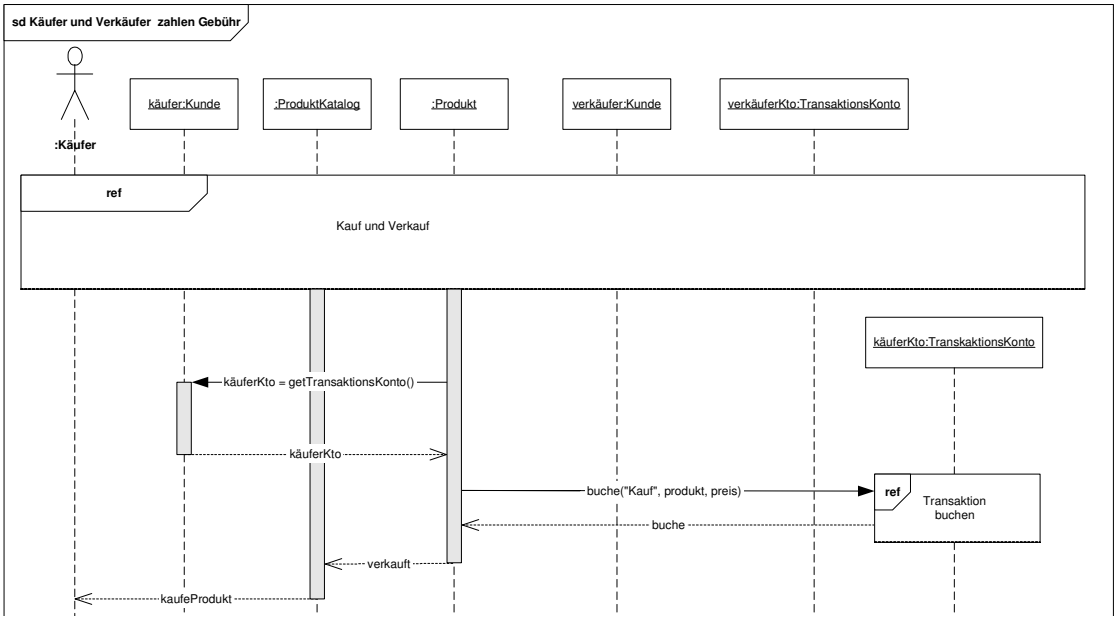


Abbildung 6-74: Angepasstes Sequenzdiagramm zum Verkauf/Kauf mit Gebührenbuchung

Zunächst wird als Interaktionsreferenz auf das Sequenzdiagramm *Kauf und Verkauf* der vorherigen Variante verwiesen. Dieses ist in Abbildung 6-70 auf Seite 203 zu sehen. Dort wurde bei einem Verkauf eines Produktes diese Aktion auf dem Transaktions-Konto des Verkäufers verbucht. Diese Aufgabe übernimmt dort die Methode `verkauft` der Klasse `Produkt`.

Der Ablauf kann unverändert übernommen werden[14] und muss in der vorliegenden Variante lediglich ergänzt werden. Nach dem Verbuchen der Verkaufstransaktion muss die Methode `verkauft` hier zusätzlich den Vorgang des Kaufens auf dem Konto des Käufers protokollieren. Dazu ermittelt sie zunächst dessen `TransaktionsKonto` (`getTransaktionsKonto`) und ruft dort anschließend die Methode `buche` auf. Als Transaktionsart wird der feste Wert *Kauf* übermittelt, so dass die zugehörige Gebühr ermittelt und festgehalten werden kann.

6.3.3 Variante Käufer als Kostenträger

Die letzte Veriante ist aus aus Sicht der Systemarchitektur identisch mit der Variante, in der der Verkäufer die Gebühren bezahlt und sie ist somit bereits beschrieben worden. Die Rechnungsstellung erfolgt hier jedoch lediglich für den Käufer. Aufgrund dieser nur marginalen Abweichung wird hier auf eine weitere Beschreibung dieser Variante verzichtet.

14. In der angesprochenen Interaktionsreferenz ist nach der Durchführung der Buchung auf dem Transaktions-Konto des Verkäufers die Aktionssequenz der Methode `verkauft` beendet und es erfolgt ein Rücksprung auf die Lebenslinie des Produktkatalogs, der das Ende der Methode `verkauft` signalisiert. Dieser Rücksprung darf in der vorliegenden Version nicht erfolgen, da die Methode noch nicht beendet wurde, sondern weitere Aktionen durchführt, wie die verlängerte Aktionssequenz in dem hier verwendeten Diagramm zeigt. Der Rücksprung erfolgt somit erst (wie dargestellt) nach dieser verlängerten Aktionssequenz. Um dennoch das vorhandene Diagramm wiederzuverwenden und nicht redundant abzubilden, wird diese Inkorrektheit in Kauf genommen.

6.4 Referenzarchitektur für das Erlösmodell Profilhandel

Dieses Erlösmodell richtet sich nicht an die Anwender des Internetauftritts, sondern an andere Unternehmen. Diese bezahlen dem Betreiber des Systems Geld für Anwenderprofile, die möglichst umfassende Informationen über die Kunden erhalten. Aus Sicht der Architekturbeschreibung wird unterschieden zwischen identifizierten und anonymen Anwendern.

6.4.1 Variante Identifizierter Kunde

Dieses Erlösmodell wurde in Abschnitt 4.2.4.1 auf Seite 79 hergeleitet und beschrieben. Die Kriterienausprägungen zeigt die folgende Grafik.

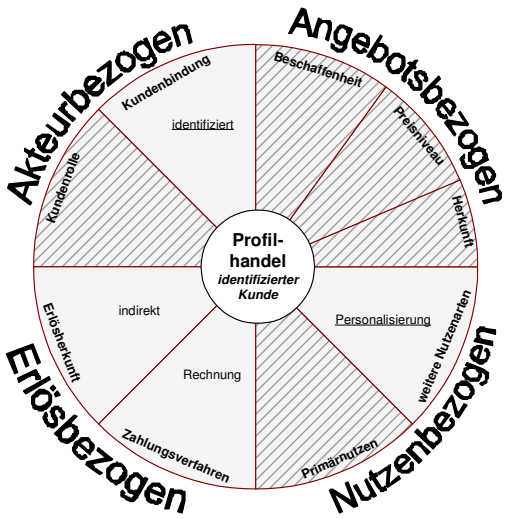


Abbildung 6-75: Erlösmodellklassifikation Profilhandel Variante identifizierter Kunde

Die schraffierten Kriterien spielen für diese Variante keine Rolle und wurden somit nicht berücksichtigt. Der folgende Abschnitt betrachtet zunächst wieder die Anforderungen, bevor darauf aufbauend die Softwarearchitektur hergeleitet und vorgestellt wird.

6.4.1.1 Anforderungsanalyse und Systemabgrenzung

Dieses Erlösmodell basiert auf der Generierung und Auswertung von Kundenprofilen. Das Angebot selber ist dagegen aus Architektursicht eher sekundär und beeinflusst lediglich die Art der gewonnenen Informationen sowie die Art der Informationsgewinnung. Werden beispielsweise Produkte verkauft, sind die Interessen des Kunden anhand der erworbenen Produkte einfach zu klassifizieren und anhand der Kauftransaktion einfach zu erheben. Bietet ein Unternehmen lediglich Informationen für den Nutzer an, so müssen die Interessen anhand der betrachteten Informationen ermittelt werden, wodurch auch die Informationserhebung eher implizit und damit aufwendiger durchzuführen ist.

Unabhängig von der konkreten Art des Angebotes werden in dieser Variante Profile von identifizierten Nutzern erhoben. Somit ergeben sich Anforderungen an eine Nutzerverwaltung. Darüber hinaus werden die Anwendungsfälle im Rahmen des Profiling explizit betrachtet, wobei dieses Szenario eng verbunden ist mit der Nutzerverwaltung, da dort bereits die Nutzerprofile der Kunden verwaltet werden. Darüber hinaus wird das Szenario der Angebotsnutzung hinsichtlich der Anwendungsfälle betrachtet, die dazu verwendet werden können, das Profiling durchzuführen.

Die Nutzerverwaltung mit den zugehörigen Anwendungsfällen ist im folgenden Diagramm ersichtlich.

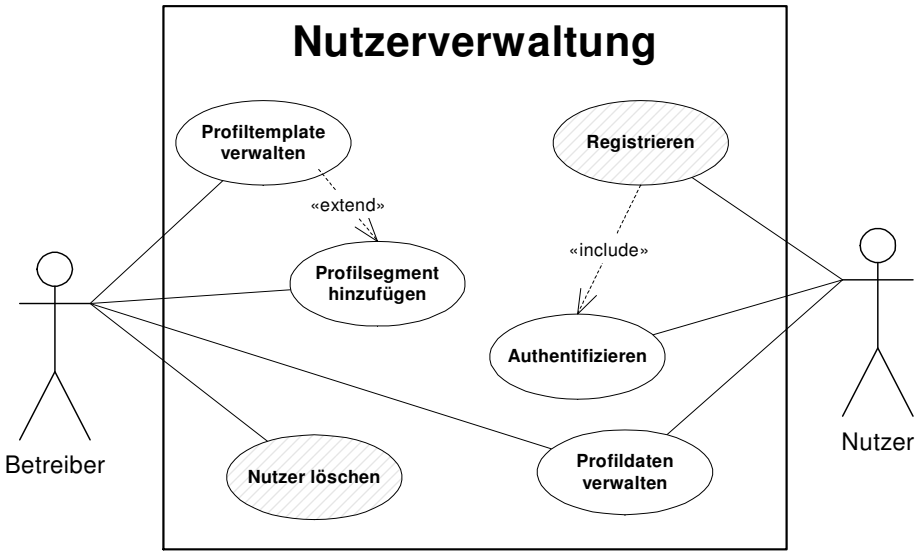


Abbildung 6-76: Die Anwendungsfälle zur Nutzerverwaltung

Die zentrale Aufgabe des Betreibers besteht darin, die Profiltemplates des Kunden so zu gestalten, dass die darin enthaltenen Informationen[15] ausreichen, möglichst umfangreiche und detaillierte Kenntnisse über die Nutzer zu erheben. Das Verwalten der Profiltemplates kann auch dazu führen, völlig neue, zusätzliche Profilsegmente hinzuzufügen. Anders als in den bisherigen Nutzerverwaltungen können sich die Profile der Kunden in diesem Erlösmodell häufiger ändern, sofern sie veränderten Marktsituationen angepasst werden müssen. In diesem Fall können Profilsegmente, die eine Reihe von Attributen zu einem speziellen Themengebiet umfassen, neu hinzugefügt werden.

Die wesentliche Aufgabe des Nutzers im Rahmen dieses Szenarios besteht darin, seine Profildaten zu verwalten. Eine Registration vorausgesetzt, die ihm ein persistentes Nutzerprofil zuweist, gibt er je nach Gestaltung des Angebots explizit Informationen über sich preis und hinterlegt sie in seinem Profil. Ein Beweggrund dafür kann beispielsweise die

15. Die Informationen werden strenggenommen nicht im Template enthalten sein, sondern in der konkreten Instanz eines solchen Templates.

Konfiguration der GUI gemäß seiner Interessen sein, wozu er diese dem System mitteilen wird. Auf diesen Anwendungsfall wird im nächsten Szenario der Angebotsnutzung eingegangen, das in Abbildung 6-77 zu sehen ist.

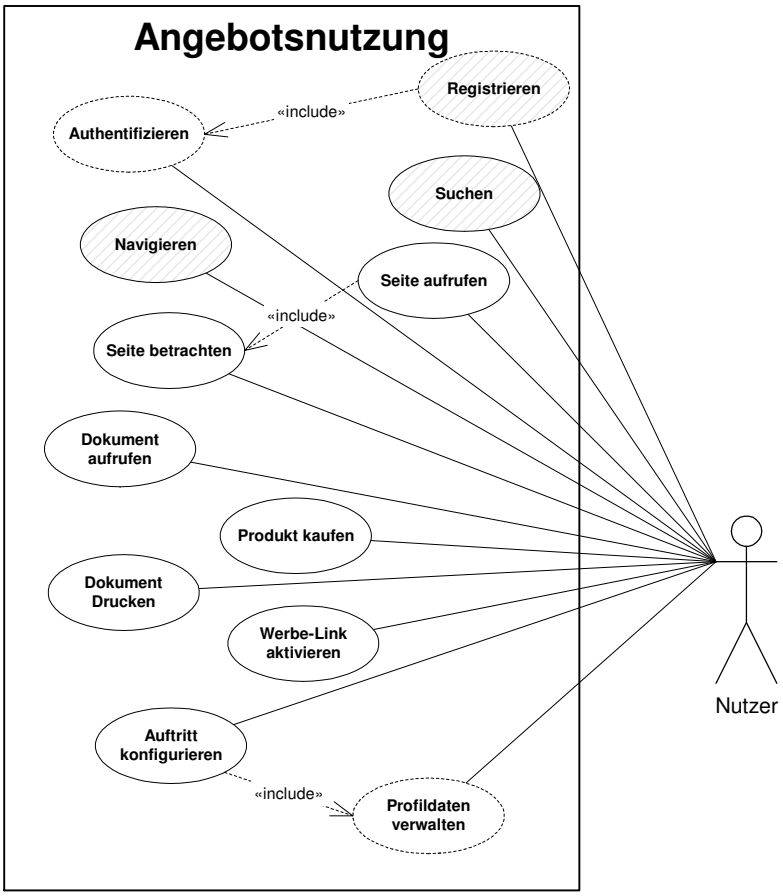


Abbildung 6-77: Anwendungsfälle der Angebotsnutzung

Das Diagramm enthält den bereits angesprochenen Anwendungsfall Auftritt konfigurieren und verdeutlicht, dass diese Konfiguration immer in Verbindung mit der Profildatenverwaltung des Nutzers erfolgt.

Ein Portalbetreiber bietet auf seiner Website umfangreiche Informationen zum Thema Börse und Finanzen an. Den Kunden, die sich für priviligierte Informationen wie Geschäftsberichte und Real-Time-Kurse registrieren müssen, steht eine Option zur Verfügung, mit der sie die Einstiegsseite individuell anpassen können. So können sie eine Liste von Unternehmen angeben, zu denen sie direkt nach der Authentifizierung umfangreiche Informationen wie den aktuellen Börsenkurs, vorliegende Analysteneinschätzungen sowie Ad-hoc-news angezeigt bekommen können. Dazu müssen sie diese Unternehmen sowie die gewünschten Informationsarten zuvor in ihrem Profil (beispielsweise in einem separaten Profilsegment *Beobachtungsliste*) hinterlegen.

Die übrigen Anwendungsfälle Seite aufrufen, Seite betrachten, Dokument abrufen, Dokument drucken, Produkt kaufen und Werbelink aktivieren sind Beispiele, die je nach konkreter Gestaltung des Angebotes variieren können und nur einen Eindruck vermitteln können, in welcher Art die Nutzung zum Ausdruck kommen kann. Diese Aktionen des Nutzers bilden die Grundlage für den Betreiber, implizite Informationen über ihn zu erhalten und so die Profile anzureichern, mit denen er handelt.

Der Anwendungsfall Seite aufrufen beinhaltet immer auch den Fall Seite betrachten. Da es für Auswertungszwecke eine wichtige Information ist, *welche* Seiten der Kunde betrachtet und *wie lange* er diese betrachtet, werden hier beide Fälle aufgeführt. Durch die Unterscheidung wird sichergestellt, dass beide Anforderungen in der Architekturgestaltung berücksichtigt werden.

Die Anforderungen in Form der Anwendungsfälle des Profiling werden in einem eigenen Szenario betrachtet, dessen Diagramm in Abbildung 6-78 gezeigt wird.

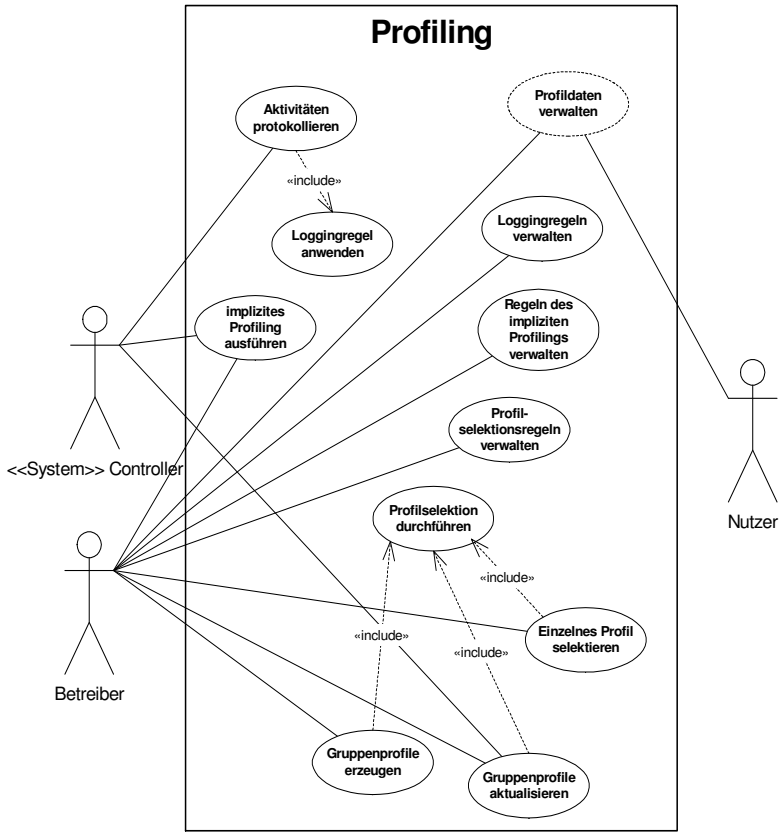


Abbildung 6-78: Anwendungsfälle des Profiling

Es wurde bereits erwähnt, dass dieses Szenario sehr eng verbunden ist mit der Nutzerverwaltung, jedoch wird es aufgrund der besonderen Bedeutung für dieses Erlösmodell separat betrachtet. Die wesentlichen Akteure sind der Betreiber sowie das System als automatisierter Controller.

Der Betreiber muss zunächst die Regeln im System hinterlegen, die die Basis der zu beobachtenden Aktionen sowie die Basis der Auswertungen der Daten zur impliziten Profilanreicherung bilden. Im Anwendungsfall Loggingregeln verwalten legt er fest, welche Aktionen des Nutzers protokolliert werden sollen. Diese Loggingregeln werden anschließend vom Controller angewendet, wenn das Logfile analysiert und die Aktivitäten protokolliert werden. Um aus dieser Datenmenge Rückschlüsse für das Profil zu gewinnen, muss der Betreiber Regeln des impliziten Profilings verwalten, bevor er diese anwenden kann und das implizite Profiling ausführt. Dieses kann fortlaufend automatisiert erfolgen oder auch manuell angestoßen werden. Die gespeicherten (geloggten) Daten des Nutzers werden dabei unter Anwendung der Regeln analysiert und die Ergebnisse in den Profilen hinterlegt.

Aus der Menge der Profile können anschließend einzelne Profile selektiert werden oder aber entsprechend einer Selektionsregel Gruppenprofile erzeugt werden. Die Profilselektionsregel kann je nach Anforderung eingerichtet, angewendet, gespeichert, verändert und erneut verwendet werden (Profilselektionsregeln verwalten). Die gewonnenen Gruppenprofile können darüber hinaus aktualisiert werden, wodurch beispielsweise neue Kunden, die dem Profil entsprechend automatisch hinzugefügt werden.

Die Aufgabe des Kunden beschränkt sich im Zusammenhang mit dem Profiling wie bereits erwähnt auf die explizite Verwaltung seiner Profildaten.

Die gesammelten Anforderungen müssen durch die Architektur berücksichtigt und durch entsprechende Funktionalitäten erfüllt werden. Der nächste Abschnitt befasst sich mit der Herleitung dieser Architektur.

6.4.1.2 Architekturbeschreibung

Das folgende Diagramm enthält die Klassen und Komponenten zur Realisierung des Erlösmodells PROFILHANDEL. Es ist direkt ersichtlich, dass sich die Software-Referenzarchitektur erheblich von den bisherigen Architekturen unterscheidet. Zwar weist das Diagramm auch hier eine Komponente zur Verwaltung von Kunden auf, jedoch weicht sie stark von den bisherigen Varianten ab. Grund dafür ist vor allem, dass es sich hier um ein Erlösmodell mit *indirekter* Erlösquelle handelt und ein Kundenobjekt somit nicht mehr über die erlösrelevanten Eigenschaften wie ein Debitorenkonto verfügen muss. Vielmehr kommt es hier darauf an, das Profil des Kunden möglichst umfassend und dynamisch erweiterbar zu gestalten.

Darüber hinaus müssen seine Aktivitäten während einer Sitzung sehr genau protokolliert werden, um eine implizite Auswertung seines Verhaltens zu ermöglichen. Gleich zwei Komponenten übernehmen diese Aufgabe: Die `Logging`-Komponente ist für die Identifizierung und Selektion der relevanten Nutzeraktionen zur Laufzeit verantwortlich, eine weitere Komponente speichert diese Daten vorübergehend in einem Sitzungsprotokoll ab.

Damit die so gewonnenen Daten ausgewertet und relevante Informationen in verdichteter Form dem Kundenprofil hinzugefügt werden können, verwaltet eine weitere `Profiling`-Komponente eine Sammlung von Auswertungsregeln und wendet diese auf den Daten des Sitzungsprotokolls an.

Die Selektion der Daten zur Aufbereitung der einzelnen Profile zu anonymisierten Gruppenprofilen, die je nach Wunsch der zahlenden Geschäftskunden zu gestalten sind, wird ebenfalls von einer eigenständigen Komponente übernommen.

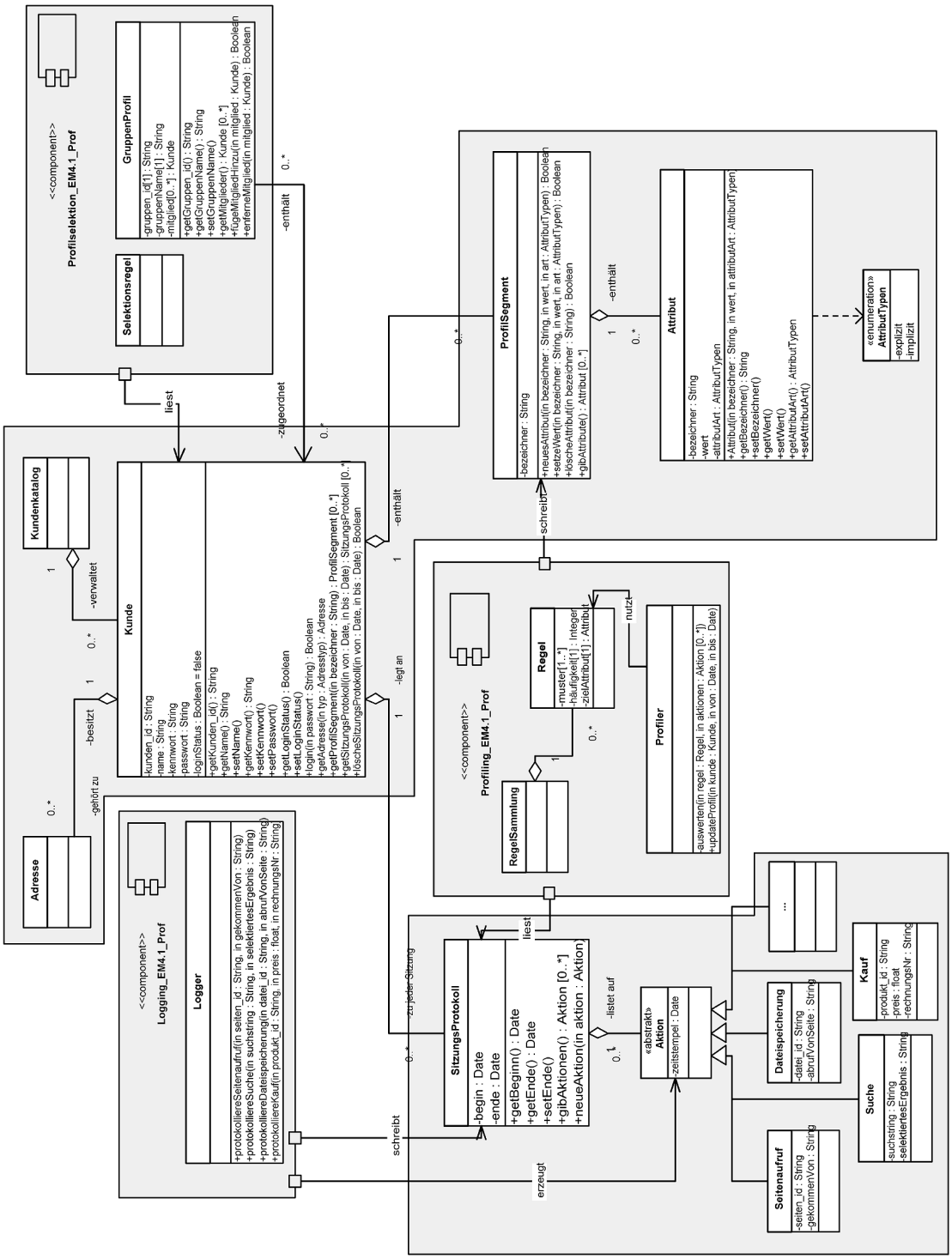


Abbildung 6-79: Statische Architektur des Erlösmodells Profilhandel für identifizierte Kunden

Die Architekturübersicht zeigt, dass in diesem Diagramm die Komponenten `Logging`, `Profiling` und `Profilselektion` nicht vollständig mit den enthaltenen Klassen dargestellt sind, sondern als Einheit, die über Ports mit der Restarchitektur interagiert. Eine Begründung für diese Verwendung wurde bereits in der Einleitung des Kapitels auf Seite 120 gegeben. Neben diesen Komponenten besteht die Architektur aus den Komponenten `Kunde` und `SitzungsProtokoll`. Sie werden im weiteren folgenden Verlauf vorgestellt.

Die Komponente zur Verwaltung des Kunden wird in der Version `Kunde_EM4.1_Prof` verwendet, die besonders die Anforderungen hinsichtlich eines umfangreichen und erweiterbaren Nutzerprofils berücksichtigt. Die Komponente enthält die Klassen `KundenKatalog`, `Kunde`, `ProfilSegment`, `Attribut` und `Adresse`, wie in Abbildung 6-80 dargestellt ist.

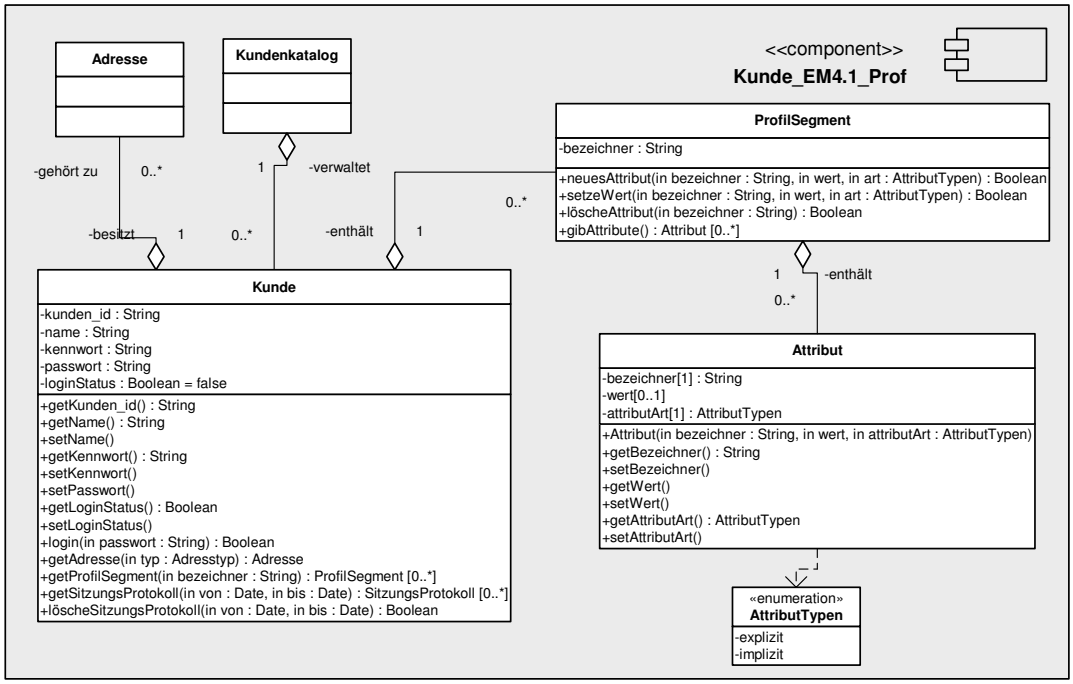


Abbildung 6-80: Komponente Kunde_EM4.1_Prof mit ihren Klassen

Die Aufteilung des Profils in die Klassen `Kunde`, `ProfilSegment` und `Attribut` ermöglicht den flexiblen Umgang mit den Nutzerprofilen. Ein Nutzer wird durch eine Instanz der Klasse `Kunde` repräsentiert. Zu jedem Kundenobjekt können 0..n Profilsegmente (Instanzen der Klasse `ProfilSegment`) verwaltet werden, zu denen jeweils 0..m Attribute (Instanzen der Klasse `Attribut`) angelegt werden können. So ist es für den Betreiber sehr einfach, seinen registrierten Nutzern zusätzliche Profilsegmente hinzuzufügen und diese bei Bedarf um beliebige zusätzliche Attribute zu erweitern.

Ein Nachrichtenmagazin teilt seinen Nutzern je nach Rubrik seines Auftritts ein eigenes Segment zu: Politik, Sport, Kultur, Weltgeschehen usw. Jedes dieser Segmente umfasst angepasste Attribute, die Auskunft geben über die Interessen und das Verhalten des Kunden. Das Segment Politik beinhaltet etwa die Attribute `durchschnittlicheLesezeitPolitik`, `häufigstBeachteteParteiinteressensAusrichtung` (mögliche Ausprägungen sind *Innenpolitik* und *Außenpolitik*) oder `häufigstesRessort` (mögliche Ausprägungen sind *Finanzen*, *Bildung*, *Soziales*, *Militär* etc). Diese Attribute werden allesamt implizit anhand des beobachteten Nutzerverhaltens erhoben. Das Segment Sport beinhaltet dagegen die Attribute `favorisierteSportart` und `Lieblingsclub`, wobei das letzte Attribut im Rahmen einer direkten Befragung des Kunden von ihm explizit angegeben wurde. Der Betreiber des Nachrichtenmagazins kann also umfangreiche Strukturen der Profile aufbauen und je nach Bedarf anpassen.

Die Klassen werden mit ihren Attributen und Methoden im folgenden noch detaillierter vorgestellt.

Diese Version der Komponente zur Verwaltung der Kunden verfügt nicht mehr über ein Debitoren-Konto, wie es die bisher diskutierten Versionen enthielten. Grund dafür ist die Tatsache, dass das Erlösmodell PROFILHANDEL die Einnahmen nicht mehr direkt von den Kunden bezieht und somit auch kein Konto notwendig ist.

Die Klassen `KundenKatalog` und `Adresse` sind unverändert aus der Version `Kunde_EM1.1_ET` (vgl. Abbildung 6-12 auf Seite 133) übernommen worden, weshalb ihre Attribute und Methoden nicht erneut aufgeführt und vorgestellt werden.

Die Klasse `Kunde` weist die bereits bekannten Standardattribute und die zugehörigen `get`- und `set`-Methoden auf. Darüber hinaus verfügt sie über die Methoden `getProfilSegment`, `getSitzungsProtokoll` und `löscheSitzungsProtokoll`.

Der Methode `getProfilSegment` wird als Parameter ein Bezeichner übergeben, der das zu überliefernde Segment kennzeichnet und identisch sein muss mit dem Attributwert von `bezeichner` der Klasse `ProfilSegment`. Sofern als Bezeichner ein definierter Wert (z.B. *alle*) übergeben wird, liefert die Methode sämtliche Profilsegmente zurück. Als Rückgabewert wird ein Listobjekt geliefert, das das oder die Segment(e) enthält.

Die Methode `getSitzungsProtokoll` erhält als Parameter einen Zeitraum (von, bis) und liefert als Rückgabewert ein Listobjekt aller in diesem Zeitraum vorhandenen Protokolle über die durchgeführten Sitzungen zurück. Die entsprechende Klasse `SitzungsProtokoll` wird weiter unten vorgestellt.

Beim Aufruf der Methode `löscheSitzungsProtokoll` wird ebenfalls ein Zeitraum (von, bis) als Parameterliste übergeben. Diese Methode löscht die vorhandenen Protokolle dieses Zeitraums, um die gesammelte Datenmenge zu reduzieren. Je nach Protokollierungsumfang und Nutzerzahlen können durch das Logging erhebliche Datenmengen anfallen,

die einerseits für die Auswertung persistent gespeichert werden, andererseits aber auch zu löschen sind, sofern sie nicht mehr benötigt werden. Die Methode kann somit manuell bei Bedarf aufgerufen oder automatisiert werden, sobald die gespeicherte Datenmenge eines Nutzers einen parametrisierbaren Höchstwert übersteigt.

Die Instanzen der Klasse `ProfilSegment` sind wie bereits erwähnt durch ihr Attribut `bezeichner` eindeutig identifizierbar. Darüber hinaus enthält die Klasse keine weiteren Attribute, da diese durch die Klasse `Attribut` abgebildet werden. Jede Instanz dieser Klasse enthält ein Set von drei Attributen: einen `bezeichner`, einen `wert` und eine `attributArt`. Die Attributmultiplizitäten zeigen an, dass dabei der Wert zunächst auch leer sein kann. Die Attribut-Art ist von einem Enumeration-Datentyp und kann die Ausprägungen *implizit* oder *explizit* annehmen. Auf diese Weise wird zu jedem Attribut festgehalten, ob es durch eine explizite Angabe des Nutzers oder aber mittels einer impliziten Erhebung durch das System gewonnen wurde.

Um den Profilsegmenten neue Attribute hinzuzufügen, verfügt `ProfilSegment` über die Methode `neuesAttribut`. Ihr werden die Parameter *bezeichner*, *wert*, und *attributArt* übergben, wobei der Parameter für den Wert auch leer bleiben kann. Die Methode ruft daraufhin den Konstruktor der Klasse `Attribut` auf, der eine neue Instanz erzeugt und die Parameter zuweist.

Um eine Wertzuweisung oder Wertänderung eines bestehenden Attributes durchzuführen, steht darüber hinaus die Methodes `setzeWert` (Klasse `ProfilSegment`) bereit. Ihr werden der Bezeichner des Attributes sowie der neue Wert als Parameter übergeben und sie ruft die `setWert`-Methode der zugehörigen Instanz der Klasse `Attribut` auf. Die Verwendung dieser Methode im Zusammenhang mit der Aktualisierung eines Profils ist im Sequenzdiagramm in Abbildung 6-85 auf Seite 224 zu sehen.

Die Methode `gibAttribute` liefert als Rückgabewert alle enthaltenen Attributobjekte eines Profilsegments als Listobjekt.

Durch diese komplexe Gestaltung des Kundenprofils in Form mehrerer einzelner Klassen ist die Anforderung nach einer dynamischen Erweiterung der Profile umgesetzt worden. Eine beliebige Menge neuer Segmente mitsamt ihren Attributen kann zu bestehenden Profilen hinzugefügt oder auch wieder gelöscht werden.

Für die Gewinnung impliziter Informationen hält die Komponente `SitzungsProtokoll` persistente Daten als Basis bereit. Sie wird nun beschrieben und ist in der nächsten Abbildung zu sehen.

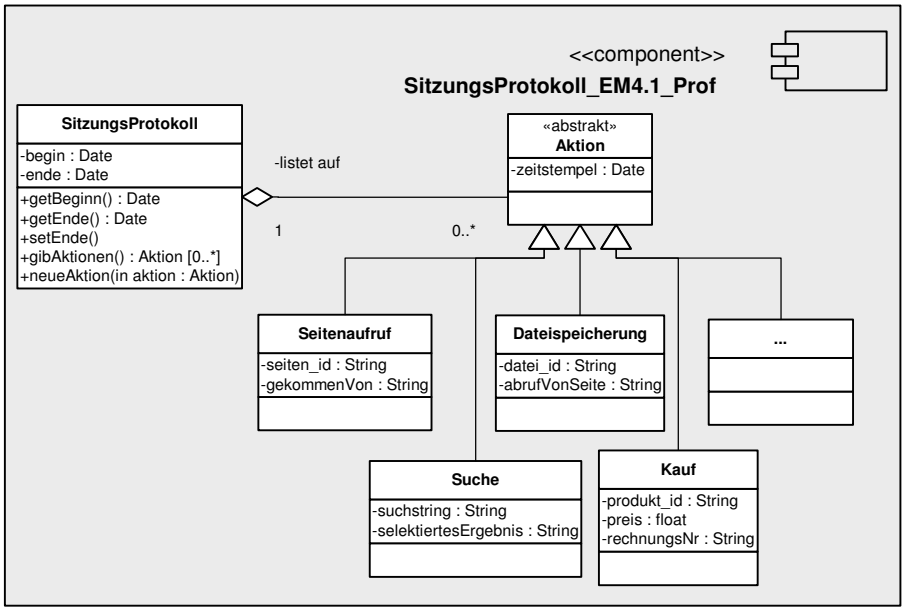


Abbildung 6-81: Komponente SitzungsProtokoll_EM4.1_Prof

Wie im Übersichtsdiagramm in Abbildung 6-79 zu sehen ist, verfügt jeder Kunde über 0..n Sitzungsprotokolle, wobei zu jeder Sitzung eine zusätzliche Instanz angelegt wird. Dadurch ist es möglich, jede Sitzung isoliert als Protokoll zu erfassen, auszuwerten und auch wieder zu löschen. Das Sitzungsprotokoll dient somit lediglich als vorübergehender Speicher von Verhaltensdaten des Nutzers, die zu einem späteren Zeitpunkt in interpretierter Form den dauerhaften Nutzerprofilen hinzugefügt werden. Durch diese Trennung wird das Nutzerprofil also nicht unnötig mit riesigen implizit gewonnenen Datenmengen angereichert, sondern enthält nur aufbereitete Informationen. Aus diesem Grund verfügt die Refernzarchitektur über beide Komponenten.

Das Sitzungsprotokoll speichert sessionübergreifend Informationen über einen Nutzer, die aus seinem Verhalten während einer Sitzung hergeleitet werden können. Da die konkrete Umsetzung eines solchen Protokolls sehr stark vom konkreten Anwendungsfall abhängt, kann hier als Referenz nur ein Beispiel aufgezeigt werden, wie die Klassenstruktur einer solchen Komponente aussehen kann. Ein `SitzungsProtokoll` wird wie bereits erwähnt zu Beginn einer neuen Sitzung für einen Kunden angelegt. Diese Klasse wird im weiteren Verlauf von der Komponente `Logging` dazu verwendet, bestimmte Aktionen des Nutzers zu speichern. Die Aktionen werden als Instanzen der (abstrakten) Klasse `Aktion` gespeichert. Im Rahmen dieser Referenzarchitektur dienen die Spezialisierungen `Seitenaufruf`, `Suche`, `Dateispeicherung` und `Kauf` als mögliche Repräsentanten von zu protokollierenden Aktionen. Je nach Spezialisierung sind geeignete Attribute zu speichern, wie in Abbildung 6-81 zu sehen ist. Durch die Darstellung einer Klasse mit der Bezeichnung „...“ wird angedeutet, dass dies jedoch nur ein Ausschnitt der Möglichkeiten ist.

Sofern die Komponente `Logging` (Beschreibung weiter unten) die Protokollierung einer Aktion veranlassen soll, ruft sie ihrerseits die Methode `neueAktion` vom `SitzungsProtokoll` auf und übergibt die zu buchende Aktion als Parameter. Eine Ausgabe aller Aktionen einer Sitzung erfolgt mit Hilfe der Methode `gibAkionen`.

Die Komponente `Logging` wird in der folgenden Abbildung dargestellt. Sie sorgt dafür, dass genau die Interaktionen des Benutzers mit dem System identifiziert und gespeichert werden, die für ein Unternehmen im Sinne der Profilbildung von Relvanz sind. Diese Komponente ist sehr eng mit dem `SitzungsProtokoll` verbunden, jedoch wird sie bewusst eigenständig realisiert. Während das Sitzungsprotokoll vordergründig als Speicher der Daten dient, enthält die Logging-Komponente die Logik der Verhaltensprotokollierung. Bei Bedarf kann diese Komponente an veränderte Verfahren angepasst werden, ohne dass das Sitzungsprotokoll davon direkt betroffen ist.

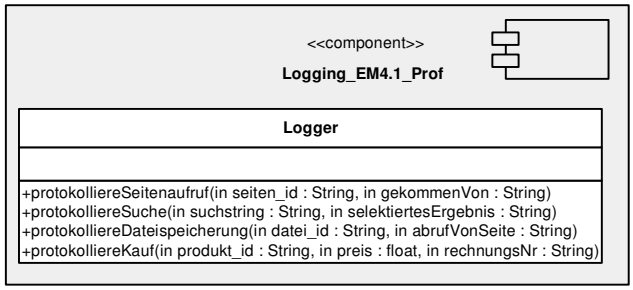


Abbildung 6-82: Komponente Logging_EM4.1_Prof

Wie einleitend in dem Kapitel bereits erläutert wurde, wird diese Art[16] der Komponente nicht mit einem vollständigen Klassendiagramm beschrieben, sondern nur anhand der *wichtigsten* Klassen und Methoden, die im Zusammenspiel mit der Restarchitektur von Relevanz sind.

Die Komponente bietet die Dienste `protokolliereSeitenaufruf`, `protokolliereSuche`, `protokolliereDateispeicherung` und `protokolliereKauf`, die als Methoden der Klasse `Logger` bereitgestellt werden. Der Methodenaufruf erfolgt je nach konkretem Anwendungsfall von Komponenten, die eine dieser Aktionen durchführen und in deren Programmlogik der Aufruf dieser Dienste implementiert ist. Da diese Komponenten außerhalb einer Referenzarchitektur sind, werden sie hier nicht näher betrachtet und sich stattdessen auf den Ablauf nach den Methodenaufruf konzentriert.

Jede der vier erwähnten Methoden ruft ihrerseits die Methode `neueAktion` der Klasse `SitzungsProtokoll` auf, die für die entprechende Buchung und persistente Speicherung sorgt. Die je nach Aktion übergebenen Parameter werden dabei weitergereicht. Da

16. *diese Art* bedeutet hier, dass bereits in der Architekturübersicht die Komponente als Einheit dargestellt wurde, die über ihre Ports mit der Restarchitektur kommuniziert, und nicht, wie die übrigen Komponenten, eine Darstellung aller realisierender Klassen beinhaltet.

das Zusammenspiel der Komponenten `Logging` und `SitzungsProtokoll` die Basis der impliziten Profilgewinnung bilden, wird ihre Funktionsweise im folgenden Sequenzdiagramm in Abbildung 6-83 detailliert aufgezeigt..

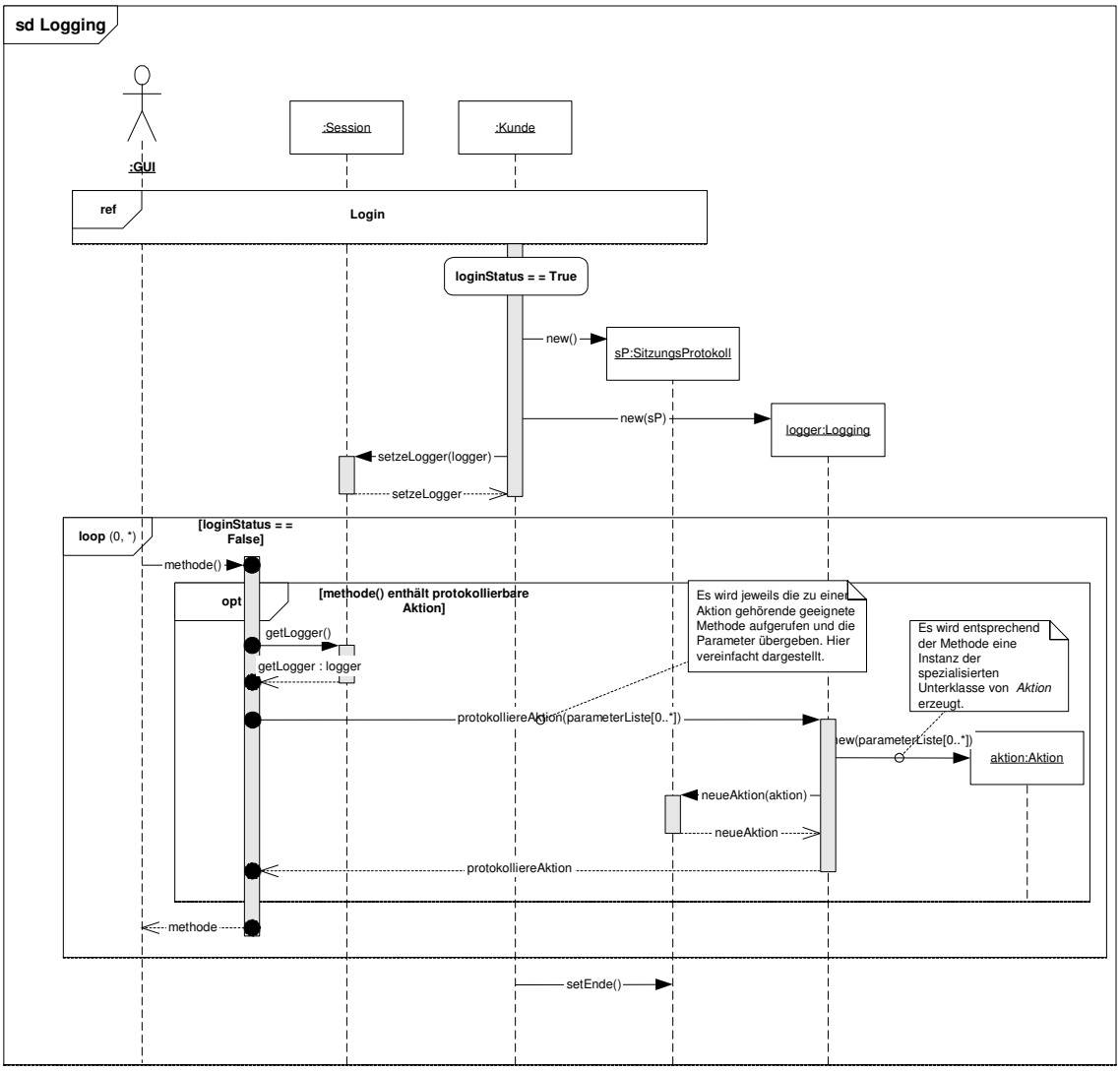


Abbildung 6-83: Sequenzdiagramm der Protokollierung von Nutzeraktionen

Nach der Authentifizierung eines Nutzers kann dessen Sitzung protokolliert werden, so dass das Kundenobjekt die Erzeugung neuer Instanzen der Klasse `SitzungsProtokoll` und der Klasse `Logging` veranlasst. Eine Referenz auf das Loggingobjekt wird der Session mitgeteilt, damit sie sämtlichen Komponenten zum Abruf zur Verfügung steht.

Die eigentliche Protokollierung wird hier im Rahmen eines Schleifenfragments dargestellt, dessen Abbruchbedingung das Abmelden des Nutzers ist. Solange der Nutzer innerhalb seiner Sitzung Methoden aufruft, die bestimmte Funktionalitäten bereitstellen und darüber hinaus die Protokollierung dieser Aktion implementiert haben (diese zusätzliche Bedingung wird durch das optionale Fragment repräsentiert), werden diese Methoden aktiv.

Zunächst ermitteln sie über die Session das aktuelle Loggingobjekt des Kunden, bevor sie dort die entsprechende Methode aufrufen und die notwendigen Parameter überreichen. `Logging` erzeugt daraufhin eine neue Instanz der spezialisierten Unterklasse von `Aktion` und übergibt die erhaltenen Parameter. Im Anschluss daran übergibt sie dem `SitzungsProtokoll` einen Verweis auf dieses neue Objekt, indem sie die zugehörige Methode `neueAktion` aufruft.

Die bis hierher beschriebene Protokollierung dient dazu, die Profile des Nutzers mit implizit erhobenen Daten anzureichern. Dazu müssen die protokollierten Daten anhand von Regeln analysiert und ausgewertet werden. Für die Durchführung dieser Analyse und Auswertung ist die Komponente `Profiling_EM4.1_Prof` zuständig, die nun betrachtet wird.

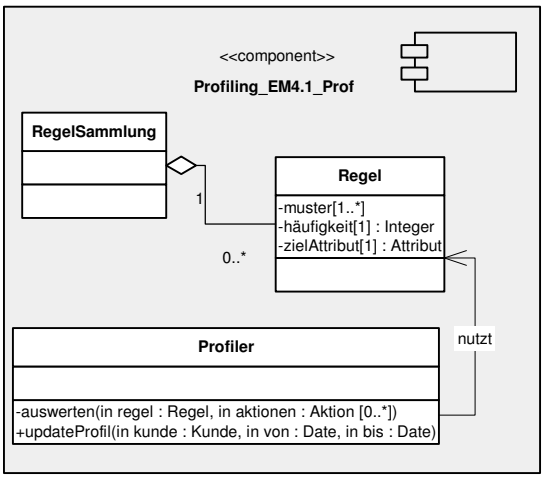


Abbildung 6-84: Komponente Profiling_EM4.1_Prof

Die Komponente `Profiling_EM4.1_Prof` wird ebenfalls nur mit ihren wichtigsten Klassen und Methoden beschrieben und nicht vollständig spezifiziert. Die wesentliche Aufgabe der Komponente besteht darin, die gesammelten Informationen aus den Sitzungsprotokollen zu analysieren und das Profil des zugehörigen Nutzers um die gewonnenen Informationen zu ergänzen.

Für eine sinnvolle Nutzung der Verhaltensbeobachtung benötigt die Komponente eine Reihe von Regeln zur Bewertung der Interaktionen. Beispielsweise können die Aktionen Selektion, Seitenbetrachtung mit Dauer, Speicherung und Drucken eines Dokuments dienen. Das Regelwerk legt fest, auf welche Weise diese Aktionen das Nutzerprofil beeinus-

sen sollen. So kann davon ausgegangen werden, dass die Speicherung oder eine längere Sichtung eines Dokuments ein hohes Interesse seitens des Benutzers zum Ausdruck bringt[17].

Aus diesem Grund ist eine `Regelsammlung` notwendig, die eine Reihe von `Regeln` enthält. Jede Regel kann dabei beschrieben werden durch ein bestimmtes Verhaltensmuster (`muster`) des Nutzers und eine Häufigkeit (`häufigkeit`), in der dieses Verhaltensmuster auftritt, damit eine Schlussfolgerung gezogen werden kann. Diese Schlussfolgerung wird in einem ebenfalls definierten Attribut `zielAttribut` festgehalten.

> Ein Beispiel für eine Profiling-Regel ist die Ermittlung der favorisierten Sportart eines Nutzers, der die Webanwendung eines Nachrichtenmagazins besucht. Das Verhaltensmuster kann dabei ausgedrückt werden in dem Verhalten, welche Unterrubrik der Kunde als erstes nach seinem Einstieg in die Seiten der Sport-Rubrik aufruft. Führt er diese Navigationsfolge mindestens zu *n* Prozent aller Besuche innerhalb der Sport-Rubrik aus, wird ab einer signifikanten Erhebungsgröße davon ausgegangen, dass dem zugehörigen Attribut `favorisierteSportart` seines Profils der entsprechende Wert zugewiesen werden kann.

Die Klasse `Profiler` verwendet diese Regeln, um die Profile zu aktualisieren. Ihre Methode `auswerten` erhält als Eingabeparameter dazu eine solche Regel und eine Menge von auszuwertenden Aktionen, die von der Klasse `SitzungsProtokoll` angefordert wurden. Die Methode wird im Rahmen einer Profilaktualisierung verwendet, wozu die Methode `updateProfil` aufgerufen werden kann, die ebenfalls von der Klasse `Profiler` bereitgestellt wird. Ihr werden der zu aktualisierende Kunde sowie ein Zeitraum (von, bis) übergeben, über den die Sitzungsprotokolle analysiert werden sollen.

Das Sequenzdiagramm in Abbildung 6-85 zeigt das komplexe Zusammenspiel der Klassen während der Auswertung der Sitzungsprotokolle und Anreicherung der Nutzerprofile.

17. Heuristische Metriken zur impliziten Auswertung des Nutzerverhaltens werden in den Arbeiten [KO98] und [KOR00] vorgestellt.

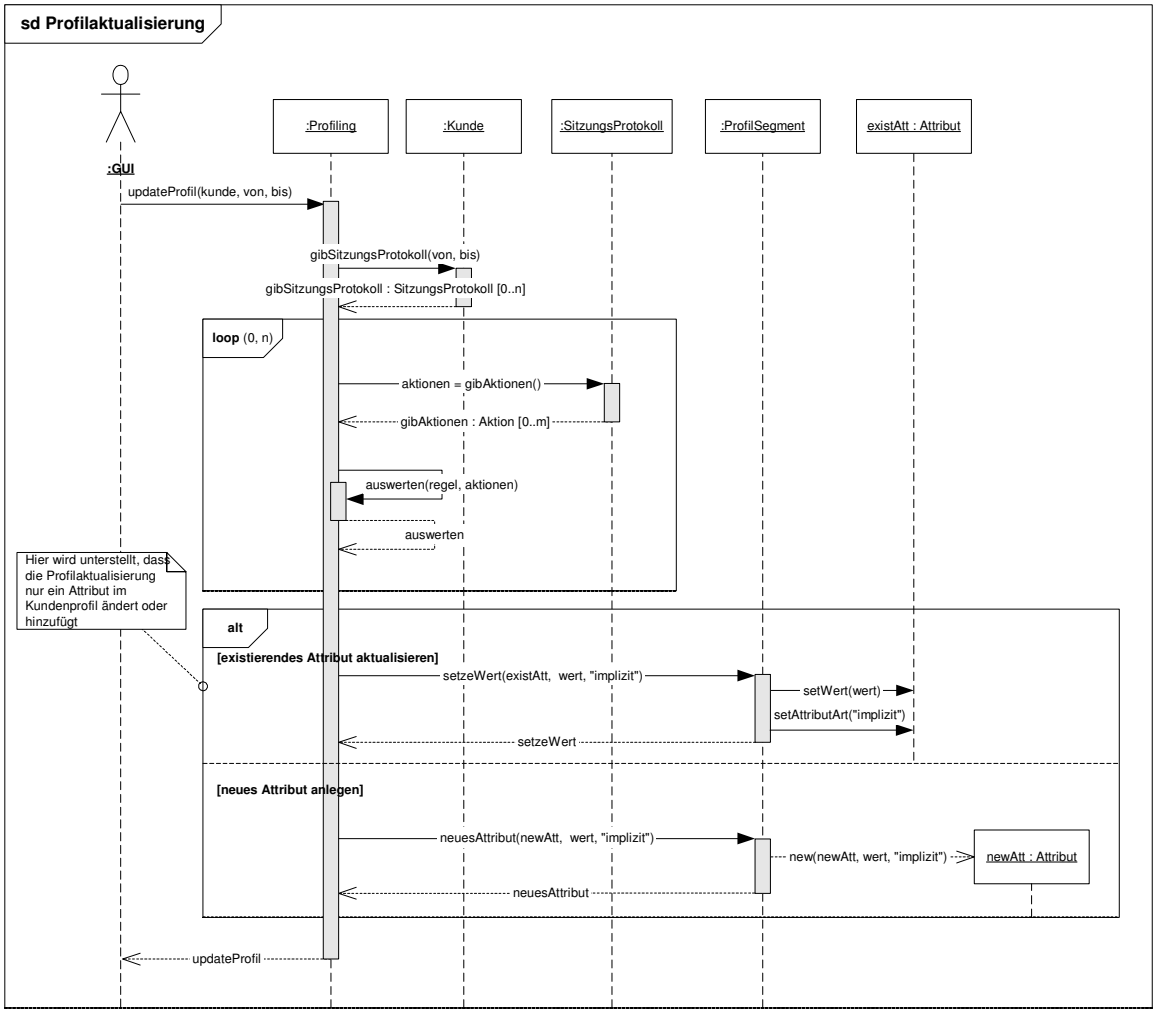


Abbildung 6-85: Sequenzdiagramm zur impliziten Datenerhebung und Aktualisierung der Nutzerprofile

Nach dem Methodenaufruf `updateProfil` ermittelt diese zunächst das `SitzungsProtokoll` anhand des übergebenen Kundenobjekts (`getSitzungsProtokoll`). Für jedes einzelne Protokoll (ersichtlich am Schleifenfragment) werden die Aktionen als Listobjekt vom `SitzungsProtokoll` mit Hilfe der Methode `gibAktionen` angefordert und anschließend unter Verwendung einer Regel durch die Methode `auswerten` analysiert[18]. Die dabei anzuwendende Regel muss zuvor innerhalb der Komponente `Profiling` entsprechend aktiviert werden. Die so gewonnenen Erkenntnisse werden anschließend im Profil des Kunden persistent gespeichert, wobei unterschieden wird, ob das Attribut bereits

18. Auch dieser Ablauf kann je nach konkreter Anwendung von dieser Referenz abweichen. Es ist ebenso denkbar, dass die Auswertung der Aktionen nicht zu jeder Sitzung erfolgt, sondern die Menge aller Aktionen über alle Sitzungen innerhalb des Zeitraums analysiert werden.

existiert und nur aktualisiert werden muss, oder ob das Attribut noch nicht existiert und somit neu anzulegen ist. In beiden Fällen wird jedoch der Typ des Attributs als *implizit* festgelegt.

Die letzte noch nicht vorgestellte Komponente der Architektur dient dazu, einzelne Profile oder Gruppenprofile zu extrahieren, damit diese als Basis des Erlösmodells gehandelt werden können. Die Komponente `ProfilSelektion_EM4.1_Prof` ist in der nächsten Abbildung zu sehen.

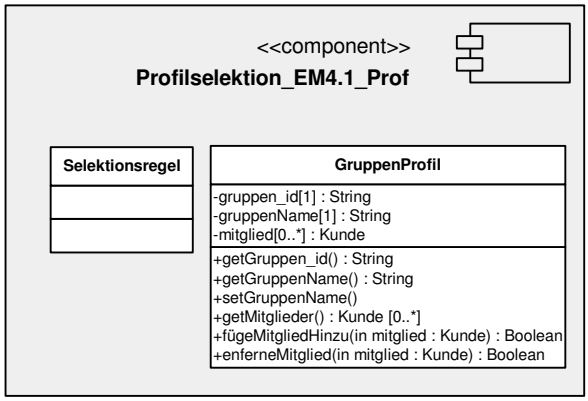


Abbildung 6-86: Komponente ProfilSelektion_EM4.1_Prof

Auch diese Komponente kann hier nur exemplarisch wiedergegeben werden, da ihre konkrete Gestaltung wiederum sehr stark von einem konkreten Anwendungsfall abhängig ist. Sie enthält im Wesentlichen Funktionalitäten zur Beschreibung und Eingrenzung der zu selektierenden Profile und darüber hinaus die selektierten Profile selber. Die Dienste zur Konfiguration der Selektierungsregeln werden hier nicht betrachtet. Es wird mit der Klasse `Selektionsregel` lediglich angedeutet, dass diese Regeln zu verwalten und die Grundlage der Filterung sind. Aufgrund der Art der zu erhebenen Daten ist eine mögliche Variante, dass SQL-Statements über eine GUI einzurichten und zu verwalten sind und diese auf einen spezifizierbaren Datenbestand ausgeführt werden können. Das Ergebnis einer Anwendung der Selektionsregeln sind Instanzen der Klasse `GruppenProfil`. Diese enthält im Wesentlichen Verweise auf die Kunden, die gemäß der Selektionskriterien dieser Gruppe zugewiesen wurden.

6.4.2 Variante Anonymer Kunde

In dieser Variante werden keine persistenten Kundenprofile gespeichert, die einem Kunden eindeutig zuzuweisen sind. Stattdessen wird je Sitzung ein anonymes Profil erzeugt und das Benutzerverhalten anonym protokolliert. Details sind der Beschreibung in Abschnitt 4.2.4.2 auf Seite 81 zu entnehmen.

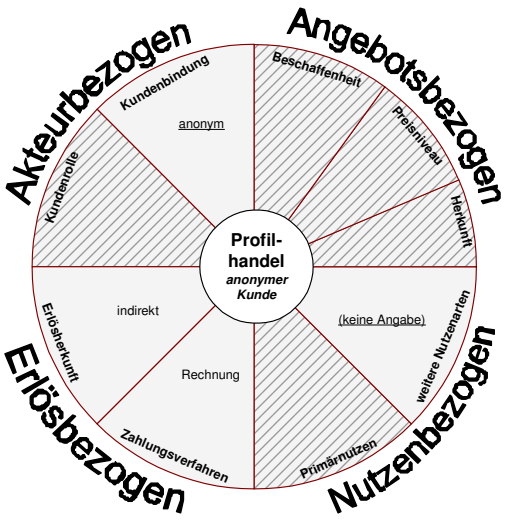


Abbildung 6-87: Erlösmodellklassifikation Profilhandel Variante anonymer Kunde

Auch hier sind die nicht relevanten Kriterien durch die schraffierte Darstellung ausgeblendet worden.

6.4.2.1 Anforderungsanalyse und Systemabgrenzung

Auch in dieser Variante gibt es weiterhin Profile der Nutzer, nur sind diese nicht mehr identifizierbar und nicht zuzuordnen. Die Verwaltung der Profile umfasst das Verwalten der Templates, die mit implizit erhobenen Daten gefüllt werden. Auch das Hinzufügen von Profilsegmenten bleibt als Anforderung bestehen, wie in der folgenden Abbildung dargestellt wird.

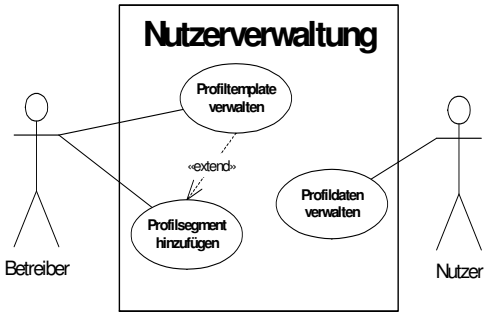


Abbildung 6-88: Nutzerverwaltung für anonyme Kunden

Es ist vorstellbar, dass der Nutzer auch in der anonymen Variante explizit Daten für sein (anonymes) Profil angibt. Ein Beispiel dafür ist eine Umfrage, an der der Nutzer teilnimmt und in der er Aussagen trifft, die in seinem Profil gespeichert werden. Aus diesem Grund ist der in der vorherigen Version verwendete Anwendungsfall Profildaten verwalten auch hier gültig, wobei er hier eben nur in diesem eingeschränkten Umfang möglich ist.

Die Angebotsnutzung erfolgt in dieser Variante anonym. Dennoch ist die implizite Datengewinnung anhand des Benutzerverhaltens identisch. Die folgende Abbildung zeigt die Anwendungsfälle, die zu einer Protokollierung führen können.

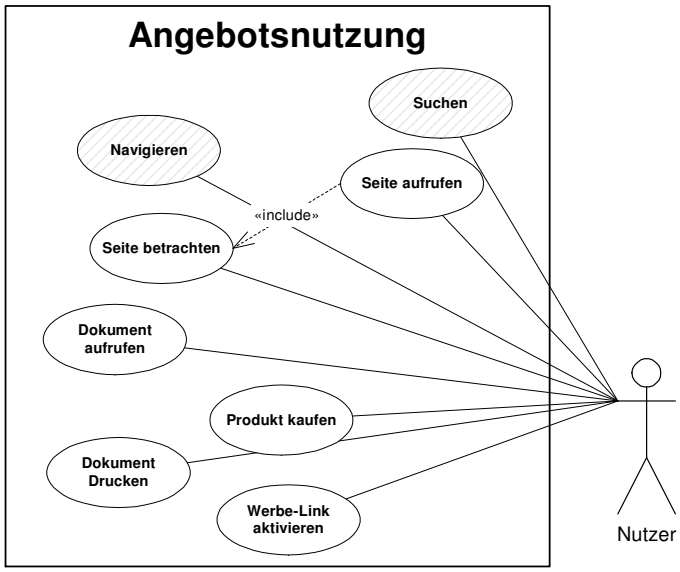


Abbildung 6-89: Anwendungsfälle der Angebotnutzung mit anonymen Kunden

Das Profiling ist identisch mit der vorherigen Variante. Die Darstellung ist in Abbildung 6-78 auf Seite 213 zu sehen.

Zusammenfassend lässt sich festhalten, dass die Anforderungen weitgehend unverändert bestehen und lediglich die Identität des Nutzers nicht gegeben ist und die Erhebung der nun anonymen Profile lediglich auf Basis einzelner Sitzungen erfolgt. Auf die Thematik, mittels Cookies sitzungsübergreifende Profile zu ermitteln, wird an dieser Stelle nicht eingegangen.

6.4.2.2 Architekturbeschreibung

Hinsichtlich der Architektur, die in Abbildung 6-90 als Übersicht dargestellt ist, ändert sich die Komponente zur Verwaltung der Kunden in geringem Umfang, der Ablauf des Loggings sowie der Ablauf des Profilings, wodurch die Dienste der zugehörigen Komponenten anzupassen sind.

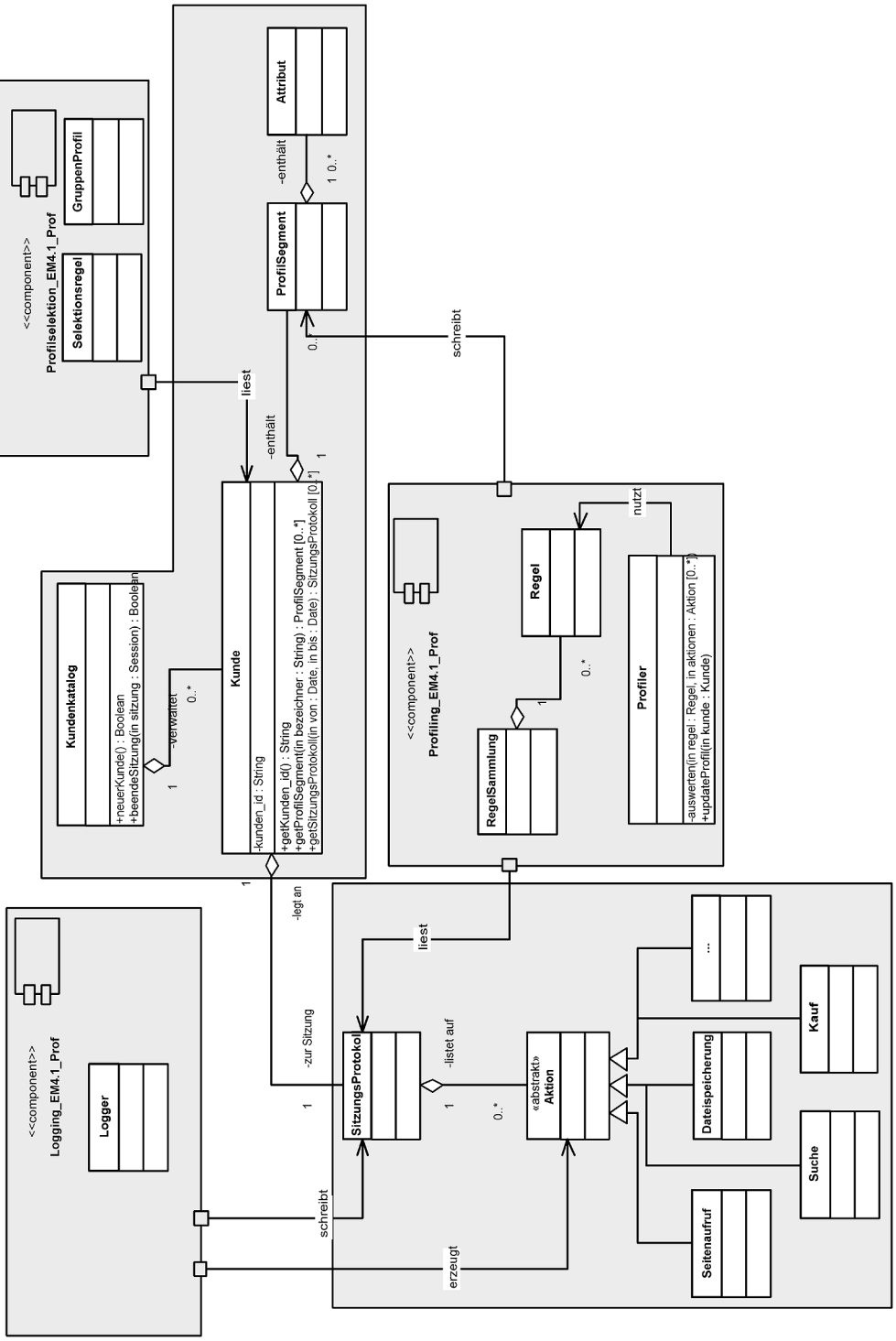


Abbildung 6-90: Statische Architektur des Erlösmodells Profilhandel für anonyme Kunden

Die wesentliche Änderung zur vorherigen Variante besteht darin, dass hier kein *identifizierbares* Kundenprofil angelegt und verwaltet wird. Dennoch wird ein Kundenprofil angelegt, wobei dieses eben nicht personalisiert und nicht eindeutig einem Nutzer zuzuweisen ist. Stattdessen ist es vielmehr sitzungsbezogen, da für jede Sitzung eines Nutzers ein neues Profil angelegt wird. Die Komponenten der Software-Referenzarchitektur müssen gemäß dieser Veränderungen angepasst werden. Darüber hinaus sind die Anforderungen an die Gestaltung der Architektur jedoch weitgehend identisch mit der vorherigen Variante.

Die Architektur zeigt, dass die Komponente zur Verwaltung der Kunden angepasst worden ist. Ein Kunde enthält keine persönlichen Attribute mehr, so dass auch die `Adresse` weggefallen ist. Die Komponente `Kunde_EM4.2_Prof` zeigt die folgende Abbildung.

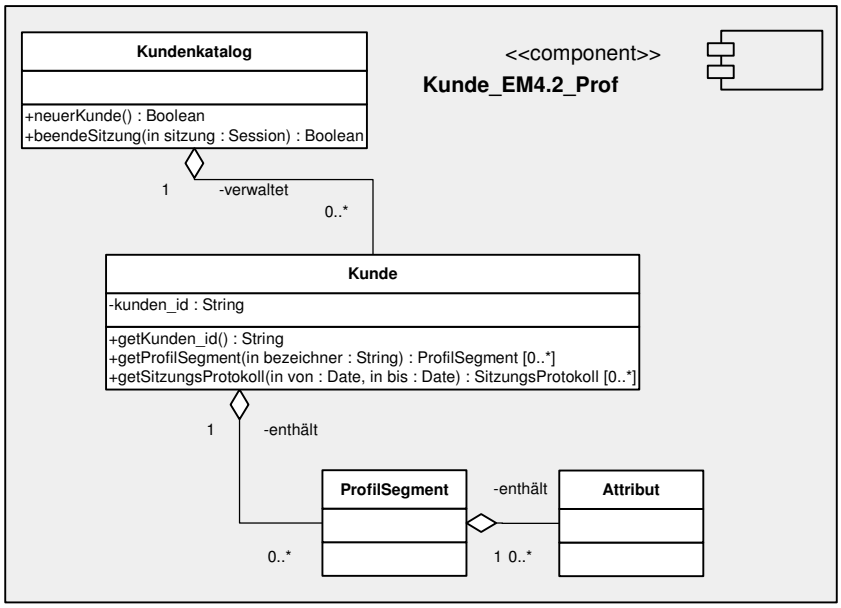


Abbildung 6-91: Komponente Kunde_EM4.2_Prof

Die Klassen `ProfilSegment` und `Attribut` sind identisch mit der Version `Kunde_EM4.1_Prof` aus der Variante für identifizierte Nutzer und werden hier deshalb nur durch den Klassennamen repräsentiert. Die Klasse `Kunde` enthält nur ein einziges Attribut zur eindeutigen Kennzeichnung für die Dauer einer Session. Ihre Methoden beschränken sich auf den Umgang mit dem `ProfilSegment` und dem zugehörigen `SitzungsProtokoll`.

Die Klasse `KundenKatalog` enthält zwei Methoden. `neuerKunde` legt zu Beginn einer Session automatisch eine neues Kundenobjekt an. Da der Kunde sich hier nicht wie in der vorherigen Variante authentifiziert und ihm kein bestehendes Profil zugewiesen werden kann, wird bei jeder Session ein neues Profil angelegt. Dies ist notwendig, damit die Komponente Logging, die hier unverändert in der Version `Logging_EM4.1_Prof` übernommen werden kann, die Protokollierung der Aktionen des Nutzers festhalten kann. Der Ablauf ist im folgenden Sequenzdiagramm in Abbildung 6-92 zu sehen, dass das Logging

von Aktionen anonymer Nutzer zeigt. Die Unterschiede zur vorherigen Variante werden dabei offensichtlich. Anders als dort ist hier nicht die erfolgreiche Authentifizierung eines registrierten Nutzers die ausschlaggebende Aktion zum Starten der Protokollierung, sondern bereits das Starten einer neuen Sitzung. Ein Kundenobjekt wird daraufhin neu instanziiert.

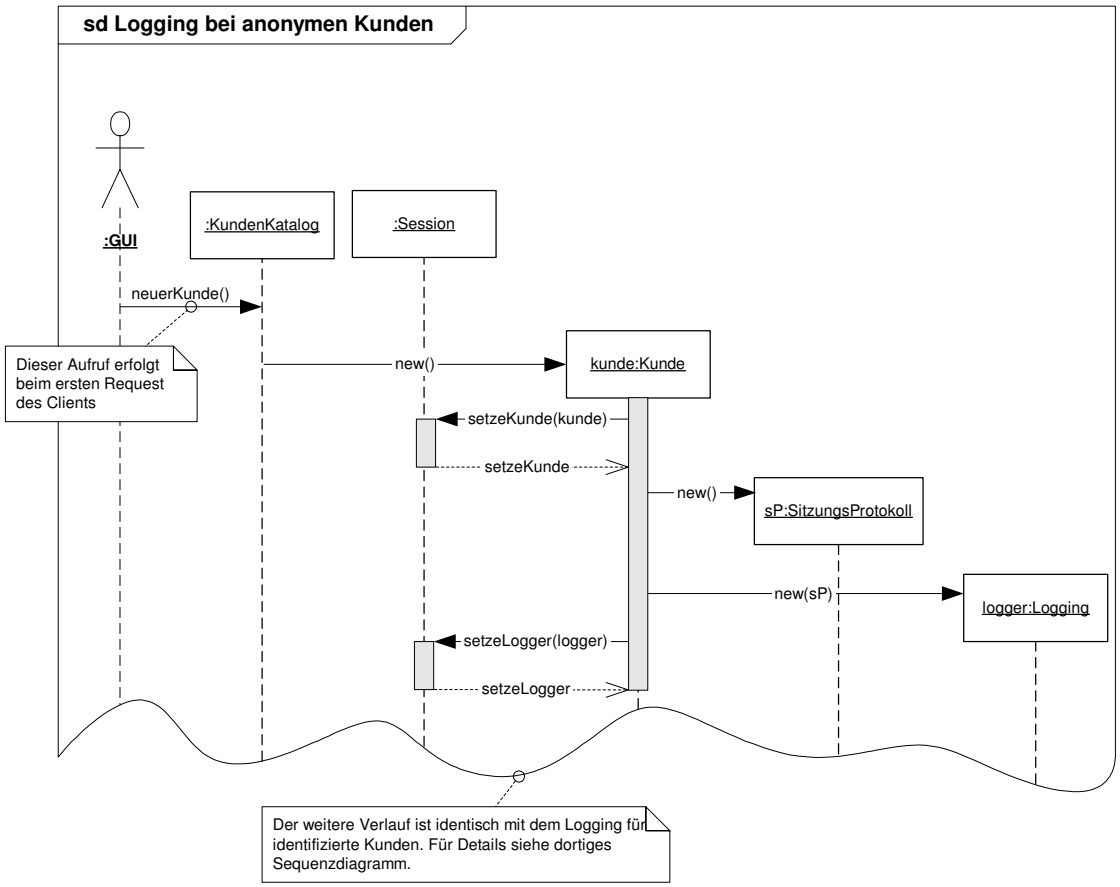


Abbildung 6-92: Auszug aus dem Sequenzdiagramm für das Logging bei anonymen Kunden

Der Konstruktor der Klasse `Kunde` übernimmt nun die Aufgabe, das `SitzungsProtokoll` sowie eine Instanz von `Logging` zu erzeugen und diese der Session mitzuteilen. Darüber hinaus teilt die Methode der Session auch eine Referenz auf sich selbts mit. Dies ist notwendig, um nach Beendigung der Session die Auswertungen im Rahmen des Profilings durchführen zu können. Dazu dient die Methode `beendeSitzung` vom `KundenKatalog`.

Bevor auf diese Methode eingegangen wird, ist noch festzuhalten, dass die auch die Komponente zur Verwaltung des Sitzungsprotokolls unverändert aus der vorherigen Variante in der Version `SitzungsProtokoll_EM4.1_Prof` übernommen werden kann.

Lediglich die Assoziation zum Kunden hat sich geändert, da ein Kundenobjekt nun genau über ein Sitzungsprotokoll verfügt und nicht, wie zuvor, über mehrere verfügen kann. Diese Abhängigkeit ist im Übersichtsbild der Architektur auf Seite 228 zu sehen.

Der weitere Ablauf des Loggings ist somit identisch mit der vorherigen Version, da die Interaktion der Komponente `Logging` mit der Restarchitektur einschließlich der Komponente `SitzungsProtokoll` identisch verläuft.

Die noch zu beschreibende Methode `beendeSitzung` der Klasse `KundenKatalog` dient wie bereits erwähnt dazu, am Ende einer Sitzung dafür zu sorgen, dass die Aktionen des Nutzers ausgewertet und die implizite Datengewinnung durchgeführt wird. Als Paramter erhält die Klasse eine Referenz auf die zu beendende Sitzung. Da der Kunde anonym interagiert, wird er sich nicht selbstständig vom System abmelden. Somit wird die Sitzung nur durch die Inaktivität über einen bestimmten Zeitraum automatisch vom System beendet. In diesem Zusammenhang wird die Methode aufgerufen.

Das Interaktionsdiagramm in Abbildung 6-94 zeigt den Ablauf des Profilings für anonyme Nutzer. Zuvor darauf eingegangen wird, wird die angepasste Komponente `Profiling_EM4.2_Prof` vorgestellt.

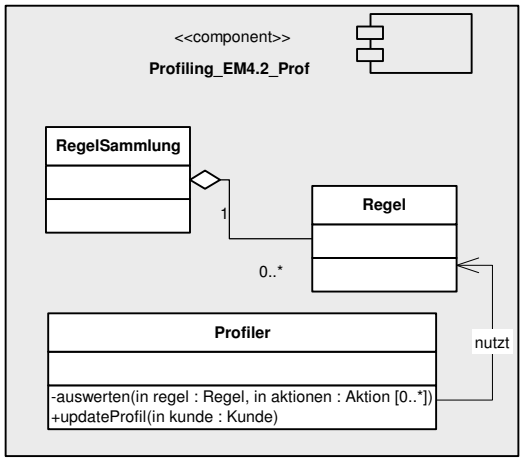


Abbildung 6-93: Komponente Profiling_EM4.2_Prof

Die Komponente enthält erneut die Klassen `RegelSammlung` und `Regel`, die unverändert zur vorherigen Variante übernommen wurden. Die Klasse `Profiler` wurde jedoch leicht verändert. Ihre Methode `updateProfil` erhält, anders als in der vorherigen Variante, nun keinen Zeitraum mehr als Übergabeparameter, da sich die Auswertungen immer auf die gesamte Sitzung eines Kunden beziehen. Der Aufruf der Methode muss in diese Variante ebenfalls angepasst werden, was im folgenden Sequenzdiagramm in Abbildung 6-94 zu sehen ist.

Hier zeigt ein Vergleich mit dem Profiling für identifizierte Kunden den wesentlichen Unterschied dieser Variante: Die Aktivierung des Profiling erfolgt nicht losgelöst von einer Session zu einem beliebigen Zeitpunkt, sondern automatisch sofort nach Beendigung einer Sitzung. Da der Kunde nicht identifiziert interagiert und sich somit auch nicht explizit abmeldet, übernimmt ein Controller die Aufgabe der Aktivierung.

Zunächst wird der `KundenKatalog` über das Beenden einer Sitzung informiert. Dazu wird der Methode `beendeSitzung` eine Referenz auf die Sitzung übergeben. Von dieser Sitzung ermittelt die Methode zunächst das zugehörige Kundenobjekt, was zu Beginn der Sitzung dort durch den Konstruktor des Kunden hinterlegt wurde. Dazu siehe das Sequenzdiagramm in Abbildung 6-92 auf Seite 230. Den so ermittelten Kunden kann die Methode nun an die Klasse `Profiling` übergeben.

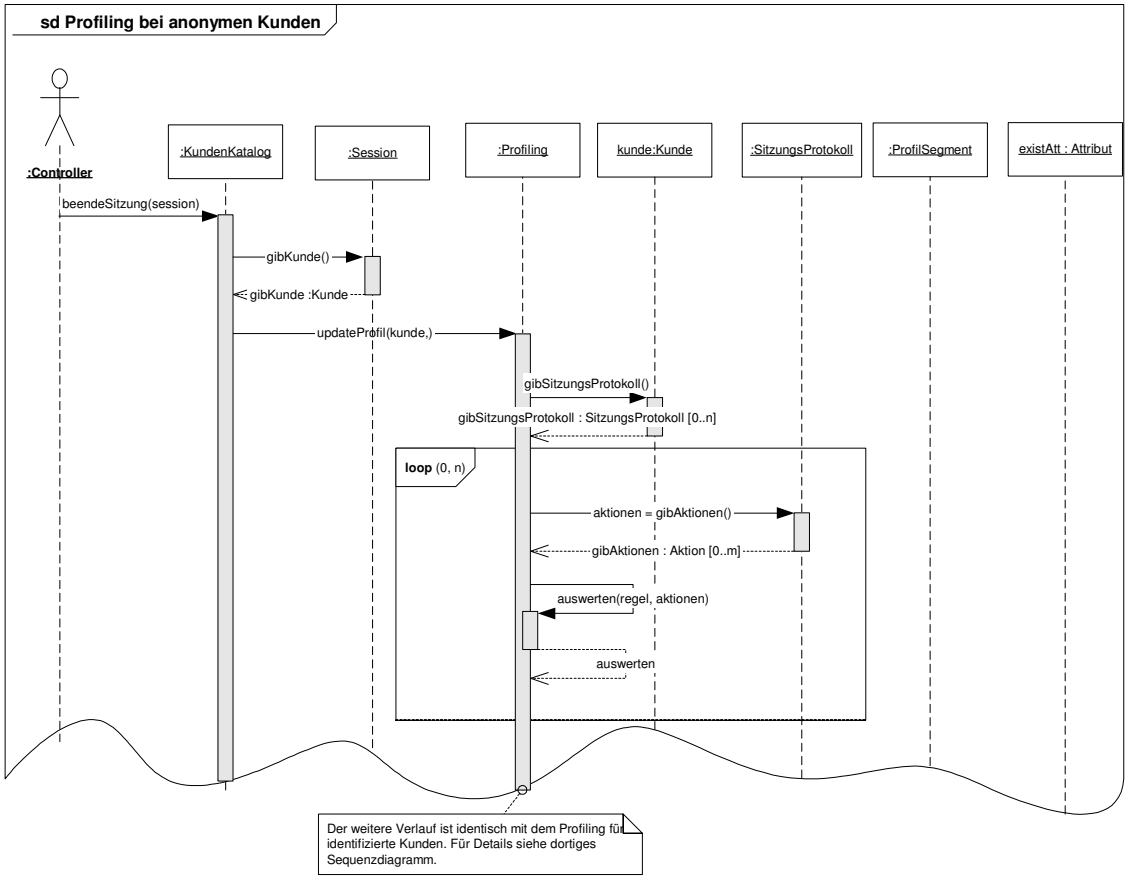


Abbildung 6-94: Sequenzdiagramm für das Profiling von anonymen Nutzern

Die Methode `updateProfil` fordert als erstes das `SitzungsProtokoll` des Kunden ab. Anders als in der vorherigen Variante wird hier kein Zeitraum als Parameter überliefert, so dass die Methode getSitzungsProtokoll sämtliche Einträge zurückliefert.

Der weitere Verlauf ist identisch mit dem Profiling für identifizierte Benutzer, wie es im Diagramm Abbildung 6-85 auf Seite 224 dargestellt wurde.

Es zeigt sich somit, dass die Anpassungen für die Variante der anonymen Nutzer nicht sehr umfangreich sind und ein Großteil der Architekturkomponenten wiederverwendet werden kann.

6.5 Referenzarchitektur für das Erlösmodell Provision

Es werden zwei Varianten des Erlösmodells PROVISION unterschieden. Es gibt Provisionen für eine anonyme und für personifizierte Vermittlungen. Beide Varianten werden in den folgenden Abschnitten separat betrachtet.

6.5.1 Variante der Vermittlung anonymer Kunden

Diese Variante für anonyme Kunden wurde in Abschnitt 4.2.5.1 auf Seite 84 detailliert vorgestellt.

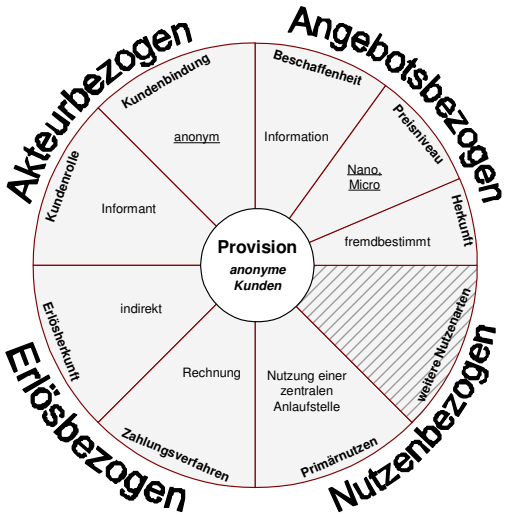


Abbildung 6-95: Erlösmodellklassifikation Provision Variante anonymer Kunde

Der folgende Abschnitt untersucht die zugehörigen Anwendungsfälle des Erlösmodells, bevor die Softwarearchitektur als Referenz vorgestellt wird.

6.5.1.1 Anforderungsanalyse und Systemabgrenzung

Da die Kunden anonym interagieren, muss in der Architektur keine Kundenverwaltung berücksichtigt werden. Das Erlösmodell basiert auf der Vermittlung von anonymen Kunden an externe Anbieter, die als Gegenleistung Provisonen zahlen. Deshalb sind zunächst die Anbieter zu verwalten.

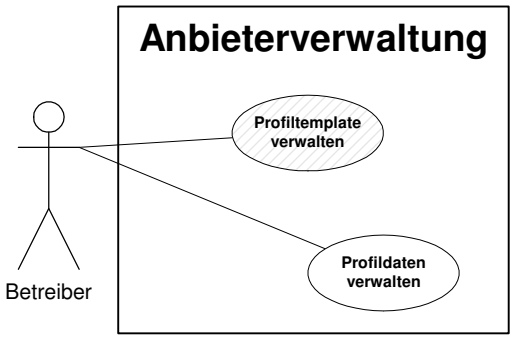


Abbildung 6-96: Anwendungsfälle der Anbieterverwaltung

Mit jedem Anbieter besteht eine vertragliche Vereinbarung, in der die Aufnahme des Angebotes auf der Plattform des Betreibers, die Art und Weise der Aktualisierung des Angebotes sowie die Provisionsvereinbarungen geregelt werden. Ein Anbieter erhält innerhalb des Systems ein Profil, dass seine wesentlichen Daten umfasst und die zu bezahlenden Vermittlungen als Grundlage der Rechnungsstellung festhält. Das grundlegende Profiltemplate ist somit zunächst vom Betreiber zu verwalten und mit Daten zu hinterlegen. Diese Profildaten sind fortwährend zu pflegen (Profildaten verwalten).

Die Produktverwaltung ist in diesem Erlösmodell in gewisser Weise vergleichbar mit dem Erlösmodell TRANSAKTIONSGEBÜHR. Das Angebot wird hier ebenfalls nicht vom Betreiber selber, sondern durch externe Anbieter bereitgestellt. Anders als im Erlösmodell Transaktionsgebühr werden hier jedoch die Produkte innerhalb des Produktkataloges durch den Betreiber festgelegt. Die relevanten Informationen wie beispielsweise Produktbeschreibungen, Produktpreis oder Verfügbarkeit werden dagegen von den Anbietern geliefert. Somit muss ihnen ein Zugang zu dem System eingeräumt werden, über den sie die Daten aktualiseren können. Der Betreiber beschränkt sich neben der Katalogverwaltung auf die Kontrolle der Daten und die Bereitstellung der Infrastruktur.

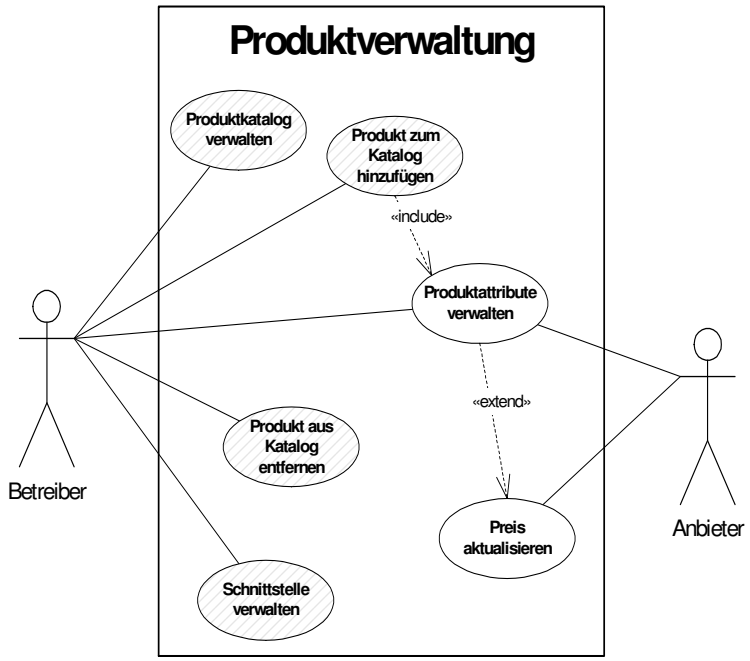


Abbildung 6-97: Anwendungsfälle der Produktverwaltung bei anonymer Vermittlungsprovision

In Abbildung 6-97 sind deshalb die Anwendungsfälle des Betreibers Produktkatalog verwalten, Produkt zum Katalog hinzufügen und Produkt aus Katalog entfernen aufgeführt. Sie verdeutlichen die verwaltende und kontrollierende Rolle des Betreibers hinsichtlich des Angebotes. Er ist auch als Akteur des Anwendungsfalls Produktattribute verwalten aufgeführt, jedoch beschränken sich seine Aufgaben in diesem Bereich ebenfalls auf die Betrachtung und Kontrolle der Daten. Der Anwendungsfall Verwaltung der Schnittstelle umfasst die Aufgaben des Betreibers hinsichtlich der Anbindung der Anbieter zur Aktualisierung ihres Produktangebotes.

Die Anbieter haben über diese Schnittstelle die Möglichkeit, die Produktattribute ihres Angebotes zu verwalten. Sofern es sich um Konsumgüter mit häufigen Preisschwankungen handelt, wird vor allem die Aktualisierung der Preise durchzuführen sein.

Neben der Anbieterverwaltung und der Produktverwaltung ist die Angebotsnutzung das dritte Szenario der Anforderungsanalyse. Es ist in der nächsten Abbildung 6-98 zu sehen.

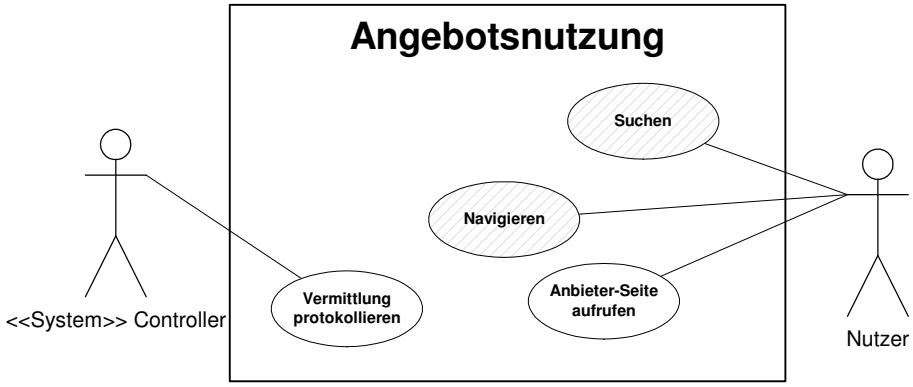


Abbildung 6-98: Anwendungsfälle der Angebotsnutzung bei anonymer Vermittlungsprovision

Die Angebotsnutzung durch die Kunden beschränkt sich vorrangig auf die Suche beziehungsweise Navigation nach Produkten sowie auf die Nutzung des Vermittlungsdienstes, indem sie die Anbieter-Seite zu einem selektierten Produkt aufrufen. Das System muss diese Vermittlung protokollieren, da es die Basis der Rechnungsstellung ist.

Die Rechnungsstellung wird an dieser Stelle nicht aufgeführt, da sie wie aus den bisherigen Erlösmodellen bekannt ist. Ein Anpassung erfolgt lediglich aus technischer Sicht dadurch, dass nun nicht die Nutzer, sondern die Anbieter der Produkte eine Rechnung erhalten, die die Vermittlungsprovisionen umfassen.

Zusammenfassend lassen sich die folgenden wesentlichen Anforderungen dieser Variante des Erlösmodells PROVISION an die Softwarearchitektur festhalten: eine Verwaltung der Anbieter, eine Produktverwaltung, die eine externe Manipulation der Produktattribute unterstützt, und eine Protokollierung der Vermittlungen als Basis der Rechnungsstellung. Im folgenden Abschnitt werden diese Anforderungen bei der Diskussion der Architektur berücksichtigt.

6.5.1.2 Architekturbeschreibung

Das Klassendiagramm Abbildung 6-99 zeigt eine Übersicht über die Klassen und Komponenten der statischen Architektur. Als besonderes Merkmal dieses Erlösmodells ist zunächst die Anforderung an eine Protokollierung der tatsächlichen Vermittlungen zu erkennen. Die Softwarearchitektur muss berücksichtigen, dass ein aktivierter Link zu einem externen Anbieter registriert werden kann, da dies die Grundlage der Erlösgenerierung darstellt. Gleich zwei Komponenten übernehmen diese Aufgabe: der `LinkController` zur Steuerung des Ablaufs und das `VerkaufsProtokollKonto` zur persistenten Speicherung und Ausgabe der Informationen.

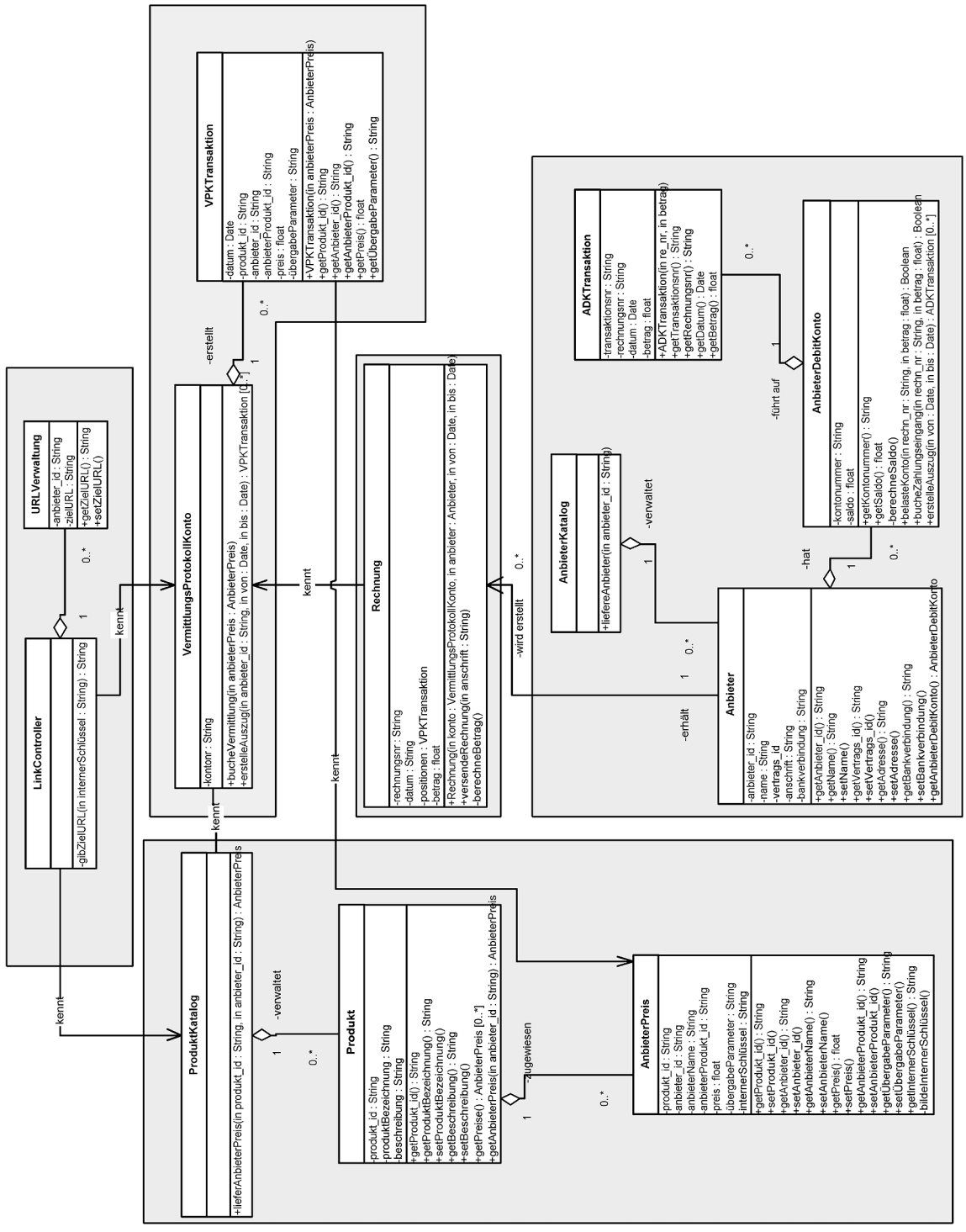


Abbildung 6-99: Statische Architektur für das Erlösmodell Vermittlungsprovision für anonyme Kunden

Eine weitere Auffälligkeit stellt die Produktverwaltung dar, die hier als *fremdbestimmt* charakterisiert worden ist. Die Besonderheit besteht darin, dass zu einem Produkt verschiedene Anbieter selbstständig ihre Preise und auch die Referenzen[19] verwalten müssen. Das Produkt mit seinen allgemeingültigen Stammdaten dagegen wird vom Betreiber des Geschäftsmodells eingerichtet. Es besteht somit eine *1:n* Relation zwischen (selbstbestimmtem) Produkt und (fremdbestimmten) Preisen und Referenzen.

Als weitere hervorzuhebende Anforderung müssen in diesem Erlösmodell keine Kunden, sondern externe Anbieter verwaltet werrden, denen periodisch die vermittelten Nutzer in Rechnung zu stellen sind. Auch diese Aufgabe wird von einer Komponente übernommen. Insgesamt umfasst die Architektur die Komponenten `Produkt`, `Anbieter`, `VermittlungsProtokollKonto`, `LinkController` sowie `Rechnung`, wobei alle in der Version `EM5.1_Prov` verwendet werden.

Bevor im Folgenden die Komponenten und ihre Funktionsweisen näher beschrieben werden, wird auf eine Designentscheidung der Architektur eingegangen, die im Klassendiagramm in Abbildung 6-99 zu erkennen ist. Die Realisierung der Protokollierung erfolgt über eine zentrale, von den Anbietern losgelöste Komponente. Da die Vermittlungen den externen Anbietern eindeutig zuzuweisen sein müssen (jeder Anbieter bezahlt nur für die Vermittlungen, die zu seiner Website führen und verlangt darüber detaillierte Nachweise), hätte man auch je Anbieter ein eigenes Protokollierungskonto einrichten können. Die hier gewählte Aufteilung hat jedoch den Vorteil, dass sämtliche Vermittlungen an zentraler Stelle durchgeführt werden und somit auch zentral zugreifbar und auswertbar sind. Erst die periodisch laufende Rechnungsstellung muss eine Zuteilung der Vermittlungen an die Anbieter durchführen. Eine Voraussetzung für diese zentrale Verwaltung der Vermittlungen ist jedoch, dass alle relevanten Details zum Zeitpunkt der Protokollierung mit gespeichert werden, die später in aufbereiteter Form an den externen Anbieter weitergereicht werden sollen.

Zunächst wird die Komponente zur Produktverwaltung detailliert betrachtet. Sie muss gewährleisten, dass der Betreiber in seinem Produktkatalog eine Menge von Produkten verwalten kann, dass jedoch zu jedem Produkt mehrere Anbieter jeweils abweichende Informationen hinterlegen können. Als wesentliches Unterscheidungsmerkmal dient der Preis des Produktes je Anbieter. Die Komponente `Produkte_EM5.1_Prov` kommt diesen Anforderungen nach, indem sie zu jedem Produkt eine Vielzahl von Anbieter-Preisen verwalten kann. Abbildung 6-100 zeigt die Komponente und ihre Klassen.

19. Als Referenz wird hier die URL zu dem Anbieter mitsamt notwendigen produktspezifischen Parametern bezeichnet.

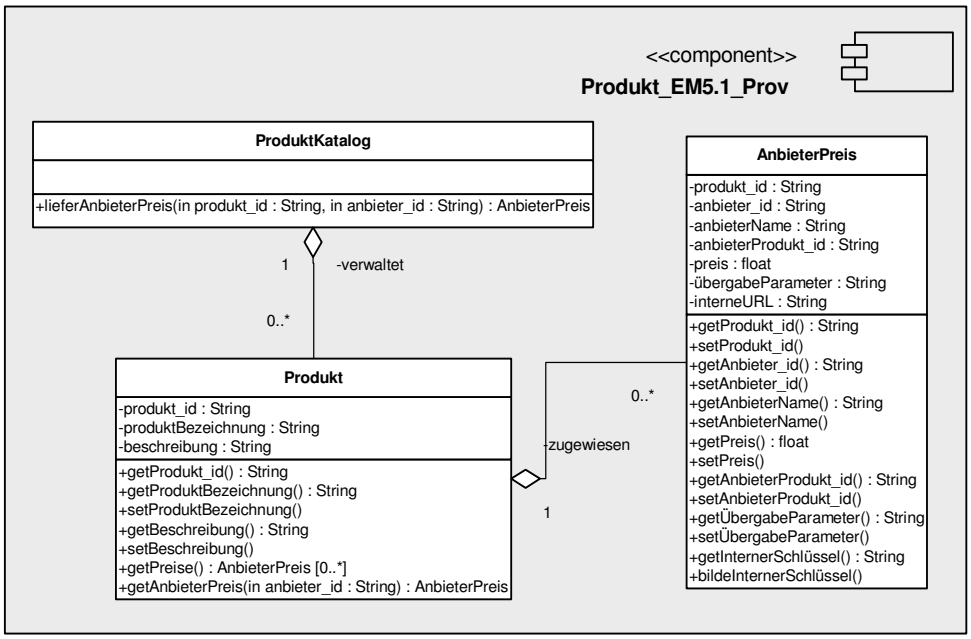


Abbildung 6-100: Komponente Produkt_EM5.1_Prov

Die Komponente umfasst die Klassen `ProduktKatalog`, `Produkt` und `AnbieterPreis`. Die Klasse `AnbieterPreis` beinhaltet dabei nicht ausschließlich den Preis, sondern auch weitere Informationen, jedoch ist dieses Attribut das wichtigste Unterscheidungskriterium für die Nutzer, so dass dieser Klassenname gewählt wurde. Durch die *1:n* Aggregation zwischen `Produkt` und `AnbieterPreis` wird sichergestellt, dass mehrere externe Anbieter ihre Preise zu einem vorgegebenen Produkt verwalten können.

Ein Produkt ist in dieser Referenzarchitektur ein standardisiertes Gut, so dass die gleichnamige Klasse über Attribute zur eindeutigen Benennung (`produkt_id` und `produktBezeichnung`) und zur Beschreibung (`beschreibung`) enthält. Neben den zugehörigen `get`- und `set`-Methoden verfügt die Klasse über die Methoden `getPreise` und `getAnbieterPreis`. In beiden Fällen werden Objekte der Klasse `AnbieterPreis` zurückgeliefert. Die Methode `getPreise` liefert jedoch sämtliche Instanzen dieser Klasse als Listobjekt zurück, während die Methode `getAnbeiterPreis` nur die Instanz liefert, deren Attribut `anbieter_id` identisch ist mit dem empfangenen Parameter.

Einem Produkt können beliebig viele Preise von verschiedenen Anbietern zugewiesen werden[20]. Die Klasse AnbieterPreis enthält neben dem Preis (`preis`) Angaben zu dem Anbieter (`anbieter_id`), dessen eigene Kennzeichnung des Produktes (`anbieterProdukt_id`) sowie weitere Parameter (`übergabeParameter`), die bei einer Vermittlung als Parameter an den Web-Server des Anbieters weitergeleitet werden sollen. Diese Parameter ermöglichen es dem Anbieter, den Ursprung des Nutzers zurückzuverfolgen und somit seinerseits eine

20. Es wird unterstellt, dass ein Anbieter immer genau einen Preis zu einem Produkt anbietet.

Protokollierung der Vermittlung vorzunehmen. Es wird hier unterstellt, dass das Attribut `übergabeParameter` auch immer die `anbieterProdukt_id` enthält, damit der Anbieter anhand der Request-Parameter weiß, welches Produkt der Kunde aufrufen möchte.

Die besondere Anforderung an die Durchführung einer Vermittlung besteht darin, dass die dargestellte URL nicht direkt auf die Website des Anbieters verweisen darf, da sonst keine Protokollierung einer getätigten Vermittlung möglich wäre; das System des Geschäftsmodellbetreibers würde den Aufruf der externen URL nicht erkennen können. Deshalb muss die Vermittlung zunächst über einen Umweg erfolgen, damit eine Protokollierung ermöglicht wird.

Für den Ablauf der Vermittlung ist vor allem das Attribut `internerSchlüssel` von Bedeutung, wie das Sequenzdiagramm in Abbildung 6-103 auf Seite 244 zeigt. Dieses Attribut stellt den Link dar, der innerhalb der GUI für dieses Produkt und diesen Anbieter eingebunden wird. Er enthält eine URL, die um die Attribute `produkt_id`, `anbieter_id` und `übergabe-Parameter` ergänzt ist. Diese URL, die zusätzlicher Bestandteil des Schlüssels ist, verweist intern auf die Adresse des `LinkControllers`, der weiter unten in diesem Abschnitt vorgestellt wird. Sie verweist nicht direkt auf die Website des Anbieters. Die Generierung des internen Schlüssels übernimmt die private Methode `bildeInternerSchlüssel`, die immer dann aufgerufen wird, wenn mindestens eine der eben erwähnten Attribute verändert wird. Der Aufruf erfolgt innerhalb der jeweiligen `set`-Methoden der drei Attribute.

Es wurde bereits erwähnt, dass zum Zeitpunkt der Vermittlung alle notwendigen Details über das vermittelte Produkt und seinen Anbieter protokolliert werden müssen. Die Informationen, die der auf der GUI dargestellte Link enthält, reichen dazu jedoch nicht aus. Um zum Zeitpunkt der Buchung an diese Details zu gelangen, die in der Klasse `Anbieter-Preis` hinterlegt sind, bietet die Klasse `ProduktKatalog` einen entsprechenden Service an. Er wird von der Komponente `LinkController` in Anspruch genommen, die als zentrale Steuereinheit für die Buchung zuständig ist und weiter unten beschreiben wird. Zu diesem Zweck stellt die Klasse `ProduktKatalog` die Methode `lieferAnbieter-Preis` bereit. Sie erhält als Parameter sowohl eine Produkt-ID und eine Anbieter-ID überliefert und ruft ihrerseits die Methode `getAnbieterPreis` des Produktobjektes auf, das mit der Produkt-ID übereinstimmt. Die Methode `getAnbieterPreis` erhält als Parameter die Anbieter-ID übergeben und liefert als Rückgabewert eine Referenz auf die Instanz der Klasse `AnbieterPreis`, deren Attribut `anbieter_id` mit diesem Wert übereinstimmt. Dieser Rückgabewert wird von der Methode `lieferAnbieterPreis` ebenfalls an das aufrufende Objekt weiter gereicht. Auch dieser Ablauf ist in dem Sequenzdiagramm in Abbildung 6-103 auf Seite 244 beschrieben.

Mit der Trennung von `Produkt` und `AnbieterPreis` ist die Voraussetzung dafür geschaffen, dass die Anbieter über eine zur Verfügung zu stellende Schnittstelle selbstständig die Produktattribute aktualisieren können. Neben dem Preis, der in der Klasse bereits berücksichtigt wurde, sind auch Angaben wie beispielsweise die Verfügbarkeit, anfallende Versandkosten oder Lieferdauer vorstellbar. Aufgrund des Referenzcharakters wurden diese spezifischen Angaben hier nicht mit aufgeführt. Eine konkrete Gestaltung der Schnittstelle wird hier nicht näher betrachtet, jedoch ist beispielsweise ein File-Import möglich, der Aktualisierungen zu den angebotenen Produkten des Anbieters umfasst.

Um die Protokollierung der Vermittlungen persistent speichern und auch wieder ausgeben zu können, ist ein entsprechendes Konto nötig. Die Komponente `VermittlungsProtokollKonto` stellt dieses bereit und ist in Abbildung 6-101 dargestellt. Wie bereits aus anderen Kontoverwaltungs-Komponenten bekannt ist, wird auch hier aus Revisionsgründen jede einzelne Transaktion festgehalten und als eine neue Instanz einer Klasse angelegt. Dazu besteht hier eine *1:n* Aggregation zwischen `VermittlungsProtokollKonto` und `VPKTransaktion`.

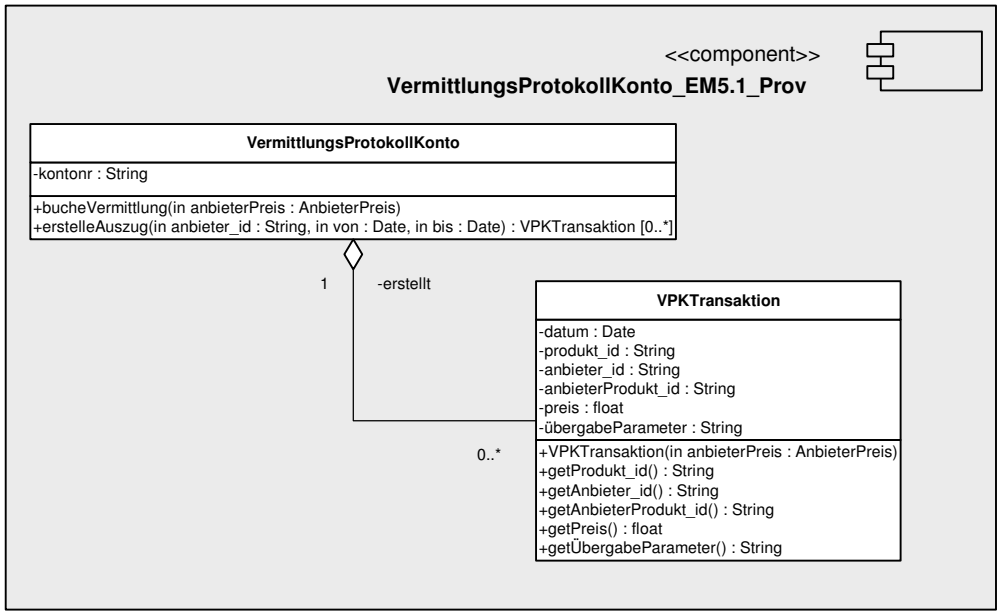


Abbildung 6-101: Komponente VermittlungsProtokollKonto_EM5.1_Prov

In dieser Architektur genügt ein zentrales Konto, das sämtliche Vermittlungen protokolliert. Um eine Zuordnung der Vermittlungen zu den Anbietern zu ermöglichen, hält jede einzelne Transaktion diese Information fest. Das Attribut `anbieter_id` der Klasse `VPKTransaktion` übernimmt diese Aufgabe. Die übrigen Attribute der Klasse sind selbsterklärend und wurden bereits bei der Vorstellung der Klasse `AnbieterPreis` besprochen. Dass die Werte der Attribute von der Klasse `VPKTransaktion` auch aus der Klasse `AnbieterPreis` stammen, wird im Sequenzdiagramm in Abbildung 6-103 ersichtlich.

Damit für jede Vermittlung eine neue Instanz von `VPKTransaktion` angelegt wird, steht die Methode `bucheVermittlung` aus der Klasse `VermittlungsProtokollKonto` zur Verfügung. Ihr wird eine Referenz auf `AnbieterPreis` übergeben, die sie an den Konstruktor `VPKTransaktion` weiterrreicht. Der Aufruf der Methode erfolgt von der Komponente `LinkController`, die nun betrachtet wird.

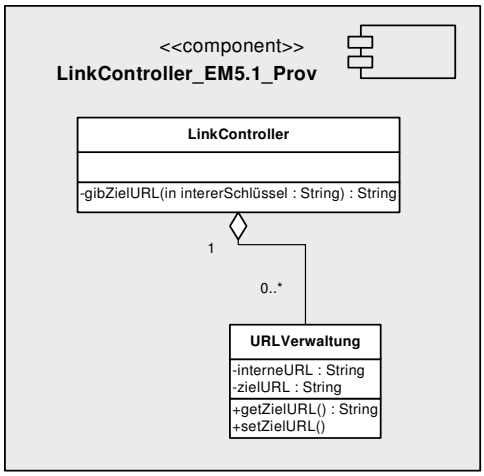


Abbildung 6-102: Komponente LinkController_EM5.1_Prov

Der `LinkController` ist dafür zuständig, die Vermittlung eines Nutzers zu einer externen Web-Adresse zu registrieren und durchzuführen. Es wurde bereits im Rahmen der Vorstellung des Attributs `internerSchlüssel` (Klasse `AnbieterPreis`) erwähnt, dass in der GUI als Link zu einem Produkt eines externen Anbieters nicht dessen direkte URL, sondern ein verschlüsselter String eingeblendet wird. Dieser Link führt dazu, dass die Klasse `LinkController` bei Aktivierung der URL aufgerufen wird.

Die Klasse `URLVerwaltung` enthält je Instanz ein Attributpaar `interneURL` / `zielURL`. Es können beliebig viele Instanzen zu einem `LinkController` angelegt werden, wie die Multiplizitäten der Assoziation zeigt. Diese Klasse übernimmt somit lediglich die Aufgabe, für jeden externen Anbieter dessen URL zu speichern und während der Vermittlung aufzuschlüsseln. Sollte sich die URL des Anbieters ändern, muss lediglich dieser Eintrag in dieser Klasse angepasst werden. Sämtliche Produkte, die der Anbieter beim Betreiber dieses Geschäftsmodells offeriert, bleiben dagegen von dieser Veränderung unberührt.

Sobald ein Nutzer über die GUI einen Link aktiviert, wird die Methode `giebZielURL` der Klasse `LinkController` aufgerufen und als Parameter der String `internerSchlüssel` übergeben. Diese Methode sorgt zunächst für die Zerlegung des überlieferten String in seine Einzelteile[21], so dass auch der Bestandteil der Anbieter-ID verfügbar ist. Mit dieser Information ermittelt `gibZielURL` anhand der zugehörigen Instanz (übereinstimmendes Attribut `anbieter_id`) von `URLVerwaltung` die Zieladresse (`zielURL`) des Anbieters.

Bevor `LinkController` einen Request an den Web-Server der Zieladresse sendet und als Parameter die `übergabeParameter` mitliefert, sorgt sie noch für die Buchung der Vermittlung auf dem `VermittlungsProtokollKonto`.

21. Die Einzelteile sind die Attribute `produkt_id`, `anbieter_id` und `übergabeParameter`, wobei der Übergabeparameter auch die Produktkennung des Anbieters (`anbieterProdukt_id`) beinhaltet und später als Request-Parameter an den Web-Server des Anbieters weitergereicht wird.

Das Zusammenspiel der zahlreichen involvierten Klassen während der Protokollierung einer Vermittlung wird am ehesten deutlich, wenn man das Sequenzdiagramm in der folgenden Abbildung 6-103 betrachtet.

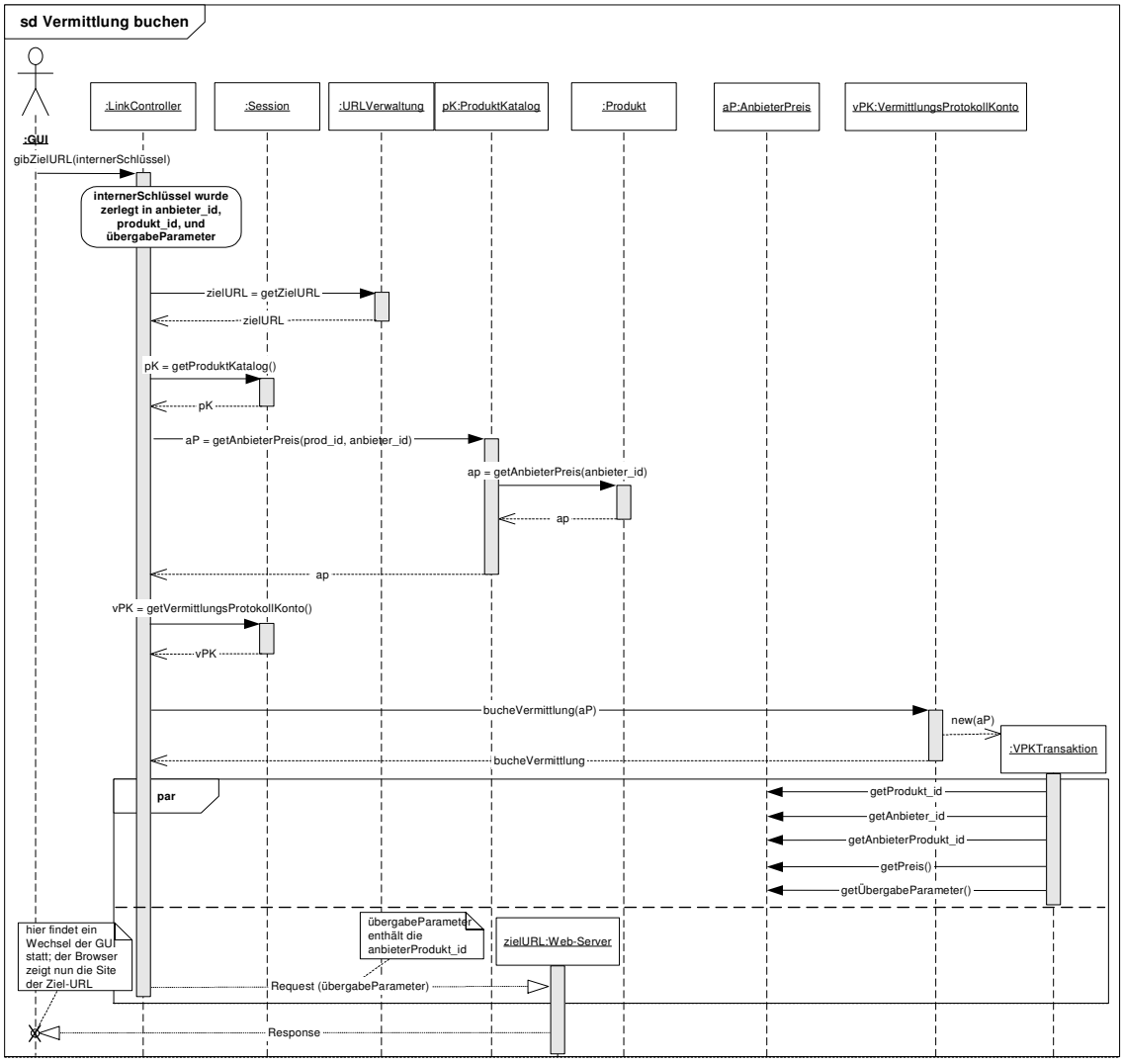


Abbildung 6-103: Sequenzdiagramm zur Buchung der Vermittlung

Es ist ersichtlich, dass `LinkController` zunächst die Bestandteile des Links (`interner-Schlüssel`) ermittelt. Als Bestandteile werden die `anbieter_id`, `produkt_id` sowie die `übergabeParameter` identifiziert. Unter Verwendung der Anbieter-ID wird die URL des Anbieters bestimmt, zu der später eine Verbindung hergestellt wird (am unteren Ende des Diagramms ersichtlich)[22].

22. Es wird im Diagramm darauf hingewiesen, dass die GUI mit dem Response des Anbieters wechselt, da nun die Website dieses Anbieters angezeigt wird.

Um die Vermittlung zu buchen, muss die Methode `bucheVermittlung` der Klasse `VermittlungsProtokollKonto` aufgerufen werden. Als Parameter erwartet die Methode eine Referenz auf die Klasse `AnbieterPreis`, so dass diese herausgefunden werden muss.

Dazu wird über die Session vom `LinkController` zunächst der `ProduktKatalog` ermittelt (getProduktKatalog), dessen Methode `getAnbieterPreis` diese Referenz zürückliefert. Die notwendigen Parameter liegen bereits entschlüsselt vor und können somit übergeben werden. Nach der Ermittlung des Kontos (`getVermittlungsProtokollKonto`) über die `Session` kann die Methode zur Buchung der Vermittlung aufgerufen werden.

Diese Methode erzeugt eine neue Instanz von `VPKTransaktion`. Der Konstruktor erhält die Referenz auf `AnbieterPreis`, von wo er die erforderlichen Attributwerte anfordert.

Bei der Weiterleitung des Links an den Web-Server des Anbieters übergibt der `LinkController` abschließend die `übergabeParameter`.

Nachdem die Schritte zur Protokollierung der Vermittlungen gezeigt wurden, sind noch die Komponenten der Anbieterverwaltung und der Rechnungsstellung zu betrachten.

Die Komponente Anbieter_EM5.1_Prov ist in der folgenden Abbildung 6-104 mitsamt ihren Klassen zu sehen.

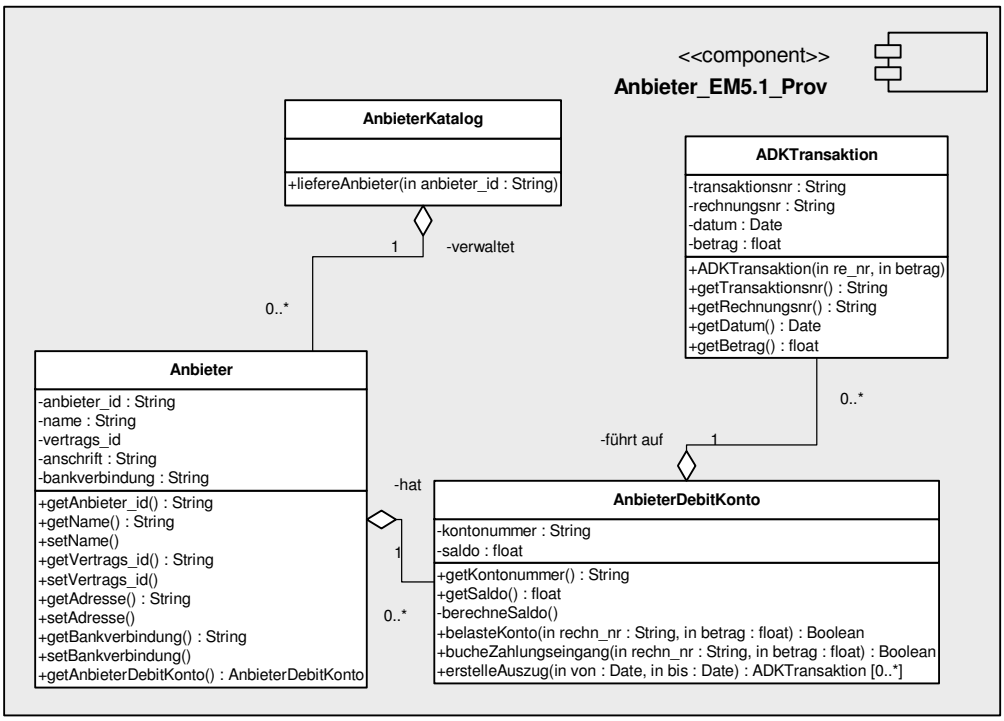


Abbildung 6-104: Komponente Anbieter_EM5.1_Prov

Es ist ersichtlich, dass sie in Anlehnung an bereits verwendete Komponenten zur Kundenverwaltung erstellt wurde. Da es sich bei diesem Erlösmodell um eine externe und nicht interne Erlösherkunft handelt, wurde sie jedoch auf die externen Anbieter angepasst. Die Komponente umfasst die Klassen `AnbieterKatalog`, `Anbieter`, `AnbieterDebitKonto` und `ADKTransaktion`. Bereits die Aufteilung und Namensgebung der Klassen zeigt an, dass diese Komponente sehr stark angelehnt wurde an die bisher bekannten Komponenten der Nutzerverwaltung. Der `AnbieterKatalog` verwaltet die Menge der `Anbieter`, die im System eingerichtet worden sind. Die Belastung des Kontos der Anbieter sowie die Verbuchung der Zahlungseingänge erfolgt mit den Klassen `AnbieterDebitKonto` und `ADKTransaktion`.

Aufgrund der starken Übereinstimmung mit den bereits bekannten Komponenten wird hier auf eine detailliertere Behandlung der Klassen verzichtet. Als Referenz kann bei Bedarf die Beschreibung der Komponente `Kunde_EM1.1_ET` in Abschnitt 6.1.1.2 ab Seite 133 herangezogen werden.

Für die Erstellung der Rechnungen ist die Komponente Rechnung_EM5.1_Prov zuständig, die in Abbildung 6-105 zu sehen ist. Die wesentliche Anforderung an diese Komponente ist die Zuweisung der geleisteten Vermittlungen aus dem Vermittlungsprotokoll-Konto zu den einzelnen externen Anbietern sowie die Kalkulation und Buchung des entsprechenden Rechnungsbetrages.

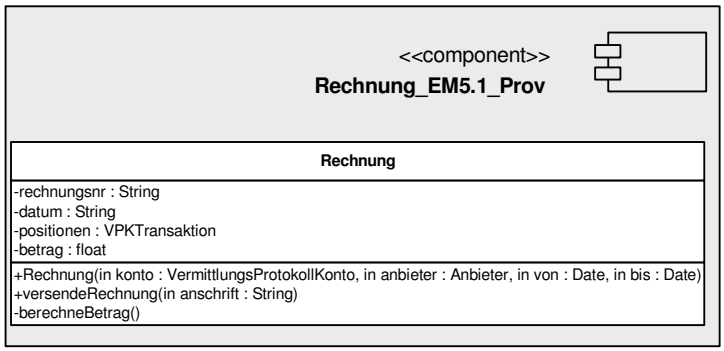


Abbildung 6-105: Komponente Rechnung_EM5.1_Prov

Dem Konstruktor `Rechnung` werden als Parameter Referenzen auf das `VermittlungsProtokollKonto` und auf den `Anbieter` sowie der abzurechnende Zeitraum übergeben. Der Ablauf der Rechnunsstellung und das Zusammenspiel der Klassen ist im folgenden Interaktionsdiagramm ersichtlich.

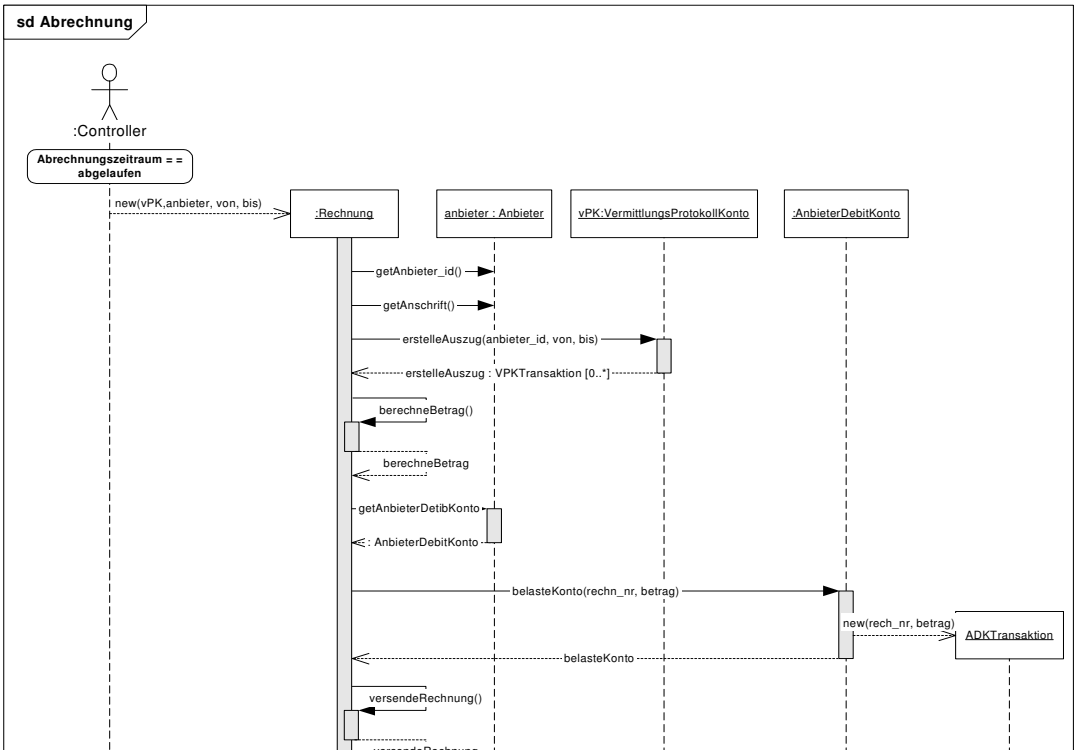


Abbildung 6-106: Abrechnung der Vermittlungsprovision

Die Methodenaufrufe des Konstruktors sind weitgehend aus den bisherigen Architekturbeschreibungen bekannt. Anhand des Anbieters wird dessen eindeutige Kennung (`anbieter_id`) ermittelt, damit das `VermittlungsProtokollKonto` die Transaktionen herausfiltern kann, die zu diesem Anbieter gehören. Diese Filterung ist hier notwendig, da es nur ein systemweites Protokoll-Konto gibt und nicht ein Konto je Anbieter. Die Methode `erstelleAuszug` liefert die geforderten Transaktionen innerhalb des abzurechnenden Zeitraums. Die Ermittlung des Rechnungsbetrages (`berechneBetrag`) kann hier nicht allgemeingültig wiedergegeben werden, jedoch wird grundsätzlich die Anzahl der protokollierten Vermittlungen sowie die vertraglich vereinbarte Provision je Vermittlung berücksichtigt werden.

Die Belastung des Debitoren-Kontos des Anbieters und die Versendung der Rechnung schließen die Rechnungsstellung ab.

6.5.2 Variante der Vermittlung identifizierter Kunden

Abschnitt 4.2.5.2 auf Seite 85 behandelt diese Erlösmodellvariante ausführlich.

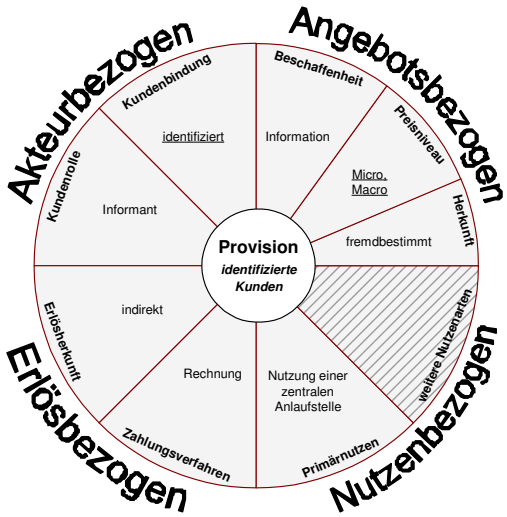


Abbildung 6-107: Erlösmodellklassifikation Provision Variante identifizierter Kunde

Diese Variante ist technisch betrachtet eng verbunden mit dem Erlösmodell PROFILHANDEL. Das Angebot, also die Produkte, werden verwendet, um Informationen über den Kunden zu erhalten. Auf diese Weise wird dessen Profil erzeugt, das das Unternehmen an seine Partner, die Anbieter der Produkte, weiterleitet und als Gegenleistung eine Vermittlungsprovision erhält.

6.5.2.1 Anforderungsanalyse und Systemabgrenzung

Abweichend zur Variante der anonymen Vermittlung gibt es hier eine Nutzerverwaltung, mittels der man die Profile der Nutzer einrichtet und pflegt. Sie ist identisch mit der bereits vorgestellten Nutzerverwaltung für identifizierte Kunden aus dem Erlösmodell PROFILHANDEL. Das Anwendungsfalldiagramm wurde bereits in Abbildung 6-76 auf Seite 210 vorgestellt und wird hier noch einmal abgebildet.

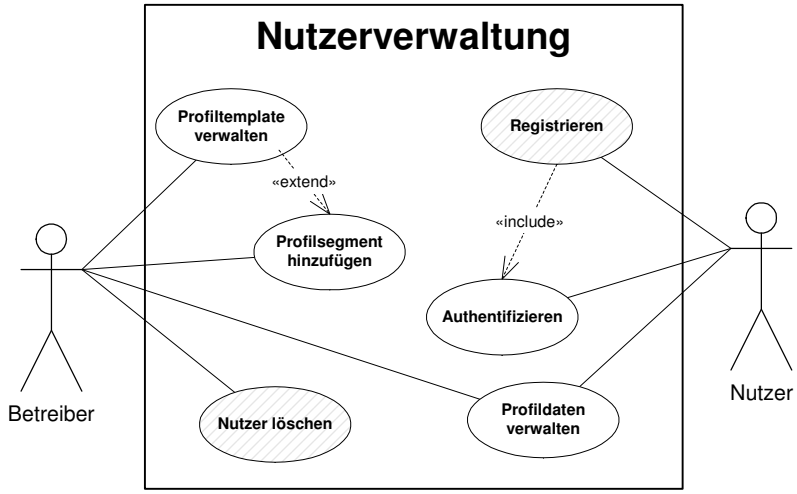


Abbildung 6-108: Anwendungsfälle der Nutzerverwaltung

Die Produktverwaltung weicht von der vorherigen Variante ab. Als Referenz wird unterstellt, dass die Produkte nicht tagesaktuell verändert werden, sondern eher selten zu aktualisieren sind. Das Beispiel einer Versicherungspolice verdeutlicht dies. Der Preis ist kein isoliertes Attribut des Produktes, sondern das Ergebnis einer komplexen Preisberechnung, die zahlreiche Attributwerte mit einbezieht. Das Produkt besteht somit im Wesentlichen aus einem Rechenkern, der vom Anbieter zur Verfügung gestellt und bei Bedarf angepasst wird. Der Betreiber des Geschäftsmodells verwaltet die grundlegenden Produktkategorien, zu denen er jeweils mehrere Produkte unterschiedlicher Anbieter aufnimmt. Darüber hinaus ist sicherzustellen, dass die notwendigen Daten für die Produkte und deren Preisberechnung vorhanden sind. Diese werden innerhalb der Geschäftslogik implementiert und stellen sich dem Kunden gegenüber als Fragen dar, die er zu beantworten hat. Das folgende Diagramm zeigt die Anwendungsfälle in diesem Zusammenhang.

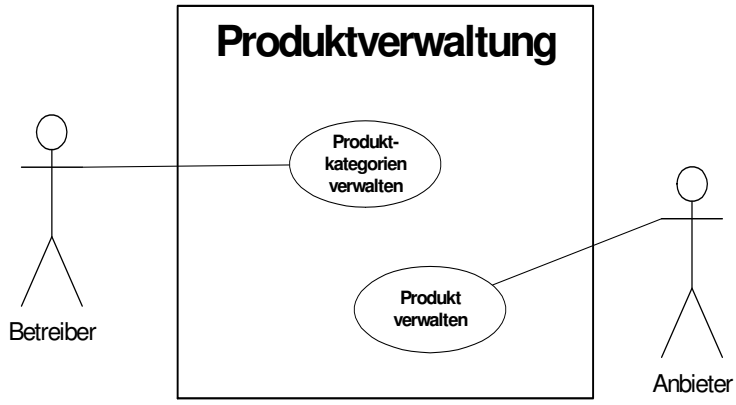


Abbildung 6-109: Anwendungsfälle der Produktverwaltung

Der Anwendungsfall Produkt verwalten umfasst die Bereitstellung der Rechenkerne aus Sicht der Anbieter. Der Betreiber beschränkt sich dagegen auf die Verwaltung der Produktkategorien, die auch die produktrelevanten Fragen beinhalten.

Die Angebotsnutzung wird in Abbildung 6-110 dargestellt.

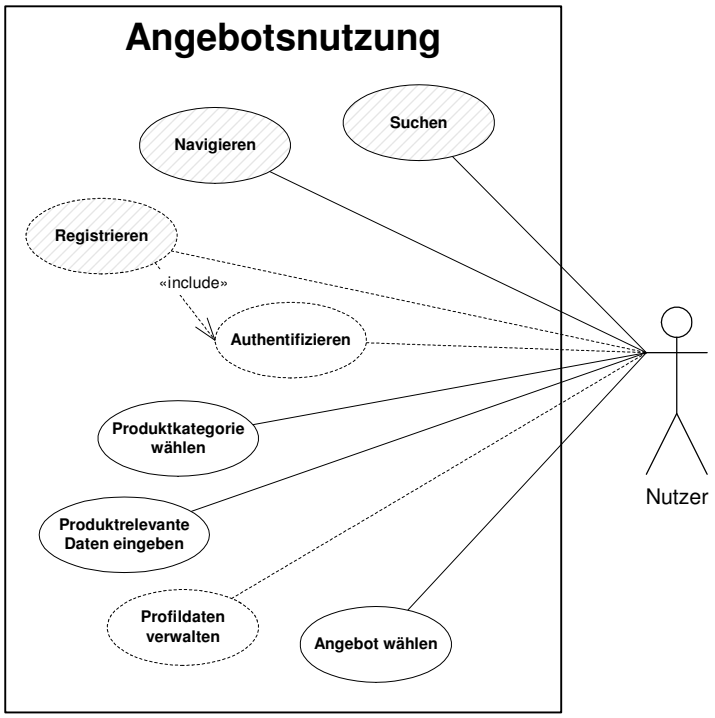


Abbildung 6-110: Anwendungsfälle der Angebotsnutzung

Neben den bereits bekannten Anwendungsfällen treten in dieser Variante des Erlösmodells PROVISION einige sehr spezifische auf, die der Nutzer ausübt. Zunächst entscheidet er sich für eine bestimmte Produktkategorie, die er aus dem Gesamtangebot wählt. Dies kann beispielsweise ein bestimmter Typ von Versicherung wie Lebens- oder Unfallversicherung sein. Zu jedem Typ sind im weiteren Verlauf der Interaktion die zugehörigen relevanten Fragestellungen zu beantworten. Der Nutzer gibt also die produktrelevanten Daten ein. Innerhalb dieser Beantwortung der Fragen kann es sein, dass persönliche Daten erhoben werden, die seinem (persönlichen) Profil zuzuordnen sind. Deshalb wird der Anwendungsfall Profildaten verwalten hier als Referenz aufgeführt. Er wurde hauptsächlich dem Szenario der Nutzerverwaltung zugeordnet. Als Ergebnis seiner Angaben erhält der Kunde eine Übersicht über die verschiedenen Produktanbieter, die sich durch ihr Leistungsspektrum sowie über den Preis voneinander unterscheiden. Aus diesem Angebot kann der Nutzer nun seine Präferenz wählen.

Es wird im Rahmen der Architekturgestaltung nicht näher untersucht, wie der weitere Verlauf der Interaktion gestaltet ist. Mögliche Varianten sind, dass der Kunde im Anschluss an seine Wahl direkt auf die Website des Anbieters geleitet wird, oder aber dass das Angebot (mit oder ohne der online erfassten Daten) in aufbereiteteter Form zum Ausdruck angebo-

ten wird, oder aber lediglich ein Hinweis erfolgt, dass sich der Anbieter mit dem Kunden in Verbindung setzen wird und ihm die Informationen zugestellt werden. Aus Sicht des Erlösmodells sind zu diesem Zeitpunkt die relevanten Informationen verfügbar, um ein Profil des Kunden zu erstellen, dieses an den oder die Anbieter weiterzuleiten und die Vermittlungsprovision zu berechnen.

Das Profiling ist ähnlich wie im Erlösmodell PROFILHANDEL, allerdings gibt es hier kein *Logging*, kein *implizites Profiling* und keine *Gruppenprofile*. Abbildung 6-111 zeigt die zugehörigen Anwendungsfälle.

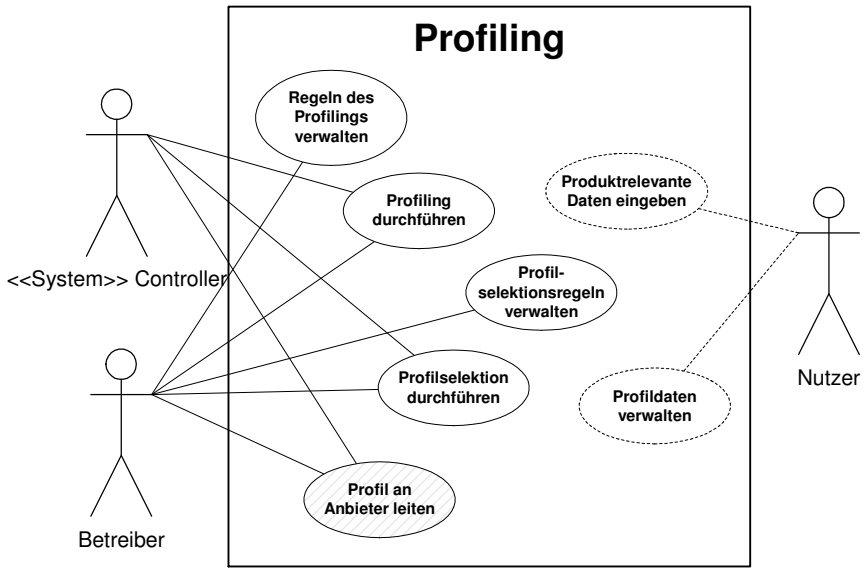


Abbildung 6-111: Anwendungsfälle zur Profilbildung

Die Überschneidung der Szenarien Profiling und Angebotsnutzung zeigt sich auch hier, da die wesentlichen Aktionen des Nutzers innerhalb der Angebotsnutzung natürlich auch die Basis des Profilings bilden und deshalb hier als Referenzen noch einmal aufgeführt wurden.

Ausgehend von dieser Datenbasis kann das System selbstständig das Profiling durchführen. Gemäß der Regeln, die der Betreiber zunächst einstellen muss (Regeln des Profilings verwalten), werden die eingegebenen Daten des Anwenders analysiert und in seinem Profil gespeichert. Wie bereits erwähnt, findet dabei in der Regel keine implizite Erhebung statt, sondern es werden die Antworten des Kunden auf die ihm gestellten produktspezifischen Daten direkt zugehörigen Attributen zugewiesen. Die Selektion der Profile (Profilselektion durchführen) führt dazu, dass die erhobenen Daten eines Anwenders in aufbereiteter Form (gemäß der Profilselektionsregeln, die der Betreiber zu verwalten hat) vorliegen und dem Anbieter zugestellt werden können (Profil an Anbieter leiten).

Die in diesem Abschnitt erhobenen Anforderungen werden in der folgenden Architektur berücksichtigt.

6.5.2.2 Architekturbeschreibung

Diese Variante des Erlösmodells PROVISION ist grundsätzlich vorstellbar für Geschäftsmodelle, die Produkte vermitteln,...

- ...die die Eingabe individueller Daten verlangen,
- ...zu denen mehrere Anbieter existieren,
- ...die sich in großen Bereichen des Leistungsspektrums ähneln oder gar übereinstimmen,
- ...und die im Detail und im Preis je nach Anbieter variieren.

Somit wird die Referenzarchitektur hier am Beispiel einer Versicherung betrachtet. Die Produktverwaltung wird also auf diese Anforderungen hin angepasst und stellt die wesentliche Neuerung dieser Erlösmodellvariante dar. Je nach konkretem Anwendungsfall muss sie entsprechend umgestaltet werden.

Darüber hinaus ähnelt diese Variante des Erlösmodells PROVISION und somit auch die Software-Referenzarchitektur weitgehend dem Erlösmodell PROFILHANDEL, da hier identifizierte Nutzerprofile angelegt und gegen eine Vermittlungsprovision an Geschäftskunden weitergereicht werden. Der Unterschied liegt vor allem darin, dass sich das Profil hier ausschließlich aus den produktspezifischen Angaben eines Nutzers generiert und nicht etwa aus der Interpretation seines Verhaltens. Somit können einige Komponenten übernommen werden. Die Anbieter der Produkte werden mit der Komponente `Anbieter` aus der vorherigen Variante dieses Erlösmodells abgebildet; ebenso kann die `Rechnung` wiederverwendet werden. Das Vermittlungsprotokoll-Konto der vorherigen Variante konnte als Vorlage verwendet werden, jedoch musste es auf die hier vorliegenden Rahmenbedingungen angepasst werden. Das folgende Diagramm zeigt als Übersicht die Klassen und Komponenten der Architektur.

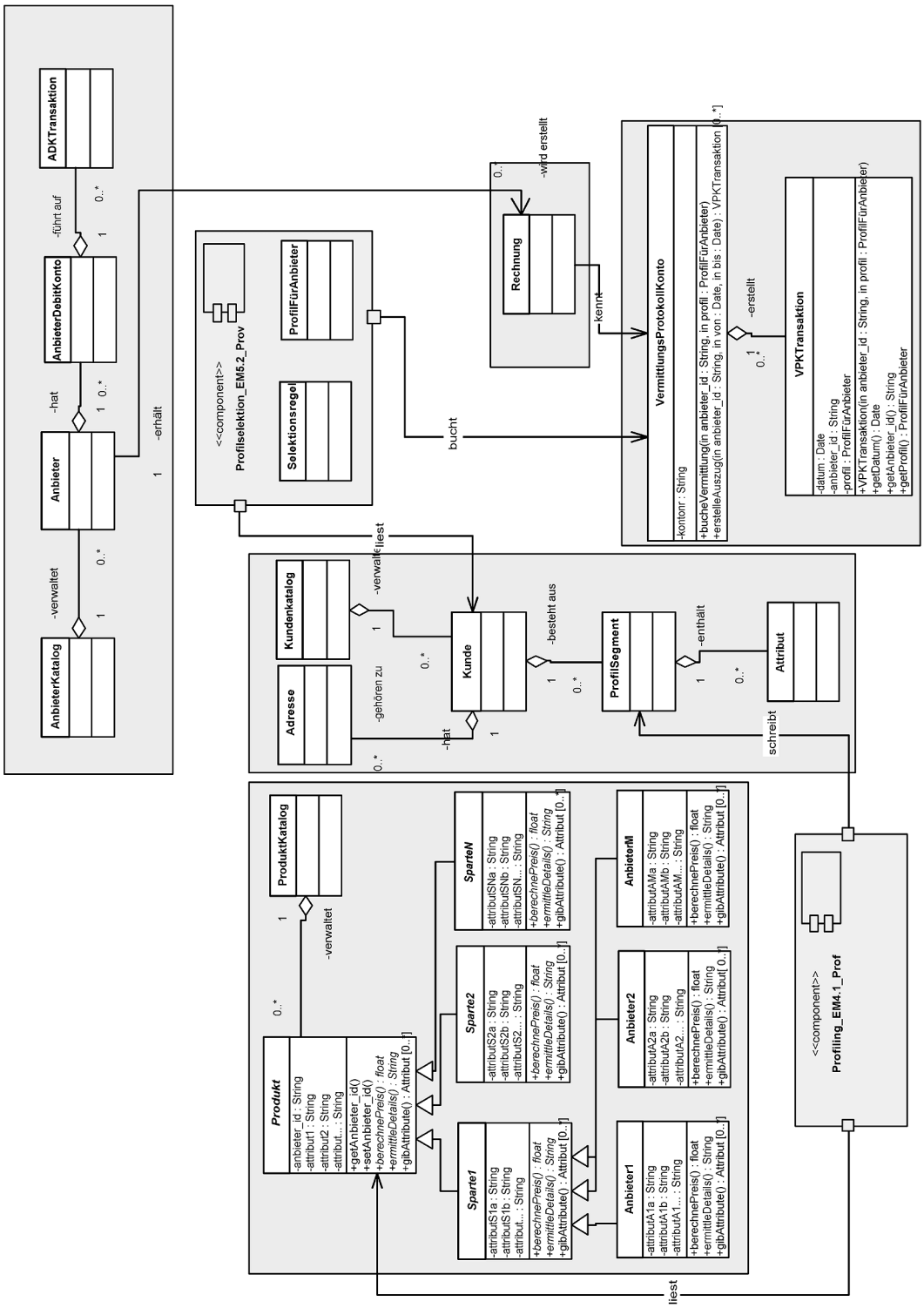


Abbildung 6-112: Statische Architektur des Erlösmodells Provision für die Vermittlung identifizierter Kunden

Die statische Architekturbeschreibung in Abbildung 6-112 zeigt die Komponenten Produkt, Kunde, Profiling, Profilselektion, Rechnung, VermittlungsProtokollKonto und Anbieter. Einige dieser Komponenten sind bereits in vorherigen Erlösmodellarchitekturen behandelt worden und können hier wiederverwendet werden. Die Komponenten für die Kundenverwaltung und für das Profiling sind aus dem Erlösmodell PROFILHANDEL übernommen worden und somit jeweils in dieser Version eingesetzt. Eine Beschreibung ist im Abschnitt 6.4.1.2 zu finden, wobei die Komponente Kunde_EM4.1_Prof auf Seite 216 und Profiling_EM4.1_Prof auf Seite 222 beschrieben werden.

Die Verwaltung der Anbieter sowie die Komponente zur Rechnungsstellung werden aus der vorherigen Variante übernommen und in den Versionen Anbieter_EM5.1_Prov (siehe Abbildung 6-104 auf Seite 245) und Rechnung_EM5.1_Prov (siehe Abbildung 6-105 auf Seite 246) wiederverwendet.

Die Komponente Produkt_EM6.2_Prov für die Verwaltung der Produkte ist dagegen wie bereits angekündigt neu. Die folgende Abbildung 6-113 zeigt als Referenz ein Klassendiagramm für die Realisierung von Versicherungspolicen. Die Produkte werden hier durch eine zweifache Spezialisierung abgebildet. Neben der abstrakten Klasse Produkt gibt es ja nach Sparte weitere spezialisierte (abstrakte) Klassen, die ihrerseits eine konkrete Klasse für jeden Anbieter als Spezialisierung aufweist. Exemplarisch zeigt die Komponente drei Sparten sowei drei Anbieter der Sparte1 auf.

Je nach Ebene werden dabei die Attribute angeordnet, die entweder allgemeingültig über alle Sparten, allgemeingültig über alle Anbieter oder aber anbieterspezifisch sind. Anhand des Beispiels der Versicherungspolicen stellen die Fragen zur Ermittlung der Preise und des Leistungsumfangs die Attribute dar.

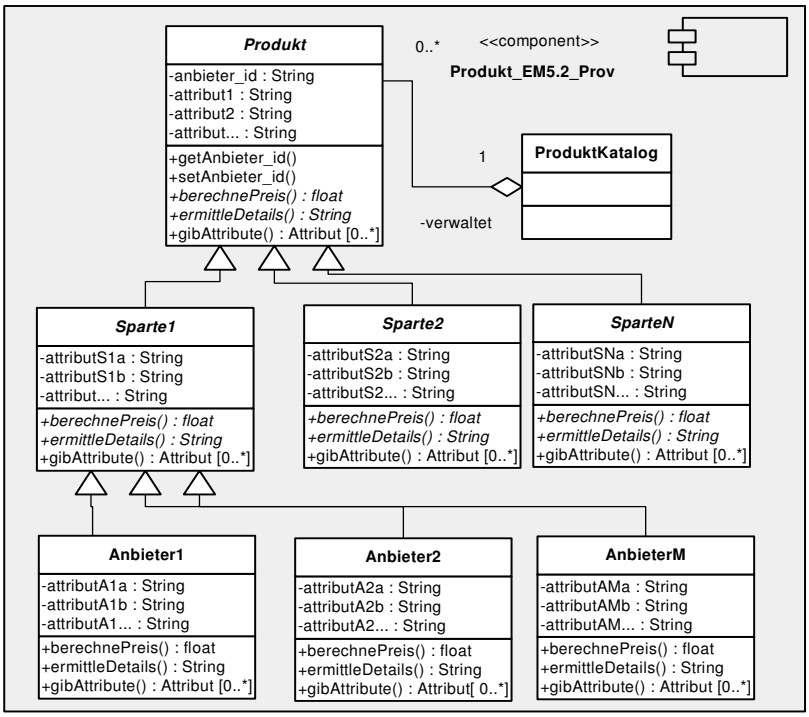


Abbildung 6-113: Komponente Produkt_EM5.2_Prov

Ein Beispiel verdeutlicht die Verwendung der Attribute auf den unterschiedlichen Ebenen.

Ein Versicherungsportal bietet als Dienst die Identifizierung der optimalen Versicherungspolicen für seine Kunden an. Er bietet die Versicherungstypen (Sparten) KFZ, Risikoleben, Unfall, Krankenzusatz und Berufsunfähigkeit an. Für die KFZ-Versicherungen hat er als Anbieter vier verschiedene Unternehmen in seinem Portfolio.

Die Attribute, die zur Berechnung der Preise und der Leistungsdetails benötigt werden, sind die Fragen der Versicherungsunternehmen. Unabhängig von der Sparte wird jede Police die Angabe des Alters erfordern. Nur die KFZ-Policen benötigen dagegen den Fahrzeugtyp und nur ein Anbieter bietet einen Sondertarif für Fahrzeuge, die in Garagen abgestellt werden.

Entsprechend der Modellierung wäre das Alter des Kunden ein Attribut der Klasse Produkt, der Fahrzeugtyp ein Attribut der Klasse Sparte und die Abfrage bezüglich des Abstellungsortes ein Attribut der Klasse Anbieter.

Zu besseren Übersicht wurde auf die `get`- und `Set`-Methoden der einzelnen Attribute verzichtet. Neben den Attributen existieren die Methoden `berechnePreis` sowie `ermittleDetails`, die den eigentlichen Rechenkern der Produkte darstellen.

Darüber hinaus steht die polymorphe Methode `gibAttribute` in der Klasse `Produkt` sowie in den Spezialisierungen zur Verfügung. Die Methode wird verwendet, um der Profiling-Komponente sämtliche Attribute zu liefern, die diese für die Profilbildung benötigt. Als Rückgabewert wird ein Listobjekt vom Typ `Attribut` geliefert, dass jeweils ein Schlüssel-Wert-Paar sowie eine `attributArt` enthält. Dieser wird von der Methode als konstanter Wert *explizit* belegt, da die Daten direkt vom Kunden angegeben wurden. Für eine detaillierte Beschreibung der Klasse innerhalb der wiederverwendeten Komponente `Kunde_EM4.1_Prof` siehe Seite 216.

Diese Komponente ist sehr stark an das hier verwendete Szenario von Versicherungspolicen angepasst worden. Grundsätzlich gilt jedoch für Geschäftsmodelle, die diese Variante des betrachteten Erlösmodells anwenden, dass die Produkte die Abfrage einer Reihe von Attributen mit sich bringen. Diese Daten werden für die Profilbildung genutzt und stellen somit die Basis der Provisionszahlungen dar. Dies bedeutet, dass auch andere konkrete Anwendungsfälle außerhalb von Versicherungspolicen eine ähnliche Struktur der Produktkomponente aufweisen werden. Diese kann in der Hierarchie der Produktgestaltung sowie in den konkreten Methoden abweichen, jedoch wird sie grundsätzliche Attribute enthalten, die für die Profilbildung herangezogen werden. Die Komponente, die die Profilbildung durchführt, wird nun vorgestellt.

Wie bereits in der Anforderungsanalyse beschrieben, erfolgt das Profiling in diesem Erlösmodell ausschließlich auf explizit erhobenen Daten (die Angaben des Nutzers gemäß der zu erhebenen Attributwerte), die in den Profilsegmenten (Klasse `ProfilSegment`) des Nutzers gespeichert werden. Die Komponente `Profiling` ist für diese Generierung der

Profile zuständig, indem sie die Informationen (Attribute und Attributwerte) über die Methode `gibAttribute` von der Klasse `Produkt` anfordert und in den Profilsegmenten speichert.

Die Selektion der Profile zur Weitergabe an die Anbieter wird auf dieser Basis von der Komponente `Profilselektion_EM5.2_Prov` übernommen.

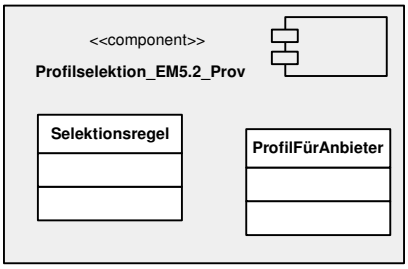


Abbildung 6-114: Komponente Profilselektion EM5.2_Prov

Wie bereits die Version aus der vorherigen Variante wird auch diese Komponente hier nur ansatzweise dargestellt. Die Komponente verfügt über `Selektionsregeln`, die Einstellungen erlauben über die zu filternden Daten und die Art der Aufbereitung. Die Ergebnisse stellen die Profile dar, die an die Anbieter übermittelt werden können. Die Klasse `ProfilFürAnbieter` deutet diese an.

Ebenfalls angepasst wurde die Komponente zur Buchung der Vermittlungen, `VermittlungsProtokollKonto_EM5`.2_Prov, die in der folgenden Darstellung zu sehen ist.

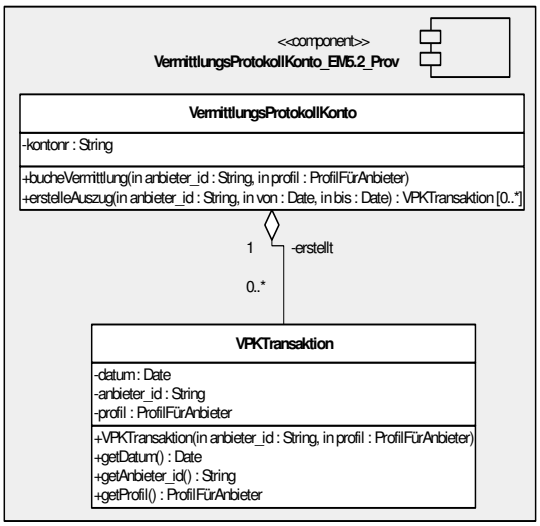


Abbildung 6-115: Komponente VermittlungsProtokollKonto_EM5.2_Prov

Das Erlösmodell basiert in dieser Variante auf der Vermittlung der Kundenprofile. Somit muss die Weitergabe der Profile an den Anbieter verbucht werden. Die Erstellung und Weitergabe wird von der Komponente `Profilselektion` übernommen. Zur Protokollierung dieser Aktion ruft sie die Methode `bucheVermittlung` des `VermittlungsProtokollKontos` auf. Als Parameter werden dabei eine Referenz auf den Anbieter sowie auf das Profil übergeben. Die Methode ruft ihrerseits den Konstruktor der Klasse `VPKTransaktion` auf und leitet die erhaltenen Parameter weiter. Der Konstruktor weist die Werte den Attributen zu und speichert zusätzlich einen Zeitstempel der Aktion. Auf diese Weise können die Rechnungen für jeden Anbieter erstellt werden.

Der Ablauf der Rechnungsstellung ist identisch mit der vorherigen Variante, wie im Sequenzdiagramm in Abbildung 6-106 auf Seite 247 gezeigt wurde. Es ist lediglich zu beachten, dass die Methode `berechneBetrag` von `Rechnung` in dieser Variante nicht die URL-Aufrufe, sondern die vermittelten Profile berücksichtigt, die in den Transaktionen verbucht wurden. Trotz dieser Anpassung wird jedoch auf der hier betrachteten Ebene die Komponente zur Rechnungsstellung unverändert wiederverwendet.

6.6 Referenzarchitektur für das Erlösmodell Werbung

Die Werbung als solche lässt sich unterteilen in die Vermietung des Werbeplatzes, der durch die Häufigkeit der geschalteten Werbebanner berechnet wird (AdImpression), sowie in den Verkauf der Klicks auf die Werbebanner (AdClick).

6.6.1 AdImpressions und AdClicks

Da beide Varianten durch den Einsatz eines AdServers technologisch umgesetzt werden können, werden hier beide gemeinsam in einem Abschnitt betrachtet.

Abbildung 6-116: Erlösmodell Werbung

In Abschnitt 4.2.6 auf Seite 87 wurde bereits darauf eingegangen, dass die Umsetzung dieses Erlösmodells den Einsatz eines AdServers benötigt, der bereits als fertiges Produkt erworben werden kann und deshalb nicht von Grund auf neu zu entwickeln ist. Somit werden im Rahmen der Referenzarchitektur die wesentlichen Komponenten identifiziert und vorgestellt, ohne jedoch ein vollständiges Klassendiagramm dieses komplexen Systems zu liefern. Zunächst wird im nächsten Abschnitt jedoch auf die Anwendungsfälle eingegangen.

6.6.1.1 Anforderungsanalyse und Systemabgrenzung

Die Anwendungsfälle beschränken sich auf die Verwaltung des AdServers. Das zugehörige Diagramm ist in der folgenden Abbildung 6-117 zu sehen. Die Akteure der Anwendungsfälle sind der Betreiber, das System als Controller und der Geschäftskunde.

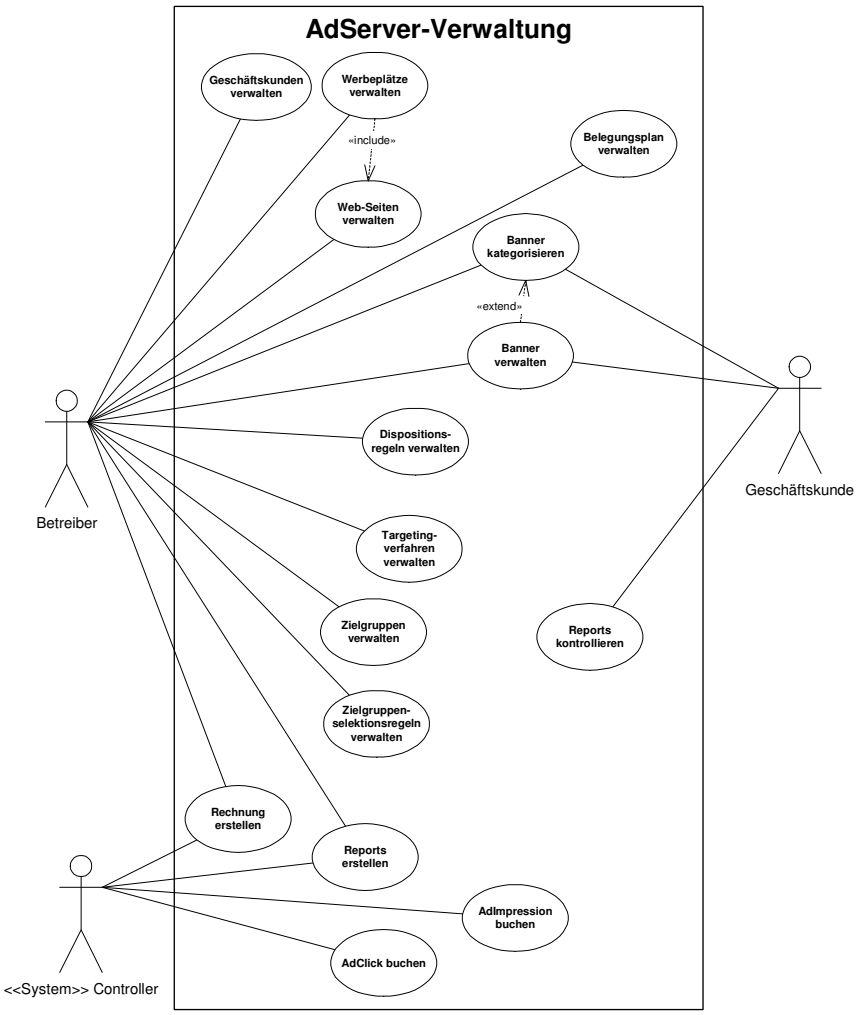


Abbildung 6-117: Anwendungsfälle der AdServer Verwaltung

Der AdServer dient dazu, sämtliche Aktivitäten im Rahmen der Werbeflächen und Banner abzudecken und somit als selbstständiges System das Erlösmodell WERBUNG vollständig abzudecken. Ein AdServer wird somit als eine eigenständige Komponente unabhängig von der zusätzlichen Systemlandschaft eines Geschäftsmodells eingesetzt. Somit ist es losgelöst vom *eigentlichen* Angebot des Geschäftsmodells. Dieses können sowohl frei zugänglicher Content, Produkte eines e-Shops oder Beiträge innerhalb einer Community sein. Sobald ein Unternehmen Werbeflächen vermieten will, stellt ein AdServer die Funktionalitäten bereit, diese zu verwalten und dem Geschäftspartner gegenüber abzurechnen.

Innerhalb des AdServers sind zunächst die Werbeflächen innerhalb der Websites sowie die Geschäftskunden zu verwalten, an die diese Flächen vermietet werden. Die Geschäftskunden buchen Werbekampagnen beim Betreiber. Dies bedeutet, dass der Betreiber einen Belegungsplan verwalten muss, der über einen bestimmten Zeitraum die Schaltung bestimmter Banner in einer bestimmten Häufigkeit umfasst. Die zu schaltenden Banner werden dabei nicht direkt benannt, sondern nur deren Kategorie. Auf diese Weise wird verhindert, dass immer ein und der selbe optische Banner eingeblendet wird. Stattdessen können mehrere, optisch unterschiedliche Banner einer Kategorie zugewiesen werden (Banner kategorisieren), die dann abwechselnd auf der Website eingeblendet werden. Um die Schaltung der Anzeigen zu steuern, kann der Betreiber Dispositionsregeln verwalten.

Die Bereitstellung der Banner (Banner verwalten) sowie deren Klassifikation in bestimmte Kategorien kann sowohl durch den Geschäftspartner als auch durch den Betreiber erfolgen, weshalb beide Rollen als Akteure definiert werden.

Ein wesentliches Ziel der Online-Werbung besteht darin, die Werbung möglichst zielgruppengenau zu platzieren und somit die Streuverluste zu minimieren. Diese Anforderung möglichst optimal umzusetzen ist die Aufgabe des Targeting. Es gibt zahlreiche Targetingverfahren, die durch den Betreiber zu verwalten sind. Einige Beispiele von Targetingverfahren sind nach [Wer98]:

- HTTP-Targeting: Bei diesem Verfahren wird die Belegung der Werbeflächen entsprechend der Betriebssysteme der Nutzer, der verwendeten Browser und der Domain-Namen geschaltet.
- Session-Targeting: Hier wird das Verhalten des Nutzers innerhalb der Session ausgewertet und dazu verwendet, geeignete Banner anzuzeigen. Einem Kunden wird dabei beispielsweise ein Banner in Abhängigkeit der besuchten Seitenaufrufe angezeigt.
- Cookie-Targeting: Hierbei handelt es sich um ein sessionübergreifendes Verfahren, indem das Verhalten der Nutzer in Cookies hinterlegt und bei erneuten Besuchen des Web-Auftritts mit berücksichtigt wird.
- Content-Targeting: Bei diesem Verfahren wird versucht, eine möglischst hohe Übereinstimmung des Werbemittels und dem Inhalt der Website zu erzielen. Als Voraussetzung müssen somit sowohl die Inhalte der Seiten als auch die Banner so detailliert wie möglich klassifiziert werden.
- Zeitraum-Targeting: Dieses Verfahren belegt die Werbeflächen in Abängigkeit mit der Tageszeit mit unterschiedlichen Bannern. Auf diese Weise können Banner explizit während allgemeiner Bürozeiten sowie früh morgens oder abends geschaltet werden.

Die Beispiele zeigen, dass einige Targetingverfahren (z.B. Session- und das Cookie-Targeting) Zielgruppen benötigen. Die Nutzer werden anhand ihres Verhaltens oder ihrer Daten analysiert und zunächst klassifiziert, damit ihnen geeignete Banner angezeigt werden können. Die Klassifikation, also die Zuordnung in eine Zielgruppe, setzt folglich eine Verwaltung der Selektionsregeln und eine Verwaltung der Zielgruppen selbst voraus.

Darüber hinaus muss ein AdServer Reports erstellen können, die tagesaktuell Auswertungen über die Kampagnen liefern, um den Erfolg zu messen und gegebenenfalls eingreifen zu können. Auch der Geschäftskunde wird diese Reports kontrollieren. Neben dem Reporting ist die Rechnungsstellung eine der Aufgaben, die der Server abdecken muss.

Das System selber übernimmt vollautomatisch die Protokollierung der Schaltung der Werbebanner (AdImpression buchen) sowie die Klicks der Nutzer auf dieses Banner (AdClick buchen).

Der Nutzer tritt im Rahmen der Angebotsnutzung auf, indem er die Banner aufruft oder auf einen Banner klickt. Beide Aktionen lösen eine entsprechende Buchung aus, die durch den (AdServer-)Controller erfolgt und thematisch bereits im Szenario der AdServer Verwaltung beachtet wurde. Das Anwendungsfalldiagramm in Abbildung 6-118 zeigt das Szenario der Angebotsnutzung.

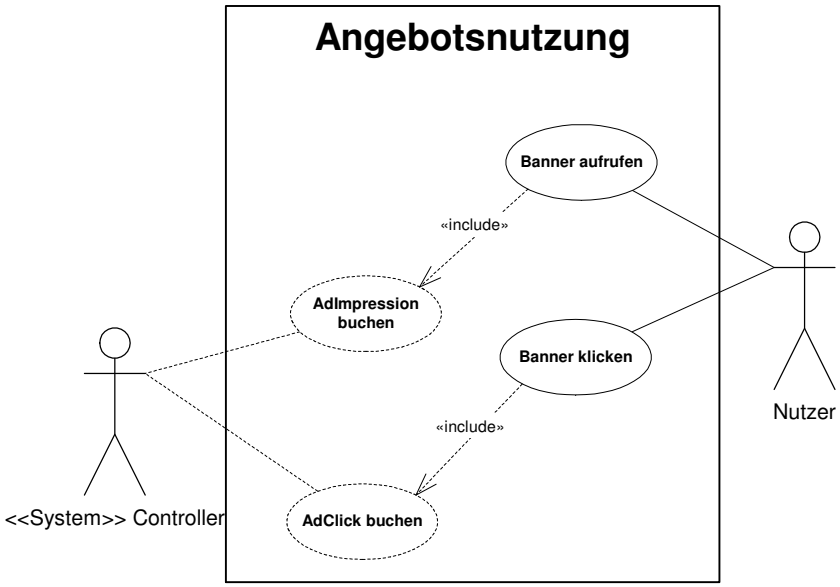


Abbildung 6-118: Anwendungsfälle der Angebotsnutzung

Im nächsten Abschnitt werden die grundlegenden Komponenten eines AdServers beschrieben, der wie bereits erwähnt die Anforderungen an eine Umsetzung des Erlösmodells WERBUNG vollständig abdeckt und bereits als fertiges System auf dem Markt zu erwerben ist.

6.6.1.2 Architekturbeschreibung

Die Architekturbeschreibung eines AdServers erfolgt hier nur grob anhand von grundlegenden Komponenten. Es werden einige grundlegende Dienste der Komponenten beschrieben, indem einige Klassen mit ihren Methoden dargestellt werden.

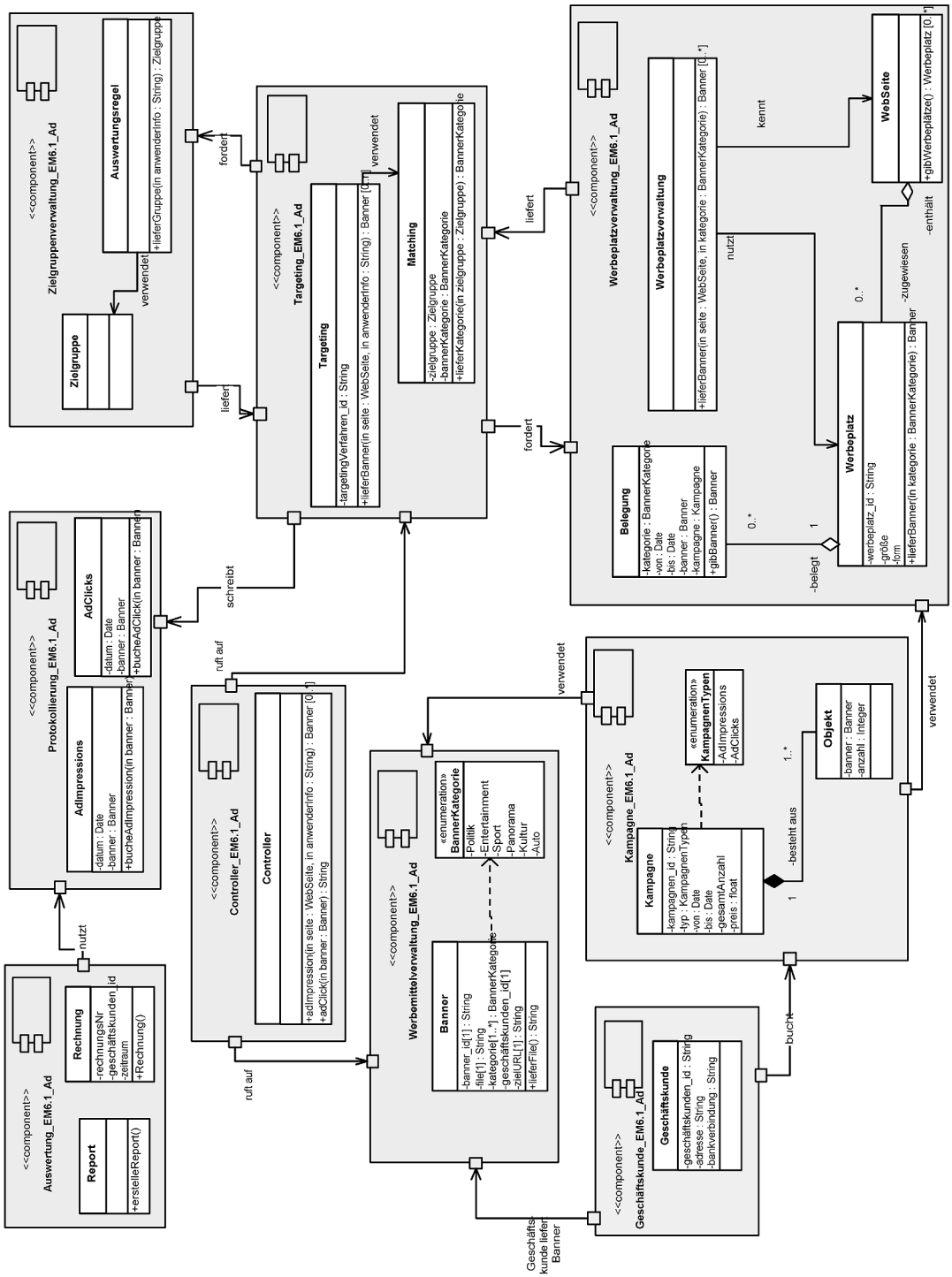


Abbildung 6-119: Statische Architektur eines AdServers

Die Komponenten des AdServes sind `Geschäftskunde`, `Werbemittelverwaltung`, `Kampagne`, `Werbeplatzverwaltung`, `Targeting`, `Zielgruppe`, `Protokollierung`, `Auswertung` und ein alles steuernder `Controller`. Alle Komponenten werden mit der Versionsbezeichnung `EM6.1_Ad` gekennzeichnet.

Die Komponente Geschäftskunde_EM6.1_Ad übernimmt die Verwaltung der Geschäftskunden. Sie ist somit weitgehend identisch mit den bisher vorgestellten Komponenten zur Verwaltung der Kunden (vgl. etwa `Kunde_EM1.1_ET` auf Seite 133) oder der Anbieter (vgl. `Anbieter_EM5.1_Prov` auf Seite 245). Sie ist deshalb hier nur sehr eingeschränkt dargestellt worden, da sie darüber hinaus für die Kernprozesse dieses Erlösmodells nur eine geringe Bedeutung hat.

Die Werbebanner werden als Files mit Hilfe der Komponente Werbemittelverwaltung_EM6.1_Ad verwaltet, die in Abbildung 6-120 dargestellt ist.

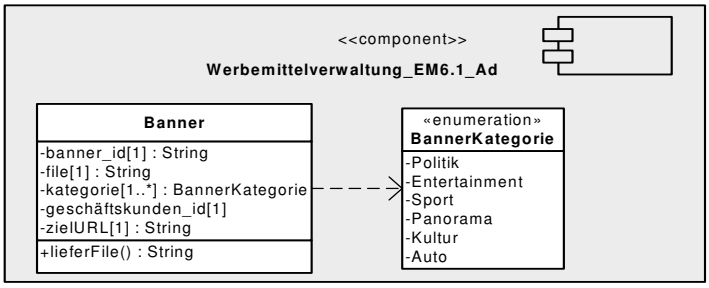


Abbildung 6-120: Komponente Werbemittelverwaltung_EM6.1_Ad

Die Kategorisierung der Banner erfolgt über einen Enumeration-Datentyp und muss im konkreten Anwendungsfall angepasst werden. Ein Banner kann mehreren Kategorien zugewiesen werden, was durch das mengenwertige Attribut `kategorie` ermöglicht wird. Als wesentlichen Dienst bietet die Komponente die Lieferung der URL, unter der ein Banner gespeichert ist (Methode `leiferFile`). Die Methode wird von der Werbeplatzverwaltung aufgerufen.

Die Komponente `Kampagne_EM6.1_Ad` dient dazu, die Werbeplätze an die Geschäftskunden zu verkaufen. Ein Geschäftskunde kann innerhalb eines Zeitraums eine bestimmte Anzahl von Bannern platzieren und bezahlt je nach `KampagnenTyp` für die *AdImpressions* oder für die tatsächlichen *AdClicks*. Jeder Kampagne ist eine Menge von `Objekten` zugewiesen, die jeweils Banner in einer vereinbarten Anzahl umfassen. Die folgende Abbildung Abbildung 6-121 zeigt die Komponente.

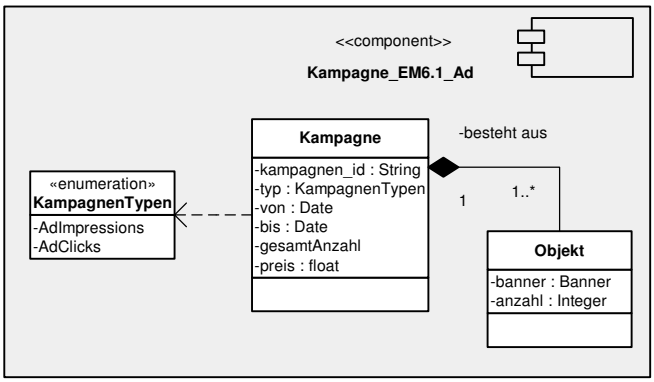


Abbildung 6-121: Komponente Kampagne_EM6.1_Ad

Während die bisher betrachteten Komponenten eher für die statische Verwaltung der Objekte zuständig sind, werden die folgenden Komponenten stärker während der Protokollierung der Aktivitäten mit einbezogen. Der Ablauf wird im Sequenzdiagramm in Abbildung 6-128 auf Seite 269 gezeigt. Zunächst werden jedoch die dort verwendeten Komponenten vorgestellt.

Die zentrale Steuerung wird von der Komponente Controller_EM6.1_Ad übernommen, die in der folgenden Abbildung zu sehen ist.

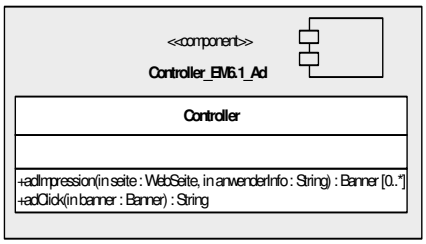


Abbildung 6-122: Komponente Controller_EM6.1_Ad

Die Komponente bietet über die Klasse Controller die beiden Methoden adImpression und adClick an. Die Methode `adImpression` ist für die Anzeige der Werbebanner auf einer Website sowie die Protokollierung dieser Aktion verantwortlich. `adClick` wird verwendet, um den Klick auf einen Banner festzuhalten.

Der Methode `adImpression` werden als Parameter die anzuzeigende Website sowie Informationen über den Nutzer überliefert. Es wird hier nicht näher darauf eingegangen, auf welche Art diese Nutzerinformationen erhoben werden. Dies kann sich je nach Targeting-Verfahren unterscheiden, wie in den Beispielen auf Seite 261 verdeutlicht wurde. Aus diesem Grund ist die Erhebung und Weitergabe dieser Daten je nach Anwendungsfall spezifisch festzulegen. Die Methode liefert als Rückgabewert eine Liste von Bannern, die im Rahmen des Response auf der Website eingebunden werden.

Um die Banner gemäß der Parameter zu ermitteln, verwendet die Methode die Komponente `Targeting`, die den Dienst `lieferBanner` bereitstellt. Dabei ist zu unterscheiden, welches Targeting-Verfahren anzuwenden ist. Für jedes Verfahren steht eine entsprechende Klasse zur Verfügung, die die spezifischen Auswertungsregeln beinhaltet. Das jeweils gültige Verfahren muss über die Komponente zunächst aktiviert werden. Die folgende Abbildung zeigt die Komponente.

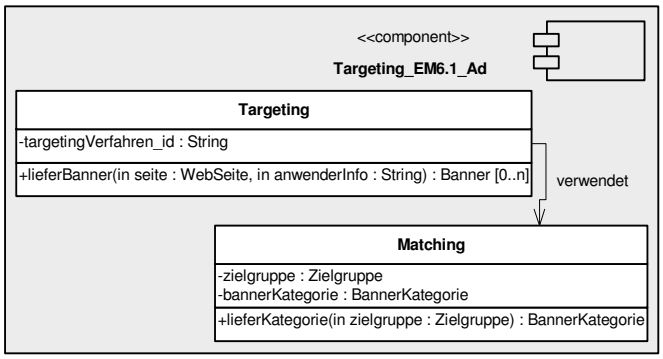


Abbildung 6-123: Komponente Targeting_EM6.1_Ad

Die Komponente muss zunächst die Zielgruppe identifizieren, der der Nutzer angehört. Dazu bedient sie sich des Dienstes `lieferGruppe` der Komponente `Zielgruppenverwaltung`. Diese Methode wird aufgerufen und liefert anhand der übergebenen Anwenderinformationen die `Zielgruppe`, in die der Nutzer einzuordnen ist.

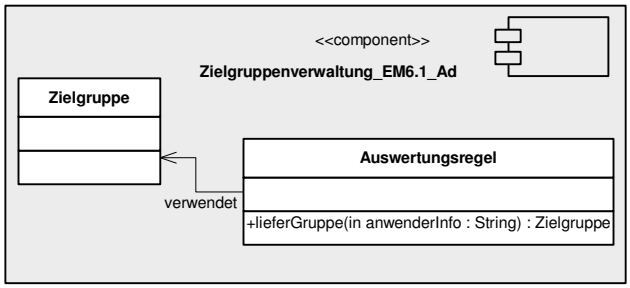


Abbildung 6-124: Komponente Zielgruppe_EM6.1_Ad

Abbildung 6-124 zeigt die Komponente mitsamt den Klassen `Zielgruppe` und `Auswertungsregel`. Die Klasse `Auswertungsregel` verwaltet die Regeln, nach denen die Anwenderinformationen auszuwerten sind und eine Zielgruppe als Ergebnis zurückgeliefert wird.

Nachdem die Zielgruppe identifiziert worden und an die aufrufende Komponente `Targeting` zurückgeliefert worden ist, muss dort anhand des verwendeten Verfahrens ermittelt werden, welche Kategorie von Bannern für den Nutzer anzuzeigen ist. Dazu verfügt die Komponente über die Klasse `Matching`, die für jede Zielgruppe die zugehörige Kategorie bereithält. Die Methode `lieferKategorie` stellt das angeforderte Ergebnis bereit.

Um nun konkrete Banner zu erhalten, die zu dieser Kategorie gehören und die gleichzeitig auch entsprechend der Kampagnen anzuzeigen sind, wird die Komponente `Werbeplatzverwaltung` verwendet. Sie ist in Abbildung 6-125 dargestellt.

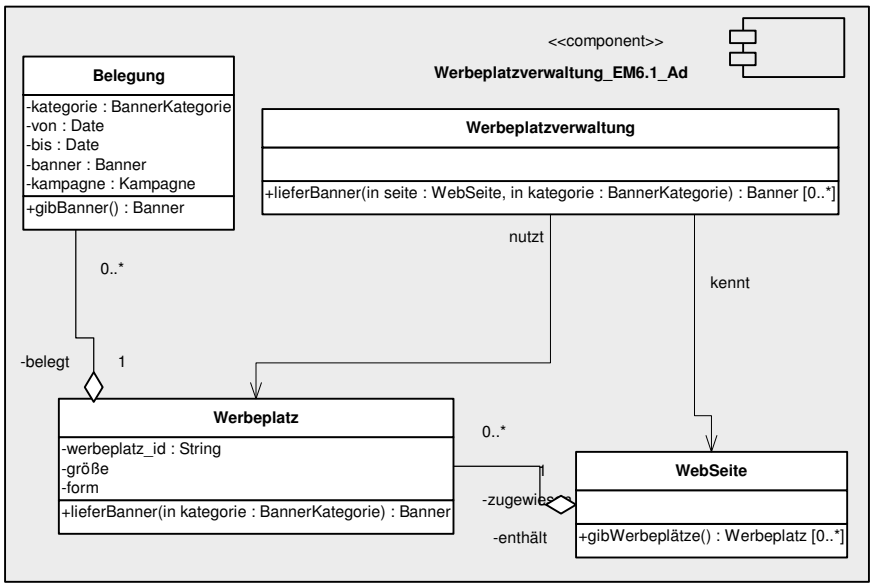


Abbildung 6-125: Komponente Werbeplatzverwaltung_EM6.1_Ad

Die Komponente verfügt über eine Verwaltung der Websites (`WebSeite`), der jeweils eine Menge von Werbeplätzen (`Werbeplatz`) zugewiesen sind, die wiederum eine Belegung (`Belegung`) aufweisen, die sich nach den gebuchten Kampagnen (Komponente `Kampagne`) richtet.

Als Schnittstelle steht die Methode `lieferBanner` zur Verfügung, die zu einer überlieferten Webseite und einer Banner-Kategorie eine Liste von Bannern zurückliefert. Der Aufruf dieser Methode erfolgt, wie bereits erwähnt, durch die `Targeting`-Komponente, die zuvor anhand der Nutzerinformationen und der Zielgruppe die relevante Kategorie der Banner ermittelt hat. Die Methode `lieferBanner` der `Werbeplatzverwaltung` ermittelt zunächst die zur Verfügung stehenden Werbeplätze der Seite, die angezeigt werden soll. Dazu nutzt sie die Methode `gibWerbeplätze` der `WebSeite` und erhält ein Listobjekt mit allen Werbeplätzen zurück. Im nächsten Schritt ruft die Methode von jeder Instanz der erhaltenen Werbeplätze die Methode `lieferBanner` auf. Diese Methode von `Werbeplatz` berücksichtigt ihre `Belegung`, indem sie eine Instanz der Klasse ermittelt,

die hinsichtlich der Kategorie übereinstimmt (Attribut `kategorie`) und einen aktuell gültigen (Attribute `von`, `bis`) Banner (Attribut `banner`) enthält. Als Rückgabewert wird eine Referenz auf diesen Banner zurückliefert.

Die Methode `lieferBanner` der Klasse `Werbeplatzverwaltung` ermittelt auf diese Weise alle Banner der Website und liefert das Ergebnis als Listobjekt an die aufrufende Methode von `Targeting` zurück. Das folgende Sequenzdiagramm zeigt diesen Ablauf mitsamt den beteiligten Klassen.

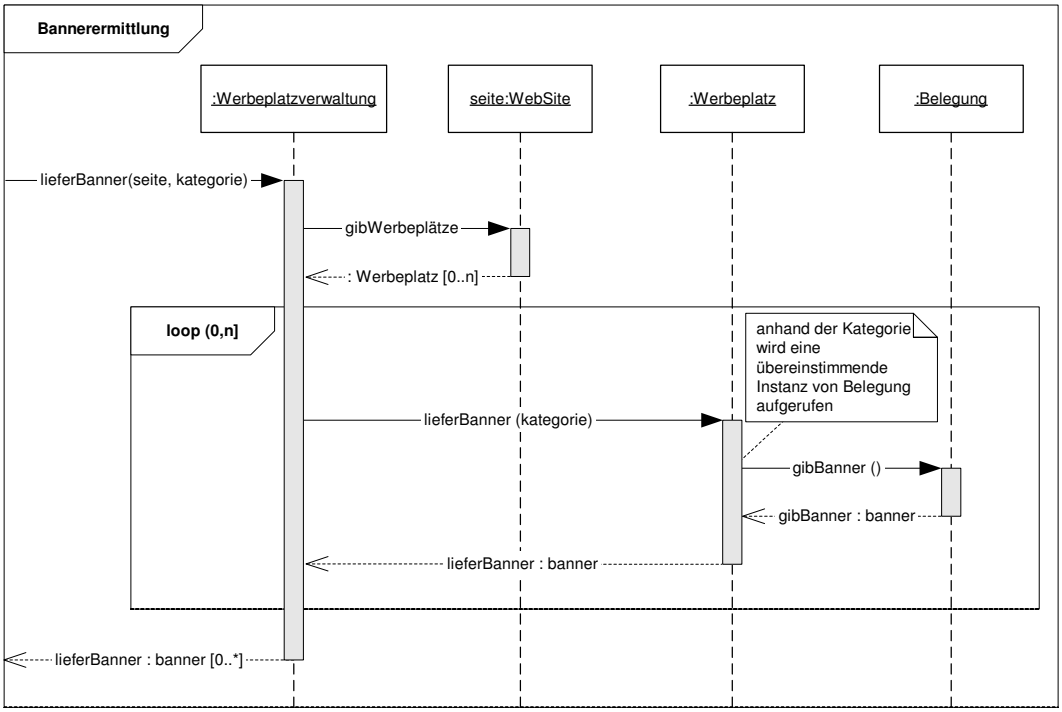


Abbildung 6-126: Sequenzdiagramm der Komponente Werbeplatzverwaltung, die die Identifikation der anzuzeigenden Banner als Dienst bereitstellt

Das Diagramm wird als Interaktionsreferenz auch in Abbildung 6-128 auf Seite 269 adressiert, wodurch die Einbindung in den Gesamtkontext offensichtlich wird.

Als letzte Aktivität muss das `Targeting` anschließend noch die URLs der Banner ermitteln und für die Buchung der Anzeige sorgen. Während die URLs über die `Werbemittelverwaltung` angefragt werden, dient die Komponente `Protokollierung` dazu, die AdImpressions zu protokollieren. Abbildung 6-127 zeigt die beiden relevanten Klassen zur Buchung der AdImpressions sowie der AdClicks.

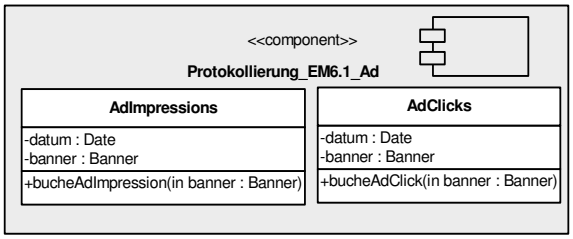


Abbildung 6-127: Komponente Protokollierung_EM6.1_Ad

Als Ergebnis wird dem Controller ein Listobjekt mit Bannern zurückgeliefert, dass dieser an die GUI weiterleiten kann, damit dort die Werbeobjekte eingeblendet werden können. Das folgende Sequenzdiagramm in Abbildung 6-128 zeigt den soeben beschriebenen Prozess mit seinen Interaktionen, die für die Anzeige der Werbebanner auf einer Website durchlaufen werden.

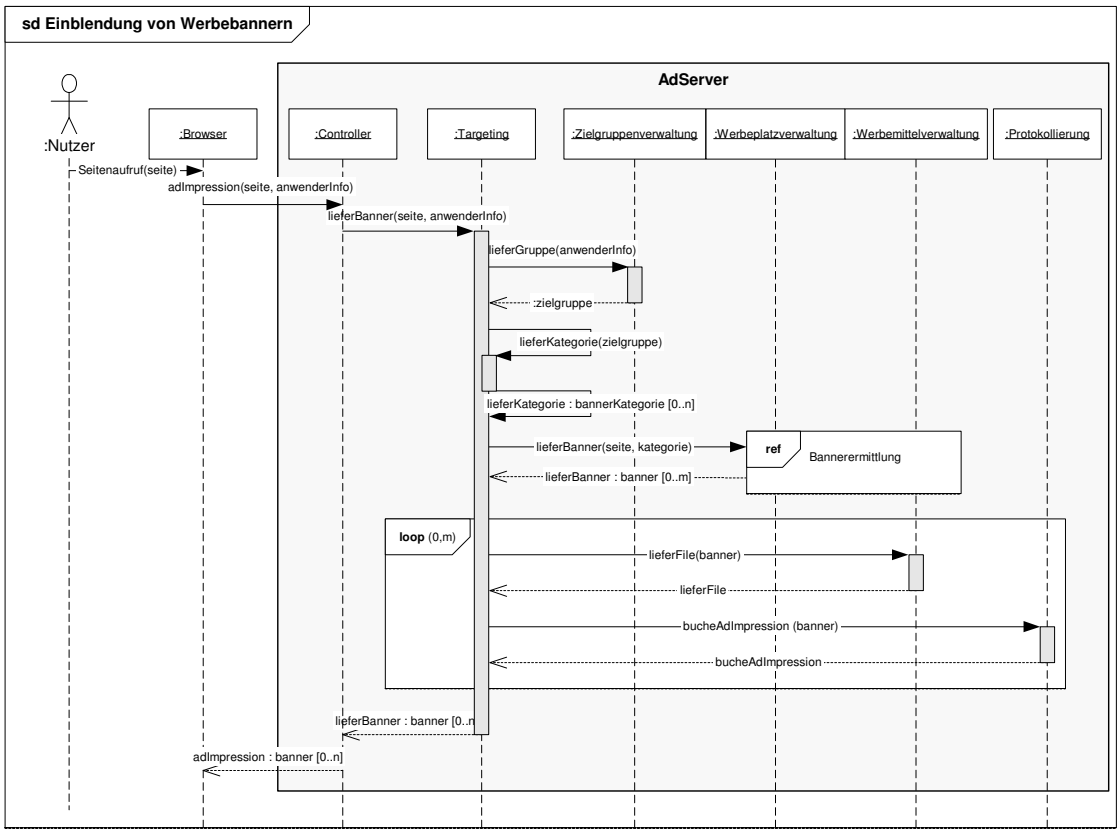


Abbildung 6-128: Sequenzdiagramm zur Buchung von AdImpressions

Das Verbuchen der AdClicks erfolgt mit den selben Komponenten, wobei die Interaktionen abweichen. Die folgende Abbildung 6-129 zeigt das zugehörige Sequenzdiagramm.

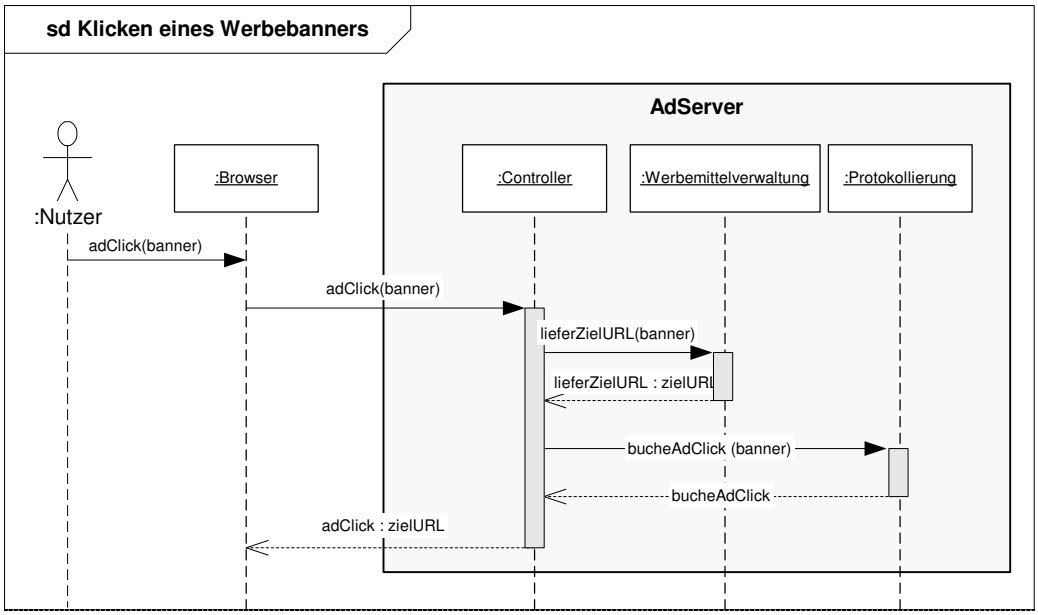


Abbildung 6-129: Sequenzdiagramm zur Buchung der AdClicks

Zur Verbuchung der AdClicks wird die gleichnamige Methode der Klasse `Controller` aufgerufen. Ihr wird als Parameter der relevante Werbebanner mitgeteilt; das heißt, dass der Werbebanner nicht direkt auf die Ziel-URL verweist, sondern zunächst eine interne URL enthält, die zum Aufruf des Controllers führt. Die Methode `adClick` ermittelt daraufhin zunächst die Zieladresse des Banners anhand der Werbemittelverwaltung und sorgt anschließend für die Protokollierung, indem sie den Dienst `bucheAdClick` verwendet. Als Rückgabewert an den Browser wird die Ziel-URL geliefert.

6.7 Schlussfolgerung

Die in diesem Kapitel definierten und beschriebenen Software-Referenzarchitekturen stellen das wesentliche Ergebnis der vorliegenden Arbeit dar. Ihre Gesamtheit wird als *Katalog* von Software-Referenzarchitekturen für Erlösmodelle des e-Business bezeichnet. Mit diesem Katalog steht eine Sammlung von Entwurfsvorlagen zur Verfügung, die dazu verwendet werden kann, die Herleitung einer konkreten Softwarearchitektur für ein konkretes Geschäftsmodell wesentlich zu erleichtern und zu beschleunigen. Die gewählte Granularität der Architekturen mit ihren Komponenten sowie den Klassen und deren Beziehungen zueinander gewährleistet dabei einen anwendungsnahen Einsatz.

Inwiefern dieser Katalog der Software-Referenzarchitekturen während der Herleitung einer konkreten Architektur nützlich ist, zeigt seine Verwendung im Rahmen eines realen Anwendungsfalls. Die Beschreibung dieses Anwendungsfalls findet im nächsten Kapitel statt.

7 Validierung

In Kapitel 6 wurden Software-Referenzarchitekturen zur Realisierung von Erlösmodellen des e-Business vorgestellt. In diesem Kapitel wird nun aufgezeigt, wie anhand dieser Referenzarchitekturen eine konkrete Implementierung eines realen Geschäftsmodells erfolgte. Im Rahmen der Umsetzung einer deutschen Internetlotterie-Plattform wurden die Erkenntnisse dieser Arbeit als Ausgangspunkt verwendet, um die Softwarearchitektur herzuleiten und an die spezifischen Anforderungen anzupassen. Die Betrachtungen in diesem Kapitel konzentrieren sich dabei auf die Umsetzung der Architekturbestandteile, die zur Realisierung des Erlösmodells notwendig waren. Es wird nicht die gesamte Softwarearchitektur der Lotterie-Plattform vorgestellt.

Im folgenden Abschnitt wird das zugrundeliegende Geschäftsmodell zunächst beschrieben und das enthaltene Erlösmodell wird anhand der Klassifikationskriterien aus Kapitel 4 charakterisiert. Anschließend erfolgt im Abschnitt 7.2 die Einordnung des Fallbeispiels in die Software-Referenzarchitekturen, die in dieser Arbeit identifiziert und vorgestellt wurden. Die Definition der Softwarearchitektur durch die Übernahme und Anpassung der vorhandenen Komponenten wird im Abschnitt 7.3 durchgeführt. Abschließend erfolgt im letzten Abschnitt eine vergleichenden Betrachtung der Softwarearchitektur für die Lotterie-Plattform mit den Referenzarchitekturen.

7.1 Das Geschäftsmodell einer Internetlotterie-Plattform

Das Fallbeispiel für die Validierung bildete eine internetbasierte Plattform eines deutschen Lotteriebetreibers. Das Geschäftsmodell basiert im Wesentlichen darauf, den Kunden die Möglichkeit zu bieten, über das Internet an staatlichen Lotterien teilzunehmen. Den Kunden stehen dazu unterschiedliche Lotterien zur Verfügung, die sich grundsätzlich in die Kategorien der Sofort- und der Zahlen-Lotterien sowie Sportwetten unterteilen lassen. Diese Lotterien und Wetten stellen somit das Angebot beziehungsweise die Produkte des Geschäftsmodells dar. Da diese Produkte sehr spezifische Eigenschaften aufweisen, werden sie zunächst detaillierter vorgestellt.

Eine Sofort-Lotterie ist dadurch gekennzeichnet, dass einzelne Lose zu einem festen Preis erworben werden und diese Lose einen vorbestimmten Gewinn[1] ausweisen. Die Anzahl der Lose pro Spiel sowie die genaue Verteilung der gewinnenden Lose je nach Gewinnklasse und die zugehörigen Gewinnbeträge sind für jede Sofort-Lotterie fest definiert und in einem Gewinnplan hinterlegt. Dieser Gewinnplan legt anhand dieser Verteilungen auch fest, dass ein fester Anteil der Erträge als Gewinne ausgeschüttet wird, wobei dieser Anteil häufig 50% ausmacht.

Das folgende Beispiel eines Gewinnplans ist den Spielregeln einer realen Sofort-Lotterie entnommen worden:

1. Ein *Gewinn* kann dabei auch 0 € betragen - zur Vereinfachung wird hier nur dieser Begriff und nicht etwa der Begriff einer *Niete* verwendet.

Die Gewinnausschüttung beträgt 50 % der Spieleinsätze. Eine Serie von 1 Mio. Losen enthält folgende Gewinne:

Anzahl der Gewinne x Gewinn in Euro = Gesamtgewinne in Euro und prozentualer Anteil

2 x 10.000,00 = 20.000,00 = 4 %

10 x 500,00 = 5.000,00 = 1 %

4.500 x 10,00 = 45.000,00 = 9 %

40.000 x 2,00 = 80.000,00 = 16 %

100.000 x 1,00 = 100.000,00 = 20 %

Die Gewinnpläne unterliegen einer staatlichen Kontrolle. Für die Realisierung einer Sofort-Lotterie in digitaler Form stellen die Gewinnpläne die Grundlage für die Generierung der Lostrommel dar, die wiederum die einzelnen Lose enthält.

Ein Kunde kann ein solches Los zu einem festen Betrag erwerben. Anschließend wird er über eine GUI zu einer Interaktion aufgefordert, die beispielsweise das Freirubbeln von verdeckten Spielflächen beinhaltet, so dass unterschiedliche Gewinnsymbole sichtbar werden, die wiederum verschiedene Gewinne repräsentieren. Die Lose werden deshalb auch als *Rubbellose* bezeichnet. Die grafischen Animationen und die zugehörigen Interaktionen richten sich jeweils nach den unterschiedlichen Spielideen einer Sofort-Lotterie, wobei diese jedoch unabhängig sind von der vordefinierten Aufteilung der Lose und ihrer Gewinne. Der Spieler hat generell keinen Einfluss auf das Ergebnis. Auch das Ziehen (der Kauf) eines Loses erfolgt durch einen Zufallsgenerator.

An dieser Stelle wird bereits ersichtlich, dass die Produkte dieses Geschäftsmodells sehr spezifisch sind und eine Eigenschaft aufweisen, die in den Referenzarchitekturen nicht berücksichtigt wurde: die Lose können Gewinne enthalten, die dem Käufer gutgeschrieben und ausgezahlt werden müssen. Bevor auf die Schlussfolgerungen dieser Besonderheit eingegangen wird, werden zunächst noch die Zahlen-Lotterien und Sportwetten als die weiteren angebotenen Spielarten vorgestellt.

Bei den Zahlen-Lotterien und Sportwetten gibt der Kunde anhand eines Spielscheins seine persönlichen Vorhersagen im Rahmen der Spielregeln ab. Ein Spielschein erhält in der Regel mehrere Tippreihen, auf denen der Kunde jeweils eine vollständige Vorhersage treffen kann; und die somit auch einzeln zu bezahlen sind. Die Ergebnisermittlung in Form einer Ausspielung erfolgt zu fest definierten Zeitpunkten. Die Ausspielungen können entweder Ziehungen von Zahlen (Beispiele sind die Lotterien *Lotto - 6 aus 49* oder *Glücksspirale*) oder aber Austragungen von Sportereignissen (Beispiel ist die Sportwette *Oddset*) sein. Der Gewinn des Kunden richtet sich grundsätzlich nach der Übereinstimmung seiner Vor-

hersagen mit den Ergebnissen der Ausspielung und nach den insgesamt geleisteten Einzahlungen aller Teilnehmer innerhalb einer Ausspielung. Ein fester Prozentsatz dieser Einzahlungen wird verteilt über alle Gewinnklassen wieder ausgeschüttet.

Eine Besonderheit dieser Spielarten ergibt sich aus Sicht des Internet-Portals dadurch, dass diese Spiele sowohl über das Internet, als auch über den terrestrischen Vertrieb anderer deutscher Lotteriegesellschaften gespielt werden. Die Lotterien und Sportwetten werden somit nicht über die Plattform des Geschäftsmodellbetreibers abgewickelt. Stattdessen übernimmt die Internetlotterie vorwiegend die Entgegennahme und Aufbewahrung der Spielscheine sowie die Auszahlung der Gewinne an die Kunden. Die Vorhersagen und Einsätze der Spieler werden an das System der ausführenden Lotteriegesellschaft weitergereicht. Dieses meldet ihrerseits die Ergebnisse der später stattfindenden Ausspielung an die Internetlotterie-Plattform.

Aufgrund dieses sehr speziellen Charakters der Spiele, deren Durchführung eben nicht auf dem Portal stattfindet, werden diese Produkte im Rahmen der Validierung nicht weiter berücksichtigt. Die Umsetzung dieser Zahlen-Lotterien und Sportwetten in Bezug auf die Softwarearchitektur ist im hohen Maße durch die Gestaltung der Schnittstellen zu den Fremdsystemen gekennzeichnet. Die Gestaltung der Softwarearchitektur zur Realisierung des Erlösmodells der Internetlotterie wird dagegen vorrangig durch die Umsetzung der Rubbellose beeinflusst, weshalb sich auf diese konzentriert wird.

Die Kunden des Geschäftsmodells können als Käufer der Rubbellose charakterisiert werden. Es ist dabei notwendig, dass die Kunden identifiziert agieren. Dies ergibt sich zum einen daraus, dass aufgrund der Teilnahmebedingungen die Kunden bei der Registrierung eine Adresse hinterlegen müssen, deren Gültigkeit vom System überprüft wird. Darüber hinaus müssen die Kunden über ein plattforminternes Onlinekonto verfügen, auf dem die potenziellen Gewinne gutgeschrieben und zur Auszahlung an ein Bankkonto verwaltet werden können.

Das Erlösmodell basiert auf den verkauften Losen und wird somit direkt von den Kunden erhoben. Je nach Spiel fällt ein bestimmter Anteil der Rubbellosgebühren an die Lotterie ab. Dieser Anteil stellt das Einkommen der Lotterie dar und muss auf einem separaten Konto verbucht werden. Die Art der Zahlungsmittel stellt ebenfalls eine Besonderheit des Geschäftsmodells dar. Da die Preise für die Produkte zum Teil sehr gering sind (etwa 0,50 € bis 5,00 € für ein Rubbellos) und die Kunden aufgrund der Gewinnoption über ein identifiziertes Profil mitsamt monetärem Onlinekonto verfügen, werden auch die Kosten für die Spiele von diesem Konto abgebucht. Die Teilnahmebedingungen der Lotterie setzen dabei voraus, dass die Lotterie nicht in Vorleistung tritt, sondern dass das Konto einen positiven Saldo mindestens in Höhe des Einsatzes aufweisen muss. Bevor ein Kunde das erste Los oder den ersten Spielschein kaufen kann, muss er sein Onlinekonto auf dem Portal *aufladen*. Dies kann mittels einer Überweisung erfolgen. Um auch Daueraufträge (Sonderform der Zahlen-Lotterien) zu unterstützen, werden darüber hinaus Lastschriften angeboten, so dass die Lotterie im Voraus für eine ausreichende Deckung des Onlinekontos sorgt. Eine weitere Option wird durch firmenspezifische Prepaid-Karten gewährleistet. Diese Karten beinhalten einen bestimmten Betrag (etwa 5 €, 10€ oder 20 €) und können von einem Kunden in den terrestrischen Geschäften der Lotterie erworben und über eine Karten-Identifi-

zierungsnummer auf dem Onlinekonto der Internet-Plattform gutschreiben werden. Das Onlinekonto stellt somit die Schnittstelle aller finanziellen Transaktionen des Kunden mit dessen externen Bankkonto dar. Es ist somit aus Revisionsgründen zwingend erforderlich, dass sämtliche Onlinekonto-Transaktionen protokolliert werden. Auch eine kundenspezifische Historie der durchgeführten Spiele, die diese Transaktionen beeinflussen, muss zu Revisionszwecken aufgezeichnet werden.

Abbildung 7-1 zeigt zusammenfassend die Klassifikationskriterien zur Charakterisierung des Erlösmodells des Lotteriebetreibers.

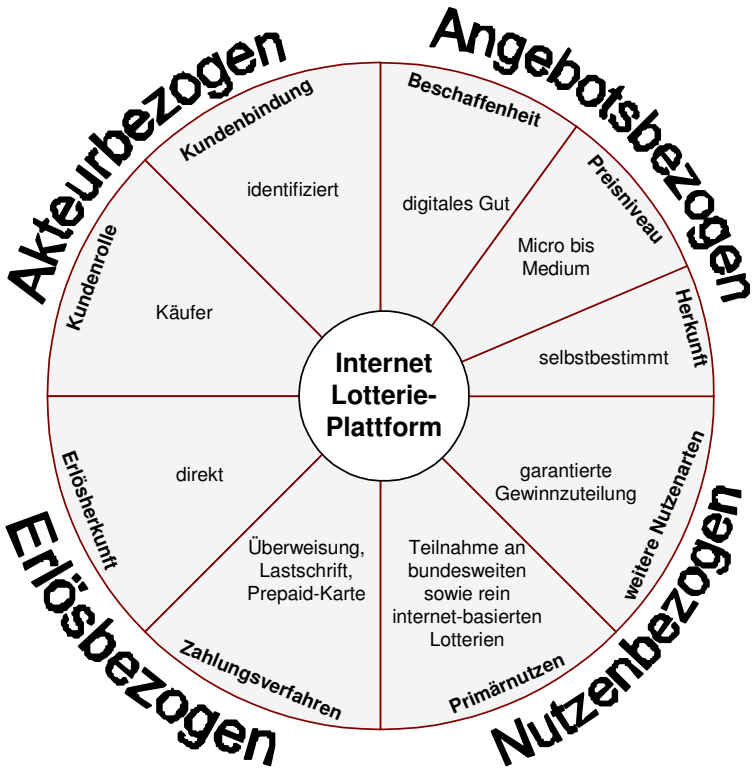


Abbildung 7-1: Erlösmodellklassifikation einer Internetlotterie-Plattform

Für den Kunden ergeben sich unterschiedliche Nutzenaspekte bei der Spieleabwicklung über das Internet. Zum einen werden ihm einige, in dieser Form nur über das Internet zu spielende Sofortlotterien offeriert. Die graphische Animation und die Interaktionsmöglichkeiten bieten dabei ein Spielerlebnis, das sich nicht auf die physische Variante dieser Spiele übertragen lässt. Darüber hinaus hat der Kunde bei einer Teilnahme an den Zahlen-Lotterien und Sportwetten jedoch auch den Vorteil, dass potenzielle Gewinne garantiert gutgeschrieben werden, da der Kunde identifiziert ist und seine Spielscheine digital hinterlegt sind. Dadurch entfällt die Gefahr durch den Verlust einer physischen Spielquittung, ohne die ein Gewinn nicht auszahlbar ist. Diese Apekte stellen somit Nutzenarten dar, die sich für den Kunden nur unter Verwendung der Internet-Plattform ergeben.

Nachdem das Geschäfts- und sein Erlösmodell nun vorgestellt wurden, wird im nächsten Abschnitt dieses Erlösmodell in Relation zu den Software-Referenzarchitekturen gesetzt, die im vorherigen Kapitel vorgestellt wurden. Ziel dabei ist es, Komponenten zu identifizieren, die identisch übernommen oder als Grundlage für eine Anpassung an die spezifischen Anforderungen verwendet werden können.

7.2 Einordnung in die Software-Referenzarchitekturen

Um eine Einordnung des Erlösmodells der Internetlotterie-Plattform in die vorhandenen Software-Referenzarchitekturen vornehmen zu können, werden zunächst noch einmal die wesentlichen Merkmale dieses Geschäftsmodells zusammengefasst:

- Es werden digitale Produkte angeboten, wobei grundsätzlich die verschiedenen Typen Sofort- und Zahlen-Lotterien sowie Sportwetten zu unterscheiden sind. Die Spieltypen unterscheiden sich wesentlich; Im Rahmen der Validierung werden lediglich die Sofort-Lotterien berücksichtigt.
- Die Produkte weisen als Besonderheit eine potenzielle Gewinnausschüttung auf.
- Die Kunden interagieren identifiziert als Käufer.
- Die Kunden verfügen über ein monetäres Onlinekonto. Dieses dient zur Gewinnverbuchung und zur Bezahlung der Lose beziehungsweise Spielscheine.
- Die Lotterie tritt nicht in Vorleistung, so dass keine Rechnungen erstellt werden. Statt dessen muss das Onlinekonto des Kunden vor einem Kauf als Mindestdeckung den Spieleinsatz aufweisen.
- Sowohl die Onlinekonto-Transaktionen als auch die durchgeführten Spiele sind aus Revisionsgründen vollständig mit den relevanten Informationen zu protokollieren.
- Die Lotterie erhält einen Anteil der Verkaufserlöse, der auf einem separaten Konto zu buchen ist.

Es wird anhand dieser Merkmale ersichtlich, dass keine eindeutige Zuordnung zu einer der identifizierten Referenzarchitekturen möglich ist. Der extrem spezifische Charakter der Produkte mit der Berücksichtigung von Gewinnen ist hierfür bereits ein hinreichendes Argument. Aus diesem Grund wurde bei der Herleitung der Softwarearchitektur untersucht, inwiefern einzelne Komponenten aus verschiedenen Referenzarchitekturen übernommen werden konnten.

Das Lotterie-Portal weist eine weitgehende Übereinstimmung mit dem Erlösmodell EINZELTRANSAKTION auf. Die Kunden der Plattform kaufen (*Käufer* als Kundenrolle) einzelne Produkte in Form der Lose (*digitales Gut* als Produktbeschaffenheit) und bezahlen dafür den jeweils anfallenden Betrag (*Micro- und Medium* Preisniveau). Das Erlösmodell richtet sich somit direkt an die Kunden (*direkte* Erlösherkunft). Die Produkte werden von der Lotterie bereitgestellt (*selbstbestimmte* Produktherkunft).

Anhand der Auflistung dieser Erlösmodellklassifikation wird ersichtlich, dass beide Varianten der EINZELTRANSAKTION zu berücksichtigen sind. Die zugehörigen Software-Referenzarchitekturen wurden in den Abschnitten 6.1.1 und 6.1.2 beschrieben. Die Varianten richten sich nach physikalischen Gütern und nach digitalen Gütern, für die eine anonyme Kundenbeziehung zugrunde gelegt wurde. Hintergrund für diese Annahme ist die Tatsache, dass digitale Produkte aufgrund ihrer Zustellbarkeit erst die Voraussetzung beinhalten, eine anonyme Kundenbeziehung und Kaufdurchführung zu realisieren. Diese Option wird in dem Fallbeispiel jedoch durch die notwendige Identifikation der Kunden gegenüber der Lotterie verhindert. Aus diesem Grund wird zunächst die Variante EINZELTRANSAKTION für physikalische Güter betrachtet. Die folgende Abbildung 7-2 zeigt grob die enthaltenen Komponenten der Software-Referenzarchitektur. Das detaillierte Klassendiagramm ist in Abbildung 6-11 auf Seite 132 zu sehen.

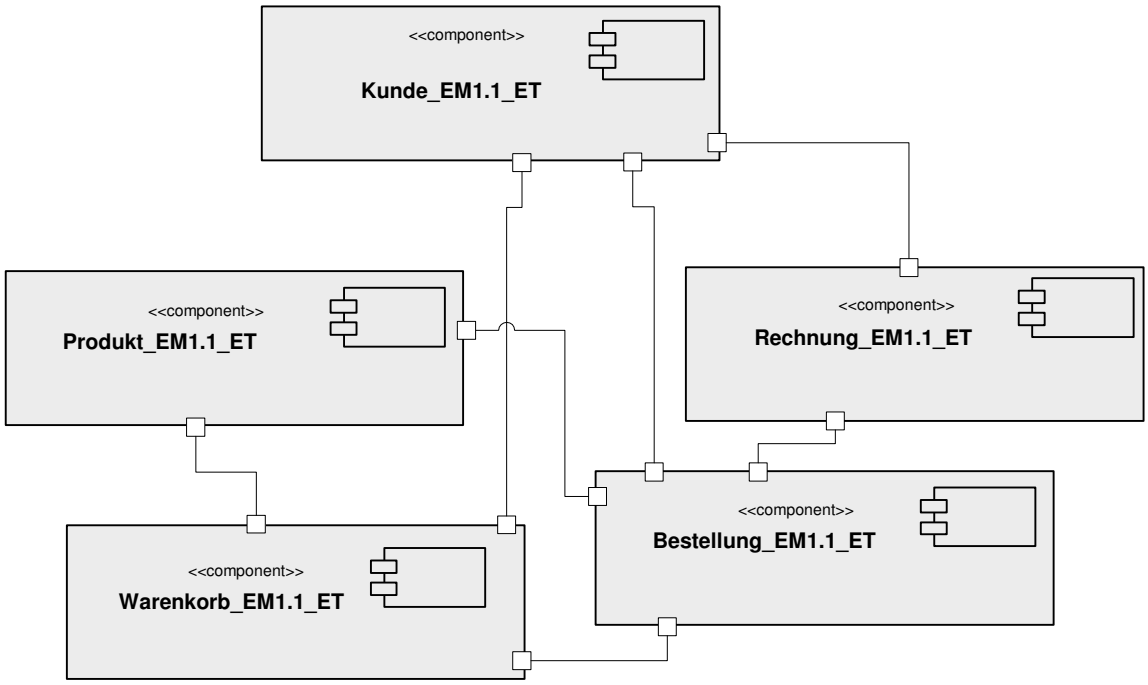


Abbildung 7-2: Komponenten des Erlösmodells Einzeltransaktion für physikalische Güter

Die verwendeten erlösmodellrelevanten Komponenten sind `Kunde`, `Produkt`, `Warenkorb`, `Bestellung` und `Rechnung`. Die Kundenkomponente kapselt dabei auch ein Debitorenkonto des Kunden.

Grundsätzlich werden für die Lotterie-Plattform die Komponenten `Kunde` und `Produkt` benötigt. Bevor auf diese Komponenten mit ihren Klassen im Detail eingegangen wird, werden die übrigen Komponenten betrachtet.

Der Warenkorb dient dazu, mehrere Produkte zunächst temporär abzulegen beziehungsweise zwischenzuspeichern, bevor der tatsächliche Kauf erfolgt. Dieses Modell weicht jedoch vom Ablauf auf der Lotterie-Plattform ab, da dort ein Produkt (ein Los) direkt gekauft wird, ohne dass zunächst ein weiteres Produkt betrachtet werden kann. Dies ergibt sich aufgrund der Anforderung, die Spieltransaktionen vollständig oder gar nicht durchzuführen. Ein Abbruch der Transaktion ist dabei bis zur Bestätigung des Kaufs möglich, jedoch kann nicht zwischen mehreren Produkten *gewechselt* werden. Aus diesem Grund ist der Warenkorb nicht erforderlich.

Ebenso wird keine Bestellung benötigt. Diese diente dazu, die Produkte des Warenkorbs endgültig zu kaufen und ihre Auslieferung und Zustellung zu veranlassen. Da hier jedoch kein Warenkorb eingesetzt wird, sondern die Lose unmittelbar gekauft werden und es sich darüber hinaus um digitale, direkt zustellbare Produkte handelt, besteht keine Notwendigkeit für diese Komponente.

Auch die Rechnung wird für die Realisierung des Erlösmodells der Lotterie-Plattform nicht benötigt. Es wurde bereits erwähnt, dass der Kunde auf seinem monetären Onlinekonto vor dem Kauf der Produkte eine ausreichende Deckung aufweisen muss und die Lotterie nicht in Vorleistung tritt. Somit entfällt die Rechnung als Zahlungsaufforderung.

Die folgende Abbildung 7-3 zeigt das vollständige Klassendiagramm der Komponenten `Kunde` und `Produkt` jeweils in der Version `EM1.1_ET` des Erlösmodells EINZELTRANSAKTION in der Variante für physikalische Güter.

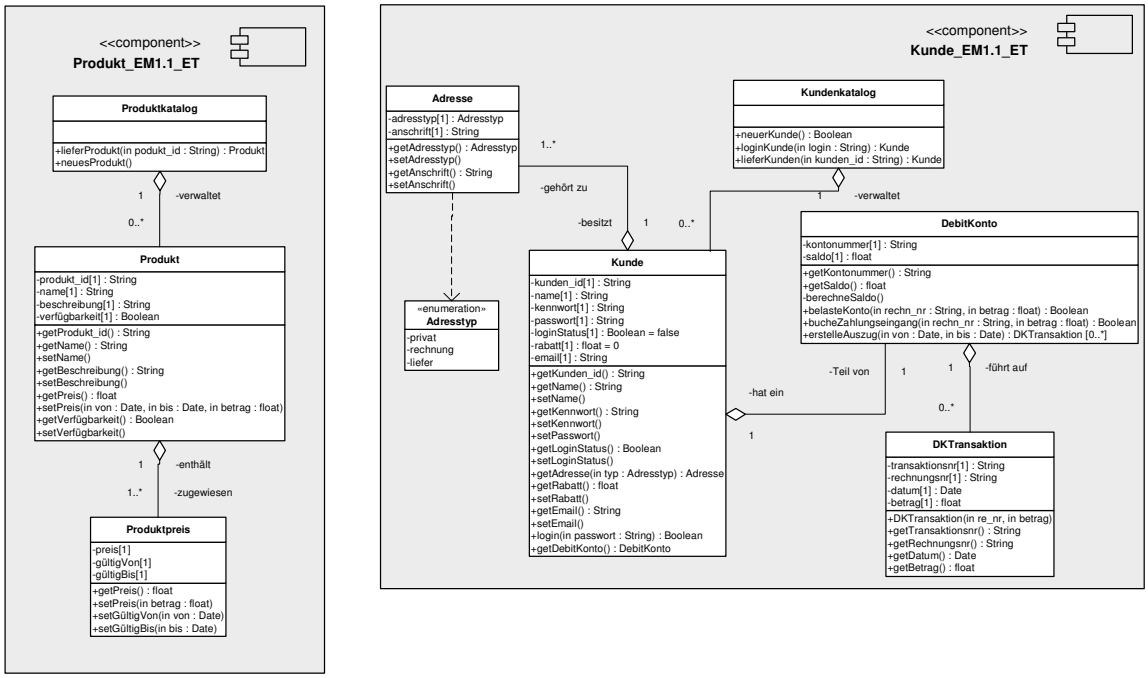


Abbildung 7-3: Klassendiagramm der Komponenten Kunde und Produkt

Die Komponene `Kunde_EM1.1._ET` kapselt die Klassen `Kundenkatalog`, `Kunde`, `Adresse`, `DebitKonto` und `DKTransaktion`. Sämtliche Klassen werden auch für das Erlösmodell der Lotterie-Plattform benötigt und können somit wiederverwendet werden. Eine identische Übernahme der Klassen mit ihren aufgelisteten Attributen ist verständlicherweise nicht möglich, so dass eine geeignete Adaption erfolgen muss. Dies ergibt sich aufgrund der Wahl eines eShops als Referenzanwendung, die dem Erlösmodell EINZELTRANSAKTION und der zugehörigen Softwarearchitektur zugrundegelegt wurde. So werden beispielsweise den Lotteriekunden keine Rabatte eingeräumt und es gibt lediglich eine private Adresse des Spielers. Diese Anpassungen auf Attributebene ergeben sich somit zwangsläufig bei der Verwendung einer Referenzarchitektur zu Herleitung eines konkret zu implementierenden Anwendungsfalls.

Auch das Debitorenkonto der Referenzarchitektur kann als monetäres Onlinekonto für das Lotterie-Portal wiederverwendet werden, wobei zum besseren Verständnis eine Umbenennung erfolgt. Die beiden Methoden `belasteKonto` und `bucheZahlungseingang` sind sehr gut dazu geeignet, die Abbuchung der Losgebühren einerseits und die Zahlungseingänge durch Überweisungen von externen Bankkonten andererseits durchzuführen. Lediglich eine Anpassung der Parameter wird erforderlich. Darüber hinaus müssen eine Methode zur Durchführung von Überweisungen auf ein externes Bankkonto und eine eigenständige Methode für das Verbuchen der Gewinne implementiert werden, um diesen speziellen Geschäftsvorfall abwickeln zu können.

Eine Betrachtung der Komponente `Produkt_EM1.1_ET` mit ihren Klassen und ein Vergleich zu den bereits erwähnten spezifischen Merkmalen der Lose zeigt, dass hier eine umfangreichere Anpassung notwendig wird. Da es sich nicht um Produkte im Sinne eines eShops handelt, sondern um digitale Lose, sind diese sehr spezifisch zu gestalten. Dennoch kann der grundsätzliche Aufbau durch die Klassen `Produktkatalog und Produkt` für die Realisierung der Sofort-Lotterie Lose wiederverwendet werden, wie im nächsten Abschnitt zu sehen sein wird.

Ein Vergleich der Lotterie-Plattform mit der Software-Referenzarchitektur für Einzeltransaktionen physikalischer Güter zeigt somit, dass einige erforderliche Komponenten übernommen werden können. Diese Variante weist jedoch keine Komponente auf, die für das Verbuchen der Erlösanteile verwendet werden kann, die der Lotterie zustehen. Dazu wird ein zentrales Konto benötigt, auf dem bei jeder erfolgten Transaktion der entsprechende Betrag gutzuschreiben ist. Der Begriff *zentral* drückt dabei aus, dass lediglich eine Instanz der Komponente innerhalb der Plattform benötigt wird, die dem Lotteriebetreiber und nicht etwa jedem Kunden zugeordnet wird. Ein solches zentrales Konto wurde in der Software-Referenzarchitektur des Erlösmodells EINZELTRANSAKTION in der Variante für digitale Produkte eingesetzt. Dort übernimmt das zentrale Konto die Aufgabe, die verkauften Produkte zu protokollieren, um eine Kontrolle der Abrechnungen mit dem externen Payment-Dienstleister zu ermöglichen. Die folgende Abbildung 7-4 zeigt die Komponente zur Realisierung dieses zentralen Kontos. Eine detaillierte Darstellung und anschließende Beschreibung der Referenzarchitektur ist in Abschnitt 6.1.2 auf Seite 148 gegeben[2].

2. das Klassendiagramm ist in Abbildung 6-28 auf Seite 155 zu sehen

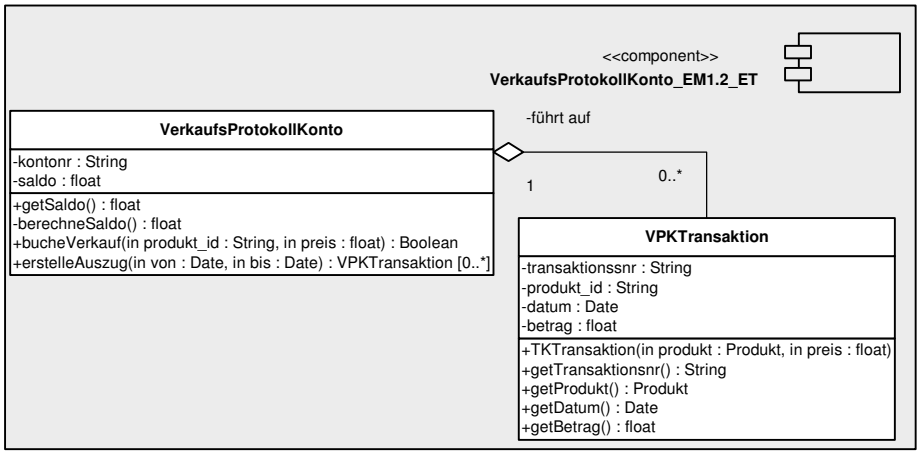


Abbildung 7-4: Komponente VerkaufsProtokollKonto aus der Referenzarchitektur Einzeltransaktion für digitale Güter

Die Komponente wurde in diesem Erlösmodell als `VerkaufsProtokollKonto` bezeichnet, so dass der Name für den Einsatz innerhalb der Lotterie-Plattform in ERLÖSKONTO geändert wurde. Die Methode `bucheVerkauf` stellt die wesentliche Schnittstelle dar, die ebenfalls an die veränderten Rahmenbedingungen der Lotterie angepasst wurde, was im folgenden Abschnitt näher erläutert wird.

Eine letzte offene Anforderung der Lotterie-Plattform stellt somit die Protokollierung der durchgeführten Spiele eines Kunden dar. Wie im vorherigen Abschnitt erläutert wurde, dient dieses Konto dazu, sämtliche Loskäufe eines Kunden aus Revisionsgründen detailliert festzuhalten. Anders als das zentrale Erlöskonto der Lotterie wird hier also ein kundenspezifisches Transaktionsprotokollkonto benötigt. Ein solches kundenspezifisches Konto wurde in der Software-Referenzarchitektur `Subskription` von digitalen und physikalsichen Gütern verwendet. Abbildung 7-5 zeigt eine vereinfachte Darstellung der Komponente und ihr Zusammenspiel mit der Systemumgebung. Die vollständige Architektur wurde in Abschnitt 6.2.2 auf Seite 180 erläutert.

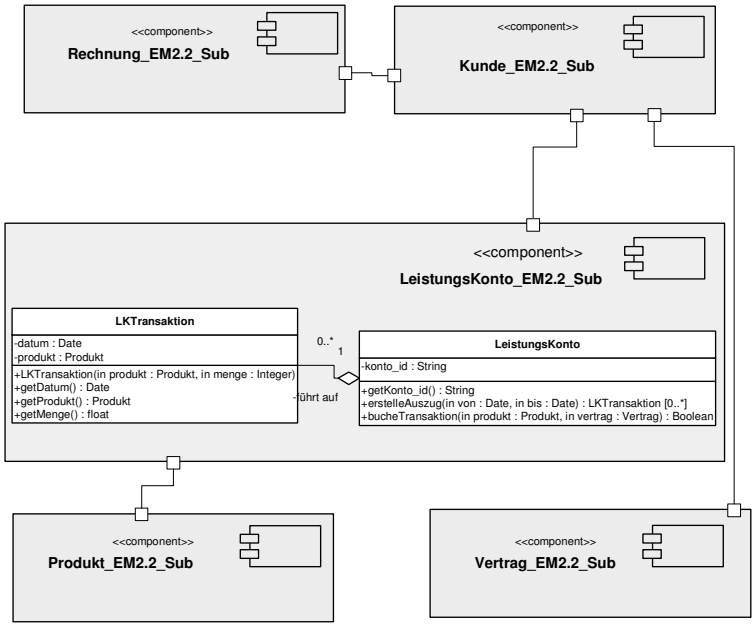


Abbildung 7-5: Die Komponente LeistungsKonto und ihr Zusammenspiel mit den übrigen Komponenten der Software-Referenzarchitektur Subskription von digitalen Gütern

Die Referenzarchitektur sieht vor, dass jedem Kundenobjekt ein Leistungskonto zugewiesen ist, dass die von ihm in Anspruch genommenen Produkte protokolliert. Dieses Konstrukt kann für die Lotterie-Plattform wiederverwendet werden, wobei das Konto in `Spielhistorie` umbenannt wird. Die grundsätzliche Gestaltung der Komponente mit ihren Methoden kann ebenfalls in angepasster Form erneut eingesetzt werden.

Insgesamt wurden somit Komponenten aus den zwei Erlösmodellen EINZELTRANSAKTION (in beiden Varianten) und SUBSKRIPTION identifiziert, die für die Realisierung der Lotterie-Plattform eingesetzt werden können. Aufbauend auf diesen zu übernehmenden Komponenten der Software-Referenzarchitekturen wird im folgenden Abschnitt die angepasste Softwarearchitektur der Lotterie-Plattform vorgestellt.

7.3 Softwarearchitektur der Lotterie-Plattform

Das Klassendiagramm mitsamt den Komponenten ist in Abbildung 7-6 zu sehen.

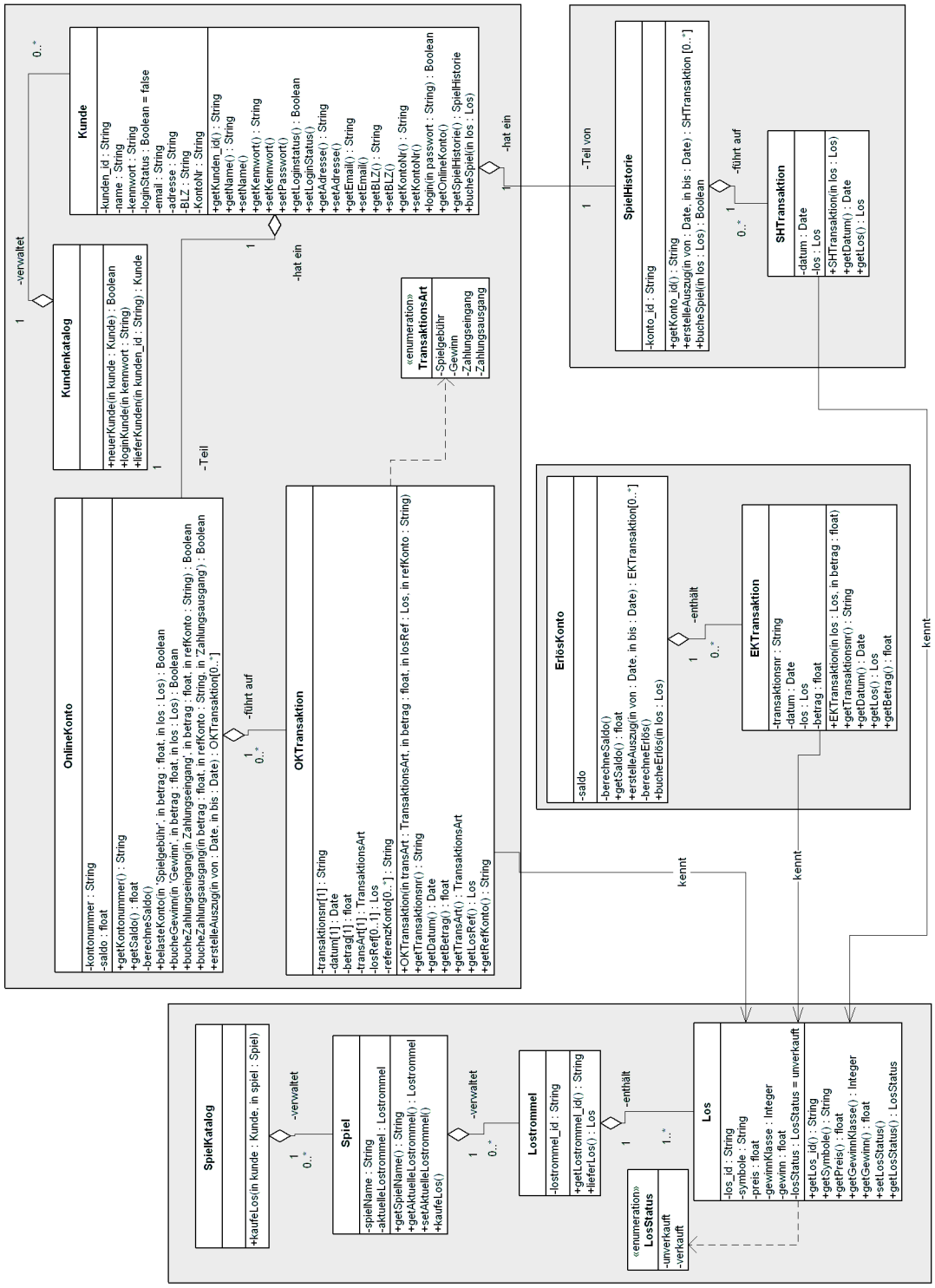


Abbildung 7-6: Statische Architektur des Erlösmodells der Internetlotterie-Plattform

Die Softwarearchitektur umfasst die Komponenten `LotterieKunde`, `RubbelLos`, `SpielHistorie` und `ErlösKonto`. Die folgende Abbildung zeigt die Komponente des Kunden mit ihren Klassen.

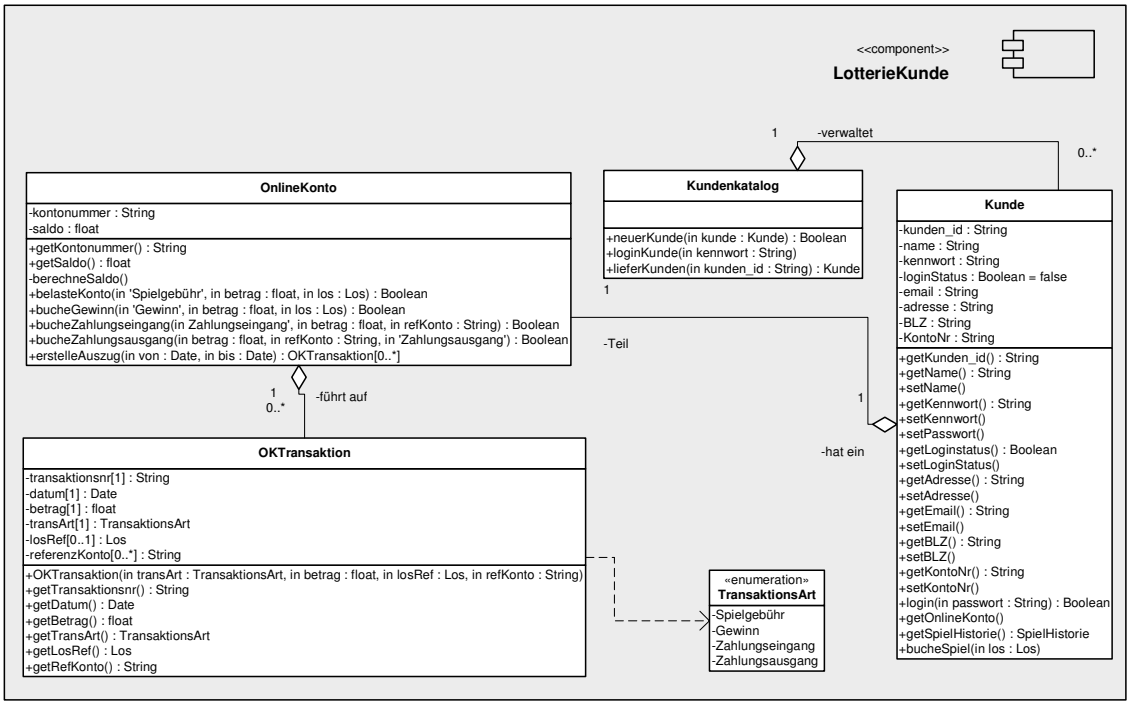


Abbildung 7-7: Klassendiagramm der Komponente LotterieKunde

Der Kunde weist die vertrauten Klassen `Kundenkatalog` und `Kunde` auf, wobei erste unverändert übernommen wurde und die zweite auf die Lotterie angepasste Attribute aufweist. Die zentrale Methode zur Buchung der Kundentransaktionen bildet `bucheSpiel`. Die Methode ist weitgehend von der Methode `bucheTransaktion` aus der Referenzarchitektur Subskription von digitalen und physikalischen Gütern übernommen worden (eine Dastellung der Klasse und Beschreibung der Methode ist in der Architekturbeschreibung ab Seite 182 nachzulesen), wurde hier jedoch auf das Verbuchen der Rubbelloskäufe angepasst. Ihre Funktionsweise wird im Sequenzdiagramm in Abbildung 7-12 auf Seite 286 ersichtlich.

Das aus der Referenzarchitektur bekannte Debitorenkonto wurde hier als `OnlineKonto` umbenannt und geringfügig erweitert. Um unterschiedliche Transaktionsarten buchen zu können, stehen die Methoden `belasteKonto` (Abbuchung der Losgebühr), `bucheGewinn` (Gutschreiben eines Gewinns), `bucheZahlungseingang` (Gutschrift von Zahlungseingängen vom Bankkonto des Kunden) sowie `bucheZahlungsausgang` (zum Überweisen von Beträgen auf das externe Bankkonto des Kunden). Buchungen des Lospreises sowie von Gewinnen speichern zusätzlich eine Referenz auf das entsprechende Los mit ab, um die von der Revision geforderte lückenlose Nachvollziehbarkeit zu gewähr-

leisten. Ein Beispiel einer Instanz von `OnlineKonto` zeigt der Screenshot in Abbildung 7-14 auf Seite 287. Bevor darauf detailliert eingegangen wird, werden jedoch zunächst noch die übrigen Komponenten behandelt.

Das Produkt der Lotterie in Form der Rubbellose musste weitgehend neu gestaltet werden, da es sehr spezifische Eigenschaften aufweist. Dazu dient die Komponente `Rubbel-Los`, die in Abbildung 7-8 zu sehen ist. Die Lotterie bietet mehrere unterschiedliche Spiele an. Jedes Spiel wird in Serien realisiert, wobei eine Serie immer genau eine Umsetzung des Gewinnplans ist, der dem Spiel zugrunde liegt. In jeder Serie sind also eine genau festgelegte Anzahl von Losen, aufgeteilt in die unterschiedlichen Gewinnklassen. Die Umsetzung einer Serie erfolgt durch eine Lostrommel, die durch eine eigene Klasse zu realisieren ist. Eine Instanz eines Spielobjektes muss ermitteln können, welche Lostrommel die derzeit gültige ist, aus der die Lose über ein Zufallsverfahren zu ziehen sind. Eine Lostrommel verwaltet die einzelnen Lose. Ein Losobjekt enthält Auskunft über seinen Preis, seine graphische Repräsentation, seine Gewinnklasse und den zugehörigen Gewinn. Ein Status gibt darüber hinaus Auskunft, ob es bereits verkauft wurde oder noch nicht.

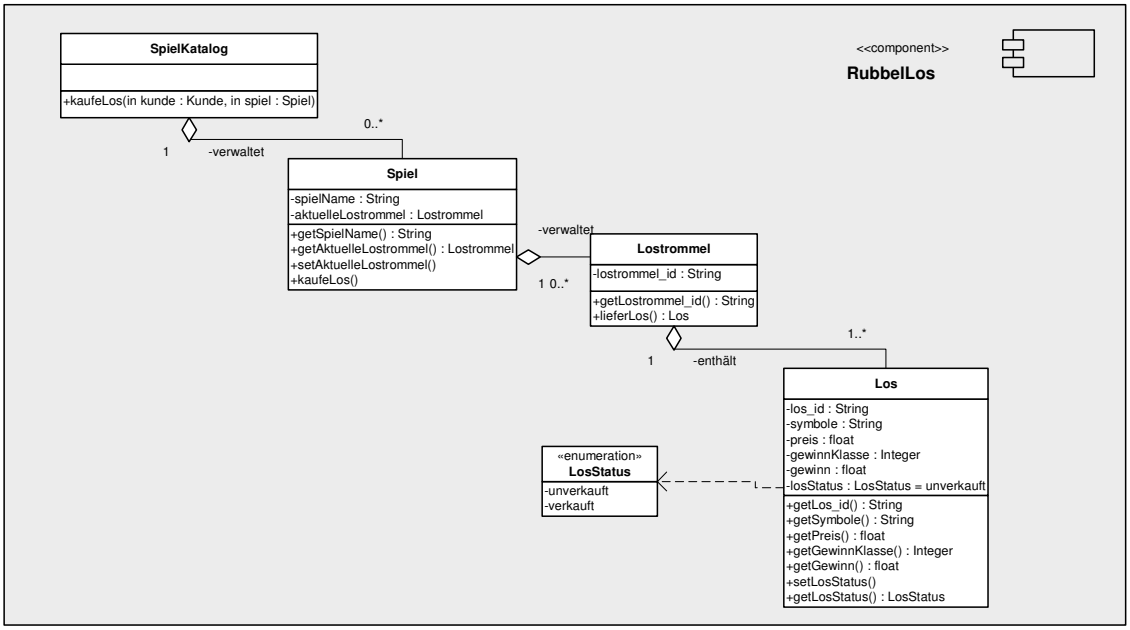


Abbildung 7-8: Klassendiagramm der Komponente RubbelLos

Die folgenden Abbildungen Abbildung 7-9 und Abbildung 7-10 zeigen die graphische Umsetzung eines Loses innerhalb der GUI. Sie sind als Screenshots von der implementierten Anwendung erstellt worden.

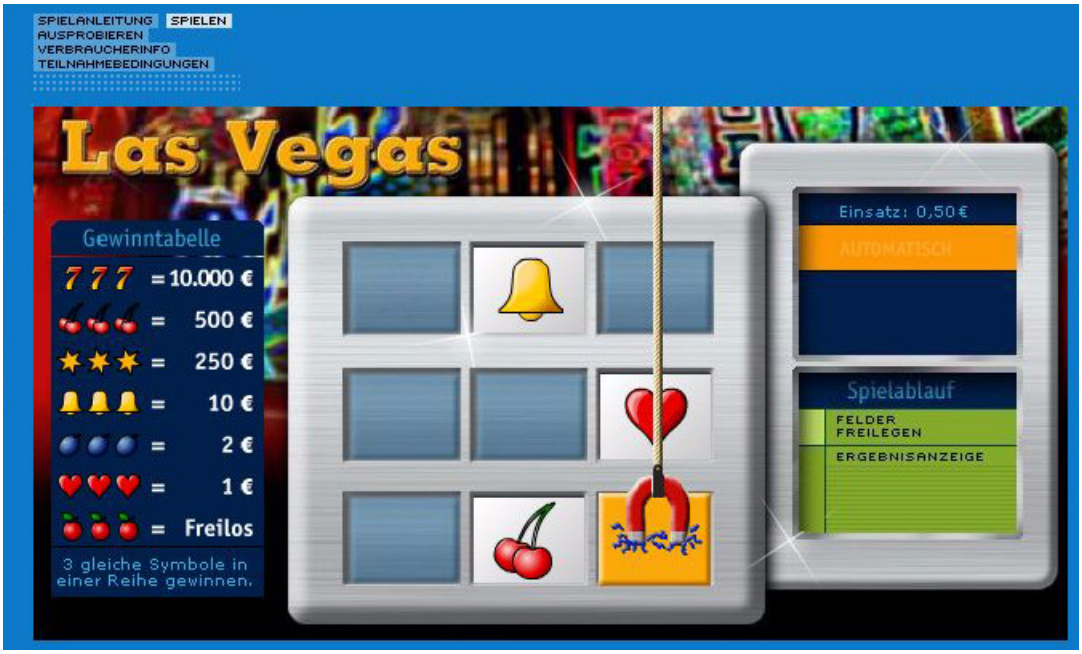


Abbildung 7-9: Visualisiertes Freirubbeln eines Loses der Sofort-Lotterie 'Las Vegas'

Abbildung 7-10: Visualisiertes Freirubbeln eines Loses der Sofort-Lotterie 'win2'

Die Komponenten `ErlösKonto` als ein zentrales und die `SpielHistorie` als kundenspezifische Protokollkonten sind wie bereits erwähnt aus den Referenzarchitekturen EINZELTRANSAKTION und SUBSKRIPTION (jeweils in der Variante für digitale Güter) übernommen und nur gering angepasst worden. Auf eine detaillierte Vorstellung wird hier des-

halb verzichtet. Ihre Funktionsweise wird sehr gut ersichtlich, wenn man den Ablauf eines Loskaufs betrachtet. Dieser Ablauf wird durch die folgenden Sequenzdiagramme wiedergegeben.

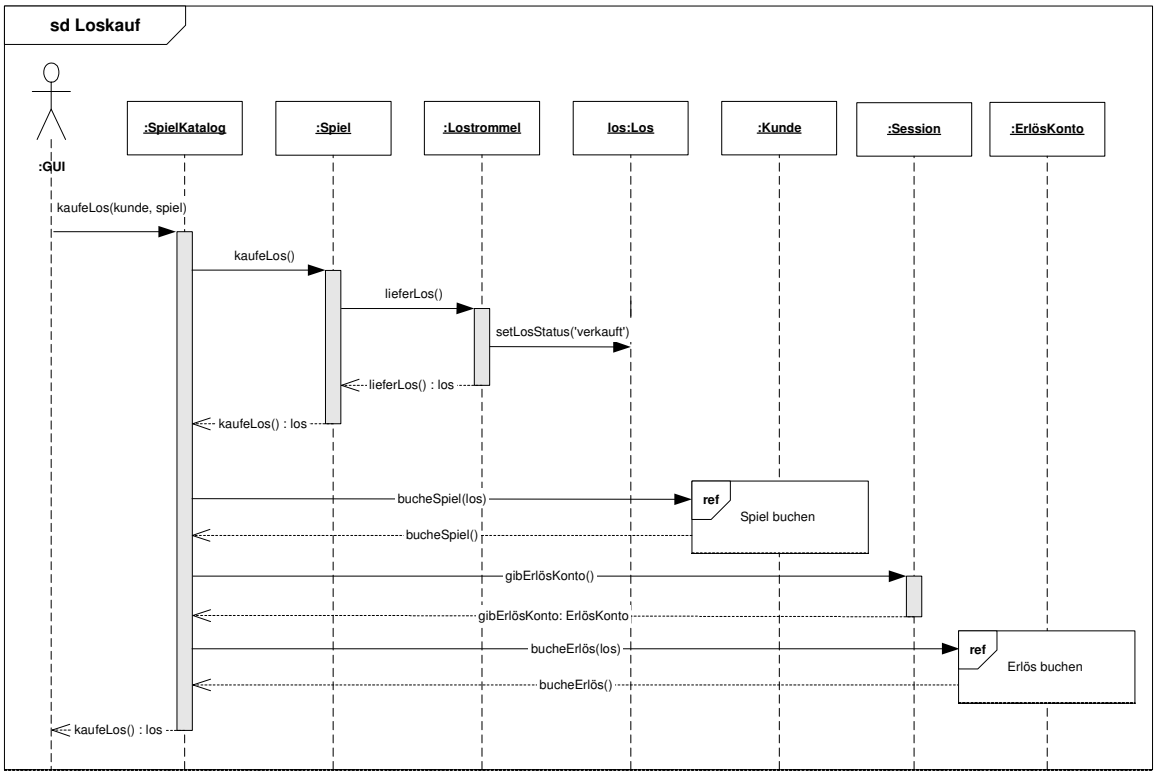


Abbildung 7-11: Sequenzdiagramm Loskauf

Das Diagramm in Abbildung 7-11 zeigt das Zusammenspiel der beteiligten Klassen. Ein Aufruf der GUI veranlasst den `SpielKatalog` unter Einbeziehung der Klassen `Spiel` und `Lostrommel`, ein Los zu ermitteln.

Bevor dieses jedoch an die GUI zurückgeliefert wird, wo eine graphische Animation die Ergebnisermittlung suggeriert, werden alle relevanten Buchungen durchgeführt. Die Belastung des monetären Kontos des Kunden und eine etwaige Gewinngutschrift erfolgen somit bereits vor dieser Darstellung, um bei einer eventuellen Unterbrechung der Transaktion während dieser Animation keinen Datenverlust zu erleiden. Hieran wird deutlich, dass das Spielerlebnis im Zusammenhang mit den Rubbellosen lediglich animiert wird und der Kunde keinen Einfluss auf das Ergebnis ausüben kann. Neben der monetären Verbuchung des Loses erfolgt auch ein Eintrag in die Spielhistorie des Kunden. Die folgende Abbildung 7-12 zeigt die Verbuchung des Loses für den Kunden.

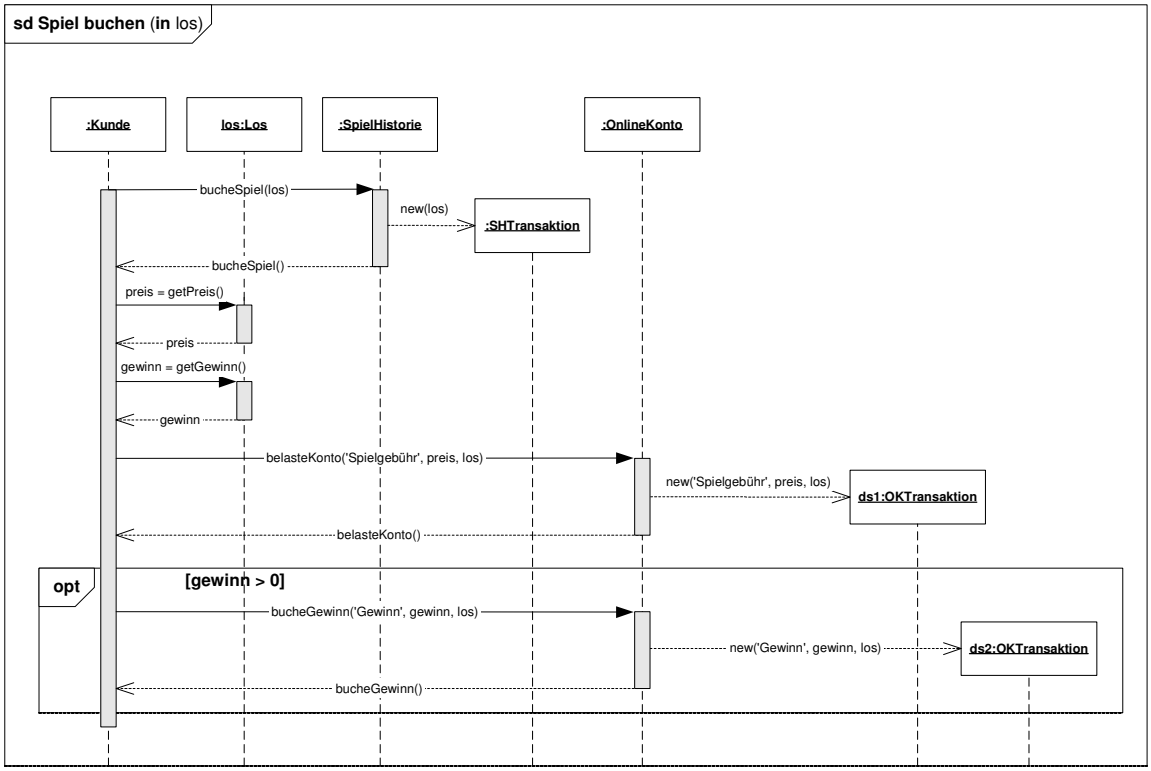


Abbildung 7-12: Sequenzdiagramm zur Spielbuchung auf dem monetären OnlineKonto und in die Spielhistorie des Kunden

Zunächst erfolgt ein Eintrag in der Spielhistorie des Kunden. Neben einem genauen Zeitstempel wird eine Referenz auf das Los gespeichert, so dass sämtliche Daten verfügbar sind. Das Ergebnis als Umsetzung innerhalb des Portals zeigt der folgende Screenshot.

Spielarchiv

SPIELFILTER Sofortlotterie

Vor- und Zuname **Thorsten Weber**
Referenznummer **110000898137**

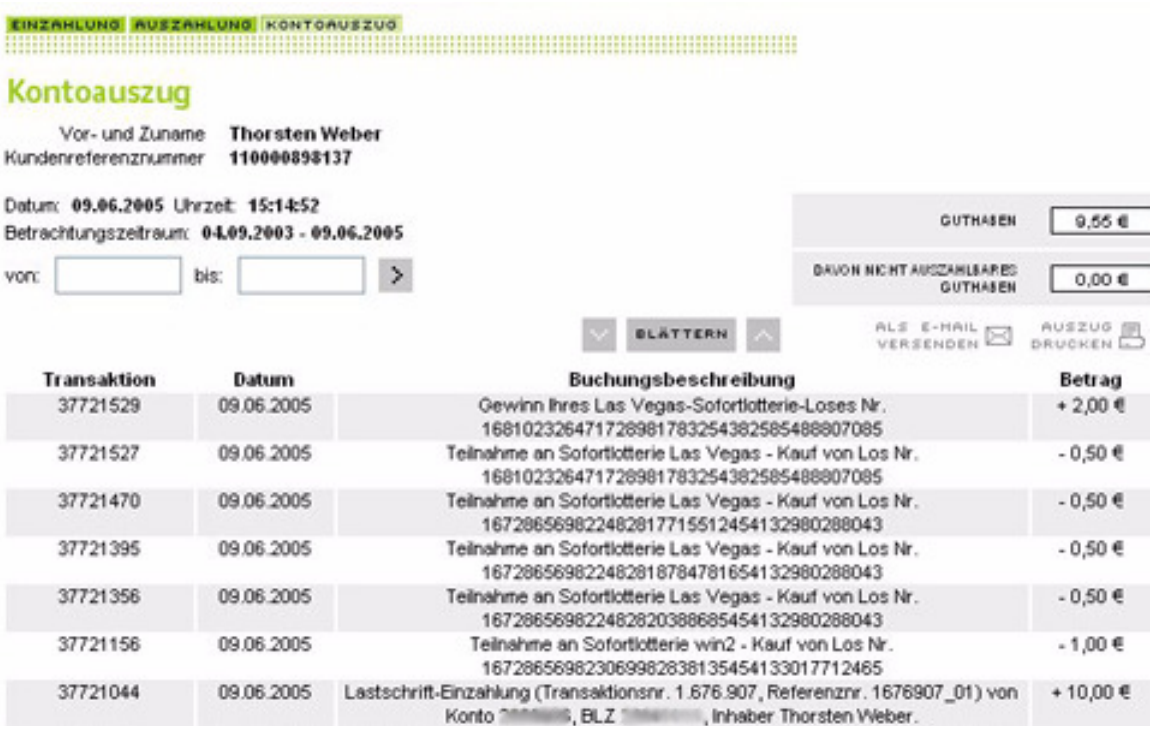
ARCHIV DRUCKEN

Spieldatum	Spielzeit	Spielart	Einsatz	Gewinn	Spielstatus	Spieldetails
09.06.2005	15:16:43	Sofortlotterie - Win2	1,00 €	Niete	archiviert	
09.06.2005	15:13:50	Sofortlotterie - Las Vegas	0,50 €	2,00 €	archiviert	
09.06.2005	15:12:36	Sofortlotterie - Las Vegas	0,50 €	Niete	archiviert	
09.06.2005	15:10:12	Sofortlotterie - Las Vegas - mit Freilosgewinn	0,50 €	Niete	archiviert	
09.06.2005	15:09:16	Sofortlotterie - Las Vegas	0,50 €	Niete	archiviert	
09.06.2005	15:05:30	Sofortlotterie - Win2	1,00 €	Niete	archiviert	

Abbildung 7-13: Screenshot der Spielhistorie eines Spielers

Es ist erkennbar, dass zu jedem Los die Kosten (Einsatz) sowie der eventuelle Gewinn ausgewiesen werden.

Nachdem der Eintrag in der Spielhistorie gebucht ist, wird der Betrag vom OnlineKonto des Kunden abgebucht, wozu ein neuer Datensatz angelegt wird. Falls das Los einen Gewinn ausweist, wird dieser unmittelbar nach der Kontobelastung gutgeschrieben, wozu ebenfalls ein neuer Datensatz angelegt wird. Alle Buchungen enthalten eine Referenz auf das Los, um eine vollständige Protokollierung sicherzustellen. Die folgende Abbildung 7-14 zeigt einen Screenshot der implementierten Anwendung.



Transaktion	Datum	Buchungsbeschreibung	Betrag
37721529	09.06.2005	Gewinn Ihres Las Vegas-Sofortlotterie-Loses Nr. 168102326471726981783254382585488807085	+ 2,00 €
37721527	09.06.2005	Teilnahme an Sofortlotterie Las Vegas - Kauf von Los Nr. 168102326471726981783254382585488807085	- 0,50 €
37721470	09.06.2005	Teilnahme an Sofortlotterie Las Vegas - Kauf von Los Nr. 167286569822482817715512454132980288043	- 0,50 €
37721395	09.06.2005	Teilnahme an Sofortlotterie Las Vegas - Kauf von Los Nr. 167286569822482818784781654132980288043	- 0,50 €
37721356	09.06.2005	Teilnahme an Sofortlotterie Las Vegas - Kauf von Los Nr. 167286569822482820388685454132980288043	- 0,50 €
37721156	09.06.2005	Teilnahme an Sofortlotterie win2 - Kauf von Los Nr. 167286569823069982838135454133017712465	- 1,00 €
37721044	09.06.2005	Lastschrift-Einzahlung (Transaktionsnr. 1.676.907, Referenznr. 1676907_01) von Konto [illegible], BLZ [illegible], Inhaber Thorsten Weber.	+ 10,00 €

Abbildung 7-14: Screenshot des monetären Onlinekontos des Spielers

Jeder Datensatz wird mit einer eigenen Transaktionsnummer versehen. Anhand der obersten beiden Buchungen ist ersichtlich, dass das Konto zunächst mit dem Preis des Loses belastet (Transaktion 37721527) und anschließend ein Gewinn des selben Loses gutgeschrieben wurde (Transaktion 37721529). Das Beispiel zeigt mit dem untersten Datensatz auch die Buchung eines externen Zahlungseingangs, der mittels Lastschrift eingezogen wurde (Transaktion 37721044). Dieser Eintrag ist das Ergebnis der Methode `bucheZahlungseingang` der Klasse `OnlineKonto`.

Analog zu der Buchung der Einsätze und Gewinne auf dem Onlinekonto des Kunden erfolgt die Buchung der Erlöse für die Lotterie auf dem zentralen Konto. Den Ablauf zeigt die folgende Darstellung in Abbildung 7-15.

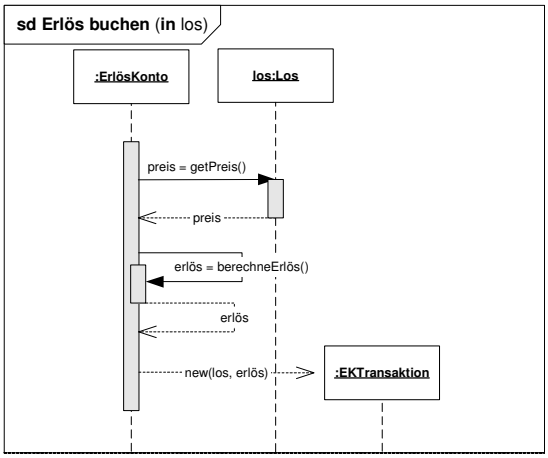


Abbildung 7-15: Sequenzdiagramm zur Buchung der Erlöse

Die Ermittlung des Erlösanteils wurde in der Methode `berechneErlös` realisiert, um eventuelle Änderungen nicht in den Produkten selbst zu hinterlegen, sondern zentral verwalten zu können.

7.4 Schlussfolgerung

In diesem Kapitel wurde eine Validierung der Ergebnisse dieser Arbeit vorgenommen. Wie einleitend in der Arbeit erläutert wurde, wurden zwei Zielstellungen verfolgt: eine Klassifikation von Erlösmodellen des e-Business und eine Aufstellung eines Kataloges von Software-Referenzarchitekturen für jedes der klassifizierten Erlösmodelle. Die Umsetzung dieser beiden Ziele wurde in den Kapiteln 4 und 6 vorgestellt und bildet das Ergebnisse der Arbeit.

Die Validierung wurde anhand des Geschäftsmodells einer Internetlotterie-Plattform durchgeführt. Dieses Geschäftsmodell weist einige sehr spezifische Eigenschaften auf. Vor allem die Berücksichtigung von Gewinnen bei der Umsetzung der Produkte und die daraus resultierenden Buchungen auf den Konten sind hierfür Beispiele. Darüber hinaus besteht eine Besonderheit darin, dass die Kunden der Lotterie zunächst ihr Onlinekonto mit Guthaben versehen müssen, bevor sie an den Spielen teilnehmen können. Der Geschäftsmodellbetreiber tritt also nicht in Vorleistung und es wird auch kein externer Dienstleister zur Zahlungsdurchführung eingesetzt.

Entsprechend des Ansatzes der Arbeit soll auch für ein solch spezifisches Geschäftsmodell eine Einordnung des Erlösmodells in die hier vorgestellte Klassifikation möglich sein. Diese Einordnung wurde in diesem Kapitel erfolgreich durchgeführt. Zwar war es nicht möglich, eine eindeutige 1:1 Zuordnung vorzunehmen, jedoch zeigte eine Betrachtung der Klassifikation der Internetlotterie-Plattform, dass eine Überschneidung mit drei verschiedenen Erlösmodellvarianten vorlag. Anhand der Referenzarchitekturen dieser Varianten konnten Komponenten identifiziert werden, die entweder weitgehend übernommen werden konnten oder gute Entwurfsvorlagen darstellten. Die Software-Referenzarchitekturen konnten somit

in ihrer Rolle als abstrakte Entwurfsvorlagen eingesetzt werden und erleichterten auf diese Weise den Entwurf eines wesentlichen Bestandteils der konkreten Softwarearchitektur erheblich.

Darüber hinaus wurde auch deutlich, dass die Klassifikation des Erlösmodells anhand einer Beschreibung des Geschäftsmodells und somit ohne technische Kenntnisse erfolgte. Auf diese Weise wurde die Einbeziehung unterschiedlicher Personengruppen mit abweichenden Kenntnissen während des Entwicklungsprozesses wesentlich erleichtert. Dies war eine der zu lösenden Problemstellungen dieser Arbeit, wie in der Einleitung in Kapitel 1 erläutert wurde.

8 Bewertung und Ausblick

Die Themengebiete der vorliegenden Arbeit sind Geschäfts- und Erlösmodelle des e-Business sowie Software-Referenzarchitekturen als Grundlage der Realisierung dieser Modelle. Das e-Business hat aufgrund der technologischen Möglichkeiten durch den kommerziellen Einsatz des Internet zahlreiche neue Geschäftsmodelle hervorgebracht, und es ist zu erwarten, dass diese Entwicklung auch in Zukunft anhält. Dabei wurden zum einen bestehende Geschäftsmodelle der traditionellen Geschäftswelt auf die neuen Rahmenbedingungen des e-Business angepasst und zum anderen gänzlich neue Formen von Geschäftsmodellen erfunden. Ein Grund für das Entstehen neuer Geschäftsideen und deren Konkretisierung durch Geschäftsmodelle ist die Möglichkeit, im e-Business neue, zuvor nicht realisierbare Erlösmodelle umzusetzen. Die Erlösmodelle sind somit nicht nur ein wesentlicher Teil der Geschäftsmodelle, sondern bilden in einigen Fällen darüber hinaus das entscheidende Element einer neuartigen Geschäftsidee im e-Business. Um von einem Geschäftsmodell zu einer lauffähigen Anwendung zu gelangen, werden Softwarearchitekturen benötigt. Sie stellen den Bauplan der Softwareanwendung dar, die im e-Business als Realisierung des Geschäftsmodells interpretiert werden kann. Um die Gestaltung dieser Architekturen zu erleichtern und zu beschleunigen, können entsprechende Referenzarchitekturen als abstrakte Entwurfsvorlagen verwendet werden, sofern sie existieren. Mit der Aufstellung eines Kataloges von Software-Referenzarchitekturen für Erlösmodelle des e-Business als wesentlicher Bestandteil eines Geschäftsmodells befasst sich diese Arbeit.

Detaillierter betrachtet stellen sich bei der Umsetzung eines konkreten Geschäftsmodells in eine lauffähige Softwareanwendung folgende Problemstellungen, die bereits im einleitenden Kapitel aufgeführt worden sind: Wie können fachlich und kaufmännisch orientierte Personen in die Konzeption von Softwarearchitekturen mit einbezogen werden und anhand welcher Instrumente kann mit ihnen über wichtige Designentscheidungen diskutiert werden? Wie können neuartige Geschäftsmodelle umgesetzt werden, ohne dass die Softwarearchitektur gänzlich neu entworfen, sondern indem auf bestehende Ansätze zurückgegriffen wird? Inwiefern können fachlich orientierte Beschreibungen der Geschäftsmodelle dazu verwendet werden, technische Komponenten zu ermitteln, die für die umsetzende Softwareanwendung benötigt werden? Wie können Bestandteile der Geschäftsmodelle identifiziert werden, die mittels Komponenten in anderen Softwarearchitekturen bereits umgesetzt wurden und die damit wiederverwendet werden können?

Um diese Problemstellungen zu lösen, wurden zwei Ziele formuliert. Das wesentliche Ziel bestand in der Aufstellung des bereits erwähnten Kataloges von Software-Referenzarchitekturen für Erlösmodelle des e-Business. Da die Erlösmodelle ein wesentlicher Bestandteil des Geschäftsmodells sind, können die zugehörigen Software-Referenzarchitekturen als Entwurfsvorlage für einen bedeutenden Teil der gesamten Softwarearchitektur des Geschäftsmodells verwendet werden. Als ein erstes Teilziel war dazu jedoch zunächst eine Klassifikation der Erlösmodelle aufzustellen. Diese Klassifikation musste derart gestaltet werden, dass sie Schlussfolgerungen auf die Softwarearchitekturen ermöglicht und somit technische Aspekte berücksichtigt. Um jedoch der Problematik nachzugehen,

dass auch Personengruppen mit fachlichen und kaufmännischen Kenntnissen in die Entwicklung der Softwarearchitektur frühzeitig mit einzubinden sind, sollten die Klassifikationskriterien als nicht-technische Beschreibungen der Geschäftsmodelle aufgestellt werden.

Für die Umsetzung beider Ziele wurde zunächst das Teilziel der Klassifikation angegangen. Dazu wurde in Kapitel 3 anhand einer umfangreichen Literaturrecherche analysiert, welche Ansätze zur Klassifikation von Geschäfts- und Erlösmodellen bereits existieren. Es wurde dabei untersucht, inwiefern sie für die hier vorliegende Aufgabenstellung hilfreich sind oder gar übernommen werden können. Zusammenfassend hat sich gezeigt, dass es einen breiten Konsens darüber gibt, dass Erlösmodelle ein wesentliches Element von Geschäftsmodellen bilden, die gelieferten Klassifikationen der Geschäfts- oder der Erlösmodelle jedoch nicht vollständig ausreichen, um Aussagen über die Softwarearchitektur ableiten zu können. Dieser Aspekt einer technischen Umsetzung der Geschäfts- und Erlösmodelle in Form einer Softwareanwendung mitsamt der dazu benötigten Softwararchitektur wurde bisher in der Literatur nur unzureichend berücksichtigt. Aufbauend auf diesen Erkenntnissen wurden in Kapitel 4 zunächst eigene Klassifikationskriterien aufgestellt und anschließend die Klassifikation der Erlösmodelle durchgeführt. Als erstes Ergebnis dieser Arbeit ergaben sich sechs verschiedene Erlösmodelle des e-Business in insgesamt 16 verschiedenen Varianten.

Hervorzuheben ist besonders die Eigenschaft dieser Klassifikation, dass sie vor dem Hintergrund aufgestellt wurde, von ihr ausgehend Software-Referenzarchitekturen abzuleiten. Wie die vorherige Literaturanalyse gezeigt hat, stellt sie somit eine wesentliche Neuerung im Gegensatz bisheriger Klassifikationen dar. Gleichzeitig orientieren sich die Klassifikationskriterien jedoch an einer nicht-technischen Beschreibung des jeweiligen Geschäftsmodells, dass eine der Erlösmodellvarianten aufweist. Somit wurde mit der Klassifikation ein Instrument geschaffen, dass den Übergang vom *fachlich* orientierten Geschäftsmodell auf die zugrundeliegende *technisch* orientierte Softwarearchitektur unterstützt. Es trägt somit dazu bei, Personengruppen unterschiedlicher thematischer Ausrichtung in den Herleitungsprozess einer Softwarearchitektur zu integrieren.

Ein weiterer Vorteil der Klassifikation ergibt sich daraus, dass die identifizierten Erlösmodelle zunächst unabhängig von konkreten Geschäftsmodellen des e-Business sind und dass somit eine Anwendbarkeit in großem Umfang unterstützt wird. Während sich bestehende und neue Geschäftsmodelle in vielen Details voneinander unterscheiden, weisen viele von ihnen ein gemeinsames und hier klassifiziertes Erlösmodell auf, wie die Analyse der existierenden themenverwandten Arbeiten bestätigt hat. Somit kann für einen großen Umfang von Geschäftsmodellen im e-Business mit dem jeweiligen Erlösmodell ein wesentlicher Bestandteil in der hier aufgestellten Klassifikation identifiziert werden. Anhand des zweiten und wesentlichen Ergebnisses dieser Arbeit existieren auch die Entwurfsvorlagen für die notwendige Softwarearchitektur dieser Erlösmodelle.

Ausgehend von diesem Teilergebnis wurde in Kapitel 6 der Katalog der Software-Referenzarchitekturen aufgestellt, indem zu jeder Erlösmodellvariante eine geeignete Software-Referenzarchitektur aufgestellt wurde. Die Architekturen wurden dabei sowohl hinsichtlich ihrer statischen Struktur als auch hinsichtlich ihres dynamischen Verhaltens mit verschiedenen Diagrammtypen beschrieben. Dabei wurden Komponenten identifiziert, die jeweils

einen bestimmten Funktionsbereich umsetzen und ihre Dienste der Restarchitektur zur Verfügung stellen. Es wurde darauf geachtet, dass wiederverwendbare Komponenten in Referenzarchitekturen anderer Erlösmodellvarianten erneut eingesetzt wurden. Konnten die Komponenten nicht identisch übernommen werden, wurden ähnliche Anforderungen durch Anpassungen dieser Komponenten umgesetzt. Auf diese Weise enthält der Katalog eine Sammlung von Komponenten, die gemeinsam als Bausatz der Software-Referenzarchitekturen angesehen werden kann.

Eine Bewertung des Kataloges der Software-Referenzarchitekturen ergibt sich nur durch den Zusammenhang mit der Klassifikation der Erlösmodelle. Da diese Klassifikation, wie bereits erwähnt, ein breites Spektrum potenzieller Erlösmodelle berücksichtigt, stehen mit den Referenzarchitekturen Entwurfsvorlagen für zahlreiche unterschiedliche Geschäftsmodelle zur Verfügung. Die detaillierte Granularität der Software-Referenzarchitekturen mit den aufgeführten Komponenten und Klassen kann dabei sehr gut verwendet werden, um einfacher und schneller eine konkrete Architektur abzuleiten, als wenn diese von Grund auf neu entwickelt werden müsste.

Dies hat die Validierung in Kapitel 7 bestätigt. Die Entwicklung der konkreten Softwarearchitektur für eine Internetlotterie-Plattform auf Grundlage der Software-Referenzarchitekturen hat gezeigt, dass auch ein Geschäftsmodell mit sehr sepzifischen Eigenschaften und Anforderungen in die vorhandene Erlösmodellklassifikation eingeordnet werden kann und dass die dafür vorliegenden Software-Referenzarchitekturen eine geeigete Vorlage für die konkrete Umsetzung gebildet haben. Die Einordnung des Erlösmodells anhand der hier aufgestellten Klassifikationskriterien hat darüber hinaus gezeigt, dass ein fachliches Verständnis des Geschäftsmodells dazu ausreicht und dass auf diese Weise auch nicht-technisch geschulte Personen in die Herleitung der Softwarearchitektur mit eingebunden werden können.

Zusammenfassend lässt sich somit sagen, dass die Ergebnisse dieser Arbeit einen wesentlichen Beitrag dazu leisten, die Realisierung von Geschäftsmodellen des e-Business in vereinfachter und beschleunigter Form durchzuführen und die unterschiedlichen, daran beteiligten Personengruppen zu unterstützen.

Ausblickend stellt sich die Frage, inwiefern die hier aufgestellten Software-Referenzarchitekturen für Erlösmodelle zusätzlich um weitere Bestandteile der Geschäftsmodelle erweitert werden können. Es wurde in dieser Arbeit mehrfach darauf eingegangen, dass ein Geschäftsmodell zahlreiche Bestandteile aufweist und dass das Erlösmodell nur ein Element bildet. Somit könnte untersucht werden, inwiefern darüber hinaus weitere Teile allgemeingültige Bestandteile existieren, die ebenfalls klassifiziert und mit geeigneten Software-Referenzarchitekturen beschrieben werden könnten. Auf diese Weise könnte der Herleitungsprozess der Gesamtarchitektur weiter beschleunigt werden. Die Schwierigkeit dieser Aufgabe ergibt sich vor allem in der notwendigen Allgemeingültigkeit dieser Bestandteile, da nur so gewährleistet werden kann, dass ein breites Spektrum von Geschäftsmodellen abgedeckt wird.

Darüber hinaus kann es in Zukunft notwendig sein, die hier vorgestellte Klassifikation der Erlösmodelle an eine veränderte Realität anzupassen. Aufgrund der fortschreitenden technologischen Entwicklung einerseits sowie sich ändernde Verhaltensweisen der Nutzer andererseits können neue, bisher noch nicht bekannte und hier auch nicht berücksichtigte Erlösmodelle entstehen, die dann in die Klassifikation mit aufgenommen und für die geeigente Software-Referenzarchitekturen aufgestellt werden müssten. Diesbezüglich werden erst die nächsten Jahre zeigen, wie langlebig die hier aufgestellte Klassifikation ist beziehungsweise wie schnell eine Veränderung der Rahmenbedingungen eintreten wird.

Literatur

[AGI02] Arbeitsgemeinschaft Internet Research (AGIREV): Der Online-Reichweiten-Monitor 2002 I, 2002, Online unter http://www.ems.guj.de/download/Monitor2002.pdf, zuletzt abgerufen am 04.06.2002

[All97] Allen, R.J.: A Formal Approach to Software Architectures. Technical Report CMU-CS-97-144, Carnegie Mellon University, Mai 1997.

[AZ00] Amit, R.; Zott, C.: Value Drivers of E-Commerce Business Models. In: 20th Annual International Conference of the Strategic Management Society, Vancouver Canada, 2000

B

[Bau99] Baudisch, P.: Joining collaborative and content-based ltering. In Interacting with Recommender Systems; Online Proceedings of the CHI Workshop, 1999.

[BCK97] Bass, L.; Clements P.C., Kazman, R.: Software Architecture in Practice, Addison Wesley, 1997

[BGH00] Bröckers, A.; Gruhn, V.; Hartmann, A.; Rudolf, R.: Komponentenbasierte Softwareentwicklung - Von der Vision zur Architektur. Innovationsbericht 2. adesso AG. ISSN 1437-948, 2000

[BJR99] Booch, G.; Jacobson, I.; Rumbraugh, J.: Unified Modelling Language, Addison Wesley, 1999

[BL00] Bartelt, A; Lamersdorf, W.: Geschäftsmodelle des Electronic Commerce: Modellbildung und Klassifikation. In: Verbundtagung Wirtschaftsinformatik 2000, S. 17-29; Shaker, Aachen, 2000

[BLS00] Boulton, R.; Libert, B.; Samek, S.: A Business Model for the new economy. In: Journal of Business Strategy, Vol. 21 (2000), S. 29-35, 2000

[BPS99] Barua, A.; Pinnell, J.; Shutter, J.; Whinston, A.B.: Measuring Interneteconomy: an exploratory paper. Working paper, University of Texas, Austin, July 1999, Online unter http://cism.bus.utexas.edu/works/articles/internet-economy.pdf., zuletzt abgerufen am 04.05.2002

[BS97] Balabanovic, M.; Shoham, Y.: Combining Content-Based and CollaborativeRecommendation. Communications of the ACM, Vol. 40(3), March 1997.

[Buc01] Buchholz, W.: Netsourcing Business Models - Geschäftsmodelle für Einkaufsplattformen. In: Dangelmaier, W., Pape, U., Rüther, M. [Hrsg.]: Die Supply Chain im Zeitalter von E-Business und Global Sourcing, Paderborn 2001, S.37-52

[BW98] Brown, A. W.; Wallnau, K. C.: The Current State of CBSE. In: IEEE Software, Vol. 15, Nr. 5, Seite 37–46, 1998.

[BWL00] Bartelt, A.; Weinreich, H.; Lamersdorf, W.: Kundenorientierte Aspekte der Konzeption von Online-Shops. In: Proceedings Virtuelle Organisation und Neue Medien, Vol. 10, S. 159-172, 2000

[BZF01] Bartelt, A; Zirpins, C; Fahrenholz, D.: Geschäftsmodelle des Electronic Information: Modellbildung und Klassifikation. In: Proceeding Papers Informatik 2001 - Wirtschaft und Wissenschaft in der Network Economy, S. 902-908, 2001, Online unter http://vsis-www.informatik.uni-hamburg.de/papers/bartelt2001_gm_ei.pdf; zuletzt abgerufen am 29.05.2003

C

[CHC01] Clemons, E.K.; Hitt, L.M.; Croson, D.C.: The future of Retail Financial Services: Transparecy, Bypass and Differential Pricing. In: The BRIE-IGCC Economy Projekt, Tracking a Transformation – E-Commerce and the Term of Competition in Industries, Brookings, Washington, 2001

[Cho97] Choi, S.:Strategien von Banken im globalen Wettbewerb; Dt. Univ.-Verl., Wiesbaden, 1997

[CK94] Clements, P.C; Kogut, P.: Features of Architecture Description Languages. Draft of a CMU/SEI Technical Report, Dezember 1994.

[Cle95] Clements, P.C.: A Survey of Architecture Description Languages. In: Eigth InternationalWorkshop on Software Specification and Design. Paderborn, Germany, 1996

[Cle96a] Clements P.C: A Survey of Architecture Description Languages; In: Proc. 8th International Workshop on Software Specification and Design, 1996.

[Cle96b] Clements P.C: From Subroutines to Subsystems: Component-Based Software Development. In: Component-Based Software Engineering, Selected Papers from the Software Engineering Institute, S. 36; IEEE Computer Society Press, 1996.

[CN96] Clements P.C.; Nothrop, L.; Software Architecture: an Executive Overview. In: Component-Based Software Engineering: Selected Papers from the Software Engineering Institute, S. 55-68. IEEE Computer Society Press, 1996.

[Com02] N.N.: ComCult Panel-Report: Online-Nutzung 2001. Online unter http://www.comcult.de/ic/download/comcultpanelreport.pdf, zuletzt abgerufen am 04.06.2002

[CR99] Coar, K. A.; Robinson, D. R. T.: The WWW Common Gateway Interface. Online unter http://www.cgi-spec.golux.com/draft-coar-cgi-v11-03.txt, 1999, zuletzt abgerufen am 16.07.2002

D

[Dac02] Dach, Christian: Vorteile einer Multi-Channel Strategie: Eine nüchterne Betrachtung, ECC-Handel (Hrsg.); Online unter www.ecc-handel.de, 2002 (Auszug in: „Stationärer Handel: Einkaufserlebnis Vorteil gegenüber Online-Shops“, www.einzelhandel.de, 05.03.2002, Online unter: http://www.einzelhandel.de/servlet/PB/menu/1004272_l1/index.html, zuletzt abgerufen am 13.06.2002)

[Den91] Denert, E.: Software Engineering. Springer Verlag, 1991

[DOP01] Dubosson-Torbay, M.; Osterwalter, A.; Pigneur, Y.: eBusiness Model Desing, Classification and Mesurement, Online unter http://citeseer.ist.psu.edu/dubosson.torbay01ebusiness.html, zuletzt abgerufen am 5.10.2004

[DP00] Denninger, S.; Peters, I.: Enterprise JavaBeans. Addison-Wesley, 2000

[DPC00] Deutsche Post und ComConsult: Studie eCommerce Facts 2.0; 2000

[DS94] Davenport, T.H.; Short, J.E.: Information technology and business process redesign. In: Galliers, R.D.; Baker, B.: Strategic Information Management – challenges and strategies in managing information systems, Butterworth-Heinemann, Oxford, 1994

[DW97] Davis, M. J.; Williams R. B.: Software Architecture Characterization. In: Proceedings of the 1997 Symposium on Software Reusability; Boston, 1997

E

[Ear92] Earl, M.J.: Putting Information Technology in its place: a polemic for the nineties. In: Journal of Information Technology, Vol. 7, S. 100-108, 1992

[EG00] Eimeren, Birgit; Gerhard, Heinz: ARD/ZDF-Online-Studie 2000: Gebrauchswert entscheidet über Internetnutzung. In: Media Perspektiven, Vol. 8/2000. S. 338-349, 2000, Online unter http://www.das-erste.de/studie/ardonl00.pdf, zuletzt abgerufen am 04.06.2002

[EGF01] Eimeren, Birgit; Gerhard, Heinz; Frees, Beate: ARD/ZDF-Online-Studie 2001: Internetnutzung stark zweckgebunden. In: Media Perspektiven, Vol. 8/2001. S. 382-397, 2001, Online unter http://www.das-erste.de/studie/ardonl01.pdf, zuletzt abgerufen am 04.06.2002

[EM01] Egyed, A; Medvidovic, N.: Consistent Architectural Refinement and Evolution using the Unied Modeling Language. In: Proceedings of the 1st Workshop on Describing Software Architecture with UML, co-located with ICSE, Mai 2001.

[Ems02] N.N: Online-Monitor Welle 7. In: Electronic Media Service, Gruhner und Jahr, 2002, Online unter http://www.ems.guj.de/download/download.php?file=ems_gfk7te_erhebungswelle.pdf, zuletzt abgerufen am 04.06.2002

[Ems02a] N.N.: Glossar. Online unter http://www.ems.guj.de/glossar/index.html, zuletzt abgerufen am 10.05.2002

F

[FB01] Foegen, M.; Battenfeld, J.: Die Rolle der Architektur in der Anwendungsentwicklung. In: Informatik Spektrum, Vol. 24, Heft 5, 2001

[FL02] Fettke, P.; Loos. P.: Methoden zur Wiederverwendung von Referenzmodellen – Übersicht und Taxonomie. In (Becker, J.; Knackstedt, R. Hrsg.): Referenzmodellierung 2002. Methoden – Modelle – Erfahrungen. Arbeitsbericht Nr. 90 des Instituts für Wirtschaftsinformatik. Universität Münster. S. 9-34.

[FLM00] Forsyth, J.E.; Lavoie, J.; McGuire, T.I.: Segmenting the e-market. In: The McKinsey Quarterly, Nr. 4, 2000

[FHL99] Fugmann, T.; Heinrich, B.; Leist, S.; Winter, R.: Banking im Informationszeitalter – Formen und Gestaltungsfragen von Wertschöpfungsnetzwerken im Bankbereich. Arbeitsbericht des Instituts für Wirtschaftsinformatik an der Universität St. Gallen, Nr. BE HSG/FP BAI/02; Juni 1999; Online unter http://bai.iwi.unisg.ch/publ.htm, zuletzt abgerufen am 30.06.2003

[FS97] Fowler, M.; Scott, K.: UML Distilled – Applying the Standard Object Modeling Language. Addison-Wesley, 1997

[Fux00] Fuxman, A.D.: A Survey of Architecture Description Languages; Februar 2000; Online unter http://citeseer.nj.nec.com/fuxman00survey.html, zuletzt abgerufen am 05.08.2003

G

[GAV00a] Gordijn, J.; Akkermans, J.M.; van Vliet, J.C.: What's in an electronic business model. 12th International Conference on Knowledge Enigneering and Knowledge Management EKAW-2000, Juan-les-Prins, October 2000, Online unter http://www.cs.vu.nl/~gordijn, zuletzt aberufen am 11.07.2002

[GAV00b] Gordijn, J.; Akkermans, J.M.; van Vliet, J.C.: Business modelling is not process modelling. ECOMO-2000 Workshop on Conceptual Modelling Approaches for E-Business, Salt Lake City, October 2000, Online unter http://www.cs.vu.nl/~gordijn, zuletzt abgerufen am 11.07.2002

[GfK01] N.N.: GfK Online-Monitor – Ergebnisse der 7. Untersuchungswelle. Auftraggebergemeinschaft GfK – Online-Monitor, Gesellschaft für Konsumforschung, 2001, Online unter http://www.gfk.de, zuletzt abgerufen am 25.05.2002

[GG00] Gulati, Ranjay; Garino, Jason: Get the right mix of bricks and clicks. In: Harvard Business Review, Vol. 22, May-June 2000

[GMM98] Guttmann, R.H.; Moukas, A.G., Maes, P.: Agents as mediators in electronic commerce. In: Electronic Markets, Vol.8. No. 1, 1998

[Goe03] Goeken, M.: Grundlagen und Ansätze einer Referenzmodellierung für Führungsinformationssysteme. Fachbericht Nr. 03/02, Philips-Universität Marburg, Institut für Wirtschaftsinformatik, 2003, Online unter http://wi.wiwi.uni-marburg.de/WebSite/ web.nsf/SysWebRessources/FB0302/$FILE/FB_03_02.pdf, zuletzt aufgerufen am 05.052005

[Gri98] Griffel, F.: Componentware - Konzepte und Techniken eines Softwareparadigmas. dpunkt-Verlag. 1998.

[GS93] Garlan, D.; Shaw, M.: An Introduction to Software Architecture; In: Advances in Software Engineering and Knowledge Engineering, S. 1-39. World Scientic Publishing Company, 1993.

[GS02] Gruhn, V.; Schneider, M.: EJB 2.0 Anwendungen - Entwurf leistungsfähiger Java-Komponenten, Addison-Wesley, 2002

[GT00] Gruhn, V.; Thiel, A.: Komponentenmodelle, Addison-Wesley, 2000

H

[Han00] Hanson, W. A.: Principles of Internet Marketing, Cincinnati, 2000

[Hay94] Hayes-Roth, F.: Architecture-Based Acquisition and Development of Software: Guidelines and Recommendations from the ARPA Domain-Specic Software Architecture Program. Technical report, Teknowledge Federal Systems, Februar 1994.

[Hil99] Hilliard, R.: Using the UML for Architectural Description. In: Proceedings of UML 99, volume 1723 of Lecture Notes in Computer Science. Springer, 1999.

[HL02] Heinrich, B.; Leist, S.: Bankenarchitektur im Informationszeitalter – Zur Rolle des Geschäftsmodells. In: Business Engineering; Österle, H.; Winter, R. (Hrsg.), Vol. 2, Springer, Berlin u.a., 2002, Online unter http://bai.iwi.unisg.ch/publ.htm, zuletzt abgerufen am 30.06.2003

[Hoq00] Hoque, Faisal: e-Enterprise – Business Models, Architectures and Components; Cambridge University Press, Cambridge, 2000

[HNS95] Hofmeister, C.; Nord, R.L.; and Soni, D: Software Architecture in Industrial Applications; In: International Conference on Software Engineering, S. 196-207, 1995.

[HNS99] Hofmeister, C.; Nord, R.L.; and Soni, D: Describing software architecture with UML. In: Proceedings of Working IFIP Conference on Software Architecture, S. 145-160. Kluwer Academic Publishers, February 1999.

[HR97] Hofmann, C., Renzel, K.: Beschreibungssprachen für Software-Architekturen - Software Architectures and Design Patterns in Business Applications; Technical Report TUM-I9746, S. 201-240, Technische Universität München, Institut für Informatik, November 1997.

[HS99] Hermanns, A.; Sauter, M.: Electronic Commerce – Grundlagen, Potentiale, Marktteilnehmer und Transaktionen. In: Hermanns, A.; Sauter, M,: Management-Handbuch Electronic Commerce; Vahlen, München, 1999

[HSi99] Hagel, J.; Singer, M.: Unbundling the corporation. In: Harvard Business Review, Vol. 21, Nr. 2, 1999

J

[JT99] Jarvenpaa, S.L.; Tiller, E.H.: Integration market, technology, and policy opportunities in e-business strategy. In: Journal of Strategic Information Systems 8 (1999), S. 235-249, 1999

[JH98] Jarzabek, J. and Hitz, M.: Business-Oriented Component-Based Software Development and Evolution. In DEXA Workshop, S. 784788, 1998.

[JRH04] Jeckle, M.; Rupp, C.; Hahn, J.; Zengler, B.; Queins, S.: UML 2.0 glasklar; Carl Hanser Verlag, München, 2004

K

[KA96] Kotler, P.; Armstrong, P.: Principles in Marketing; Prentice Hall, 1996

[Kau98] Kauffels, F.-J.: E-Business; MITP, Bonn, 1998

[KGL00] Klein, S.; Güler, S.; Lederbogen, K.: Personalisierung im elektronischen Handel. Das Wirtschaftsstudium, Vol. 29(1), S. 88-94, 2000.

[KL00] Kogler, B.; Lebowitz, J.: Integrating the e-business model. In: Mortage Banking, Vol. 60, T. 6, 2000, S. 66-76, Washington, DC, 2000

[Klu00] Klug, S.: Komponenten als Basis für das E-Business. Versicherungsmagazin Vol. 10, 2000

[KO98] Kim, J., Oard, D.W.: Implicit Feedback for Recommender Systems. In: Proceedings of the AAAI Workshop on Recommender Systems, July 1998.

[KOR00] Kim, J., Oard, D.W., Romanik, K.: Using implicit feedback for user modeling in Internet and Intranet searching. In: Technical Report, College of Library and Information Services, University of Maryland at College Park, 2000.

[KPT01] Kittle, C.; Petrovic, O.; Teksten, R.D.: Developing Business Models for eBusiness. International Conference on Electronic Commerce 2001, Vienna, October 31. November 4., 2001.

[Kra00] Krause, J.: TK- und IT-Produkte erfolgreich über das Netz vertreiben; Symposion, Düsseldorf, 2000

[Kro02] Kropsch, P.: eBusiness auf dem Prüfstand: Benchmarking Europa, Arbeitskreis 4: digitale Inhalte: Contentsyndizierung; 2002. Online unter http:// www.wko.at/alpbach/bm/dok/Kropsch.pdf; zuletzt abgerufen am 21.01.2002

[Kru95] Kruchten, P.: Architectural Blueprints - The ”4+1” View Model of Software Architecture. In: IEEE Software, Vol. 12, Nr. 6, 1995

[KS00] Kaplan, Steven; Sawhney, Mohanbir: E-Hubs: The new B2B Marketplaces. In: Harvard Business Review, Vol. 22, May-June 2000

L

[Lew98] Lewandowski, S.M.: Frameworks for Component-Based Client/Server Computing. In: ACM Computing Surveys, Vol. 30, Nr. 1, Seite 3–27,1998

[LSM01] Lohmann, M.; Schmitzer, B.; Mertens, P.: Kern-Schalen-Modell mit Fokus auf E-Commerce. In: Tagungsband 3. Workshop komponentenoreientierte betriebliche Anwendungssysteme (WKBA3),Turowski, K. (Hrsg), 2001

[LW00] Leist, S.; Winter, R.: Finanzdienstleister im Informationszeitalter – Vision, Referenzmodell und Transformation. In: Dienstleistungskompetenz und innovative Geschäftsmodelle; Belz, Ch.; Bieger, T. (Hrsg.), Thexis, St. Gallen, 2000

M

[Mah00] Mahadevan, B.: Business Models for Internet based E-Commerce – an Anatomy. In: California Management Review, Vol. 42, Nr. 4, Summer 2000

[Mef01] Meffert, H.: Marketing, 9. Auflage, Gabler, 2001

[Mer99] Merz, M.: Electronic Commerce – Marktmodelle, Anwendungen und Technologien; dpunkt-Verlag, 1999

[MSD99] Morgan Stanley Dean Witter: Business Models – On a Crowded Runway, Only a Few Stand Out. Studie: The Internet and Financial Services; S. 36-41; 1999, Online unter http://www.msdw.com/techresearch/financer/info.html; zuletzt abgerufen am 15.02.2001

[MT00] Medvidovic, N.; Taylor, R.N: A Classication and Comparison Framework for Software Architecture Description Languages. In: Software Engineering, 26(1); S.70-93, 2000.

[MTL99] Merz, M.; Tu, T.; Lamersdorf, W.: Electronic Commerce – Technologische und organisatorische Grundlagen. In: Informatik Spektrum, Band 22, Heft 5, Oktober 1999

N

[Neh96] Nehl, Theodor: Einführung in die Produktionswirtschaft; Oldenbourg, München, 1996

[NFO02] N.N.: Monitoring Informationswirtschaft – 4. Faktenbericht 2002. NFO Infratest, München 2002, Online unter http://193.202.26.196/bmwi/pdf_files/2002_04de_Faktenbericht_Vollversion.pdf; zuletzt abgerufen am 10.10.2002

[Nie98] Niemeier, Joachim: Internet-Communities als Geschäftsmodell. In: Zeitschrift Führung und Organisation, 4/1998, S. 220-223, 1998

[NL97] Nierstrasz, O.; Lumpe, M.: Komponenten, Komponentenframeworks und Gluing. In: HMD Praxis der Wirtschaftsinformatik, 1997

[NN96] N.N.: Other important IT Business Model Changes in the New Era. In: Computer Industry Report, Vol 30, Nr.23/24, Framingham, Massachusets, 1996

O

[OEC99] Organisation for economic co-operation and development: The econimical and social impact of electronic commerce; 1999

[OHE96] Orfali, R.; Harkey, D.; Edwards, J.: The Essential Distributed Objects Survival Guide. John Wiley & Sons, 1996.

[OP02] Osterwalder, A.; Pigneur, Y.: An e-Business Model Ontology for Modeling e-Business. In: 15th Bled Electronic Commerce Conference e-Reality: Constructing the e-Economy, June 2002

[Ost04] Osterrieder, C.: Komponentenmodelle für Web-Anwendungen, Diplomarbeit am Institut für Computerwissenschaften, Universität Salzburg, 2004. Online unter http://suntrec.salzburgresearch.at/projects/componentWeb/diplomarbeit.pdf, zuletzt abgerufen am 20.03.2005

[Öst96] Österle, H.: Das Geschäftsmodell im Informationszeitalter. In: Wirtschaftsinformatik, Vol. 38, Heft 4, 1996

[Öst00] Österle, H.: Geschäftsmodell des Informationszeitalters. In: Winter, R.; Österle, H.; Business Engineering – Auf dem Weg zum Unternehmen des Informationszeitalters; 2000

[Öst99] Österle, H.: Geschäftsmodell des Informationszeitalters. Vortrag zur Konferenz „3.Medien-Biennale – Envisioning Knowledge“ der Burda Akademie zum dritten Jahrtausend, 1999

[Ott01] Ott, Hans Jürgen: e-Business-Kompendium von Prof. Dr. Hans Jürgen Ott. Skript-System von Prof. Ott, Stand: 18.07.2001, Online unter http://www.kecos.de/script/, zuletzt abgerufen am 13.06.2002

P

[PM85] Porter, M.E.; Millar, V.E.: How information gives you competitive advantage. In: Harvard Business Review, Vol. 63, 1985

[Por99] Porter, M.E.: "Wettbewerbsvorteile: Spitzenleistungen erreichen und behaupten" 5. Auflage, Campus Verlag, 1999

[PR97] Peppers, D.; Rogers, M.: Enterprise One to One – Tools for Competing tn the Interactive Age; Doubleday, 1997

[PR99] Paul, C.; Runte, M.: Wie ziehe ich den Kunden an? - Virtuelle Communities“. In: Albers, S.; Clement, M.; Peters, K.; Skiera, B. (Hrsg.) eCommerce, Frankfurt am Mein, 1999

[Pri01] Pringle, David: Just looking: Europeans avoid online shopping. In: Convergence, Vol. 7 (2001), Nr. 2, S.22-26, 2001, Online unter http://www.kecos.de/script/26shvorteile.htm, zuletzt abgerufen am 12.06.02

[PS96] Pfister, C.; Szyperski, C.: Why Objects are Not Enough. In: International Component Users Conference, München, 1996. SIGS.

R

[Rad01] Radetzky, Gerda von:Mit der schnellen Nachricht auf Abo-Fang. In: Computerwoche, Nr. 24 vom 15.06.2001

[Rap01] Rappa, M.: Business Models on the Web, Online unter http://ecommerce.ncsu.edu/business_models.html, zuletzt abgerufen am 04.05.2002

[Rau99] Rautenstrauch, C.: Fachkomponenten zur Gestaltung betrieblicher Anwendungssysteme. In: IM Information Management & Consulting, Vol. 14, Nr. 2, S. 25-34, 1999

[Ray99] Rayport, J. F.: The Truth about Internet Business Models. Booz, Allen & Hamilton, 1999, Online unter http://www.strategy-business.com/briefs/99301/page1.html, zuletzt abgerufen am 01.08.2000

[Rei01] Reif, W.: Was ist E-Commerce? Vorlesungsunterlagen des Lehrstuhls für Softwaretechnik und Programmiersprachen - SS 2001, Universität Augsburg, 2001, Online unter http://www.uni-augsburg.de/lehrstuehle/info1/ lehre/ss01/e-commerce/folien/Definition.pdf, zuletzt abgerufen am 05.05.2003

[RFF99] Rasner, C.; Füser, K.; Faix, W. G.: In: Das Existenzgründerbuch. Verlag moderne Industrie, Landsberg/Lech 4. Auflage 1999

[RK01] Riemer, K.; Klein, S.: Personalisierung von Online Shops. In: Report Online Handel. Symposion Verlag, 2001.

[RK97] Renzel, K.; Keller, W.: Three Layer Architecture. In: Software Architectures and Design Patterns in Business Applications, 1997

[RRS99] Rumpe, B.; Radermacher, A.; Schoenmakers, M.; Schürr, A.: UML + ROOM as a Standard ADL? In: Proceedings of ICECCS'99 Fifth IEEE International Conference on Engineering of Complex Computer Systems, 1999.

[RS92] Rockart, J.F.; Short, J.E.: Information Technology in the 1990s: managing organizations interdependence. In: Galliers, R.D.; Baker, B.: Strategic Information Management – challenges and strategies in managing information systems, Butterworth, Heinemann, Oxford, 1989

[RS01] Reicheld, F.; Schefter, P.: E-Loyality: Your secret weapon on the web. In: Harvard Business Review, Heft 4 July/August, 2001

[RSG01] Rossi, G.; Schwabe, D.; Guimaraes, R.: Designing personalized web applications. In: World Wide Web, S. 275-284, 2001

S

[SBC00] Shankaranarayan, G.; Balasubramanian, P.R.; Chen, K.: Conceptualizing Architektures for E-Business Systems. In: Proceedings of the Sixth Americas Conference on Information Systems (AMCIS 2000), S. 249-252, 2000

[SBS96] Sarker, M.; Butler, B.; Steinfield, C.: Intermediaries and Cybermediaries: A continuing role for mediating players in the Electronic Market. In: Journal of Computer Mediated Communication (JCMC), Nr. 3, 1996

[Sch97] Scheer, A.-W.: ARIS House of Business Engineering. Konzept zur Beschreibung und Ausführung von Referenzmodellen. In (Becker, J. et al. Hrsg.): Entwicklungsstand und Entwicklungsperspektiven der Referenzmodellierung. Proceedings zur Veranstaltung vom 10. März 1997. Arbeitsberichte des Instituts für Wirtschaftsinformatik Nr. 52. S. 3-15.

[Sch00] Schmidt, R.: Internetplattformen und neue Geschäftsmodelle. In: Versicherungswirtschaft, Heft 20, 2000

[Sch00a] Schmidt, B. F.: Was ist neu an der digitalen Ökonomie? In: Belz, C.; Bieger, T.: Dienstleistungskompetenz und innovative Geschäftsmodelle – Forschungsgespräche der Universität St. Gallen 1999, Thexis Verlag, St. Gallen, 2000

[SDL03] Scheer, C.; Deelmann, T.; Loos, P.: Geschäftsmodelle und internetbasierte Geschäftsmodelle - Begriffsbestimming und Teilnehmermodell. ISYM Working Paper. Dezember 2003, Online unter: http://isym.bwl.uni-mainz.de/publikationen/isym012.pdf, zuletzt abgerufen am 20.02.05

[Sel99] Selz, D.: Value Webs – Emerging forms of fluid and flexible organizations. Dissertation Nr. 2310, Universität St. Gallen, Schweiz, Difo-Druck GmbH, Bamberg, 1999

[SEI02] Software Engineering Institute, Carnegie Mellon University: How to define Software Architecture, Online unter http://www.sei.cmu.edu/architecture/definitions.html, zuletzt abgerufen am 16.07.2002

[Sha95] Schaw, M.: Making Choices: A Comparison of Styles for Software Architekture. In: IEEE Software, Vol 12, Nr. 6, 1995

[Sha96] Schaw, M.: Some patterns for software architectures. In: Proceedings of the Second Pattern Languages of Program Design Workshop., 1996

[Sha00] Shaw, M.J.: Information-Based Manufacturing with the Web. In: The International Journal of Flexible Manufacturing Systems, 12 (2000), S. 115-129, Kluwer Academic Publishers, Boston, 2000

[Sim94] Sims, O.: Business Objects - Deliverings Cooperatice Objects for Client/Server. In: The IBM McGraw-Hill Series, McGraw-Hill Book Company, 1994

[Ski00] Skiera, B.: Wie teuer sollen die Produkte sein ? – Preispolitik. In: Albers, S.; Clement, M.; Peters, K.; Skiera, B. (Hrsg.); eCommerce. Einstieg, Strategie und Umsetzung im Unternehmen, Frankfurt am Main, 2000

[Ski99] Skiera, B: Preisdifferenzierung. In: Albers, S.; Clement, M.; Peters, K. (Hrsg.), "Marketing mit Interaktiven Medien. Strategien zum Markterfolg", Frankfurt am Main, 1999

[SL00] Skiera, B.; Lambrecht, A.; Erlösmodelle im Internet, Lehrstuhl für Betriebswirtschaftslehre, insbesonsere Electronic Commerce, Johann Wolfgang Goethe Univeristät, Frankfurt am Main, 2000, Online unter http://www.ecommerce.wiwi.uni-frankfurt.de/skiera/publications/Erloesmodell.pdf, zuletzt abgerufen am 13.12.2002

[SNH95] Soni, D.; Nord, R.L.; Hofmeister, C.: Software Architecture in Industrial Applications. In: 17th International Conference on Software Engineering. Seattle, Washington, USA, 1995.

[SS97] Schubert, P.; Selz, D.: Web Assessment – A Model for the Evaluation and the Assessment of successful Electronic Commerce Application. In: Schmidt, Beat F.; Selz, Dorian; Sing, Regine: EM – Electronic Product Catalogs. EM – Electronic Markets, Vol. 7, No. 3; 09/97, 1997

[SS99] Strauß, R.E.; Schoder, D.: Electronic Commerce – Herausforderungen aus Sicht der Unternehmen. In: Hermanns, A.; Sauter, M,:; Management-Handbuch Electronic Commerce; Vahlen, München, 1999

[SSK02] Schätzle, R.; Seifert, T.; Kleine-Gung, J.: Enterprise Java Beans - Kritische Betrachtung zu einer modernen Software-Architektur. In: Wirtschaftsinformatik, Vol. 44, Heft 3, 2002

[ST00] Schinzer, H.; Thome, R.: Electronic Commerce: Anwendungsbereiche und Potentiale der digitalen Geschäftsabwicklung ; Vahlen, München, 2000

[Str90] Struck, U.: Geschäftspläne. Schäffer-Verlag, Stuttgart 1990

[Syd92] Sydow, J.: Strategische Netzwerke-Evolution und Organisation; Wiesbaden, 1992

[Sym01] N.N.: Internet Shopping Report 2001 – Die große Nutzerumfrage; Symposium Publishing, Düsseldorf, 2001

[Szy98] Szyperski, C.: Component Software: Beyond Object-Oriented Programming. 2. Aufl., Addison-Wesley, 1998

T

[TTL99] Tapscott, D.; Ticoll, D; Lowy, A.: The rise of the Business Web, In: Business 2.0, Seite 198-208,November 1999

[Tim94] Timmons, J. A.: New Venture Creation – Entrepreneurship for the 21st century. 4. Ausgabe, IRWIN/McGraw-Hill, Boston 1994

[Tim98] Timmers, P.: Business Models for Electronic Markets. In: Gadient, Yves; Schmidt, Beat F.; Selz, Dorian: EM – Electronic Commerce in Europe. EM – Electronic Markets, Vol.8. No. 2; 07/98, 1998, Online unter http://www.electronicmarkets.org/modules/pub/view.php/electronicmarkets-183, zuletzt abgerufen am 10.05.2005

[Tim99] Timmers, P.: Electronic Commerce – strategies and models for business-to-business trading; John Wiley & Sons Ltd. England, 1999

[TS97] Thome, R.; Schinzer, H.: Marktüberblick Electronic Commerce. In: Electronic Commerce, Thome, R.; Schinzer, H. (Hrsg.), Vahlen, München, 1997

V

[VP96] Viscio, A.J.; Pasternack, B.A.: Toward a new Business Model. Booz, Allen & Hamilton, 1996, Online unter http://www.strategy-business.com/research/96201/page1.html, zuletzt abgerufen am 01.08.2000

[VSM02] Vassiliou, C.; Stamoulis, D.; Martakos, D.: The Process of Personalizing Web-Content: Techniques, Workow and Evaluation. International Conference: Advances in Infrastructure for e-Business, e-Education,e-Science, and e-Medicine on the Internet, 2002.

W

[WAB02] Winter, A.; Ammenwerth, E.; Brigl, B.; Aux, R.: Krankenhausinformationssysteme. In (Lehmann, T.; Meyer zu Bexten, E. Hrsg.): Handbuch der Medizinischen Informatik. München 2002, S. 473-552.

[WB02] Wirtz, B.; Becker, R.: Geschäftsmodellansätze und Geschäftsmodellvarianten im Electronic Business - Eine Analyse zu Erscheinungsformen von Geschäftsmodellen. In: WiSt - Wirtschaftswissenschaftliches Studium, Vol. 31, Nr.2, Februar 2002

[Web99] Weber, R.: Chablis - Market Analysis of Digital Payment Systems; Technical Report (TUM-I9819) am Institut für Informatik der Technischen Universität München, Version 1.5; August 1999. Online unter http://chablis.informatik.tu-muenchen.de, zuletzt abgerufen am 06.12.2002 (Version 1.0 online unter http://chablis.informatik.tu-muenchen.de/MStudy/x-a-marketpay.html; zuletzt abgerufen am 06.12.2002)

[Web99a] N.N.: Studie zur Eignung verschiedener Produktgruppen im e-Commerce, 1999, Online unter http://www.webstudie.de/studien/ergebnisse_b2c.htm, zuletzt abgerufen am 22.05.2002

[Wei02] Weitz, W.: Basisarchitekturen Web-basierter Informationssysteme. In: Wirtschaftsinformatik, Vol. 44, Nr. 3, S. 207-216, 2002

[Wes99] Weske, M.: Business-Objekte: Konzepte, Architekturen, Standards. In: Wirtschaftsinformatik, Vol. 41, Nr. 1, Seite 4–11, 1999

[Wer98] Werner, A: Ad-Server: Einsatz der Software bei den Online-Angeboten der Zeitungsverlage, Mai 1998. Online unter http://www.bdzv.de/online/adserver.htm, zuletzt abgerufen am 18.03.2005

[Wer00] Werbach, K.: Syndication – The emerging model for Business in the internet era. In: Harvard Business Review, Vol. 22, Heft 3, May-June 2000

[Wir01] Wirtz, B.W.: Electronic Business, 2. Auflage, Gabler Verlag, Wiesbaden 2001.

Z

[Zac87] Zachman, J.A.: A framework for information systems architecture. In: IBM Systems Journal, Vol. 26, Issue 3, S.276 - 292, 1987

[ZD00] Zwick, D.; Dholakia, N.: E-Business Models implicit in comparable American and German corporate websites. In: Proceedings of the Sixth Americas Conference on Information Systems (AMCIS 2000), S. 766-771, 2000

[Zim00] Zimmermann, H.-D.: Understanding the digital economy: Challenges for new Business Models. In: Proceedings of the Sixth Americas Conference on Information Systems (AMCIS 2000), S. 729-732, 2000

[Zim98] Zimmermann, H.-D.: Elements of a new Approach to develop innovative Business Models for Electronic Markets. In: Proceedings WISE 1998, New York, Dezember 11, 1998

[ZPS99] Zerdick, A.; Picot, A.; Schrape, K., u.a.: Die Internet-Ökonomie - Strategien für die digitale Wirtschaft. Springer Verlag, 1999

Abbildungsverzeichnis

Tabellenverzeichnis

Selbständigkeitserklärung

Hiermit erkläre ich, die vorliegende Dissertation selbständig und ohne unzulässige fremde Hilfe angefertigt zu haben. Ich habe keine anderen als die angeführten Quellen und Hilfsmittel benutzt und sämtlich Textstellen, die wörtlich oder sinngemäß aus veröffentlichten oder unveröffentlichten Schriften entnommen wurden, und alle Angaben, die auf mündlichen Auskünften beruhen, als solche kenntlich gemacht. Ebenfalls sind alle von anderen Personen bereitgestellten Materialen oder erbrachten Dienstleistungen als solche gekennzeichnet.

Leipzig, 01.September 2005

....

(Ort, Datum) (Unterschrift)

Werdegang

Beruflicher Werdegang

April 2000 - gegenwärtig	adesso AG, Dortmund Senior Consultant
April 1999 - März 2000	Fraunhofer-Institut für Software- und Systemtechnik (ISST), Abteilung Workflow-Management, Dortmund Wissenschaftlicher Mitarbeiter

Ausbildung

1994 - 1999	Studium der Wirtschaftsinformatik, Universität Paderborn und University of Wollongong, Australien Abschluss: Dipl. Wirt.-Inf.
1983 - 1992	Gymnasium am Markt, Bünde (NRW) Abschluss: Abitur

Danksagung

Mein besonderer Dank gilt Herrn Professor Dr. Volker Gruhn, der zum einen mit dazu beigetragen hat, dass mir die Firma adesso AG die notwendigen Freistellungen zur Durchführung der Promotion einräumte und zum anderen als *Doktorvater* maßgeblich an meiner fachlichen Betreuung dieser Arbeit beteiligt war. Seine kritische Beurteilung der Zwischenergebnisse, die konstruktiven Hinweise zum weiteren Vorgehen sowie die unmissverständlichen Durchhalteparolen waren allesamt nötig, um die vorliegende Arbeit abzuschließen.

„...promovieren heißt schreiben und leiden“

Darüber hinaus bedanke ich mich bei Professor Dr. Klaus-Peter Fähnrich sowie Professor Dr. Stefan Eicker für ihre Tätigkeiten im Rahmen der Begutachtung.

Auch bedanke ich mich ausdrücklich bei Matthias Book für seine geduldige und fortwährende Unterstützung und Beantwortung meiner zahlreichen Fragen. Bei Thomas Goesmann, Mariele Hagen und Knut Woller sowohl für die fachlichen Diskussionen und die kritischen Reviews sowie ganz besonders für das gemeinsame promotionsbezogene Philosophieren im außeruniversitären Umfeld.

Allen Mitarbeitern des Lehrstuhls für Angewandte Telematik / e-Business an der Universität Leipzig danke ich dafür, dass sie mir während meiner dortigen Zeit hilfreich zur Seite standen und für ein angenehmes Betriebsklima sorgten.

Abschließend bedanke ich mich ganz besonders bei Sille für ihre Rücksicht mir gegenüber: Niemand sonst musste so viel Geduld aufbringen, musste so häufig zuhören und war so häufig meinen Launen ausgesetzt und stand dennoch immer hinter diesem Vorhaben.

Thorsten Weber, im September 2005