Unterstützung von Integrationsdienstleistungen durch abstrakte Integrationsmuster

Von der Fakultät für Mathematik und Informatik
der Universität Leipzig
angenommene

D I S S E R T A T I O N

zur Erlangung des akademischen Grades

DOCTOR RERUM NATURALIUM
(Dr. rer. nat.)

im Fachgebiet
Informatik

vorgelegt

von Herrn Dipl.-Wirtsch.-Inf. Martin Pero (geb. Gebauer)

geboren am 19. Dezember 1978 in Halle (Saale)

Die Annahme der Dissertation wurde empfohlen von:

1. Professor Dr. Klaus-Peter Fähnrich (Universität Leipzig)
2. Professor Dr. Ronald Maier (Universität Innsbruck)

Die Verleihung des akademischen Grades erfolgte mit Bestehen
der Verteidigung am 19.02.2013 mit dem Gesamtprädikat magna cum laude

Für meinen Großvater
Herrn Studienrat Hermann Gebauer † 2007.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Quellcodeverzeichnis

1 Motivation und Abgrenzung der Arbeit

Die vorliegende Arbeit verwendet einen anwendungsorientierten Wissenschaftsbegriff. Das Ziel der Arbeit besteht nicht in der Entwicklung einer neuen Theorie, sondern in der Erforschung von Ziel-Mittel-Zusammenhängen.[1] Eine wissenschaftliche Arbeit basiert demnach auf einem Praxisproblem, welches Effizienzeinbußen zur Folge hat. Aus diesem Praxisproblem wird als Perspektive der wissenschaftlichen Auseinandersetzung die Forschungsfrage abgeleitet. Die Auseinandersetzung mit der Forschungsfrage deckt mangelhafte Erkenntnisse in verschiedenen Teilproblemen auf, welche ein Forschungsproblem definieren. Die Lösung dieses Forschungsproblems – wenn auch nur teilweise – trägt zur Beantwortung der Forschungsfrage und damit der Lösung des Praxisproblems bei und bestimmt den Gegenstand der wissenschaftlichen Arbeit.[2]

In den folgenden Unterkapiteln werden das Praxisproblem (Kapitel 1.1), die Forschungsfrage (Kapitel 1.2) und die Forschungsprobleme abgeleitet. Das Kapitel 1.4 definiert darauf aufbauend das Erkenntnisziel der Arbeit. Das sich anschließende Kapitel 2 leitet in die wissenschaftstheoretische Einordnung und bestehende Forschungsergebnisse – also den aktuellen Stand der Wissenschaft – über.

1.1 Problemstellung

Systemintegration ist eine fortwährende Aufgabe, welche in allen Unternehmen auftritt, die zur Unterstützung Ihrer Geschäftstätigkeit Informationstechnik in Form von Anwendungssystemen einsetzen. Dabei unterstützt ein Anwendungssystem – im Regelfall – nur einen Ausschnitt der betrieblichen Tätigkeit, wodurch es zum Einsatz verschiedenartiger Anwendungssysteme kommt, um einen möglichst großen Bereich der Geschäftstätigkeit abzudecken. Durch die Verkettung der betrieblichen Tätigkeiten ergeben sich Zusammenhänge der Anwendungssysteme, welche diese unterstützen. Zur Sicherstellung einer nahtlosen Aufgabenausführung werden die Grenzen zwischen den einzelnen Anwendungssystemen durch Integration überbrückt.

Die fortschreitende Durchdringung der Unternehmen mit Systemen der Informationstechnologie (IT-Systeme) in den vergangenen Jahren hat dazu geführt, dass die Einführung von neuen IT-Systemen und die Implementierung von Prozessinnovationen zunehmend verschwimmen.[3] In vielen Unternehmen entsteht deshalb durch sich ändernde Märkte, beispielsweise bei der Erschließung neuer Beschaffungs- und Absatzkanäle und der daraus resultierenden Veränderung der Geschäftsprozesse, ein fortwährender Bedarf nach Integrierung von Anwendungssystemen. Integration bleibt keine Aufgabe, die abgeschlossen werden kann, sie erwächst vielmehr zu einer kontinuierlichen Angelegenheit, um den optimalen Unterstützungsgrad durch IT-Systeme zu gewährleisten. Integration im unternehmensweiten und Zulieferer umfassenden Sinne ist die Kernkomponente bei der Transformation der Geschäftsprozesse.[4] Dabei sind die gewachsenen Anwendungslandschaften mit sich zum Teil überlagernden Funktionalitäten zu berücksichtigen und in optimaler Weise zu nutzen.

Insbesondere bei kleinen und mittelständischen Unternehmen (KMU) wird im E-Business „bei integrierten ganzheitlichen Technologien und Anwendungen noch Nachhol- und Informationsbe-

[1] Vgl. (Winter, 2009c).
[2] Vgl. (Booth; Colomb und Williams, 2003).
[3] Vgl. (Selhofer u. a., 2008, S. 10).
[4] Vgl. (Lam und Shankararaman, 2004, S. 40).

darf“[5] gesehen. Aber auch große Unternehmen halten das notwendige Know-how nur selten selbst vor. Deshalb werden Integrationsdienstleistungen am Markt bei technischen Dienstleistern nachgefragt, die „in den Phasen der technischen Konzeption und Implementierung die wichtigste Rolle“[6] spielen. Bei solchen Dienstleistern handelt es sich um „hochspezialisierte Produktentwicklungs-, Systemarchitektur- und Systemintegrationsspezialisten“[7].

Auf dem Markt der Systemintegration kann eine Verschärfung des Wettbewerbs beobachtet werden. Dies zeigen auch die regelmäßig veröffentlichten Lünendonk Listen, welche unter anderem die Entwicklung der IT-Beratungs- und Systemintegrationsunternehmen ausweisen. Die Inlandsumsätze 2011 im Vergleich zu den Umsätzen von 2010 zeigen eine Spreizung der Entwicklung von plus 30,6 Prozent bis minus 11,1 Prozent[8], was den Wettbewerb verdeutlicht. Dennoch rechnen die befragten Unternehmen für das laufende Jahr 2012 und die mittelfristige Zukunft bis 2017 mit einem jährlichen Marktwachstum von durchschnittlich unter 5 Prozent.[9] Auch in diesem Bereich setzen sich Forderungen nach immer kürzeren Projektlaufzeiten und Budgetminimierung durch. Die Änderung der IT-Landschaft – und damit die Veränderung von Integrationsbeziehungen – muss mit der Änderung der betrieblichen Tätigkeit Schritthalten. Demgegenüber steht die wiederkehrende Lösung gleichartiger Problemstellungen bei jedem Integrationsprojekt von Neuem, ohne in Vorprojekten entwickelte Lösungskomponenten wieder zu verwenden. In Folge der Spezialisierung eines Integrationsdienstleisters lassen sich Ähnlichkeiten sowohl in Bezug auf die eingesetzten Vorgehen und Methoden als auch auf die technischen Lösungen feststellen.[10] Basierend auf diesen Erkenntnissen lässt sich als Praxisproblem für einen Integrationsdienstleister das Problem der Rationalisierung der Erstellung von Integrationslösungen formulieren. Die vorliegende Arbeit wird sich mit diesem Praxisproblem auseinandersetzen und zu diesem Zweck im folgenden Abschnitt die Forschungsfragen ableiten.

1.2 Forschungsfragen

Der Begriff der Rationalisierung, wird mit unterschiedlichen Inhalten verknüpft.[11] Unterschiede bestehen zwischen verschiedenen wissenschaftlichen Disziplinen aber auch innerhalb eines Fachgebietes gibt es unterschiedliche Sichtweisen. Bereits Rauecker betont jedoch die Unzulänglichkeit der Trennung in eine wirtschaftliche und eine technische Rationalisierung.[12] Rationalisierung im wirtschaftlichen Sinne wird von Rauecker als „mit den geringsten Kosten, die höchsten Einnahmen zu erzielen“[13] unabhängig des Weges der Erreichung[14] definiert.

Bronner spricht davon, dass eine systematische Planung und das Rationalisieren zu den Urtrieben des Menschen zählen, der mit seinen Ressourcen sparsam wirtschaften muss.[15] „Unter dem Begriff Rationalisierung (d. h. 'vernünftig machen') faßt man alle Maßnahmen zusammen, die über eine Verringerung der Gesamtkosten oder über eine Steigerung des Ertrags zu einer umweltverträglichen, langfristigen Gewinnmaximierung führen sollen“[16]. Bronner betont ferner, dass es

[5] (Lorenz u. a., 2007, S. 12)
[6] (ebenda, S. 12)
[7] (Bundesministerium für Bildung und Forschung, 2007, S. 10)
[8] Vgl. (Lünendonk GmbH, 2012, S. 2).
[9] Vgl. (ebenda, S. 3).
[10] Vgl. (Gebauer; Thränert u. a., 2008), (Gebauer und Thränert, 2008).
[11] Vgl. (Voigt, 2012).
[12] Vgl. (Rauecker, 1925, S. 683).
[13] (ebenda, S. 684)
[14] Rauecker spricht von mit „guten oder mit schlechten Stiefeln erreicht werden kann“ in Anlehnung an Sombart siehe auch (Krüger, 1988) und (Sombart, 1913).
[15] Vgl. (Bronner, 1992, S. 1).
[16] (ebenda, S. 1)

sich damit primär nicht um ein technisches, sondern um ein wirtschaftliches und soziologisches Problem handelt, zu dessen Lösung aber technische Mittel eingesetzt werden.[17] Grundsätzlich wird zwischen Mechanisierung und Automatisierung als möglichen Mitteln unterschieden. Bei der Mechanisierung wird eine Substitution menschlicher Leistung durch technische Hilfsmittel erreicht, bei der Automatisierung die vollständige Substitution, so dass der Mensch obsolet wird.[18] Diese Sichtweise spiegelt sich in einer durch das RKW Rationalisierungs- und Innovationszentrums der Deutschen Wirtschaft e.V. durchgeführten, anonymisierten Zielgruppenanalyse wieder. Es wurden verschiedene Experten befragt, mit dem Ergebnis, dass zwei Sichtweisen überwiegen. Zum einen wird mit dem Begriff eine „Gestaltung betrieblicher Prozesse zur Steigerung der Wirtschaftlichkeit“[19] und zum Anderen „mit dem Abbau von Arbeitsplätzen“[20] verbunden. Letzteres entspricht dem Überflüssigwerden des Menschen.

Als primäre Bewertungsgrößen der Rationalisierung werden die Produktivität und die Wirtschaftlichkeit angegeben, wobei Produktivität der Quotient aus Produktionsleistung und der Summe aus Material, Arbeit und Kapital ist. Wirtschaftlichkeit ist der Quotient aus dem Erlös für das Produzierte und den Kosten des Einsatzes. Dabei gibt die Produktivität eine relative Größe wieder, wohingegen die Wirtschaftlichkeit eine absolute Größe liefert, jedoch den Schwankungen des Marktes im Sinne erzielbarer Preise unterworfen ist.[21]

Die Optimierung der Abläufe also die Anpassung des Menschen an eine mechanisierte Welt war Gegenstand der tayloristischen Bemühungen.[22] Das Ziel war eine Verbesserung von Arbeitsabläufen z. B. durch angepasste Bewegunsabläufe von Fließbandarbeitern. Die Rationalisierung durch Automatisierung ist vor dem Hintergrund der Bewertungsgröße Wirtschaftlichkeit nur dann sinnvoll, wenn – das gleiche Arbeitsergebnis vorausgesetzt – die Investitions- und Folgekosten die für eine manuelle Erledigung nötigen Lohnkosten unterschreiten.[23] Diese Betrachtungen haben in der Fortschreibung der industriellen Revolution zu einem eigenen Forschungszweig – welcher eine Systemrationalisierung des Gesamtsystems produzierender Betrieb anstrebt – dem Industrial Engineering geführt.[24]

Der Prozess zur Erstellung von Integrationslösungen – im Folgenden nach Thränert als Integrierung[25] bezeichnet – zur Anwendungssystemintegration lässt sich nur bedingt mit Produktionsprozessen vergleichen, da Bewegunsabläufe kaum eine Rolle spielen. Legt man diesen Rationalisierungsbegriff jedoch etwas weiter aus, so können Vorgehensmodelle zur Erstellung von Integrationslösungen an dieser Stelle eingeordnet werden. Sie legen mögliche Integrationsschritte und ihre Reihenfolge fest. Weiterhin kann daraus ableitet werden, dass im Bereich des Software Engineering auch durch Automatisierung und vermehrte Wiederverwendung rationalisiert werden kann. Diese Annahme wird untermauert durch Erfahrungen von Systemintegratoren, die wiederholt die Entwicklung von Teilautomatisierungen bei der Integrierung durch einzelne Teammitglieder beobachten konnten. Dabei basieren diese Rationalisierungen im Wesentlichen auf dem Erkennen von Bekanntem in einer neuen Problemstellung und der Auswahl, Anwendung, sowie ggf. Anpassung und Ergänzung erprobter individueller Lösungsmuster, die zumeist ähnlich gelagerten Projekten entstammen. Dabei wird durch die Teammitglieder eine Entlastung von redundanten, fehlerträchtigen Aktivitäten angestrebt. Diese Entwicklungen sind jedoch nicht

[17] Vgl. (ebenda, S. 1f.).
[18] Vgl. (ebenda, S. 2).
[19] (Tetzel, 2006, S. 31)
[20] (ebenda, S. 31)
[21] Vgl. (Bronner, 1992, S. 2).
[22] Vgl. (Taylor, 1993).
[23] Vgl. auch Rationalisierungsinvestition in (o.V. 2012) im Gegensatz zu einer Erweiterungsinvestition.
[24] Vgl. (Sink; Poirier und Smith, 2001).
[25] Vgl. (Thränert, 2008).

ausreichend dokumentiert und systematisiert, sodass erzielte Rationalisierungseffekte dem Unternehmen nicht dauerhaft zur Verfügung stehen und mit dem Fortgang des Entwicklers verloren gehen. Eine umfassende Werkzeugunterstützung kann als Automatisierung, wenigstens jedoch als Teilautomatisierung, gesehen werden. Aus diesen Ausführungen leiten sich die folgenden Forschungsfragen ab:

- Wie kann einmal erworbenes Lösungswissen konserviert und im Rahmen eines Werkzeuges eingesetzt werden?
- Wie kann die Erbringung von Integrationsdienstleistungen mit geeigneten Werkzeugen, durchgehend, mit einer Verbesserung des Automatisierungsgrades unterstützt werden?

1.3 Forschungsprobleme

Integrationsdienstleistungen lassen sich grob in die Phasen Analyse, Planung, Entwicklung, Übergabe und Betrieb einteilen.[26] Dabei werden derzeit die Phasen Entwicklung und Betrieb durch Hilfsmittel und Werkzeuge unterstützt. Hilfsmittel gehen in die entstehende Integrationslösung ein und automatisieren Aufgaben einer Integrationsarchitektur wie z. B. Transformationen, Kontrollflussteuerung.

Werkzeuge automatisieren die Erstellung von Konfigurationsartefakten, welche von den Hilfsmitteln für die Ausführung konkreter Integrationsaufgaben benötigt werden. Dies können spezifische Kommunikationsendpunkte und Kontrollflusstrukturen in Form von automatisierten Prozessen sein. Den Ausführungen des vorangegangenen Abschnitts zur Folge findet damit bereits eine Rationalisierung statt.

Den Schritten Entwicklung und Betrieb gehen jedoch die Phasen der Analyse und Planung voran. Diese sind bisher nicht Gegenstand von Automatisierungs- und damit Rationalisierungsbemühungen und stellen manuelle Schritte dar, die von der Qualifikation einzelner Projektbeteiligter abhängen. Darüber hinaus werden Integrationsentscheidungen selten auf einer Architekturebene getroffen sondern beziehen sich sofort auf konkrete Hilfsmittel z. B. eine spezifische Middleware.

Damit fließt das in diesem Bereich existierende und konservierte Lösungswissen nur unzureichend in die Gestaltung von Integrationslösungen ein, sodass sich in diesem Bereich Ansatzpunkte für weitere Rationalisierung bieten. Dabei ist zunächst existierendes Lösungswissen zu sichten und zu dokumentieren. Dieser Arbeit liegt die These zugrunde, dass der Abstraktionsgrad des Lösungswissens die Einsetzbarkeit in der Praxis nur unzureichend unterstützt. Dieser ist im Anschluss in geeigneter Weise anzupassen und damit die Nutzbarkeit zu verbessern.

1.4 Zielsetzung und Vorgehensweise

Das Ziel dieser Arbeit ist die Identifikation hochstehender Integrationsmuster und der Einsatz dieser in einem Werkzeug mit Framework-Charakter. Damit werden Rationalisierungseffekte durch einen erhöhten Automatisierungsgrad und zum anderen durch die strukturierte Wiederverwendung existierenden Lösungswissens erzielt. Dabei weisen hochstehende Integrationsmuster gegenüber bestehenden Ansätzen einen Abstraktionsgrad auf, der einen Einsatz in den Phasen der Analyse, Planung und Entwicklung möglich macht und damit zu besseren Architekturentscheidungen führt. Um dieses Ziel zu erreichen, wird in Kapitel 2.1 der Untersuchungsgegenstand in Form von betrieblichen Informationssystemen (BIS) eingeführt. Der aktuelle Stand der Forschung im Bereich Integration wird in 2.2 dargelegt. Dabei ist besonders dem Abschnitt 4 Aufmerksamkeit

[26] Vgl. (Gebauer und Stefan, 2011b, S. 36).

zu schenken, da hier Integrationsdienstleistungen beschrieben werden. Im Anschluss daran führt Abschnitt 2.3 ein aus der Architektur und der Softwareentwicklung bekanntes Konzept zur Konservierung von Lösungswissen ein. Die Abbildung 1.1 stellt die Struktur der Arbeit grafisch dar und verdeutlicht den Einfluss von Kapitel.3 Ein wesentlicher Beitrag ist die empirische

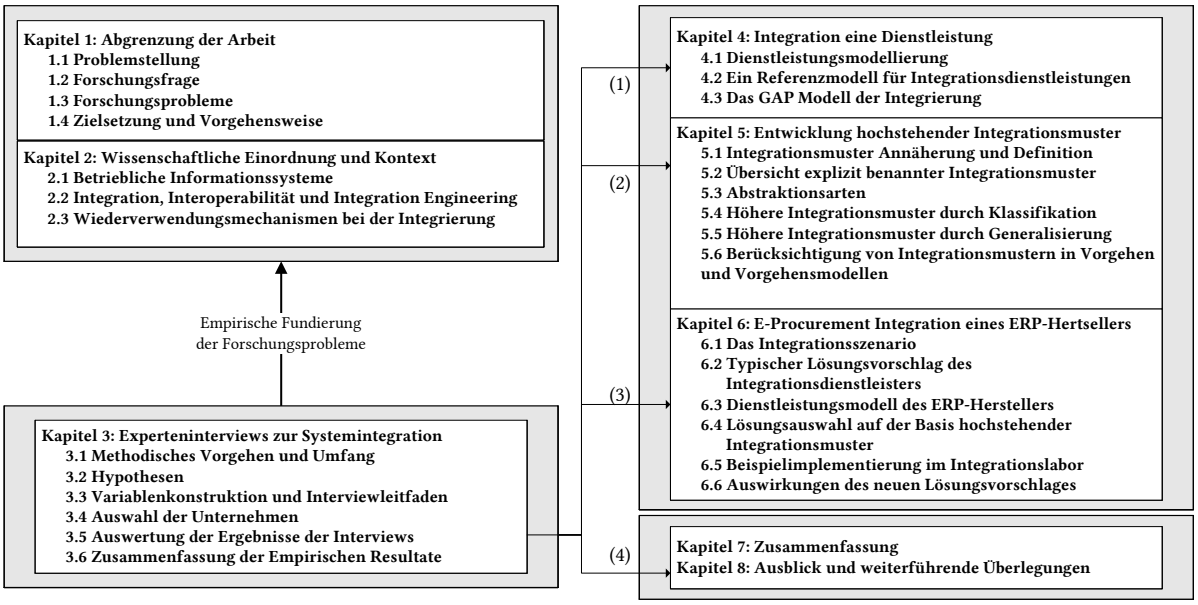


Abbildung 1.1: Struktur der Arbeit

Fundierung der Forschungsprobleme. Damit wird sichergestellt, dass es sich nicht um einzelne Phänomene handelt, sondern eine breitere Basis existiert und die Problemstellung bei mehr als einem Integrationsdienstleister anzutreffen ist. Es bleibt anzumerken, dass im Rahmen dieser Arbeit die zugrunde liegende Studie[27] nur in Teilen wieder gegeben wird, wenn ein direkter Zusammenhang zur Thematik der Arbeit besteht. Der Untersuchungsgegenstand der Studie ist breiter und beinhaltet weitere eng verwandte Themenstellungen.

Die Markierungen in Klammern innerhalb der Abbildung 1.1 kennzeichnen den weiteren Einfluss der Empirie auf die restlichen Teile der Arbeit. *(1)*: Die empirische Untersuchung stellt die Datenbasis für das in Kapitel 4 entwickelte Servicemodell für Integrationsdienstleistungen dar.

In Kapitel 5 werden hochstehende, abstrakte Integrationsmuster entwickelt, mit dem Ziel der besseren Einsetzbarkeit in der Praxis. *(2)*: Auf der Basis der Aussagen über die Musternutzung wird die Abstraktion in geeigneter Weise angepasst.

Die entwickelten Lösungen werden in Kapitel 6 zu Anwendung gebracht und in Fallstudien evaluiert. *(3)*: Die Teilnehmer der Fallstudien sind disjunkt von den Teilnehmern der empirischen Untersuchung, wodurch der Evaluationscharakter verstärkt wird.

Die Arbeit schließt mit einer Zusammenfassung in Kapitel 7, welche durch den Ausblick und zukünftigen Forschungsbedarf in Kapitel 8 ergänzt wird. Wie bereits angedeutet, hat die empirische Untersuchung einen breiteren Fokus als die vorliegende Arbeit, deshalb enthält der Ausblick auch *(4)*: durch die Studie aufgeworfene Fragen, welche in der Arbeit unbearbeitet bleiben und Gegenstand späterer Arbeiten werden.

[27] Vgl. (ebenda).

2 Wissenschaftliche Einordnung und Kontext

In diesem Kapitel wird die wissenschaftliche Einordnung der Arbeit vorgenommen und wesentliche Begriffe dargelegt. Dazu werden in 2.1 betriebliche Informationssysteme mit einem systemtheoretischen Blickwinkel vorgestellt. Weiterhin werden Integration, Interoperabilität und Integration Engineering in Kapitel 2.2, sowie in Anbetracht der Forschungsprobleme, Wiederverwendungsmechanismen bei der Integrierung in Kapitel 2.3 behandelt. Das Kapitel 3 stellt, im Anschluss, eine empirische Fundierung und gleichzeitig Detaillierung der Problemstellungen dar.

2.1 Betriebliche Informationssysteme

In diesem Abschnitt werden die begrifflichen Grundlagen für den in dieser Arbeit als Problembereich betrachteten Weltausschnitt und die Sichtweise auf diesen gelegt. Der Problembereich sind betriebliche Informationssysteme, die mit einer systemtheoretischen Sicht betrachtet werden. Zunächst wird die Sichtweise dargelegt, die Brille aufgesetzt, durch die Welt betrachtet werden soll. Im Anschluss daran wird der Problembereich näher dargelegt.

Die Systemtheorie kann nicht als eigenständiges Forschungsgebiet betrachtet und einer einzelnen wissenschaftlichen Disziplin zugeordnet werden. Neben der Informatik und den Wirtschaftswissenschaften finden sich zahlreiche weitere wissenschaftliche Disziplinen die mit dem Grundkonzept eines Systems arbeiten, aber zum Teil widersprüchliche Definitionen zugrunde legen.[1] Insbesondere in der Wirtschaftsinformatik wurde diskutiert, die Systemtheorie zur Fundierung der wissenschaftlichen Disziplin zu verwenden.[2] Im Sinne einer Disziplin übergreifenden Supertheorie[3] wurde die Systemtheorie bereits zur Beschreibung von Informationssystemen genutzt[4], weshalb sich diese Arbeit ebenfalls auf diese Theorie stützt.

Systembegriffe werden traditionell in zwei Weisen verwendet. Zum einen wird der Systembegriff verwendet um eine Einheit mit widerspruchsfreiem Zusammenhang zu kennzeichnen (logischer, analytischer Ansatz) und zum anderen ein Ganzes welches sich aus – miteinander in Beziehung stehenden – Teilen zusammensetzt.[5]

Die Teile-Ganzes-Beziehung findet sich auch in der Betriebswirtschaft: Nach Ulrich besteht ein System aus der „geordneten Gesamtheit von Elementen, zwischen denen irgendwelche Beziehungen bestehen oder hergestellt werden können“[6]. Der Zweck eines Systems kann mit der Ordnung eines Gedanken- oder Gegenstandsbereich beschrieben werden. Die Elemente bilden dabei atomare Bausteine. Das Zusammenwirken der Elemente wird durch Beziehungen zwischen diesen ausgedrückt. Elemente, die außerhalb der Systemgrenzen liegen, befinden sich im sogenannten Umsystem. Die Elemente eines Systems lassen sich gruppieren und zu Subsystemen zusammenfassen. Diese bilden eine Hierarchie zwischen den Systemteilen. Zahlreiche betriebswirtschaftliche Modelle wie Gleichgewichtsmodelle, Input-Output-Modelle basieren auf einem derartigen Systembegriff. Eines

[1] Vgl. (Robra, 2003) sowie (Robra, 2007).
[2] Vgl. (Patig, 2001).
[3] Vgl. (Luhmann, 1984, S. 16ff.).
[4] Vgl. (Alt, 1997, S. 45ff.).
[5] Vgl. (Bardmann, 2010, S. 368).
[6] (Ulrich, 2001, S. 133)

der erfolgreichsten Systemmodelle welches eine ganze Betriebswirtschaft fokussiert ist das St. Galler Management-Modell.[7]

Je nach gewähltem System unterscheiden sich auch die beschriebenen Systemelemente. Im Folgenden werden die in dieser Arbeit wichtigen Systembegriffe angeführt und die Interdependenzen in Form einer Subsystembeziehung deutlich gemacht.

Als Systemzweck kann bei betrieblichen Informationssystemen (BIS) von der – auf das Unternehmensziel ausgerichteten – Erfüllung betrieblicher Aufgaben gesprochen werden.[8] Nach Weichelt wird in Informationssystemen lediglich die Beziehungsart Information unterstützt, welche komplementär zu sonstigen Zahlungs- und Güterflüssen, sowie physischen Dienstleistungen ist.[9] Anders gesprochen befasst sich ein Informationssystem mit der Informationsverarbeitung und allen dazu notwendigen Teilaspekten und umfasst die dazu notwendigen Aufgabenträger. Die Bestandteile eines Informationssystems trennen die Definitionen in jene mit einem engeren und jene mit einem weiteren Systembegriff. Je nach Definition umfasst ein Informationssystem ein computergestütztes Anwendungssystem[10] – also ein Softwaresystem, welches betriebliche Aufgaben unterstützt bzw. ausführt[11]– die für den Betrieb notwendige Hard- und Software[12], sowie als sozio-technisches System[13] auch Menschen[14]. Die Begriffsbestimmungen mit einem sozio-technischen Systembegriff unterscheiden darüber hinaus die eingeschlossenen Gruppen von Menschen. So kann zwischen Konstrukteuren, Operateuren und den Nutzern unterschieden werden, von denen die ersten beiden Gruppen zu einem mittelweiten und eine Inklusion der letzten Gruppe zu einem allumfassenden Informationssystembegriff führen.[15]

Nach Ferstel und Sinz besteht eine betriebliche Aufgabe aus einem Lösungsverfahren, welches auf ein Aufgabenobjekt angewandt wird. Dabei muss das Lösungsverfahren den Sach- und Formalzielen der Unternehmung genügen. Eine spezifische Aufgabe ist durch jeweilige Vor- und Nachereignisse gekennzeichnet. Es kann zwischen einer Innen- und Außensicht unterschieden werden, wobei die Außensicht nur die Sach- und Formalziele, sowie die Vor- und Nachereignisse erfasst und damit von der Innensicht – dem Lösungsverfahren – abstrahiert.[16] Diese Unterscheidung ermöglicht es von maschinellen, digitalen und menschlichen Aufgabenträgern, also unterschiedlichen Lösungsverfahren, zu abstrahieren. Diese Unterscheidung ist notwendig, da streng genommen die Automatisierung einer betrieblichen Aufgabe auch durch eine Maschine erfolgen kann, welche ohne Software funktioniert. Um den IT-Bezug hervorzuheben, wird im Folgenden von digitalen Aufgabenträgern gesprochen, wenn explizit IT-Systeme gemeint sind.

Auf der Basis der eingehenden Systemdefinition aus der Systemtheorie und den Abgrenzungen eines betrieblichen Informationssystems lassen sich weitere (Sub-) Systembegriffe abgrenzen, welche den Problembereich näher detaillieren. Diese (Sub-) Systembegriffe – wie auch der Informationssystembegriff – dienen dazu die Aufgaben und insbesondere die Aufgabenträger in Subsysteme aus gleichartigen Elementen zusammenzufassen. Die Abbildung 2.1 stellt die Subsysteme eines Unternehmenssystems im Überblick dar, diese werden nachfolgend eingehender erläutert. Das *Unternehmen* bzw. Unternehmenssystem bildet die Wurzel in der Systemhierarchie

[7] Vgl. (Bleicher, 2004).
[8] Vgl. (Bullinger und Fähnrich, 1997), (Hansen und Neumann, 2002), (Ferstl und Sinz, 2008, S. 2f.), (Mandl, 2009, S. 2).
[9] Vgl. (Weichelt, 2009, S. 12).
[10] Vgl. dazu auch automatisierte Informationsverarbeitung nach (Goldammer; Huhn und Picht, 1988, S. 23ff.).
[11] Vgl. (Gabriel, 2011).
[12] Vgl. (Winter und Aier, 2011).
[13] Vgl. (Heinrich, 1990).
[14] Vgl. (Winter und Aier, 2011), (Krcmar, 2003, S. 25).
[15] Vgl. (Winter und Aier, 2011).
[16] Vgl. (Ferstl und Sinz, 2008).

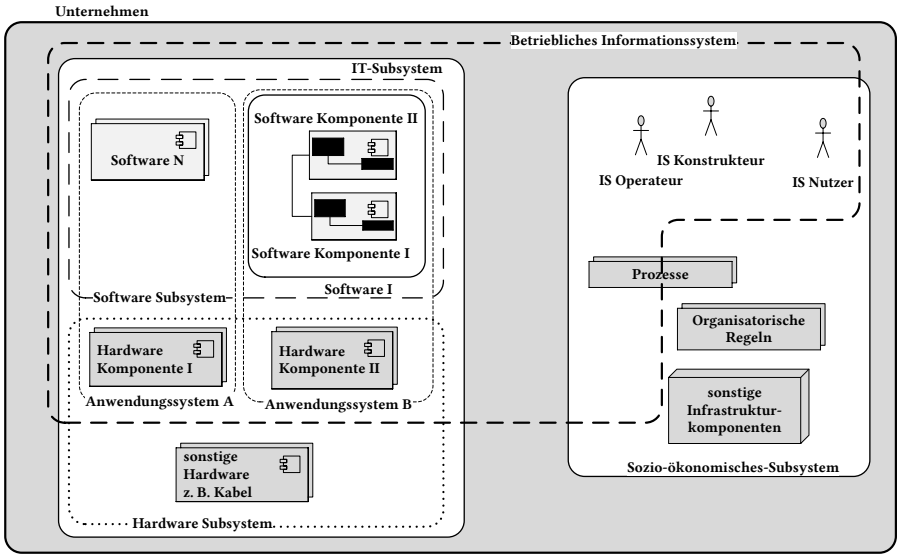


Abbildung 2.1: Subsysteme eines Unternehmens

für diese Arbeit, da es sich um eine juristische Person handelt, welche damit über die Eigenschaft der Abgrenzbarkeit hinaus, selbstständig und rechts fähig ist.

Für diese Arbeit ist es zweckmäßig, das Unternehmenssystem in 2 Subsysteme zu gliedern. Das *IT-Subsystem* und das *Sozio-ökonomische Subsystem*. Das *Sozio-ökonomische Subsystem* umfasst dabei die Menschen, Prozesse und organisatorischen Regeln eines Unternehmens sowie die physischen und sonstigen Infrastrukturkomponenten. Zu den sonstigen Infrastrukturkomponenten sind Nicht-Hardwarekomponenten, also Gebäude, Maschinen u.ä. zu zählen.

Das *IT-Subsystem* besteht aus der Gesamtheit der Hardware und Software in einem Unternehmen. Ein Unternehmenssoftwaresystem [17] ist das Gesamtsystem der Systeme und Applikationen einer Organisation.[18] Daraus ergeben sich eine mögliche weitere Gliederung des *IT-Subsystems* in ein *Hardware-Subsystem* und ein *Software-Subsystem*. Demgegenüber steht die Definition eines Anwendungssystems, welche Bestandteil der Abgrenzungen eines betrieblichen Informationssystems ist.

Nach den angeführten Definitionen umfassen Informationssysteme Menschen, Prozesse, IT-Systeme und die gesamte Infrastruktur zum Betrieb der IT-Systeme. In Abbildung 2.1 ist das *Betriebliche Informationssystem* entsprechend gekennzeichnet. Es wird deutlich, dass es sich um eine Systemdefinition handelt, welche sowohl Teile des *IT-Subsystems* umfasst, als auch Teile des *Sozio-ökonomischen Subsystems*.

Ein *Anwendungssystem* nimmt sich – im Gegensatz zu einem Informationssystem – nur einer Untermenge an Elementen an und fokussiert die Anwendungssoftware und die für den Betrieb notwendige Systemsoftware sowie die benötigten Hardwarekomponenten auf denen diese laufen.[19] Diese Definition ist nur bedingt zweckdienlich. Für die Erfüllung der betrieblichen Aufgabe kommt mehr als ein Anwendungssystem zum Einsatz. Zusätzlich können sich zwei Softwaresysteme eine Hardwareumgebung teilen, stellen aber definitionsgemäß zwei Anwendungssysteme dar.

[17] engl. enterprise software system
[18] Vgl. (Andersson und Johnson, 2001, S. 225).
[19] Vgl. (Schwarzer und Krcmar, 1999, S. 12), (Stahlknecht und Hasenkamp, 2005, S. 208f.).

Diese Konstellation ergibt sich in Folge einer Ressourcenoptimierung häufiger. Das *IT-Subsystem* besteht somit aus der Summe aller Anwendungssysteme eines Unternehmens.[20]

Die Gestaltung des *sozio-ökonomischen-Subsystems* kann der Betriebswirtschaftslehre unter anderem der Organisationstheorie und -gestaltung zugerechnet werden und ist nicht direkter Gegenstand dieser Arbeit im Sinne des zu gestaltenden Systems. In erster Linie werden durch das *sozio-ökonomische-Subsystem* Rahmenbedingungen gesetzt, die bei der Gestaltung des IT-Subsystems zu berücksichtigen sind. Die Informations- und Kommunikationstechnik hat die Eigenschaft für bestimmte Tätigkeiten erst die Voraussetzungen zu schaffen. Dadurch können sich Wechselwirkungen ergeben, bei denen das *IT-Subsystem* das *sozio-ökonomische-Subsystem* vorgebend beeinflusst. Dieser Umstand kann auch bei der Integration zutage treten, wenn vormals manuelle Prozesse abgelöst werden. Im Sinne des Gestaltungsziels dieser Arbeit steht jedoch das *IT-Subsystem* im Vordergrund.

Das IT-Subsystem unterteilt sich in die Menge der Hardwarekomponenten (*Hardware-Subsystem*) und in die Menge der Softwarekomponenten (*Software-Subsystem*). Die Gestaltung beider Subsysteme kann der Informatik zugerechnet werden. Dabei wird in dieser Arbeit das *Hardware-Subsystem* nicht als explizites Gestaltungsziel betrachtet. Es ist jedoch analog zu dem *sozio-ökonomischen Subsystem* in die Betrachtungen einzubeziehen, da sowohl existierende und zu nutzende Teilkomponenten Vorgaben für die Gestaltung des *Softwaresubsystems* machen als auch seitens des *Softwaresubsystems* durch Hardwareanforderungen Vorgaben für das *Hardwaresubsystem* entstehen. Das *Hardwaresubsystem* wird in dieser Arbeit demnach nur partiell und immer dann betrachtet, wenn sich z. B. durch eine Veränderung eine verbesserte Erfüllung von Vorgaben des *sozio-ökonomischen-Subsystems* ergibt. Dies kann eintreten, wenn aus betriebswirtschaftlicher Sicht wünschenswerte Durchlaufzeiten – von durch Software automatisierten Prozessen – nur mit einer bestimmten Ausgestaltung des *Hardware-Subsystems* zu erreichen sind.

Für die weiteren Betrachtungen der Arbeit sind nun die Elemente der Subsysteme von Bedeutung. Im Gegensatz zu den anderen Subsystemen eines Informationssystems existieren für das *Softwaresubsystem* zahlreiche Überlegungen einen Elementbegriff – im Folgenden Komponente genannt – zu etablieren. Ziel der Ausführungen ist es einen allgemeinen Element- bzw. Komponentenbegriff zu entwickeln. Als Ausgangspunkt wird deshalb das *Softwaresubsystem* verwendet, welches darüber hinaus im Fokus der Betrachtungen steht. Die Gestaltung von Softwarekomponenten – also von einzelnen Elementen des *Softwaresubsystems* – kann der Informatik zugerechnet werden, insbesondere das Software Engineering als Teildisziplin ist hier zu nennen. Teilgebiete davon beschäftigen sich mit der Gestaltung einzelner Teilkomponenten einer Softwarekomponente. Im Fall der Objektorientierung wären dies Klassen und Objekte. Bei einer Komponente im Sinne des Softwareengineering handelt es sich demnach um ein Element eines Subsystems eines Informationssystems. Die Eigenschaften einer Komponente gehen aus den folgenden Definitionen hervor.

> *„A component is a unit of composition with contractually specified interfaces and explicit context dependencies only. A software component can be deployed independently and is subject to composition by third parties.“*[21]

Dieser Definition lässt sich entnehmen, dass Komponenten zum Zweck der Komposition – also z. B. der Bildung von Systemen aus Komponenten – gemacht sind. Ein Ziel der Komposition – der übergeordnete Systemzweck – wird jedoch nicht spezifiziert. Darüber hinaus verfügt

[20] Diese Darstellung vernachlässigt, dass es Hardwarekomponenten gibt auf denen keine Software läuft, welche aber für den Betrieb unerlässlich sind. Ein Beispiel stellen alle zur Vernetzung notwendigen Kabel dar. In dieser Arbeit werden auch diese dem IT-Subsystem zugerechnet.

[21] (Szyperski, 1998, S. 34)

eine Komponente über vertraglich vereinbarte Schnittstellen und explizit deutlich gemachte Abhängigkeiten. Da sich diese Definition auf Softwarekomponenten bezieht, wird als weitere Eigenschaft angeführt, dass diese unabhängig lauffähig sein müssen. Die letztere Eigenschaft muss bei der Ableitung eines allgemeinen Komponentenbegriffes vernachlässigt werden. Die Eigenschaft der explizit definierten Schnittstellen lässt sich für Komponenten der Subsysteme eines Informationssystems – das Softwaresubsystem ausgenommen – nur im übertragenen Sinne beibehalten. Die Vor- und Nachereignisse einer betrieblichen Aufgaben können in gewissem Maße als Schnittstellen interpretiert werden. Ein Aufgabenträger reagiert auf bestimmte (Vor-) Ereignisse (Input-Schnittstelle) und trägt durch die Anwendung eines Lösungsvorgehens zum Entstehen entsprechender Nachereignisse (Outputschnittstelle) bei.

> *„A component represents a modular, deployable, and replaceable part of a system that encapsulates implementation and exposes a set of interfaces. [...] A component may be implemented by one or more artifacts, such as binary, executable, or script files“*[22].
>
> *„A component represents a modular part of a system that encapsulates its contents and whose manifestations is replaceable within its environment. A component defines its behavior in terms of provided and required interfaces. As such, a component serves as a type, whose conformance is defined by these provided and required interfaces (encompassing both their static as well as dynamic semantics)“*[23].

In den Definitionen der Object Management Group (OMG) werden weiterführende Eigenschaften einer Komponente aufgegriffen. Die Schnittstellen werden nach Ihrer Art unterschieden. Es werden benötigte und zur Verfügung gestellte Schnittstellen unterschieden. Benötigte Schnittstellen zielen darauf ab, dass eine Komponente nur mit entsprechendem Input arbeitsfähig ist. Im übertragenen Sinn lässt sich diese Eigenschaft – welche im Wesentlichen der Existenz entsprechender Vorereignisse entspricht – auch für Komponenten des sozio-ökonomischen-Subsystems übernehmen. Nimmt man eine Organisationseinheit wie eine Reisekostenstelle als Beispiel, so kann diese nur sinnvolle Ergebnisse produzieren, wenn Reisekostenabrechnungen als Vorereignis und damit als benötigte Schnittstelle vorliegen.[24] Da diese Eigenschaften jedoch nicht direkt übertragbar sind, werden sie für die allgemeine Komponentendefinition einer BIS-Komponente nicht übernommen. Die Kapselungseigenschaft lässt sich auch nicht auf die übrigen Subsysteme übertragen. Eine der wesentlichen Eigenschaften, welche übernommen wird, ist die, dass es sich bei einer Komponente um einen modularen Teil eines Systems handelt. Diese Eigenschaft lässt sich auf alle Subsysteme eines BIS übertragen. An dieser Stelle werden die wesentlichen Eigenschaften aufgegriffen und ein allgemeiner Komponentenbegriff als Abstraktion zu den verschiedenartigen Elementen der jeweiligen Subsysteme definiert.

> *Eine Komponente ist ein modularer Teil eines Systems, welcher innerhalb seines Kontextes austauschbar ist.*

In diesem Sinne ist die obige Definition als Manifestierung des Modularitätsprinzips für alle Elemente eines Informationssystems zu sehen und die bekannten Komponentendefinitionen der Informatik als eine Spezialisierung. Eine wesentliche Eigenschaft von Softwarekomponenten ist die unabhängige Einsetzbarkeit.[25] Dabei gilt dennoch: „A component can be loaded into and

[22] (Object Management Group, OMG, 2001, S. 36)
[23] (Object Management Group (OMG), 2003, S. 110)
[24] Streng genommen müssten zur vollständigen Beschreibung der Schnittstelle Protokollbeschreibungen für mögliche Interaktionen und Beschreibungen von Reisekostenabrechnungsformularen vorliegen.
[25] Vgl. (Szyperski, 1998, S. 34).

activated in a particular system“[26]. Es gibt jedoch Elemente eines Softwaresubsystems wie Bibliotheken, Frameworks u. ä. welche genau diese Eigenschaft nicht erfüllen.

2.2 Integration, Interoperabilität und Integration Engineering

Integrationsbegriff
Die Thematik der Integration ist bisher Gegenstand vielfältiger Forschungsbemühungen gewesen und wurde in unterschiedlichen Wissenschaftsdisziplinen untersucht. Zu diesen zählen neben der Informatik die Betriebswirtschaftslehre, Wirtschaftsinformatik, Soziologie, Politik, Elektrotechnik und Mathematik. Die Mathematik stellt in dieser Aufzählung eine Ausnahme dar, da mit Integration eine Rechenoperation bezeichnet wird. In allen anderen Disziplinen wird zunächst von einer etymologischen, historisch-linguistischen Analyse des Begriffes Integration ausgegangen[27], die sich auch in der gebräuchlichen Darstellung des Duden niederschlägt:

> *In|te|gra|ti|on, die; -, -en [lat. integratio = Wiederherstellung eines Ganzen]: 1. (bildungsspr.) [Wieder]herstellung einer Einheit [aus Differenziertem]; Vervollständigung. 2. Einbeziehung, Eingliederung in ein größeres Ganzes, Ggs. Desintegration (1). 3. Zustand, in dem sich etwas befindet, nachdem es integriert worden ist; Ggs. Desintegration(2.) 4. Berechnung eines Integrals(2); vgl. ...ation/...ierung.*[28]

Bereits aus dieser Definition wird deutlich, dass sich Integration sowohl als Zustand[29] – in dem sich bspw. ein System befindet – betrachten lässt, als auch als Prozess, welcher eine Veränderung eines Integrationszustandes herbeiführt[30]. Thränert führt zum Zweck einer besseren Unterscheidung beider Aspekte den Begriff Integrierung für die Betrachtung des Prozesses ein.[31] Darüber hinaus unterscheidet er auch zwischen Maßnahmen, welche die Integrationsfähigkeit eines Integrationsobjektes verändern, ohne auf eine konkrete Integration desselben abzuzielen – ex ante Integration – und jenen, welche zwei oder mehrere Integrationsobjekte tatsächlich integrieren, der ex post Integration.[32]

Gängige Betrachtungen der Integration und Einführungen in betriebliche Informationssysteme[33] weisen gemeinsame systemtheoretische Grundlagen auf. So wird die Bildung eines Systems aus eigenständigen Elementen respektive Subsystemen häufig als Integrierung verstanden.

Aufbauend auf diesen allgemeinen Grundlagen haben sich in verschiedenen Wissenschaftsdisziplinen unterschiedliche Integrationsbegriffe durchgesetzt. Für diese Arbeit sind besonders die Begriffe der Betriebswirtschaft, der Informatik und der Wirtschaftsinformatik bzw. angewandten Informatik von Bedeutung. Die folgenden Ausführungen stellen eine kurze Zusammenfassung verschiedener Arbeiten dar, die diesen Aspekt detaillierter aufbereiten[34] um einen kurzen Überblick zu geben und die – für diese Arbeit notwendigen – Grundlagen zusammenzufassen.

In der **Betriebswirtschaftslehre** (BWL) spielt Integration in den Teilgebieten der Organisationstheorie und Organisationsgestaltung eine wichtige Rolle. Bereits Chester Barnard definierte formale Organisationen als Systeme bewusst koordinierter Handlungen[35] oder Kräfte von zwei

[26] (Szyperski, 1998, S. 36)
[27] Vgl. (Thränert, 2005, S. 11) (Thränert, 2008, S. 5), (Fischer, 2008, S. 14ff.), (Schäfer, 2008, S. 8).
[28] Vgl. (Duden, 2007).
[29] Vgl. (Grochla, 1974, S. 38), (Lehmann, 1980, S. 976), (Fischer, 2008, S. 14).
[30] Vgl. (Thränert, 2008, S. 5), (Fischer, 2008, S. 14).
[31] Vgl. (Thränert, 2008, S. 20).
[32] Vgl. (ebenda, S. 40).
[33] Vgl. Abschnitt 2.1.
[34] Vgl. (Fischer, 2008), (Schäfer, 2008), (Thränert, 2008).
[35] Eine Abgrenzung von Integration und Koordination findet sich in (Alt, 1997, S. 211f.).

oder mehr Personen[36]. Kosiol sieht Organisation als „integrative Strukturierung von Ganzheit“[37]. In der Organisationslehre stehen als Integrationsobjekte Menschen und Aufgaben im Vordergrund.[38] Insbesondere Aufgabenzerlegung und -synthese spielen eine wichtige Rolle und bilden die Integrationstätigkeiten der Betriebswirtschaft.[39] In der neueren Forschung zu Unternehmenszusammenschlüssen und -akquisitionen wird als Integrationsobjekt ein ganzes Unternehmen betrachtet.[40]

Die **Informatik** betrachtet Integration hauptsächlich im Rahmen des Softwareengineering, dabei wird das Zusammenspiel einzelner Komponenten (Daten, Funktionen, Benutzerschnittstellen, Softwaresubsysteme) im Rahmen eines Softwaresystems oder einer Softwareinfrastruktur auf verschiedenen Granularitätsstufen untersucht.[41] Beispiele sind die objektorientierte Softwareentwicklung und Software aus Komponenten. Erstere befasst sich mit der Funktionsverteilung und Kommunikation von Objekten welche Instanzen von Klassen darstellen [42] und Letztere mit Komponenten auf einem gröberen Granularitätsniveau.[43] Darüber hinaus können auch Hardwaresysteme von den Betrachtungen betroffen sein.

Der Integrationsbegriff der **Wirtschaftsinformatik** ist geprägt von der Tatsache, dass diese grundlegend ein interdisziplinäres Verständnis hat und sich als Mediator zwischen Betriebswirtschaft und Informatik sieht. So wird die Wirtschaftsinformatik teilweise als Integrationsdisziplin bezeichnet[44], zumindest aber ist die integrierte Informationsverarbeitung eine der wesentlichen Forschungsströmungen dieser Disziplin. So kann eine technische und eine organisatorische Integration unterschieden werden.[45] Für eine detailliertere Analyse der Integrationsbegriffe unterschiedlicher wissenschaftlicher Disziplinen sei auf die Arbeiten von Thränert[46] und Fischer[47] verwiesen.

Interoperabilität wird häufiger im Zusammenhang mit Integration gebraucht, sodass eine Abgrenzung zu diesem erforderlich scheint. Nach Vernadat lässt sich Interoperabilität als die Eigenschaft eines Systems definieren, Teile eines anderen Systems zu nutzen. In der Sprache von Informationssystemen und Geschäftsprozessen bedeutet dies, dass ein Daten-, Informations- und Wissensaustausch oder die Nutzung von Funktionalität möglich sein muss. Dabei wird betont, dass dies ohne einen speziellen Aufwand möglich sein muss.[48] Die Abgrenzung zur Integration stellt sich wie folgt dar:

> *„Interoperability is indeed one of the many facets of enterprise integration. An integrated family of systems must necessarily be interoperable in one form or another. Interoperable systems do not need to be integrated.“*[49]

[36] (Barnard, 1938) zitiert nach (Berger und Bernhard-Mehlich, 1999, S. 134). Vgl. auch (Barnard, 1968).
[37] (Kosiol, 1962), vgl. dazu auch (Fischer, 2008, S. 16).
[38] Vgl. (Thränert, 2005, S. 13f.), (Thränert, 2008, S. 9), (Fischer, 2008, S. 15).
[39] Vgl. (ebenda, S. 15).
[40] Vgl. (Hartmann, 2001, S. 34ff.), (Bauch, 2004, S. 44ff.), (Schäfer, 2008, S. 8f.).
[41] Vgl. (Fischer, 2008, S. 15).
[42] Vgl. (Balzert, 1998, S. 512f.).
[43] Vgl. (Szyperski, 1998).
[44] Vgl. (Heinrich, 2005),(Becker u. a., 2007).
[45] Vgl. (Herden u. a., 2006, S. 11).
[46] Vgl. (Thränert, 2008).
[47] Vgl. (Fischer, 2008, S. 14ff.).
[48] Vgl. (Vernadat, 2009, S. 1531), sowie (Wegner, 1996), (Chen; Doumeingts und Vernadat, 2008).
[49] (Vernadat, 2009, S. 1531), vgl. auch (Molina u. a., 2007, S. 354). Für die Einordnung von Interoperabilität in ein Begriffsnetz zu Begriffen im Umfeld der Integration vgl. (Thränert, 2008, S. 12).

Bezogen auf die ex ante und die ex post Integration bedeutet dies, dass zwei Systeme ex post ohne Aufwand miteinander integrierbar sind, wenn sie vollständig interoperabel sind.[50] Tätigkeiten der ex ante Integration fördern demnach die Interoperabilität. Andersherum formuliert entstehen Probleme bei der ex post Integration unter anderem durch die Existenz von Interoperabilitätskonflikten, welche ihren Ursprung in der Heterogenität der Integrationsobjekte haben.

Zusammenfassend lässt sich formulieren, dass in allen drei betrachteten Disziplinen das Bilden eines Ganzen aus einzelnen Elementen im Vordergrund steht und dass die Differenzierung der Integrationsbegriffe im Wesentlichen über die Gegenstände der Integration erfolgt. Es lässt sich feststellen, dass die Integrationsgegenstände mit den in Abschnitt 2.1 gebildeten Subsystemen korrespondieren. Diese Trennung hat in der unternehmerischen Realität jedoch nur untergeordnete Bedeutung, wie das Forschungsgebiet der Unternehmensintegration zeigt, welches von mehreren Disziplinen adressiert wird. Molina schreibt über die Unternehmensintegration – im Englischen Enterprise Integration genannt – dass sie maßgeblich in den getrennten Forschungsdisziplinen der Unternehmensmodellierung und der Informationstechnologie bearbeitet wurde.[51] Dabei wurde im Bereich der Unternehmensmodellierung nach Möglichkeiten gesucht, Architekturen zu beschreiben und Ansätze zu finden, mit denen sich lokale und globale Ziele im Rahmen von Entscheidungsfindungsprozessen aufeinander abstimmen lassen. Die folgende Abbildung 2.2 verdeutlicht die Überschneidung der Integrations- und Systembegriffe. Der Übersichtlichkeit halber wurden einige Subsysteme, welche bereits in Abbildung 2.1 vorgestellt wurden, entfernt. Es wird deutlich, dass sich die Integrationsbegriffe überschneiden und keiner – für sich allein

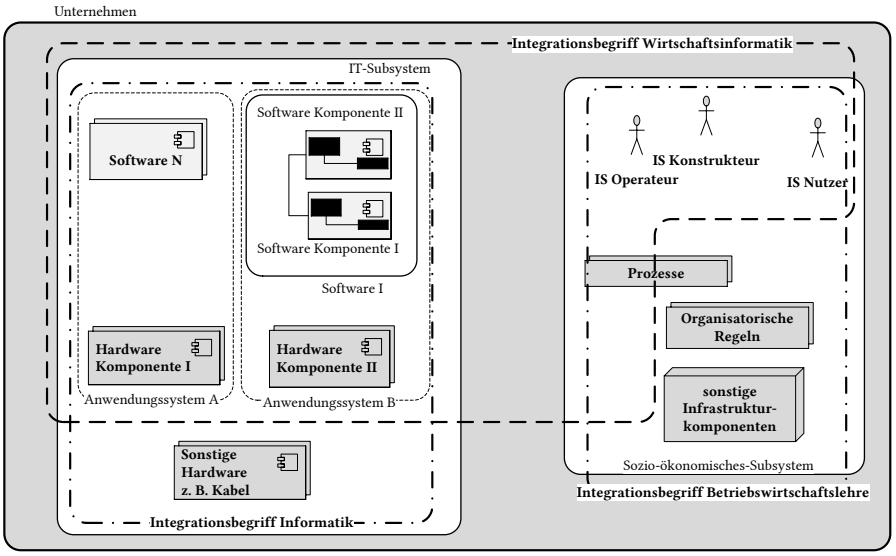


Abbildung 2.2: Integrationsbegriffe der Betriebswirtschaftslehre, Informatik und Wirtschaftsinformatik

genommen – in der Lage ist alle Integrationsfragestellungen in einem Unternehmen abzudecken. Bei der Integration im betrieblichen Kontext handelt es sich um ein interdisziplinäres Gebiet. Die Methoden und Werkzeuge, welche zum Einsatz kommen, sind nicht gänzlich technischer Natur und entstammen nicht ausschließlich der Informatik. Die Interdisziplinarität reicht bis in die Betriebswirtschaft hinein und macht Integration zu einem originären Gegenstand der angewandten

[50] Dieser Umstand ist in der Praxis nur in den seltensten Fällen gegeben.
[51] Vgl. (Molina u. a., 2007, S. 354).

Informatik, oft auch der Wirtschaftsinformatik. Hinter konkreten Integrationsanforderungen verbergen sich zumeist betriebswirtschaftliche Notwendigkeiten.

Integration betrieblicher Informationssysteme
Deshalb verwenden wird im weiteren Verlauf der Arbeit als Integrationsverständnis die Integration betrieblicher Informationssysteme verwendet. Diese geht „[...] über die Definition der Informatik für Informationssysteme (ein Informationssystem dient der rechnergestützten Erfassung, Speicherung, Verarbeitung, Pflege, Analyse, Benutzung, Disposition, Übertragung und Anzeige von Informationen) hinaus und adressieren das gesamte informationsverarbeitende soziotechnische System.“[52]

Da für die weiteren Analysen die Konzentration zunächst auf dem Softwaresubsystem liegt, wird Integration im Sinne der Geschäftsprozessintegration aufgefasst. Das Ziel ist demnach die Verknüpfung von Anwendungen um einen automatisierten Arbeitsablauf sicherzustellen, der aus einem bestimmten betrieblichen Prozess – im Sinne einer Vorgabe durch das sozio-ökonomische Subsystem – resultiert. Durch die Konzentration auf das betriebliche Informationssystem, besteht die Aufgabe in der Informationsverarbeitung. Nielsson, Nordhagen und Oftedal[53] identifizieren die Übertragung von Daten und das Auslösen von Aktionen in anderen Systemen. Funktionalitäten, welche in heterogenen Systemen implementiert sind, werden demnach nicht erneut implementiert, sondern es wird sichergestellt, dass eine dem Geschäftsprozess entsprechende Übergabe relevanter Daten (Informationen) und des Kontrollflusses erfolgt. Integration wird demnach als die Sicherstellung von Kommunikation und die dafür notwendige Überbrückung von Heterogenität verstanden. Dabei wird nicht auf das Gleichmachen der Integrationsobjekte abgezielt, sondern auf die Herstellung von Korrespondenz im Sinne des übergeordneten Prozesses des sozio-ökonomischen Subsystems.

Über diese Abgrenzungen hinaus werden im weiteren Verlauf der Arbeit noch weitere Begriffe benötigt, welche im Folgenden erläutert werden. Als Integrationsobjekte werden sämtliche im Rahmen eines betrieblichen Informationssystems abgrenzbaren Komponenten oder Subsysteme betrachtet.

Integrationsbeziehung
Zwischen zwei Integrationsobjekten (IO_1, IO_2) kann eine Integrationsbeziehung (IB_i) bestehen. Dies ist dann der Fall, wenn die Weitergabe von Daten der Zugriff auf Funktionen oder schlicht eine Weitergabe des Kontrollflusses notwendig ist. Ein Integrationsobjekt verfügt demnach über eine Menge an Funktionen und eine Menge an Informationen, welche meist in Form von Datenobjekten[54] implementiert sind. $F_{IO} = f_i, \ldots, f_n$ und $DO_{IO} = do_i, \ldots, do_n$. Dabei bezieht sich eine Integrationsbeziehung immer auf einen spezifischen Datensatz oder eine spezifische – zu nutzende – Funktionalität unabhängig davon, ob das Integrationsobjekt eine Menge an Funktionen oder Daten beinhaltet. Sind andere Daten oder Funktionen ebenfalls betroffen, so sind dies jeweils eigene Integrationsbeziehungen. Eine Integrationsbeziehung IB ist ein Tripel welches die betroffenen Integrationsobjekte und die Funktion F oder das betroffene Datenobjekte DO einander zuordnet. Am Beispiel einer Funktion würde $IB = \{IO_1, IO_2, F_i\}$ entstehen. Deshalb wird im Folgenden immer von bilateralen Integrationsbeziehungen gesprochen. Multilaterale Integrationsbeziehungen mIB werden als eine Menge von bilateralen Integrationsbeziehungen aufgefasst $mIB = \{IB_1, \ldots, IB_n\}$. Aufbauend auf diesen beiden Definitionen kann sowohl der

[52] (Hahn, 2011)
[53] Vgl. (Nilsson; Nordhagen und Oftedal, 1990) zitiert nach (Rossak und Ng, 1991, S. 99).
[54] Die Betrachtung an dieser Stelle erfolgt auf der Ebene logischer Applikationen und abstrahiert von den technischen Details der Implementierung. Zu jedem Datenobjekt gibt es auf der Ebene des Prozesses ein korrespondierendes Geschäftssobjekt – z.B. Kunde – welches von einer Implementierung abstrahiert und in der Sprache der modellgetriebenen Entwicklung auf der Ebene des Computation Independent Model (CIM) angesiedelt wäre.

Begriff eines Integrationsszenarios definiert werden, als auch Optimierungsprobleme formuliert werden, bei denen Muster für bilaterale Integrationsbeziehungen durch effizientere Muster für die Lösung von multilateralen Beziehungen abgelöst –also gleichartige Integrationsbeziehungen zusammengefasst– werden.[55]

Integrationsszenario

Im Kontext betrieblicher Informationssysteme werden die Integrationsobjekte nur in der für ein Unternehmen gegenwärtig gewünschten Art und Weise integriert. Dies bedeutet, dass Integrationsobjekte, obwohl sie bezüglich einer Funktionalität oder spezifischer Datenobjekte interoperabel sind nicht miteinander integriert werden. Oder dass die Integrationsobjekte integriert werden müssen, obwohl sie nicht vollständig interoperabel sind. Da sich die Definition eines Integrationszustandes auf ein spezifisches Integrationsobjekt bezieht[56], wird hier der Begriff eines Integrationsszenarios (ISZ) definiert.

Bisher wurde dieser Begriff zwar in einigen Publikationen gebraucht jedoch eher in einem umgangssprachlichen Sinne, ohne eine exakte Definition zu hinterlegen.[57] Einzig Vogler[58] geht tiefer auf diesen Begriff ein. In Ihrer Arbeit ist ein Integrationsszenario das Ergebnis der Voruntersuchung.[59] Es kann ein Teil des Applikationsszenarios sein, wenn bereits Integrationsbeziehungen zwischen Anwendungen bzw. Applikationen bestehen.[60] In einer Darstellung – welche Vogler als Beispiel für ein Integrationsszenario angibt – sind Komponenten des Softwaresubsystems enthalten und die jeweiligen Integrationsbeziehungen. Die Integrationsbeziehungen sind mit den auszutauschenden Datenobjekten annotiert.[61]

Auf dieser Basis wird definiert, dass ein Integrationsszenario eine Menge an Integrationsobjekten umfasst, deren gegenwärtiger Integrationszustand bezüglich eines gewünschten Zielintegrationszustandes nicht befriedigend ist. Es besteht also aus einer Menge an Integrationsbeziehungen, bei denen es sich sowohl um bilaterale als auch multilaterale handeln kann. $ISZ = \{IB\}$ mit $IB = \{IB_1, \ldots, IB_n\}$. Ein Integrationsszenario ist gewissermaßen die Problembeschreibung für ein Integrationsprojekt. Wichtig ist anzumerken, dass diese Problembeschreibung kein objektives Vollständigkeitskriterium kennt und sich nur nach den Bedürfnissen des Unternehmens richtet, was die Menge der einbezogenen Integrationsobjekte und deren Integrationsbeziehungen angeht. Es handelt sich um willkürliche Grenzen im Sinne einer Vorgabe durch das sozio-ökonomische Subsystem. Legt man fest, dass ein ISZ_{ist} nur bestehende Integrationsbeziehungen beinhalten und eine ISZ_{soll} den gewünschten Zielzustand markiert, so können über einen Vergleich die notwendigen Veränderungen bestimmt werden.

Integration Engineering

Seit längerer Zeit wird ein Fehlen einer einheitlichen Methodologie der Integration bemängelt.[62] Dies zu beseitigen ist das erkläre Ziel des Integration Engineering (IE) welches einen planmäßigen, strukturierten Entwurf einer Integrationslösung mit ingenieurmäßigen Prinzipien zum Ziel hat.[63] Es wird ein methodologischer Rahmen zur Verfügung gestellt, der dies ermöglicht. Dabei fokussiert das Integration Engineering nicht auf einzelne Integrationsobjekte, sondern versteht sich als übergeordnete anwendbare Theorie der Integration. Wird also das gesamte Unternehmen als Gestaltungsziel gewählt, so ist es erklärtes Ziel des IE das Unternehmen zu analysieren und zu

[55] Vgl. zu dieser Thematik in Form von weiterführenden Überlegungen Kapitel 8.
[56] Vgl. (Thränert, 2008, S. 20f.).
[57] Vgl. (Kornek, 2004, S. 42), (Schäfer, 2008, S. 130).
[58] Vgl. (Vogler, 2006).
[59] Vgl. (ebenda, S. 267).
[60] Vgl. (ebenda, S. 273).
[61] Vgl. Abbildung 6.38 in (ebenda, S. 274).
[62] Vgl. (Lublinsky und Farell, 2002, S. 41),(Lam und Shankararaman, 2004, S. 40)
[63] Vgl. (Rautenstrauch, 1993; Thränert, 2008).

gestalten, mit dem Ziel einen definierten Integrationszustand zu erreichen oder zu erhalten.[64] Das Gestaltungsziel des Integration Engineering ist die Integrationsarchitektur eines Unternehmens, welche Teil der Unternehmensarchitektur (UA) ist. Mit dieser Einordnung wird deutlich, dass das Integration Engineering als Gestaltungsziel die Unternehmensarchitektur hat und sich somit in den Bereich des Architekturmanagements einordnen lässt, oder zumindest die Aktivitäten dieses Bereiches als Einfluss berücksichtigen muss.

2.3 Wiederverwendungsmechanismen bei der Integrierung, Muster und Mustersysteme

Wiederverwendung ist definiert als „the process of creating software systems from existing software rather than building them from scratch“[65]. Die ersten Ideen zur systematischen Wiederverwendung wurden auf der NATO-Software-Engineering-Konferenz thematisiert.[66] Seit dem im Rahmen der Wiederverwendung unter anderem Produktivitäts- und Rentabilitätsfragen[67] untersucht. Im Bereich der Softwareentwicklung hat es zahlreiche Ansätze gegeben [68], zu denen die Objektorientierung[69], Komponentenansätze[70], domänenspezifische Entwicklung[71] und Softwareproduktlinien[72] zählen.

Eines der wesentlichen Ergebnisse dieser Forschung war die Definition des Hauptgegenstandes der Wiederverwendung dem Asset[73] „An asset is a collection of artifacts that provides a solution to a problem. The asset has instructions on how it should be used and is reusable in one or more contexts, such as a development or a runtime context“[74]. Diese Bemühungen haben die Object Management Group veranlasst die Reusable-Asset-Spezifikation zu veröffentlichen, welche auf einem abstrakten Niveau ein Asset charakterisiert und spezifiziert.[75]

Im Rahmen der modellgetriebenen Entwicklung werden Modelle als wiederverwendbare Artefakte diskutiert[76], die: "provide a unique opportunity to mitigate complexity, improve consumability, and reduce time to market“[77]. Modellgetriebene Ansätze wie die modellgetriebene Softwareentwicklung[78] (MDSD)[79], das Model-Driven-Engineering (MDE)[80] und die Model-Driven-Architecture (MDA)[81] zielen darauf ab, domänenspezifische Abstraktionen zu finden und diese durch Modelle greifbar zu machen[82]. Damit wird Wiederverwendung auf eine konzeptionelle bzw. die Entwurfsebene gehoben. Ein Umstand, der im Rahmen der Integration gewichtige Bedeutung hat, denn die bisher genannten Ansätze haben Wiederverwendung von fertig implementierten

[64] Vgl. (Molina u. a., 2007, S. 355).
[65] (Lucredio u. a., 2006)
[66] Vgl. (McIlroy, 1968).
[67] Vgl. (Shiva und Shala, 2007).
[68] Vgl. (Larsen, 2006; Shiva und Shala, 2007).
[69] Vgl. (Balzert, 2001).
[70] Vgl. (Szyperski, 1998).
[71] Vgl. (Evans, 2004).
[72] Vgl. (Northrop u. a., 2006).
[73] Ein dem Englischen entstammender Begriff der in die deutsche Sprache übernommen wurde. Am ehesten kann Asset mit Vermögens- oder Anlagegenstand übersetzt werden.
[74] (Larsen, 2006, S. 543)
[75] Vgl. (Object Management Group, 2006).
[76] Vgl. (Gebauer und Ngonga Ngomo, 2008).
[77] (Larsen, 2006, S. 542)
[78] engl. model-driven software development
[79] Vgl. (Stahl und Völter, 2006).
[80] Vgl. (Bezivin, 2005).
[81] Vgl. (Mellor u. a., 2004).
[82] Vgl. (Stahl und Völter, 2006, S. 4).

Artefakten wie Komponenten oder Bibliotheken ermöglicht.[83] Dies hat jedoch im Bereich der Integration keinen großen Nutzen, da sich Teilphasen im Vergleich zur Anwendungsimplementierung massiv verschieben. Die Bedeutung der Architektur- und Entwurfsphasen steigen und die der Implementierung tritt in den Hintergrund.[84]

Dabei bleibt anzumerken, dass der Begriff Wiederverwendung eigentlich nur einen Teil – den des wiederholten Einsatzes – von drei Fragestellungen anspricht, die mit den nachfolgend vorgestellten Konzepten der Wiederverwendung auf Entwurfs- und Architekturebene adressiert werden:

- Wie kann ein gemeinsames Verständnis über einen Problemraum oder Arbeitsbereich hergestellt werden?
- Wie kann einmal erworbenes Wissen unter Mitarbeitern verteilt werden?
- Wie kann mit der Fluktuation von Mitarbeitern umgegangen werden?

Eine der wichtigsten Fragen, die sich bei diesen Bemühungen stellt, ist die der Konservierung des notwendigen Wissens. Bisher hat es dazu im Bereich der Informationssystemintegration nur wenige Ansätze gegeben.[85] Neben Referenzarchitekturen und Frameworks findet sich unter den Ansätzen der Wiederverwendung in der Informatik und Wirtschaftsinformatik auch das Konzept von Mustern. Dabei finden Muster – besser unter dem englischen Begriff Pattern bekannt – traditionell in der Softwareentwicklung Anwendung. Referenzmodelle und damit auch Referenzarchitekturen sind hingegen ein Untersuchungsgebiet der Wirtschaftsinformatik.[86] Die eingangs dargelegte interdisziplinäre Charakteristik von Integrationsproblemen bedingt, dass beide Ansätze infrage kommen. Zumal bei eingehender Analyse deutlich wird, "wie nahe die einzelnen Ansätze beieinander liegen. Im Kern geht es um die Wiederverwendung von Artefakten im Konstruktionsprozess"[87]. Obwohl darüber hinaus weitere Mechanismen wie Produktlinien[88] bekannt sind – denen im Unterschied zu den Vorgenannten zugeschrieben wird systematische Wiederverwendung zu ermöglichen – ist deren Gebrauch im Themenfeld Integration bisher nicht bekannt. Im Folgenden wird das Konzept von Mustern detailliert vorgestellt.

In diesem Kapitel wird das Konzept von Mustern als Mechanismus der Wiederverwendung eingeführt und dargelegt, wie dieses Konzept bisher Eingang in die Informatik und Wirtschaftsinformatik gefunden hat. Dabei wird jedoch nicht die gesamte Variationsbreite der Verwendungen beleuchtet, da diese, Gegenstand des Kapitels 5 ist. Neben der Einführung des Konzeptes (Kapitel 2.3) werden darüber hinaus bestehende Ansätze zur Organisation mehrerer Muster in Form von Hierarchien, Mustersystemen und Mustersprachen erläutert (Kapitel 2.3). Weiterhin werden erste Unterscheidungen verschiedener Musterarten sowie Ansätze zur Dokumentation dargelegt, welche – im Hinblick auf die Integration – in Kapitel 5 detaillierter analysiert werden.

Musterbegriff

Das aus der Architektur von Gebäuden stammende Werk von Alexander hat zu der Prägung des Begriffes Muster geführt.[89] Die dort entwickelte Idee – bestimmte Problematiken beim Entwurf von Gebäuden und Plätzen durch Lösungsmuster zu erfassen – wurde unter anderem durch Gamma u. a. in die Software Entwicklung übertragen[90] und ist seit dem auf breiter Basis akzeptiert. Grundsätzlich wird mit der Verwendung von Mustern das Ziel verfolgt, einen individuellen, innovativen Entwurf, bei dem für gleichartige Probleme immer wieder Lösungen

[83] Zur Wiederverwendung bei der Integration vgl. auch (Gebauer, 2007).
[84] Vgl. (Lublinsky und Farell, 2002).
[85] Vgl. (Purao; Paul und Smith, 2007, S. 519).
[86] Vgl. (Winter, 2009b, S. 535).
[87] (ebenda, S. 537f.)
[88] Vgl. (Clements und Northrop, 2002).
[89] Vgl. (Alexander u. a., 1977; Alexander, 1979).
[90] Vgl. (Gamma u. a., 1995).

von Grund auf neu entwickelt werden, in einen strukturierten Entwurf zu überführen.[91] Dieser nutzt dabei Muster und löst so nicht jedes Problem von Grund auf neu. Die Annahme, die dabei mitschwingt, ist, dass sich diese Muster intersubjektiv bewährt haben und damit eine höhere Qualität erreicht wird als bei einem innovativen Entwurf. Nachteilig ist, dass sich unter den Mustern – für erstmalig auftretende Problemstellungen – kein Lösungsansatz findet.[92] In einem solchen Fall muss auf einen innovativen Entwurf zurückgegriffen werden.

Allen Arten von Mustern ist gemein, dass neben der eigentlichen Problemlösung auch das Problem selbst, sowie weitere Informationen enthalten sein sollen. Häufig erfolgt die Beschreibung von Mustern in Form eines Templates, welches zumindest für einzelne Publikationen konsistent verwendet wird. In der Literatur existiert eine Vielzahl an unterschiedlichen Varianten, auf die an dieser Stelle jedoch nicht weiter eingegangen werden soll. Ein umfangreicher Vergleich findet sich bei Mailch[93] und Schmidt[94] diskutiert spezifische, für die Integration geeignete Ausprägungen. Muster stellen keine direkt wiederverwendbaren Artefakte dar, sondern müssen stets im Kontext des aktuellen Problems implementiert werden. Sie ermöglichen Wiederverwendung auf der Entwurfsebene und nicht wie im Falle einer Code-Bibliothek um Wiederverwendung auf Implementierungsebene.

Nach Schmidt u. a.[95] können verschiedene Arten von Mustern unterschieden werden. Entwurfsmuster nach Gamma u. a.[96] sind Muster, welche sich mit den Elementen eines Softwaresystems und den Beziehungen zwischen diesen befassen. Sie beschreiben eine Struktur von kommunizierenden Objekten, welche ein generelles Entwurfsproblem in einem bestimmten Kontext lösen. Aus dieser Definition lässt sich ableiten, dass Entwurfsmuster den inneren Aufbau von Software beschreiben und die Bestandteile einer einzelnen Softwarekomponente strukturieren.

Neuere Arbeiten versuchen das Konzept des Musters, welches sich beim Entwurf von Softwarekomponenten bewährt hat, von der Konstruktion eines einzelnen Softwaresystems auf die Konstruktion von Systemen aus Systemen zu übertragen. Die so entstandenen Ansätze verwenden zur Abgrenzung häufig den Begriff eines Architekturmusters, um darauf hinzuweisen, dass es sich um komplexere Systeme handelt, bei denen die Granularität der Subsysteme – und damit auch deren Komplexität – größer geworden ist.

> *They „express the fundamental, overall structural organization of software systems and provide a set of predefined subsystems, specify their responsibilities, and include guidelines for organizing the relationships between them“*[97].

Bezug nehmend auf diese Definition erläutert Lutz die Abgrenzung von Entwurfsmustern:

> *„A key difference between the design patterns and architecture patterns , however, is that architecture patterns specify the systemwide structural properties of an application and have an impact on the architecture of its subsystems. Design patterns have no such systemwide impacts“*[98].

[91] Vgl. (Shaw und Garlan, 1996).
[92] Vgl. (Garlan und Shaw, 1993).
[93] Vgl. (Malich, 2008).
[94] Vgl. (Schmidt, 2010).
[95] Vgl. (Schmidt und Buschmann, 2003).
[96] Vgl. (Gamma u. a., 1995).
[97] (Buschmann; Meunier u. a., 1996)
[98] (Lutz, 2000, S. 66)

Darüber hinaus existieren vielfältige, weitere Ansätze dieses Konzept zu nutzen. Diese reichen von Organisationsmustern über Analysemuster[99] bis hin zu Mustern, welche auf der Ebene der Geschäftstätigkeit[100] anzusiedeln sind.

In Kapitel 5 wird dieses Konzept aufgegriffen und es wird untersucht, welche Ansätze speziell im Bereich der Integration existieren. Das sich anschließende Unterkapitel 2.3 befasst sich mit Mechanismen mehrere Muster – welche immer nur ein Teilproblem eines größeren Problems lösen – sinnvoll, gemeinsam zu organisieren und sie zueinander in Beziehung zu setzen.

Organisation von Mustern: Mustersysteme und Mustersprachen
Bereits ein einzelnes Muster kann einen hohen Nutzwert besitzen und einen Entwurfsprozess entscheidend verkürzen. Ein ungleich größerer Nutzwert wird jedoch dem planvollen, gemeinsamen Einsatz mehrerer Muster zugeschrieben. Zu diesem Zweck ist es notwendig verschiedene Beziehungen zwischen einzelnen Mustern zu erfassen, da sich Muster auch gegenseitig ausschließen können. In der Literatur lassen sich zu diesem Zweck Musterkataloge, Mustersysteme und Mustersprachen voneinander unterscheiden.[101]

Einige Publikationen listen Muster auf, die zwar einem gemeinsamen Problembereich zugeordnet werden können, jedoch ohne zwischen den einzelnen Mustern Beziehungen zu kennzeichnen. In einem solchen Fall wird von einem Musterkatalog gesprochen. Fragen nach dem gemeinsamen Einsatz – ob Muster sich gegenseitig ausschließen oder ergänzen – lassen sich so nicht beantworten. Deshalb ist ein Musterkatalog mit einem Wörterbuch vergleichbar, welches ebenfalls keine Aussagen über die gemeinsame Verwendbarkeit der Wörter trifft.[102]

Den Anspruch den Nutzwert von Mustern über die bloße Sammlung hinaus zu erhöhen, erheben Mustersysteme.

> *„A pattern system for software architectures is a collection of patterns for software architectures, together with guidelines for their implementation, combination and practical use in software development.“*[103]

Im Gegensatz zu einem Mustersystem webt eine Mustersprache nach Alexander[104] eine Menge in Beziehung stehender Muster zusammen um ein Vokabular zu definieren, dass es ermöglicht über Entwicklungsprobleme zu sprechen. Darüber hinaus stellt die Sprache einen Prozess für die geordnete Lösung dieser Probleme zur Verfügung.[105]

> *“A pattern language prescribes only the allowable configurations among small groups of patterns; it can be thought of as a way of describing dependencies between patterns.“* [106]

Ein Problem, welches im Zusammenhang mit Mustersprachen diskutiert wird, ist die Vollständigkeit der Mustersprache [107], welche sich auf die Menge der Probleme und die durch die Sprache angebotenen Lösungen bezieht. Im Wesentlichen gibt es konzeptionell keine Unterschiede in den Arten der Beziehungen, welche zwischen Mustern beschrieben werden, vergleicht man Mustersysteme mit Mustersprachen. Häufig ist die Vollständigkeitsdiskussion der Grund, den Begriff Mustersystem anstelle einer Mustersprache zu verwenden. Ein weiterer Aspekt der Vollständigkeit ist die Frage, wann eine Mustersprache als vollständig angewandt gilt. Dies ist dann der Fall,

[99] Vgl. (Kolp; Giorgini und Mylopoulos, 2003).
[100] Vgl. (Hruby, 2006; Braga; Germano und Masiero, 1999).
[101] Vgl. (Tešanović, 2004, S. 10f.).
[102] Vgl. (Salingaros, 2000, S. 155).
[103] (Buschmann; Meunier u. a., 1996, S. 361), vgl. auch (Malich, 2008, S. 123).
[104] Vgl. (Alexander u. a., 1977).
[105] Vgl. (Schmidt und Buschmann, 2003), sowie (Zwanziger und Herden, 2004).
[106] (Porter; Coplien und Winn, 2005, S. 3)
[107] (Klose, 2002, S. 110), (Buschmann; Meunier u. a., 1996, S. 360)

wenn alle vorliegenden Probleme gelöst wurden.[108] Das weist bereits auf die Relativität dieses Kriteriums hin, da hierfür die aktuelle Problemstellung die einzige Entscheidungsgrundlage liefern kann.

Da es zwischen Mustersystemen und Mustersprachen kaum Unterschiede in den beschriebenen Beziehungen gibt, werden nun mögliche Beziehungen zwischen Mustern vorgestellt. Ziel dieser Darstellung ist es, die Beziehungen später gezielt zu verwenden. In der Literatur existieren sehr wenige Arbeiten, die sich explizit mit Beziehungen zwischen Mustern beschäftigen.

Die Auswertung der wissenschaftlichen Literatur lieferte Quellen, welche sich explizit mit Beziehungen zwischen Mustern auseinandersetzen. Dies bedeutet nicht, dass es in anderen Publikationen (sekundäre) nicht auch Hinweise auf Beziehungen gibt. Diese sind aber meistens als Instanz zu sehen, da sie zwei konkrete Muster miteinander in Beziehung setzen. Die als primäre Publikationen eingestuften Quellen befassen sich explizit mit der Definition von Relationstypen zwischen Mustern. Im Folgenden werden die Inhalte der einzelnen Publikationen vorgestellt und im Anschluss daran ein gemeinsames System der Beziehungen zwischen Mustern als Synthese entwickelt.

Die Arbeit von Riehle[109] lässt sich – obwohl häufig in diesem Zusammenhang zitiert – für eine Analyse der Relationstypen zwischen Mustern nicht direkt verwenden. Muster werden in dieser Arbeit auf Basis der Wurzelobjekte nach Gamma u. a.[110] beschrieben und die Beziehungen auf dieser Basis definiert. Dies entspricht der Komposition auf einer tieferen Abstraktionsebene, als der hier gewählten. Dennoch trägt diese Arbeit den wichtigen Gedanken des zusammengesetzten Musters[111] bei. Dieses unterscheidet sich von dem konkreten Muster, welches im Englischen den gleichen Namen – Composite – trägt und bezeichnet ein Konzept. Ein zusammengesetztes Muster kann entstehen, wenn für ein spezifisches Problem immer wieder derselbe Pfad – also dieselbe Reihenfolge Mustern – aus einer Mustersprache zur Anwendung gebracht wird. Dies weist auf ein Muster hin, welches auf einer höheren Abstraktionsstufe angesiedelt ist. Dies ist der wesentliche Grundgedanke, der hinter der Beschäftigung mit Relationstypen zwischen Mustern steckt.

Zimmer[112] befasst sich in seiner Arbeit gezielt mit den Beziehungen zwischen Entwurfsmustern und kann damit als eine der ersten primären Arbeiten auf diesem Gebiet eingestuft werden. Als Grundlage dient eine Analyse der Muster nach Gamma u. a.[113]. Dabei werden die folgenden Relationstypen für eine bilaterale Beziehung zwischen 2 Mustern X und Y identifiziert: *Uses*, *Similar* und *Can be combined*. Die *Uses* Beziehung sieht im Fall von X *Uses* Y dabei vor, dass Y ein Bestandteil von X ist. Dies wird im Folgenden mit Kompositions- und Enthaltenseins-Beziehungen gleichgesetzt.

Ebenfalls auf Entwurfsmuster zielt die Arbeit von Noble[114] ab, die Grundlage der Analyse war wiederum die Arbeit von Gamma u. a.[115]. Das Ziel der Arbeit ist explizit die Klassifikation der Beziehungen zwischen den Mustern und wird damit als eine primäre Publikation eingestuft. Dabei unterscheidet Noble zwischen primären und sekundären Beziehungen. Als sekundäre Beziehungen werden von Noble alle diejenigen Beziehungen eingestuft, die sich mithilfe der primären Beziehungen ausdrücken lassen. Diese beiden Klassen werden für die angestrebte Synthese der Beziehungsdefinitionen übernommen. Die primären Relationen umfassen nach Noble

[108] Vgl. (Weiss, 2008).
[109] Vgl. (Riehle, 1997).
[110] Vgl. (Gamma u. a., 1995).
[111] Engl. composite pattern.
[112] Vgl. (Zimmer, 1995).
[113] Vgl. (Gamma u. a., 1995).
[114] Vgl. (Noble, 1998).
[115] Vgl. (Gamma u. a., 1995).

neben *Uses*, *Refines* auch *Conflicts*. Von diesen war nur die Erste bei Zimmer enthalten. Die weiteren acht, von Noble identifizierten, Beziehungen sind sekundäre Beziehungen. Unter anderem auch *Similar*, welche bereits Zimmer beschrieb. Es handelt sich häufig um Inversrelationen zu den primären Relationen. Als Beispiel können *Refined* und *Refined by* genannt werden.

Die Arbeit von Salingaros[116] ist dahin gehend bemerkenswert, als dass sie einer langjährigen Zusammenarbeit mit Christopher Alexander entspringt und sich auf den ursprünglichen Gestaltungsgegenstand bauliche Architekturen bezieht. Die Relationen nach Salingaros bedürfen einer gewissen Interpretation, da sie sich von den beiden vorherigen Arbeiten unterscheiden. Diese sind ohne einen konkreten Bezeichner beschrieben. Weshalb im Folgenden sprechende Bezeichner verwendet werden, welche sich aus den Beschreibungen ableiten lassen. *Containment/Generalization* werden im Fall von *Containment* als einen Fall von *Uses* interpretiert. *Generalization* kann auf *Refines* abgebildet werden. Die Relation *Complementary* kann als eine Erweiterung zu *Uses* aufgefasst werden. Es handelt sich um eine strikte Enthaltenseins-Beziehung. Darüber hinaus bleibt Spielraum offen, ob die Eigenschaft der Vervollständigung bidirektional ist. Bei *Uses* ist dies nicht der Fall. Gilt X *Uses* Y. Dann gilt auch Y *Complementary* X ob dahingegen daraus folgt X *Complementary* Y bleibt offen. In einem solchen Fall wäre jedoch die grundlegende Eignung von X und Y als Muster anzuzweifeln, da beide alleine keine vollständige Lösung liefern. Mit *Coexistence* beschreibt Salingaros die Beziehung zwischen zwei Mustern, welche welche unterschiedliche aber überlappende Probleme lösen und deshalb auf demselben Abstraktionsniveau koexistieren. Interpretiert man dies als die Abwesenheit einer *Conflicts* Relation, so lässt sich diese Relation ebenfalls auf die primären Relationen nach Noble abbilden. Die überlappenden Problemstellungen werden von dieser Abbildung jedoch nicht erfasst und bleiben offen. *Alternative* wird vollständig durch *Conflicts* abgebildet. Es handelt sich um zwei Muster, welche dasselbe Problem mit unterschiedlichen Lösungsansätzen adressieren. *Similar structure* verleitet dazu, diese Relation aufgrund der Namensähnlichkeit auf *Similar* zurückzuführen. Besser ist jedoch die Abbildung auf *Variants*, da es sich bei ähnlicher Struktur um Varianten eines Lösungsschemas handelt. Aus dieser Diskussion resultiert, dass die Relationen nach Salingaros auf die Relationen *Uses*, *Refines*, *Conflicts* und *Variant* nach Noble abgebildet werden.

Die Tabelle 2.1 enthält alle untersuchten primären und sekundären Publikationen, wobei analog zu der vorangegangenen Diskussion der Abbildung der Relationen auf das Schema von Noble, alle in sekundären Publikationen verwendeten Relationen ebenfalls auf das Schema nach Noble transferiert wurden. Diese Abbildungen werden jedoch nicht im Einzelnen diskutiert. Die einzige Ausnahme bildet die *May-use* oder *Can be combinded* Relation, welche bei Zimmer[117] und Kodituwakku und Bertok[118] enthalten ist. Diese werden zu den sekundären Relationstypen hinzugefügt. Es handelt sich dabei um eine Verfeinerung der *Uses* Beziehung, wobei für beide Muster gelten muss, dass sie auch allein existent sind und bereits eine vollständige Lösung liefern. Die sekundären Publikationen stammen aus sehr unterschiedlichen Gebieten. Schmidt; Stal u. a.[119] befassen sich mit Mustern für nebenläufige und vernetzte Systeme, wohingegen Derntl und Motschnig-Pitrik[120] sich mit E-Learning genauer gesagt Blended Learning befassen.

„The use relationship is the most important and most common relationship between patterns.“[121] Dies lässt sich durch die Anzahl des Auftretens in der Analyse bestätigen.[122]

[116] Vgl. (Salingaros, 2000).
[117] Vgl. (Zimmer, 1995, S. 348).
[118] Vgl. (Kodituwakku und Bertok, 2003, S. 64).
[119] Vgl. (Schmidt; Stal u. a., 2000).
[120] Vgl. (Derntl und Motschnig-Pitrik, 2004).
[121] (Kodituwakku und Bertok, 2003, S. 64)
[122] Vgl. 2.1.

	Publikation	strukturelle Relationstypen												temporale Relationstypen		
		primäre			sekundäre											
		uses	refines	conflicts	Used by	Refined by	Variant	Variant Uses	similar	combines	Requires	Tiling	may-use, can be combined	sequence of elaboration	sequence	main sequence
primäre	(Zimmer, 1995)	x							x				x			
	(Noble, 1998)	x	x	x	x	x	x	x	x	x	x	x				
	(Salingaros, 2000)	x	x	x			x									
	(Porter; Coplien und Winn, 2005)														x	
sekundäre	(Schmidt; Stal u. a., 2000)	x														
	(Lind und Goldkuhl, 2001)	x														
	(Schumacher, 2001)	x														
	(Spinellis, 2001)	x											x			
	(Conte u. a., 2002)	x	x				x				x					
	(Kodituwakku und Bertok, 2003)	x	x				x						x			
	(Rizzi u. a., 2003)	x	x													
	(Derntl und Motschnig-Pitrik, 2004)	x	x								x					
	(Avgeriou und Zdun, 2005)	x	x				x		x				x			
	(Heer und Agrawala, 2006)	x														
	(Henninger und Ashokkumar, 2006)	x		x					x		x					
	(Delessy; Fernandez und Larrondo-Petrie, 2007)	x									x					
	(Fernandez u. a., 2008)	x	x						x		x					
	(Kumar und Prabhakar, 2010)	x	x								x					

Tabelle 2.1: Mögliche Relationstypen zwischen Mustern

In verschiedenen Publikationen wird angeregt, die Muster auf verschiedenen Ebenen, häufig auch Layer genannt, anzuordnen. Meistens dient die Einführung dieser Ebenen dem Zweck der Abstraktion.

Lind und Goldkuhl[123] entwerfen auf der Basis der Arbeit von Weigand[124] eine Musterarchitektur, welche auf Layern basiert. Zwischen den Layern tritt ausschließlich der Relationstyp *consist* auf.[125] Bei den Instanzen unterscheiden sich lediglich die Kardinalitäten der beteiligten Layer.

Die Vorstufe einer Mustersprache stellen Mustersequenzen dar:[126] Bei diesen handelt es sich um eine Reihenfolge, in der Muster angewandt werden, um ein komplexeres Designproblem zu lösen.[127] Sie enthalten also bis zu einem gewissen Grad Beziehungen, die unter den einzelnen Mustern existieren. Nach allen Ausführungen zu diesem Konzept kann geschlussfolgert werden, dass jede Sequenz für sich selbst steht, ohne notwendigerweise Beziehungen zu anderen Sequenzen

[123] Vgl. (Lind und Goldkuhl, 2001).
[124] Vgl. (Weigand und Van den Heuvel, 1998).
[125] Vgl. dazu die Abbildung 8 in (Lind und Goldkuhl, 2001, S. 124).
[126] Vgl. (Buschmann; Henney und Schmidt, 2007b, S. 12ff.).
[127] Vgl. (Porter; Coplien und Winn, 2005).

aufzuweisen. Porter u. a.[128] gehen davon aus, dass eine komplexe Integrationslösung durch die Anwendung mehrerer Muster entsteht. Dabei stellt sich die Frage, in welcher Reihenfolge diese angewandt werden. Sequenzen werden von Mustern auf unterschiedlichen Abstraktionsebenen stellt dies ein wesentliches Problem dar.

Ein weiterer Beitrag zur Kombination von Mustern zu einer komplexen Design Methode wird in der Arbeit von Zimmermann; Zdun u. a.[129] behandelt. Darüber hinaus wird in einer weiteren Arbeit der Autoren das Management von Architectural-Decision-Models (ADMs) beschrieben.[130] Insbesondere die Hinweise auf Verfeinerungen von Mustern und Ansätze die Sequenzen von Mustern - also Kombinationsmöglichkeiten beschreiben - sind interessant. Der Ansatz bleibt auf einem allgemeinen Niveau stehen und beschreibt, mit welchen Ausdrucksmöglichkeiten sich die Architekturentscheidungen festhalten lassen. Diese Entwicklung ist analog zu der von Typ-II-Frameworks im Bereich der Unternehmensarchitektur zu sehen. Es wird ein Typ-I-Framework für die Integration im E-Business angestrebt. Also die Aufzeichnung von Architekturentscheidungen für die Integration im E-Business um das Warum zu erfassen. Darüber hinaus ist zu kritisieren, dass bereits Buschmann[131] auf den Begriff Muster-System abstellt. Eine Mustersprache[132] müsste vollständig sein. Diesen Anspruch erhebt der vorgestellte Beitrag nicht, sodass es sich vielmehr um die Kombination von Mustern zu Systemen mit ADMs handelt.

[128] Vgl. (Porter; Coplien und Winn, 2005).
[129] Vgl. (Zimmermann; Zdun u. a., 2008).
[130] Vgl. (Zimmermann; Koehler u. a., 2009).
[131] Vgl. (Buschmann; Meunier u. a., 1996).
[132] engl. pattern language

3 Experteninterviews zur Systemintegration

„Entweder etwas ist wissenschaftlich erforscht und in der Praxis macht es keiner oder in der Praxis werden Sachen pragmatisch gelöst und der wissenschaftliche Unterbau fehlt völlig. Es gibt ein Gap zwischen der wissenschaftlichen Community und der Praxis.“[1]

Im Rahmen der wissenschaftlichen Diskussion werden eine Vielzahl an Methoden und technischen Lösungen für Integrationsprobleme diskutiert. In der Praxis sind – im Gegensatz zur Wissenschaft – nach wie vor einfache und längst tot geglaubte Lösungen anzutreffen. Als Beispiel können serviceorientierte Architekturen (SOA) dienen, welche – sofern unter der Nutzung von Standards wie Webservices umgesetzt – helfen eine Reihe von Problemen der Integration zu lösen. Schnittstellen sind in einem einheitlichen Format beschrieben und die Kommunikation erfolgt nach einheitlichen Mechanismen. Dennoch kann in der Praxis beobachtet werden, dass sich SOA keineswegs derart als Architektur- und Integrationsprinzip etabliert hat, dass es als einziges Architekturparadigma Verwendung findet. Entgegen der wissenschaftlichen Diskussion singulärer Lösungsprinzipien besteht in der Praxis die Anforderung des Betriebes von Mischarchitekturen, welche schon allein durch Lock-In Effekte entstehen, hervorgerufen durch Investitionen in IT-Systeme. So werden Anwendungssysteme über mehrere Jahrzehnte hinweg gepflegt, was eine Ablösung sehr schwierig und zeitaufwendig macht, je nach der Bedeutung des Systems für die Geschäftstätigkeit des Unternehmens. Darüber hinaus sind Ablösungsbemühungen auch mit einem immensen Risiko des Scheiterns behaftet. Dies führt dazu, dass in der Realität von BIS andere Integrationslösungen entwickelt und genutzt werden, als der wissenschaftliche Diskurs vermuten lässt. Diese Diskrepanz zwischen Wissenschaft und Praxis ist die Ursache für die im Folgenden vorgestellte empirische Untersuchung, welche es ermöglichen soll bessere Einblicke in die Kausalketten der Praxis zu bekommen, die zur Entscheidung pro oder kontra bestimmter Integrationsprinzipien und damit zur Etablierung von praktischen Integrationslösungen führen. Dies entspricht dem, in Kapitel 1 dargestellten, anwendungsorientierten Wissenschaftsbegriff und dient der Bestätigung der Praxisrelevanz der aufgeworfenen Forschungsfrage.

Wie bereits betont, wurde für die empirische Untersuchung die Fragestellung nach den Kausalmechanismen welche zur Ausprägung einer Integrationslösung in einer bestimmten Form führen in das Zentrum gestellt. Dies lässt sich mit den folgenden Fragen beschreiben:

- Warum entscheidet sich ein Unternehmen in einer konkreten Situation für ein bestimmtes Architekturparadigma?
- Warum werden exakt die gewählten Integrationswerkzeuge und -hilfsmittel eingesetzt?
- Wenn mehr als eine Lösung existiert, aus welchen Gründen entscheidet man sich für eine bestimmte?
- Welche Schritte eines Integrationsprojektes werden durch welche Methoden und Werkzeuge unterstützt?

Es handelt sich um eine rekonstruierende Untersuchung, die auf wiederholt auftretende Einflussfaktoren abzielt, weshalb sich qualitative empirische Methoden anbieten. Darüber hinaus gilt: „Integrationswissen ist über zig Leute verteilt. Quantitative Fragen zu Integration sind unmöglich zu beantworten“[2].

[1] (Gebauer und Stefan, 2011b, S. 2), wörtliches Zitat eines Befragten.
[2] (ebenda, S. 3)

Im Bereich der qualitativen Methoden gibt es verschiedene Möglichkeiten wie Fallstudien, Grounded Theory und qualitative Querschnittsanalyse.[3] Grundsätzlich kommen neben der qualitativen Querschnittsanalyse[4] auch Fallstudien infrage. Die Grounded Theory setzt auf eine intensive Beobachtung und induktive Theoriegewinnung.[5] Aufgrund der schlechten Beobachtbarkeit von Integrationsprojekten muss diese Methode verworfen werden. Die Durchführung von Fallstudien – als Methode zur Gewinnung empirischer Daten – wurde ebenfalls verworfen. Szenarien, die in Fallstudien erhoben werden, besitzen unter Umständen für den einzelnen Dienstleister einen generellen Charakter – da er mehr als ein Projekt nach diesem Szenario abwickelt – jedoch lassen sich die gewonnenen Erkenntnisse nicht ohne Weiteres auf andere Dienstleister übertragen. Diese Anforderung sollte die Untersuchung jedoch erfüllen.

Ein weiteres Instrument sind Experteninterviews, welche je nach ausgewähltem Experten eine höhere Aussagekraft haben, als einzelne Fallstudien. Sie greifen auf aggregiertes Erfahrungswissen des jeweiligen Experten zurück. In der Regel entstammt dieses einer Fülle von durchgeführten Projekten, was jedoch stark vom individuellen Experten abhängt.

Experteninterviews sind in den Bereich der qualitativen Querschnittsanalyse einzuordnen. Methodisch gibt es im deutschsprachigen Raum zwei wesentliche Vertreter, welche die Methodik qualitativer Experteninterviews zusammen mit den sich anschließenden Auswertungsverfahren eingehender behandeln. Dies sind die Arbeiten von Mayring[6] und das Vorgehen nach Gläser und Laudel[7]. Die beiden Ansätze weisen in den Details Unterschiede auf, das Verfahren von Mayring öffnet zwar – das in den theoretischen Vorarbeiten entwickelte – Kategoriensystem, in dem es dieses am erhobenen Material überprüft, ist aber schlussendlich darauf ausgerichtet ein Kategoriensystem auf das Material anzuwenden und die Häufigkeiten des Auftretens der Kategorien zu extrahieren.[8] Das Hauptinteresse der Untersuchung war jedoch die Extraktion von Wirkungszusammenhängen, weshalb die Wahl auf das Verfahren nach Gläser und Laudel fiel. Die Methodik dazu wird in Abschnitt 3.1 umrissen.[9] Wie bereits erläutert, kommt der Auswahl der Experten eine besondere Bedeutung zu, da von ihrer Erfahrung abhängt, inwieweit die extrahierten Informationen auf aggregiertem Erfahrungswissen basieren. Deshalb wird die Auswahl der Unternehmen in Abschnitt 3.4 dargelegt. Methodisch stellt das gewählte Verfahren auf theoretische Vorüberlegungen ab, welche Gegenstand von Abschnitt 3.3 sind. Da die durchgeführten Interviews über die in dieser Arbeit betrachteten Aspekte hinausgehen, wird in Abschnitt 3.5 nur der für die Arbeit relevante Teil der Ergebnisse dargestellt.

3.1 Methodisches Vorgehen und Umfang

In der Einführung zu diesem Kapitel wurde die Motivation und die grundsätzliche Auswahl von Experteninterviews dargelegt. Darüber hinaus wurde deutlich gemacht, dass das Verfahren nach Gläser und Laudel[10] eingesetzt wurde. Dieses soll hier kurz umrissen werden, damit der Gang der Untersuchung deutlich wird. Zusätzlich wird eine kurze Übersicht über den Umfang und die zeitliche Einordnung der Untersuchung gegeben, da dies in einer schnelllebigen Branche für

[3] Vgl. (Wilde und Hess, 2007, S. 282).
[4] Die Methode fasst „[...] Erhebungstechniken wie Fragebögen, Interviews, Delphi-Methode, Inhaltsanalysen etc. [...] zusammen“(ebenda, S. 282).
[5] Vgl. auch (Becker u. a., 2007, S. 11) für die Bedeutung dieser Methode in der Wirtschaftsinformatik.
[6] Vgl. (Mayring, 2002).
[7] Vgl. (Gläser und Laudel, 2009).
[8] Vgl. (ebenda, S. 198f.), sowie (Mayring, 2002)
[9] Für alle Details sei auf das Hauptwerk (Gläser und Laudel, 2009) verwiesen.
[10] Vgl. (ebenda).

eine Bewertung der Studienergebnisse von Bedeutung ist. Das methodische Vorgehen wird in Abbildung 3.1 zusammenfassend dargestellt.

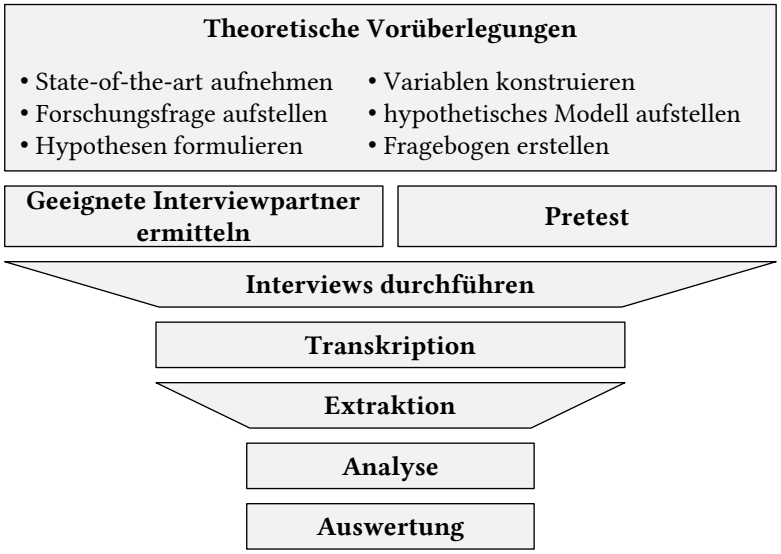


Abbildung 3.1: Schrittfolge der angewandten Methode
Quelle: (Gebauer und Stefan, 2011b, S. 5)

Ausgehend von den theoretischen Vorüberlegungen, welche in einem ausgearbeiteten Interviewleitfaden münden, wurden geeignete Interviewpartner ermittelt. Die Auswahl der Unternehmen ist Gegenstand von Abschnitt 3.4. Parallel dazu wurde der Interviewleitfaden einem Pretest unter realen Bedingungen unterzogen. Dieser wurde mit einem Probanden durchgeführt, welcher zu keiner Zeit in die Erstellung der Studie einbezogen war, bei dem aber aufgrund der Nähe zu den Durchführenden mit umfangreicher und konstruktiver Kritik gerechnet werden konnte. Letztendlich lag der verbesserte Interviewleitfaden in zwei Versionen vor. Die Vorabversion, welche an die Interviewteilnehmer zum Zwecke der Vorbereitung versandt wurde, enthielt keine Nachfragen. In der Version für den Interviewer waren Nachfragen enthalten, welche im Zweifel herangezogen werden konnten, wenn die Antworten des Experten die betreffende Frage nicht in ausreichendem Maße beantworteten.

Im Anschluss erfolgte die Durchführung der Interviews. Es wurden insgesamt 35 Interviews durchgeführt, von denen das Kürzeste 46 Minuten und das Längste 2 Stunden und 39 Minuten in Anspruch nahmen. Die Durchführung der Interviews erfolgte durch zwei Personen mit einer klaren Rollenteilung. Eine – immer dieselbe – Person führte das Interview, wobei die andere Person ein Protokoll anfertigte, um die spätere Auswertung zu erleichtern. Der Interviewer nahm zusätzlich handschriftliche Notizen vor. Alle Interviews wurden per Audioaufzeichnung mitgeschnitten, was die primäre Grundlage für die Auswertung bildete. Von denen 35 Interviews konnten 30 zur weiteren Auswertung herangezogen werden. In fünf Fällen konnte der Status des Interviewpartners als Experte für Integration nicht zweifelsfrei bestätigt werden. Das auszuwertende Audiomaterial summierte sich auf 52 Stunden und 49 Minuten und wurde als Grundlage für die qualitative Inhaltsanalyse transkribiert.

Die Transkription wurde im Zweifelsfall um die Aussagen des Protokolls und der handschriftlichen Notizen ergänzt. Zusätzlich wurde das gesamte Material anonymisiert, sodass bereits in dem Transkript keine Rückschlüsse auf die beteiligten Unternehmen mehr möglich sind. Nach Abschluss dieser Schritte lagen 553 Seiten Textmaterial in DIN A4 mit einer Schriftgröße von 12 Punkten

vor. Die sich anschließende qualitative Inhaltsanalyse[11] entnimmt – Extraktion genannt – dem Basismaterial – also dem Transkript – die Rohdaten und trennt sich so frühzeitig vom Text. Zu diesem Zweck wird ein entsprechendes Suchraster verwendet, welches interessante Aussagen in gemeinsamen Kategorien gruppiert. Diese richten sich meist nach den Variablen des hypothetischen Modells. Dabei werden die Quellenangaben in Form von Querverweisen weiter mitgeführt, sodass diese bis hin zu den Audioaufzeichnungen nachprüfbar bleiben. Auf der Basis dieses Materials erfolgte dann die weitere Verdichtung und Auswertung. Im folgenden Abschnitt werden die Hypothesen vorgestellt, welche der Untersuchung vorangestellt waren.

3.2 Hypothesen

Dieser Abschnitt spiegelt den Beginn der theoretischen Vorüberlegungen wieder, welche die Untersuchung anleiten sollen.[12]Die Hypothesen helfen bei der Formulierung der Interviewfragen. Das aus diesen Vorüberlegungen entstehende Wirkungsmodell[13] kann sukzessive verbessert und ergänzt werden. Der vorgestellten Untersuchung[14] lagen die folgenden Hypothesen zugrunde:

1. Je mehr die Integrationsdienstleister unter Zeitdruck gesetzt werden, desto unstrukturierter ist das Vorgehen.
2. Kurzfristige Ziele stehen bei Integrationslösungen im Vordergrund.
3. Je leichtgewichtiger das Integrationsvorgehen ist, um so schneller ist die Lösung erstellt und um so höher die Akzeptanz.
4. Zu Projektbeginn sind nicht alle Anspruchsgruppen der Lösung bekannt.
5. Integrationsmuster sind weitgehend unbekannt.
6. Domänenspezifische Konzepte (Systemklassen, Prozesse etc.) treten wiederholt in Integrationsprojekten auf, ohne Eingang in die Lösungsgestaltung zu finden.
7. Großrechner sind in vielen Integrationsszenarien großer Unternehmen zu finden. Da deren Ablösung nur selten eine Option ist, müssen Integrationsdienstleister diese Altsysteme[15] bei der Integration besonders berücksichtigen.

Die Hypothesen zwei, fünf und sechs sind von besonderer Bedeutung für diese Arbeit. Die zweite Hypothese unterstreicht die vorherrschenden Kriterien bei dem Entwurf der Integrationsarchitektur und der Auswahl von Integrationshilfsmitteln, welche zwangsläufig einen Einfluss ausüben. Hypothese fünf unterstellt, dass der Untersuchungsgegenstand der Arbeit keinen Eingang in die Praxis findet. Demgegenüber zielt die Hypothese sechs jedoch darauf ab, dass eine Anwendbarkeit aufgrund wiederholt auftretender Probleme sehr wohl möglich wäre.

3.3 Variablenkonstruktion und Interviewleitfaden

Die Darstellung in diesem Abschnitt erfolgt in umgekehrter Reihenfolge zu der Erstellung. Es wird nicht mit den einzelnen Variablen begonnen, sondern zunächst das hypothetische Modell beschrieben und erst danach die einzelnen Variablen. Dies hat den Vorteil, dass das gesamte Wirkungsmodell leichter überblickt werden kann. In Abbildung 3.2 ist das hypothetische Modell dargestellt. In dem vorliegenden Fall ist der eigentliche Untersuchungsgegenstand die

[11] Vgl. (Gläser und Laudel, 2009, S. 197ff.).
[12] Vgl. (ebenda).
[13] Vgl. Abschnitt 3.3
[14] Vgl. (Gebauer und Stefan, 2011b, S. 6f.).
[15] Engl. legacy systems.

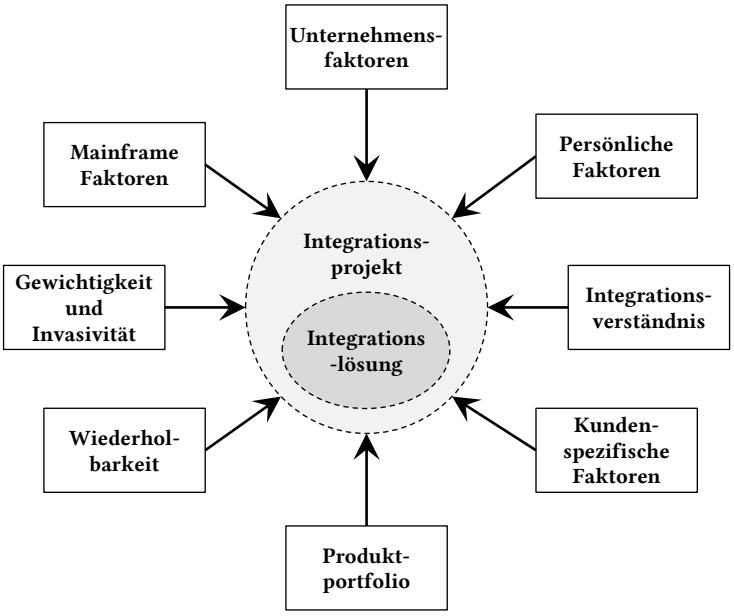


Abbildung 3.2: Hypothetisches Modell
Quelle: (Gebauer und Stefan, 2011b, S. 11)

Integrationslösung. Dadurch, dass Integrationswissen nicht zu den Kernkompetenzen der meisten Unternehmen zählt, werden Integrationsdienstleistungen am Markt eingekauft. Deshalb entstehen Integrationslösungen in Projekten, an denen unter Umständen mehrere Dienstleister und verschiedene Bereiche des Kundenunternehmens beteiligt sind. Dies bewirkt, dass der Untersuchungsgegenstand – die Integrationslösung – und deren Einflussfaktoren nicht direkt beobachtbar sind. Stattdessen müssen die Integrationsprojekte und deren Einflussfaktoren untersucht werden, um Rückschlüsse für die Forschungsfrage ziehen zu können. Die in Abbildung 3.2 dargestellten Rechtecke symbolisieren diese Einflussfaktoren und gelten als unabhängige Variablen. Es handelt sich bei der Unabhängigkeit der Variablen um eine analytische Setzung[16], welche für das Untersuchungsziel gewählt wurde. Selbstverständlich hängen auch diese Variablen wiederum von Einflussfaktoren ab, diese Zusammenhänge werden jedoch für die vorliegende Untersuchung ausgeblendet.

Die *Unternehmensfaktoren* erfassen den Einfluss eines Integrationsdienstleisters als Unternehmen auf die Integrationsprojekte und damit die Integrationslösungen. Als Beispiel kann die Größe des Dienstleisters dienen, welche die Größenordnung der Projekte limitiert, die ein Dienstleister bewältigen kann.

Persönliche Faktoren spiegeln die demografischen Basisfaktoren der Experten wieder. Mithilfe dieser Variable und ihren Unterfaktoren kann überprüft werden, ob der Anspruch Experten zu interviewen erfüllt wurde. Zusätzlich verfügen nur Personen ab einer gewissen Hierarchiestufe eines Integrationsdienstleisters über einen ausreichenden Überblick, um projektübergreifende und strategische Einflüsse benennen zu können.

Das *Integrationsverständnis*, welches in einem Unternehmen als kollektive Auffassung von Integration existiert limitiert ebenfalls die Art der Projekte, für die sich ein Unternehmen interessiert. Liegt der Fokus eher auf einer technischen Auffassung von Integration, so werden Projekte mit einem ausgeprägten Anteil der Geschäftsprozessanalyse vermutlich nur Randerscheinungen sein.

[16] Vgl. (Gläser und Laudel, 2009, S. 81).

Kundenspezifische Faktoren stellen eine Einflussgröße auf die Integrationslösung dar, da kleine und mittelständische Unternehmen andere Integrationsbedarfe haben als große Konzerne. Damit müssen auch andere Integrationslösungen umgesetzt werden. Darüber hinaus kann die Kundenstruktur auch die in einem Integrationsprojekt zu besetzenden Rollen beeinflussen.

Das *Produktportfolio*, mit dem im Wesentlichen die eingesetzten Integrationswerkzeuge und -hilfsmittel gemeint sind, beinhaltet auch die Festlegung auf bestimmte Hersteller dieser Produkte. So kann z. B. die Partizipation an einem Partnernetzwerk eines großen Herstellers die Art der in einem Integrationsprojekt erstellten Lösungen beeinflussen. Lässt das Hilfsmittel des Herstellers die Umsetzung eines bestimmten Architekturparadigmas nicht zu, so wird dieses in den Lösungen des Integrationsdienstleisters kaum anzutreffen sein. Darüber hinaus wird der Dienstleister kein Wissen in diesem Bereich aufbauen.

Die *Wiederholbarkeit* von Integrationslösungen und das wiederholte Auftreten gleichartiger Problemstellungen ist eine Variable, welche von außerordentlicher Bedeutung für die vorliegende Arbeit ist. Die erfolgreiche Anwendung von Wiederverwendungsmechanismen – wie Referenzmodellen und Mustern – und der strukturierte Entwurf nach den Prinzipien der Ingenieurskunst bedingen, dass derartige Wiederholungen in der Praxis existieren. Darüber hinaus sind wiederkehrend zu lösende Probleme auch die Voraussetzung für jedwede Art der Automation. Zusammenfassend lässt sich sagen, dass mögliche Effizienzsteigerungen von der Wiederholbarkeit der Integrationslösungen abhängen.

In der Softwareentwicklung wird seit Längerem eine Diskussion über *Gewichtigkeit und Invasivität* geführt. Diese bezieht sich auf die Art und Weise des Vorgehens, wobei zwischen schwer- und leichtgewichtigen, sowie agilen Methoden unterschieden wird.[17] Darüber hinaus wird das Maß der notwendigen Veränderung der Applikationen (Integrationsobjekten)[18] mit dem Begriff der Invasivität messbar gemacht.

Der *Mainframe* stellt ein besonderes Integrationsobjekt dar und ist – obwohl häufig nicht vermutet – in vielen Unternehmen nach wie vor präsent. Die lange Lebensdauer und die Zentralität dieses Integrationsobjektes und der zugehörigen Anwendungen machen den Mainframe zu einem besonderen Integrationsobjekt, was zu einer eigenen Klasse von Integrationsproblemen und damit Integrationslösungen führt.

Aus den hier vorgestellten Variablen und den im vorangegangenen Abschnitt dargelegten Hypothesen wurde der Interviewleitfaden erstellt, welcher hier nur im Aufbau beschrieben wird. Der vollständige Interviewleitfaden ist im Anhang enthalten.[19] Die Fragen sind bis auf wenige Ausnahmen alle offen gehalten, wobei es sich bei den Ausnahmen um halb offene Fragen handelt. Die Kategorisierung der Fragen[20] erfolgte in Anlehnung an die Variablen des hypothetischen Modells, wobei diese Ordnung keinen Einfluss auf die Auswertung der Fragen hatte. Es lassen sich folgende Teilabschnitte voneinander unterscheiden:

1. Demografische Basisdaten,
2. Verständnis zum Thema Integration,
3. Integrationsprodukte / -werkzeuge / -hilfsmittel,
4. Integrationsmuster und Automatisierung,

[17] Die Gewichtigkeit kennzeichnet den Formalisierungsgrad eines Vorgehens und die Agilität die Dauer der Zyklen und damit die Möglichkeiten auf Veränderungen zu reagieren. Beide Eigenschaften haben zwar Wechselwirkungen miteinander, sind jedoch nicht identisch.
[18] Vgl. (Fincham, 1999).
[19] Vgl. Anhang D
[20] (Gebauer und Stefan, 2011b, S. 11f.)

5. Gewichtigkeit / Invasivität,
6. Integrationsobjekt Mainframe.

Die Teilabschnitte wurden in der Reihenfolge ihrer Aufzählung durchlaufen. Die Teile, bei denen mit den wenigsten Antworten gerechnet wurde, lagen am Schluss. So mussten keine Abschnitte übersprungen werden, wenn z. B. keine Angaben zu dem Integrationsobjekt Mainframe gemacht werden konnten. In dem sich anschließenden Kapitel wird detailliert auf die Gewinnung von Interviewpartnern eingegangen und so die Auswahl der Unternehmen dargelegt.

3.4 Auswahl der Unternehmen

An dieser Stelle wird die Auswahl der Unternehmen geschildert inclusive der Darstellung der schlussendlich interviewten Menge der Unternehmen. Die der Grundgesamtheit zuzurechnenden Unternehmen wurden aus mehreren Unternehmensdatenbanken ausgewählt. Dabei kam primär die Hoppenstedt Datenbank[21] zum Einsatz, welche bei fehlenden Angaben durch Auswertung von D&B Firmenprofile Deutschland[22], bedirect Firmenkurzprofile Deutschland[23] oder nomina IT-Firmenprofile[24], sowie durch manuelle Recherche ergänzt wurde.[25] Dennoch konnten zu einigen Unternehmen keine für eine Größenklassifikation geeigneten Daten gefunden werden. Nach eigener Angabe deckt die Hoppenstedt Datenbank mehr als 85% der deutschen Wirtschaftskraft ab. Es werden alle Unternehmen mit einer Mitarbeiterzahl größer als zehn und/oder einem Umsatz von mehr als einer Million Euro pro anno verzeichnet. Die Klassifikation der Geschäftstätigkeiten und die Aktualität der Ansprechpartner werden durch eine eigene Redaktion geleistet. Aufgrund dieser Eigenschaften war die Hoppenstedt Firmendatenbank das primäre Recherchemedium.

Als Auswahlkriterium wurde das Auftauchen von Systemintegration oder verwandten Begriffen in der Beschreibung der Geschäftstätigkeit des Unternehmens festgelegt. Darüber hinaus wurden bestehende Kontakte zu Unternehmen genutzt, sodass insgesamt 544 Unternehmen als Grundgesamtheit identifiziert und angeschrieben werden konnten. Die Größenverteilung der angeschriebenen Unternehmen ist in Abbildung 3.3 dargestellt.

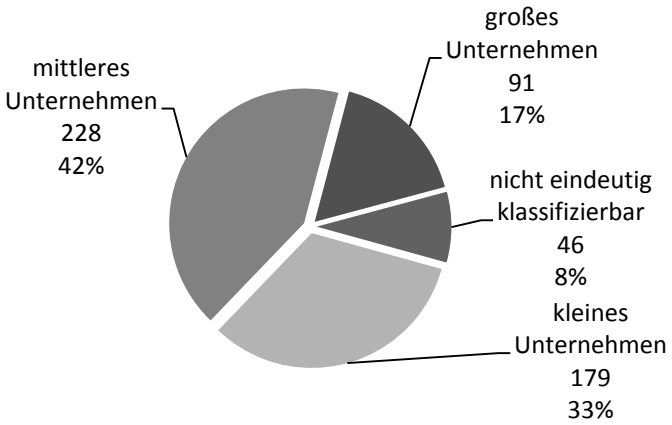


Abbildung 3.3: Größenverteilung der angeschriebenen Unternehmen
Quelle: (Gebauer und Stefan, 2011b, S. 14)

[21] (*Hoppenstedt Firmeninformationen*)
[22] (D&B, 2011)
[23] (bedirect GmbH & Co. KG, 2012)
[24] (Nomina GmbH Informations- und Marketing-Services, 2012)
[25] Diese Datenbanken sind auch über den Dienst Wiso aggregiert zugreifbar Vgl. (GBI-Genios Deutsche Wirtschaftsdatenbank GmbH, 2012).

Die Untersuchung kann nicht als repräsentativ gelten, da sich verfügbare Informationen über den Markt der Systemintegration z. B. auf die 25 führenden IT-Beratungs- und Systemintegrationsunternehmen beschränken.[26] Außerdem existieren keine allgemein anerkannten Kriterien für die Klassifikation eines Unternehmens als Integrationsdienstleister. Von den 544 angeschriebenen Unternehmen haben sich insgesamt 38 Unternehmen (7%) positiv zu einer Teilnahme geäußert. Die Differenz zur abschließend befragten Zahl der Unternehmen resultiert aus Problemen bei der Terminvereinbarung. Abbildung 3.4 stellt die Rückläuferquote grafisch dar. Zusätzlich zu

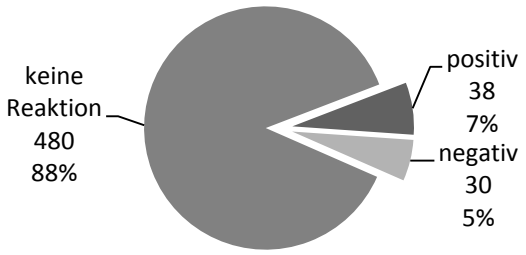


Abbildung 3.4: Rücklaufquote der angeschriebenen Unternehmen
Quelle: (Gebauer und Stefan, 2011b, S. 15)

den positiven Reaktionen, waren erstaunlicher Weise auch negative Reaktionen zu verzeichnen, welche das Ansinnen eines Interviews und den Zeitaufwand als abwegig bezeichneten und dies plastisch darstellten.

3.5 Auswertung der Ergebnisse der Interviews

Die hier vorgestellten Variablen der zugrunde liegenden Untersuchung sind nicht alle in gleichem Maße von Interesse für die vorliegende Arbeit. Deshalb werden die nachfolgenden Ausführungen auf die – für diese Arbeit – wesentlichen Variablen beschränkt. Von primärem Interesse für diese Arbeit sind die Variablen *Wiederholbarkeit* und *Portfoliofaktoren*. Darüber hinaus werden auch die Ergebnisse der Variablen *Unternehmensfaktoren*, *Kundenspezifische Faktoren* und *Integrationsverständnis*, in dem für diese Arbeit notwendigen Maße, herangezogen. Für eine vollständige Darstellung der Ergebnisse der Studie sei auf die Auswertung von Gebauer und Stefan[27] verwiesen.

3.5.1 Unternehmensfaktoren

An dieser Stelle werden Auszüge aus den Unternehmensfaktoren vorgestellt. Diese dienen zum einen der Darlegung, ob mit den befragten Unternehmen die avisierte Zielgruppe der Integrationsdienstleister getroffen wurde und zum anderen kann gezeigt werden, dass die Unternehmen über umfangreiche Erfahrungen in diesem Geschäftsfeld verfügen.

Da die Auswahl der Unternehmen auf der Grundlage einer redaktionellen Klassifikation der Geschäftstätigkeit durch die ausgewählte Unternehmensdatenbank erfolgte, war von Interesse, ob die Unternehmen sich selbst als Integrationsdienstleister sehen. Diese Selbstwahrnehmung kommt in Abbildung 3.5 zum Ausdruck. Nahezu alle befragten Unternehmen stufen sich selbst als Integrationsdienstleister ein. Ein Befragter gab seine Position als konzerninterner Dienstleister an, weshalb nur 29 Nennungen für Integrationsdienstleister verzeichnet werden. Durch die Größe des Unternehmens sind jedoch die Aussagen dieses Befragten als gleichwertig anzusehen und

[26] Vgl. (Lünendonk GmbH, 2012).
[27] Vgl. (Gebauer und Stefan, 2011b; Gebauer und Stefan, 2011a).

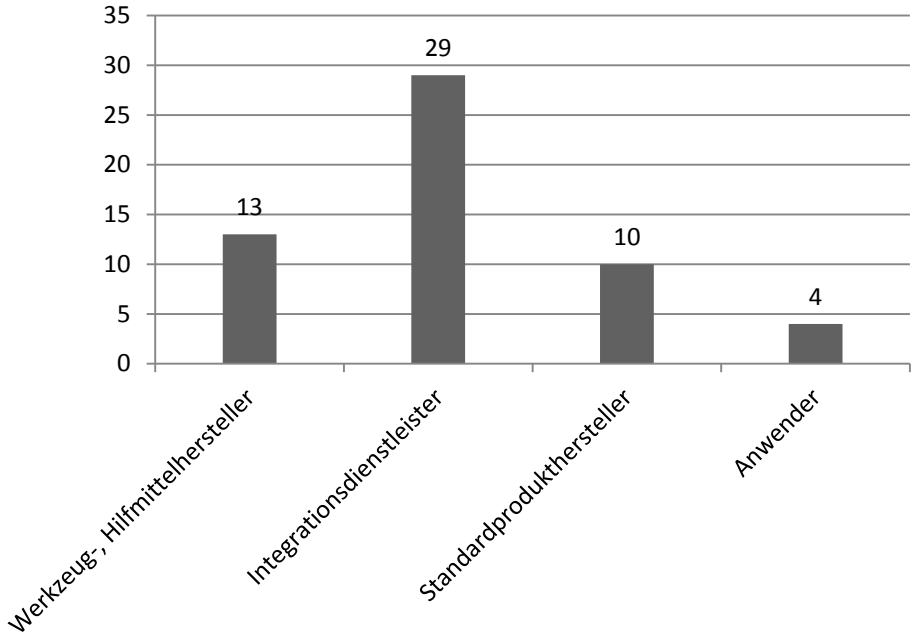


Abbildung 3.5: Selbsteinordnung
Quelle: (Gebauer und Stefan, 2011b, S. 31)

es kann davon gesprochen werden, dass die Zielgruppe der Befragung zu 100% erreicht wurde. Als weiteres Indiz für die Aussagekraft kann die Dauer der Marktaktivität der Unternehmen gewertet werden. Keines der befragten Unternehmen wurde nach 2004 gegründet. Selbst das jüngste befragte Unternehmen war also zum Zeitpunkt der Befragung bereits ca. 6 Jahre am Markt aktiv. Hinzu kommt, dass 22 der befragten Unternehmen vor 1995 gegründet wurden und somit mehr als 15 Jahre am Markt aktiv sind. Die Tabelle 3.1 listet auf, seit wann sich die befragten Unternehmen mit Systemintegration als Geschäftsfeld beschäftigen.

Jahre	**Häufigkeit**
0..10	8
11..25	18
25..30	0
31..40	4
40..	0

Tabelle 3.1: Integration ist ein Thema seit?
Quelle: (Gebauer und Stefan, 2011b, S. 30)

22 Unternehmen haben Systemintegration seit mehr als 10 Jahren als Geschäftsfeld. Kombiniert mit den Gründungsdaten – und damit der generellen Aktivität am Markt – kann konstatiert werden, dass bei allen Unternehmen umfangreiche Erfahrungen im Bereich Systemintegration vorliegen. Zählt man zudem noch hinzu, dass nur 3 Unternehmen in der Gruppe bis 49 Mitarbeiter und 10 Unternehmen der Gruppe 50 bis 249, sowie 12 in der Gruppe ab 250 Mitarbeitern eingeordnet wurden[28], so kann von einer entsprechend großen Anzahl an durchgeführten Projekten ausgegangen werden.

Die Geschäftsmodelle der Unternehmen decken ein breites Spektrum ab, was in der folgenden Aufzählung dargelegt wird:

[28] Vgl. (Gebauer und Stefan, 2011b, S. 19).

- umfassender IT-Beratungsansatz, Integration ist ein erheblicher Teil davon,
- Integration rund um ein eigenes Produkt,
- Integration rund um ein vertriebenes Produkt,
- reines Integrationsberatungshaus,
- Abdeckung einer Branche durch Standardisierung, inclusive der Integrationsprobleme,
- Spezialisierung auf spezifische Systeme und Hersteller z. B. SAP,
- Integration von individuell entwickelten Systemen,
- Software-as-a-Service Integrations-Plattform im B2B Bereich.

Für weitere detaillierte Erklärungen sei auf die vollständige Auswertung der Studie verwiesen. Treten Probleme oder Tätigkeiten, wiederholt auf, so kann ob der Breite der Geschäftsmodelle von einer gewissen Allgemeingültigkeit ausgegangen werden, da die Phänomene nicht auf ein spezifisches Geschäftsmodell zurückzuführen sind.

Neben diesen grundlegenden Eigenschaften der Unternehmen selbst wurden unter dieser Variable Einflussgrößen zusammengefasst, die Bestandteil des Unternehmenswissens sind. Zu diesem Unternehmenswissen sind z. B. Vorgehensmodelle einschließlich der enthaltenen Rollen zu zählen, sofern sie aktiv vom Unternehmen eingesetzt und die Mitarbeiter aktiv in der Anwendung geschult werden. Die Mehrzahl der Unternehmen (21) verfügen über ein derartiges Modell. 9 Unternehmen hingegen nicht. Dies lässt den Schluss zu, dass man am Markt immer noch ohne ein strukturiertes Vorgehensmodell aktiv sein kann, sich in der Regel aber die Ausarbeitung eines solchen empfiehlt. Als Synthese aller Nennungen können die in Abbildung 3.6 dargestellten Phasen verstanden werden. Jede der Phasen wurde im Rahmen der Studie durch die Interviewteilnehmer ausführlich

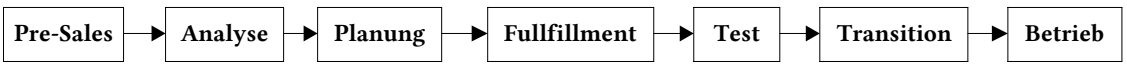


Abbildung 3.6: Übersicht: Phasen eines Integrationsprojektes
Quelle: (Gebauer und Stefan, 2011b, S. 37)

mit Teilphasen, beteiligten Rollen und Arbeitsergebnissen detailliert. Die Ergebnisse werden an dieser Stelle nicht vorgestellt, da sie in aller Ausführlichkeit in das Dienstleistungsmodell für Integrationsdienstleistungen in Abschnitt 4.2 einfließen. Für die Art des Vorgehens kann keine eindeutige Aussage getroffen werden. Lediglich in der Fullfillment-Phase – also der Erstellung der Integrationslösung – überwiegen iterative Vorgehensweisen, wohingegen für die Analysephase keine Aussage getroffen werden kann.

3.5.2 Persönliche Faktoren

Die persönlichen Faktoren besitzen neben der Tatsache, dass sie die Erstellung der Integrationslösung beeinflussen auch die Eigenschaft, dass sie eine Überprüfung der Experteneigenschaft der befragten Personen ermöglichen. Mit dieser Intention werden hier Ausschnitte aus der Studie vorgestellt, mit denen es möglich ist nachzuweisen, dass es sich bei den befragten Personen ausschließlich um Experten handelt und dass diese auf ein umfangreiches Erfahrungswissen im Bereich der Systemintegration verweisen können.

Die strategische Ausrichtung und Fragen der Unternehmensentwicklung spielen nicht auf allen Ebenen eines Unternehmen die gleich Rolle. Im Allgemeinen sind erst Mitarbeiter ab einer höheren Hierarchiestufe mit diesen Themen befasst. Das Hauptgebiet dieser Arbeit, wiederkeh-

rende Probleme mit Mustern zu adressieren, kann in diesen strategischen Bereich eingeordnet werden, da der Aufbau eines derartigen Wissens zunächst eine Investition darstellt, insbesondere wenn das Wissen systematisch weiter gegeben werden soll. Die Abbildung 3.7 verdeutlicht die Positionen der befragten Experten. Bei den wenigsten befragten Experten handelt es sich um

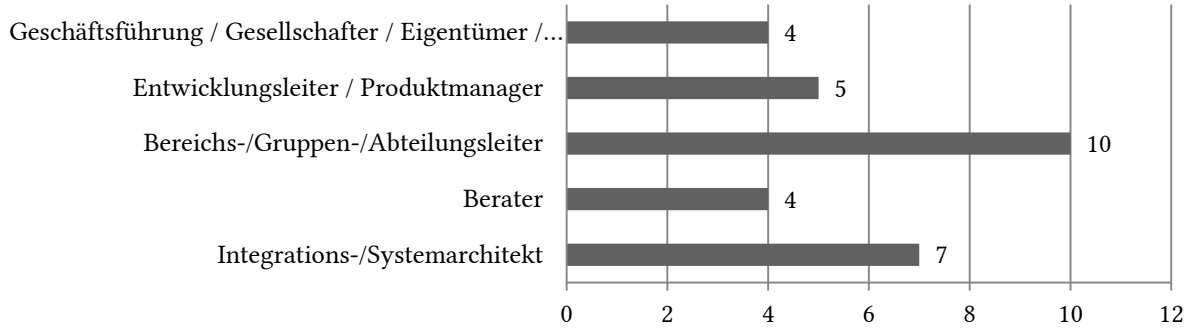


Abbildung 3.7: Verteilung der Experten nach ihrer Position im Unternehmen
Quelle: (Gebauer und Stefan, 2011b, S. 59)

Projektmitarbeiter. Es dominieren hierarchisch höher angeordnete Positionen wie Architekten, Leiter einer Abteilung bzw. eines Bereiches und Entwicklungsleiter. Darüber hinaus zählen 4 der Experten zur unmittelbaren Führung des Unternehmens oder sind sogar Teilhaber. Aus diesen Ergebnissen lässt sich schließen, dass die befragten Experten aufgrund ihrer Position zu strategischen Fragen und Fragen des strukturierten Entwurfes, sowie der Weitergabe von Unternehmenswissen umfassend Auskunft geben konnten. Ein weiteres Indiz für diese Auskunftsfähigkeit ist die Unternehmenszugehörigkeit der befragten Experten (Abbildung 3.8). Da auch die vorrangig adressierten Leitungspositionen gelegentlich mit externen Kandidaten besetzt werden, muss dieses Kriterium mit in Betracht gezogen werden.

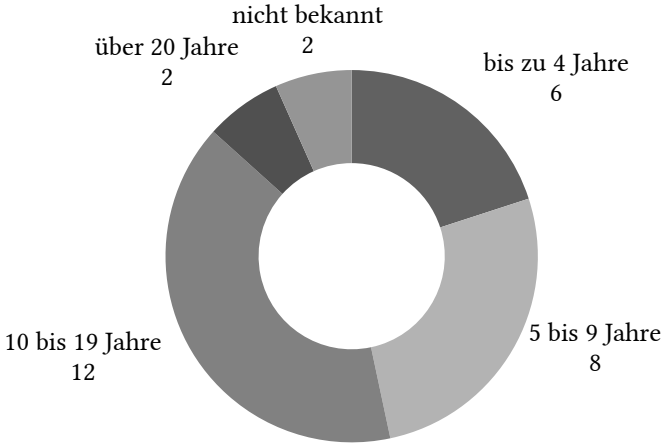


Abbildung 3.8: Verteilung der Experten nach Unternehmenszugehörigkeit
Quelle: (Gebauer und Stefan, 2011b, S.)

Etwas mehr als 73% der Befragten sind länger als 4 Jahre im aktuellen Unternehmen und mehr als 45% länger als 9 Jahre. Auf Basis dieser Angaben kann davon ausgegangen werden, dass es sich bei fast allen Befragten um langjährige Mitarbeiter handelt und diese in die Prozesse ihrer Unternehmen bestens eingebunden sind. Sofern es Instrumente gibt, mit wiederkehrenden Problemen und Herausforderungen strukturiert umzugehen, sind diese den Befragten bekannt.

Abgerundet wird das Bild der befragten Experten durch die Selbsteinschätzung der Erfahrung und Tätigkeit im Bereich Systemintegration, welche den gesamten Werdegang des Experten einschließt (Abbildung 3.9). Mit diesem Aspekt der persönlichen Faktoren kann der Expertenstatus der Interviewteilnehmer manifestiert werden. Mehr als 66% der Interviewten haben nach eigenen

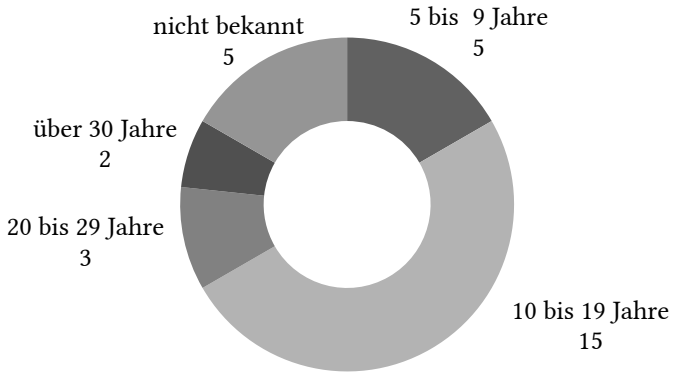


Abbildung 3.9: Erfahrungen der Experten im Bereich Systemintegration
Quelle: (Gebauer und Stefan, 2011b, S. 63)

Aussagen 10 oder mehr Jahre Erfahrung im Bereich Systemintegration. Dabei waren sie zumeist in unterschiedlichen Rollen tätig und haben in den Anfangsjahren oft auch als Entwickler an Integrationsprojekten mitgewirkt. Die dabei durchgeführte Anzahl von Integrationsprojekten reicht bei einigen Befragten bis an den dreistelligen Bereich heran.[29] Darüber hinaus wurden diese Zahlen auch gesondert für Quereinsteiger ohne einschlägige Ausbildung im IT-Bereich ausgewertet. Die überwiegende Mehrheit der Quereinsteiger gehört zu den oben angeführten 66%.[30]

Zusammenfassend können alle befragten Personen als Experten im Bereich Systemintegration und als versierte Kenner der eigenen Unternehmen bezeichnet werden, da sie das Systemintegrationsgeschäft von der Pike auf gelernt haben und bereits lange Jahre in den aktuellen Unternehmen tätig sind. Die Auswahl der Befragten kann als erfolgreich gewertet werden.

3.5.3 Faktoren der Wiederholbarkeit

Von außerordentlicher Bedeutung für diese Arbeit sind die Faktoren der Wiederholbarkeit. Diese liefern zum einen die Ansatzpunkte für Mechanismen der Wiederverwendung und zum anderen bereits im Einsatz befindliche Konzepte, mit denen wiederkehrende Herausforderungen und Tätigkeiten adressiert werden.

In der Diskussion mit Fachexperten ist zunächst teilweise strittig, ob es bei der Integration überhaupt zu wiederkehrenden Tätigkeiten kommt. Da dies jedoch eine Grundvoraussetzung für die Wiederverwendung und insbesondere die strukturierte Wiederverwendung ist, werden zunächst die Äußerungen für oder gegen das Auftreten von wiederkehrenden Tätigkeiten dargelegt. 8 Experten haben Bedenken gegen die Wiederverwendung und geben an, dass keine wiederkehrenden Tätigkeiten existieren bzw. diese nicht ausgenutzt werden können. Als Gründe werden verschiedene Positionen angeführt. Die Wiederverwendung auf Code-Ebene scheitert daran, dass dieser in das Eigentum des Kunden übergeht und so nicht nutzbar ist. Auch sind die Kosten dieser Bemühungen zu groß. Weiterhin wird angezweifelt, dass Mitarbeiter ein Interesse daran haben, da sie sich unter Umständen damit selbst gefährden, weil vormals manuelle Tätigkeiten nun automatisiert

[29] Vgl. (Gebauer und Stefan, 2011b, S. 63).
[30] Vgl. (ebenda, S. 64).

werden könnten. Am häufigsten wird jedoch angeführt, dass jede Integrationslösung individuell ist. Dies wird vor allem mit dem Zwang zur Differenzierung von Wettbewerbern begründet, die dadurch entstehende Individualität wirkt bis hinunter zu den Integrationslösungen.

Im Vergleich zu den Äußerungen einzelner wiederkehrender Tätigkeiten haben sich lediglich 4 Experten zu dieser Frage auf einem prinzipiellen Niveau geäußert. Eine Ursache besteht in Compliance-Regeln und regulatorischen Vorschriften. Diese bewirken wiederkehrende Integrationsphänomene. Der Nutzen lässt sich auf Dienstleisterseite leicht nachweisen in dem man mit der Zahl der Instanzen des Problems rechnet. Insbesondere nicht als Kerngeschäft geltende Prozesse der Kunden lassen sich mit schematischen Lösungen adressieren, da hier kein Zwang zur Differenzierung besteht.

In den vorangegangenen Ausführungen wurden die Meinungen der Experten zum Auftreten von wiederkehrenden Tätigkeiten und Herausforderungen dargelegt. Diese Überlegungen werden in Abbildung 3.10 beiseitegelassen und es wird analysiert, welche Tätigkeiten wiederholt auftreten und sich damit als Ansatzpunkt für Wiederverwendung und Automatisierung anbieten. Mit Abstand

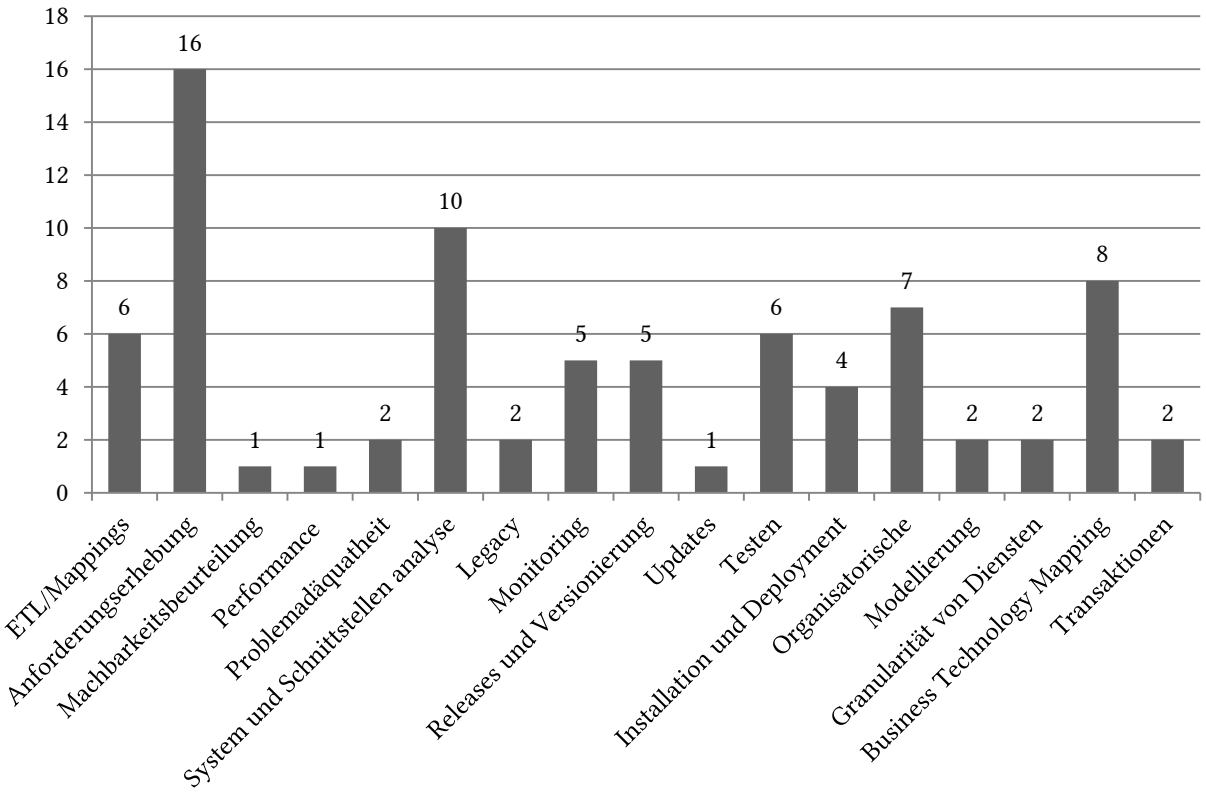


Abbildung 3.10: Wiederkehrende Aufgaben und Probleme – Verteilung der Äußerungen
Quelle: (Gebauer und Stefan, 2011b, S. 114)

am häufigsten tritt die *Anforderungserhebung* auf. Dieses Problem entsteht durch die Vielzahl der Beteiligten an einer Integrationslösung und vor allem deren unterschiedliche Erwartungen und die jeweils eigene Fachsprache. Nicht selten gestaltet sich die Kommunikation schwierig, weil die Parteien einander nicht verstehen. Besonders die strukturierte Erhebung der Anforderungen bereitet Probleme. Die *System- und Schnittstellenanalyse* ist die zweithäufigste wiederkehrende Tätigkeit. Probleme in diesem Bereich entstehen häufig dadurch, dass IT keine Kernkompetenz der Kunden ist. Insbesondere durch eine lange Betriebsdauer und schlechte Dokumentation sind bestimmte Details, welche für eine Integration relevant sind, nicht bekannt und müssen aufwendig

in Erfahrung gebracht werden. Dabei ist neben Details zu Schnittstellen auch das Verhalten der Systeme in speziellen Fällen in Erfahrung zu bringen. Die dritt häufigste Tätigkeit – und damit die Letzte die hier detailliert vorgestellt wird – ist das *Business-Technology-Mapping*. Durch die immer weiter wachsende Vernetzung steigt die Komplexität der Landschaften. Dies führt dazu, dass die Analyse und Beurteilung immer schwieriger wird. Technische und fachliche Modelle stimmen häufiger nicht überein, was die Abbildung der gewünschten betriebswirtschaftlichen Prozesse auf IT-Systeme als digitale Aufgabenträger schwierig macht. Weitere wiederkehrende Tätigkeiten können der Abbildung 3.10 entnommen werden, für deren ausführliche Erläuterung sei auf die Studie[31] verwiesen.

Mit diesen Aussagen ist aber zweifelsfrei nachgewiesen, dass es bei der Integration zu wiederkehrenden Tätigkeiten bzw. Herausforderungen auf unterschiedlichen fachlichen Ebenen kommt. Diese bieten Ansatzpunkte für methodisches Vorgehen und wiederkehrend einsetzbare Lösungselemente.

Existieren in einem Unternehmen wiederkehrende Probleme respektive Tätigkeiten und werden diese immer wieder mit speziellen Lösungsbausteinen adressiert, so ist es notwendig, dieses Wissen zu konservieren und an die Mitarbeiter weiter zu geben. Im Folgenden wird zunächst die Weitergabe – das Wie – und danach die eingesetzten Konzepte – das Was – betrachtet.

Die Abbildung 3.11 gibt die Häufigkeiten der genannten Mechanismen wieder, welche in den Unternehmen eingesetzt werden, um das vorhandene Wissen unter den Mitarbeitern zu verteilen und zu verankern. Dabei spielen häufig Überlegungen eine Rolle, die einkalkulieren, dass Wissensträger das Unternehmen auch verlassen können und damit möglicherweise Beeinträchtigungen einher gehen könnten. Neben der *individuell(en)* Verantwortung des Einzelnen lassen sich *Best-*

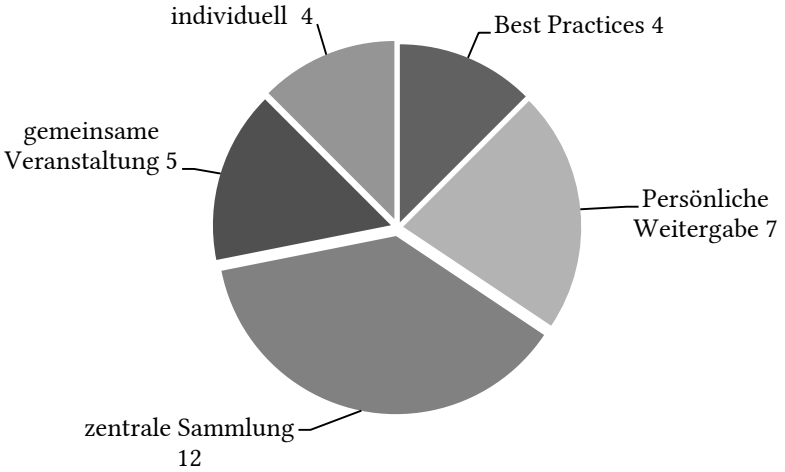


Abbildung 3.11: Häufigkeiten und Arten der Wissensweitergabe
Quelle: (Gebauer und Stefan, 2011b, S. 125)

Practices, die *persönliche Weitergabe* durch die Wissensträger, *gemeinsame Veranstaltungen* und die *zentrale Sammlung* unterscheiden. Dabei bestimmen *Best Practices* mehr die Art der Dokumentation und die anderen Punkte eher die Art der Verbreitung. Auffällig ist, dass es sich außer bei der gemeinsamen Veranstaltung nicht um systematische Verfahren handelt, sondern diese Informationen aktiv eingeholt werden müssen. Lediglich bei einer gemeinsamen Veranstaltung bei der Anwesenheitspflicht besteht kann sichergestellt werden, dass auch Mitarbeiter erreicht werden, die sich nicht um eine Nutzung dieses Lösungswissens bemühen.

[31] Vgl. (Gebauer und Stefan, 2011b, S. 113ff.).

Eine der wesentlichen Ursachen für Sprachprobleme zwischen Fach- und IT-Experten sind Fach- oder Domänenspezifika, welche zur Ausprägung einer eigenen Sprache führen. In einem erweiterten Sinn können diese Domänenspezifika auch als Muster auf der Seite der Fachlichkeit interpretiert werden. Die prinzipielle Existenz wurde von 18 Experten mit ja beantwortet und lediglich von 6 Experten verneint. Ein wesentlicher Faktor für das Erkennen von Domänenspezifika ist Zeit. Es kann sich um die lange Arbeit für einen Kunden oder um Tätigkeiten in einer Branche handeln. Die Wirkung ist, dass sich Fach- und IT-Experten beginnen auf dieser Abstraktionsebene auszutauschen. Da es sich um implizites Wissen der Experten handelt, konstatieren diese als Wirkung, dass dieses Wissen zwar vorhanden ist und in die Projekte einfließt, aber nicht bewusst genutzt wird. Gesetzlich stark regulierte Bereiche fördern das Entstehen derartigen Wissens besonders. Wurde zeitig auf die Kernkompetenzen geachtet, so hat der Kunde früh seine Domänenkonzepte formuliert. Ist das nicht der Fall, halten Kunden an individuellen Begebenheiten fest in dem Glauben sich dadurch zu differenzieren. Dem Integrationsdienstleister beleibt es dann verwehrt an diesen Stellen erprobte Lösungen einzusetzen, die Folge sind von Grund auf neu erstellte Einzellösungen.

Die von den Experten kommunizierten Domänenspezifika mit einem Bezug zur Integration lassen sich in verschiedene Klassen einteilen. Es existieren Systemklassen, die spezifische Aufgaben wahrnehmen. Diese müssen nicht branchenspezifisch sein. In Bereichen in denen wenig Möglichkeiten zur Differenzierung bestehen, treten auch branchenübergreifend einheitliche Systemklassen auf. Ein Beispiel ist die Klasse der ERP-Systeme, welche in allen Branchen einen ähnlichen Funktionsumfang und ähnliche Aufgaben hat. Die Differenzierung gegenüber dem Wettbewerb findet z. B. bei produzierenden Unternehmen in anderen Bereichen als in der Buchhaltung statt. Neben den Systemklassen existieren spezielle Geschäftsobjekte welche, wenn durch ein IT-System realisiert – ihre Abbildung durch Datenobjekte innerhalb der Systeme finden. Die Tabelle 3.2 stellt die im Rahmen der Studie erwähnten Objekte differenziert nach Branchen dar.

Branche	**Geschäftsobjekte**
branchenübergreifend	Bestellung, Produkt, Kunde
Handel	Lieferschein, Wareneingangsmeldung, Gutschrift, Zahlungsziel, Rechnung, Rechnungsnummer, Datum, Betrag, Umsatzsteuerkennung, Kunde, Lieferant,
Versicherungen	Partner, Police, Werkstatt, Kostenvoranschlag, Rechnung
Banken	Lastschrift, Gutschrift, Zahlungsanweisung, Überweisung, Transaktion, Kunde, Wertpapier
Energie-, Versorgungswirtschaft	Vertrag, Kunde, Debitor, Kreditor, Rechnungseinheit, Lieferant, Lokation, Tarif, Preise
öffentlicher Dienst, Meldewesen	Kunde
produzierendes Gewerbe	Besteller, Lieferant, Zähler, Lokation

Tabelle 3.2: Geschäftsobjekte nach Branchen
Quelle: (Gebauer und Stefan, 2011b, S. 132)

So kann ein Eindruck davon gewonnen werden, dass einige Objekte aus Dienstleistersicht einen hohen Wiederholungsgrad besitzen. Ein Objekt *Kunde* ist in seinen Grundkonzepten in allen Branchen ähnlich. Dennoch ist der Dienstleister permanent gefordert, diese Objekte zu kennen und technische Fragen in der Sprache der Fachexperten zu stellen und dabei die bekannten Objektbezeichner zu verwenden. Neben den Geschäftsobjekten tritt eine weitere Gruppe von Domänenspezifika auf. Es handelt sich um Prozesse oder ganze Prozesslandkarten. Die Kenntnis dieser, ist von besonderem Interesse für einen Integrationsdienstleister, da sie die Geschäftsobjekte mit Tätigkeiten verknüpfen und je nach Abbildung auf die IT-Systeme zu unterschiedlichen Interaktionen zwischen den Integrationsobjekten – den IT-Systemen – führen. Die Tabelle 3.3 stellt die in der Studie beschriebenen domänenspezifischen Prozesse, wiederum differenziert nach

Branchen, dar. Die domänenspezifischen Prozesse machen deutlich, dass hier eine viel größere

Branche	**domänspezifische Prozesse**
Handel	Finanzprozess, Lieferprozess
Versicherungen	Schadenmeldung, Schadensbearbeitung
Banken	Zahlungsverkehr, Clearingverfahren, Datenträgeraustauschverfahren, Risikomanagement
Energie-, Versorgungswirtschaft	Ein- und Ausbau von Zählern
Telekommunikation	Telefonat
produzierendes Gewerbe	Business-Process-Master-List

Tabelle 3.3: Prozesse nach Branchen
Quelle: (Gebauer und Stefan, 2011b, S. 133)

Differenzierung als bei den Geschäftsobjekten vorliegt. Es gibt auf den ersten Blick eine viel größere Zahl an unterschiedlichen Prozessen, welche nur selten branchenübergreifende Gemeinsamkeiten aufweisen. Eine weitere – in ihrem Reifegrad über die beiden vorgenannten hinausgehende – Art von Domänenspezifika sind Blueprints bzw. Referenzmodelle. Diese fassen Systemklassen, Objekte und Prozesse zusammen und ergänzen diese um Konfigurationsinformationen.

Mit 26 haben sich – im Vergleich zu den Pro- und Kontra-Aussagen der Wiederverwendung – relativ viele Experten zu den Wirkungen domänenspezifischer Eigenheiten positioniert. Eine wichtige Wirkung ist die Veränderung der Denk- und Kommunikationsweise, welche sich insbesondere auf die Dienstleister auswirkt. Dadurch entsteht eine neue Art der Kundenorientierung, da der Integrationsdienstleister sich anpasst und nicht der Kunde den Eindruck behält eine neue, technische – ihm fremde – Sprache erlernen zu müssen. Der wesentlichste Effekt ist jedoch der Rationalisierungseffekt für den Integrationsdienstleister. Die wiederkehrenden Anwendungsfälle in denen wiederkehrende Geschäftsobjekte auftreten, helfen dabei die Lücken in den Interaktionen zwischen den Systemen zu erkennen. Der Dienstleister beginnt auf diesem Abstraktionsniveau zu denken, und intuitiv und automatisch mit diesen Abstraktionen zu hantieren. In der extremsten Ausprägung führen diese Abstraktionen zu der Ausprägung von Branchenstandards, wie im Fall des Gesamtverbandes der Deutschen Versicherungswirtschaft (GDV). Dies kommt nicht nur dem Dienstleister zugute, sondern jedem einzelnen Kunden, da sich – durch die Verbesserung der Kommunikation – die Time-to-market der individuellen Integrationslösungen verringert.

In Abschnitt 2.3 wurde das Konzept der Muster eingeführt. Die hier vorgestellten Ergebnisse der Studie geben wieder, inwieweit dieses Konzept bei Integrationsdienstleistern im Einsatz ist und Anwendung findet. In einem ersten Schritt wurde danach gefragt, welche Muster bekannt sind. Die Abbildung 3.12 gibt Aufschluss darüber, welches Muster mit welcher Häufigkeit genannt wurde. Bei dieser Darstellung wurde bewusst darauf verzichtet, die Äußerungen in Klassen unterschiedlicher Abstraktionsniveaus zu gruppieren, da so ein realistischerer Eindruck der Praxis der Integration und der Adoption wissenschaftlich wohl diskutierter Konzepte bestehen bleibt. Die Abstraktionsniveaus der genannten Beispiele differieren teils erheblich voneinander. An dieser Stelle sollen zwei Beispiele dies verdeutlichen. Das am häufigsten genannte Muster ist die *Stern-Topologie* auch Hub-and-Spoke genannt. Streng genommen handelt es sich dabei um ein Architekturmuster, und zwar um eines der Klasse Kommunikationsmuster. Es beschreibt, wie die beteiligten Komponenten durch Kommunikationsverbindungen möglichst effizient zu verbinden sind. Vergleicht man dieses Muster mit dem zweit häufigsten (*Punkt-zu-Punkt*) so bleibt man in derselben Klasse von Mustern. Zieht man aber den Vergleich zum dritt häufigsten (*Messaging*) oder zu den viert häufigsten Mustern, so muss man feststellen, dass das Abstraktionsniveau ein vollkommen anderes ist. Unter diesen finden sich vollständige Mustersysteme (EIP Hohpe[32])

[32]EIP Hohpe bezieht sich auf den in (Hohpe und Woolf, 2003) enthaltenen Musterkatalog.

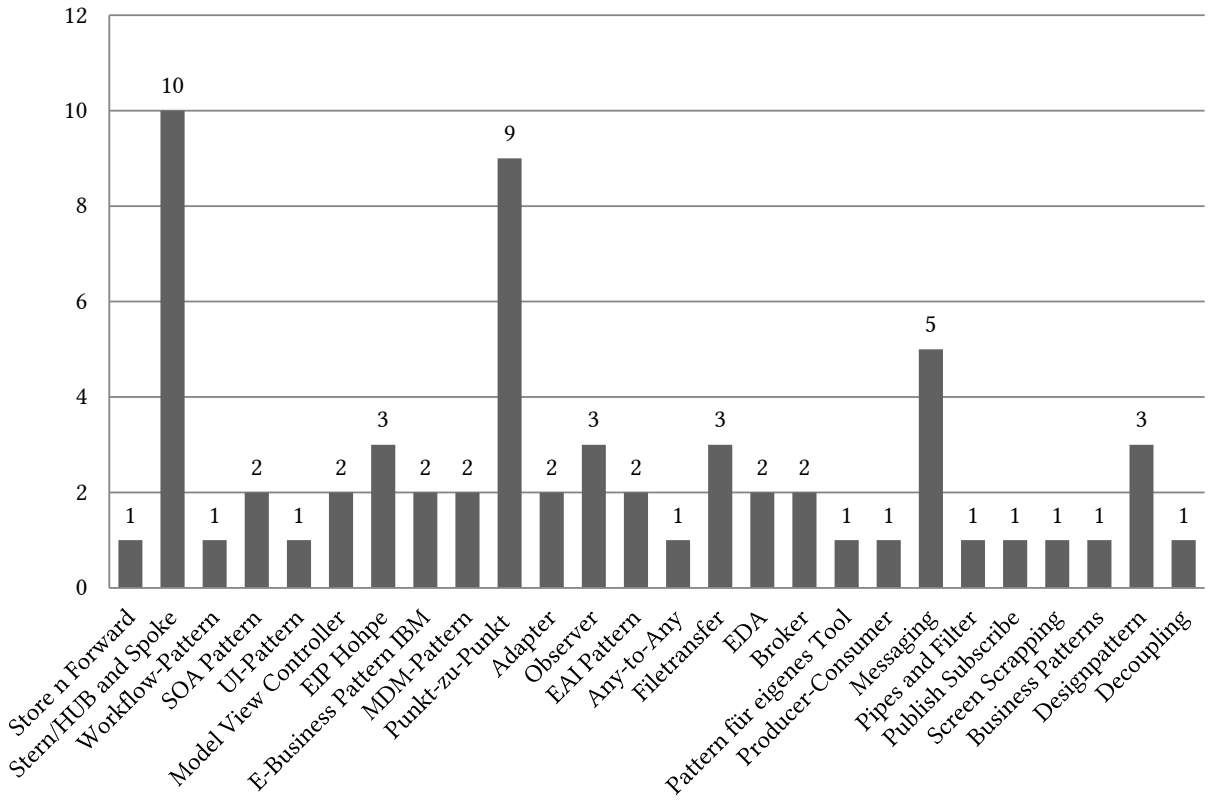


Abbildung 3.12: Häufigkeiten der Äußerungen zu bekannten Mustern
Quelle: (Gebauer und Stefan, 2011b, S. 140)

oder Entwurfsmuster, welche auf den inneren Aufbau einzelner Softwarekomponenten abzielen. Als Fazit lässt sich formulieren, dass das Wissen über Muster für die Integration sehr zerklüftet ist und keine Systematiken bekannt sind.

Dies spiegelt sich auch massiv in den Äußerungen zum Einsatz von Mustern wieder. Die vorangegangene Darstellung erfasste ja nur auf die bloße Kenntnis. Von 30 Experten haben sich nur sieben zu dieser Thematik geäußert, was überrascht. Lediglich einer der Experten benannte eine Verwendung in der Analysephase und ein weiterer die implizite – also durch ein Werkzeug gewährleistete – Anwendung. Die restlichen Äußerungen charakterisieren die Verwendungen als unbewusst – bedingt durch die Kenntnis im Rahmen einer fachbezogenen Ausbildung – und in der individuellen Gestaltungsfreiheit des einzelnen Mitarbeiters liegend.

Eine Möglichkeit Muster einzusetzen ist es, Hilfsmittel zu wählen, welche Muster implementieren und so die Verwendung implizit erzwingen[33] oder durch Konfiguration die Umsetzung von Mustern ermöglichen. Damit dies bewusst und planvoll geschehen kann, muss man sich darüber im Klaren sein, welches Muster das betreffende Hilfsmittel ermöglicht bzw. erzwingt. Es liegen in der Studie nur Aussagen zu wenigen Hilfsmitteln großer Hersteller und einem Open-Source-Hilfsmittel vor. Im Rahmen der Portfoliofaktoren wurde jedoch eine weitaus größere Anzahl an Hilfsmitteln benannt. Daraus kann geschlossen werden, dass die Integrationsdienstleister keine tieferen Kenntnisse über die implizit mit bestimmten Hilfsmitteln verwendeten Architekturmuster haben.

[33] Jede Middleware, welche Funktionen zentralisiert, erzwingt eine Hub-and-Spoke Topologie.

Im Rahmen der Studie wurden die Experten zu einem Konzept befragt, mit dem eine Brücke zwischen der IT-technischen und der Fachseite geschlagen werden könnte. Dieses Konzept zielt darauf ab, bestehende technische Muster um Domänenspezifika anzureichern und so zum einen eine sprachliche Brücke zu bauen und zum anderen Effizienzsteigerungen zu realisieren. 9 Experten befürworten diesen Ansatz, wobei angemerkt wurde, dass man den Aufwand dafür nur betreiben würde, damit man beim nächsten Projekt schneller ist.

Zum Schluss wurde von einem Experten geäußert, dass ein etablierter Kanon an Integrationsmustern in Analogie zu den Werken aus dem Bereich der objektorientierten Programmierung fehlt und vermisst wird.

3.5.4 Portfoliofaktoren

Hier werden die Portfoliofaktoren vorgestellt, bei denen jedoch nicht auf einzelne Werkzeuge oder Hilfsmittel eingegangen wird – dafür sei auf die Publikation der vollständigen Ergebnisse der Studie verwiesen[34] – sondern auf Klassen von Werkzeugen bzw. Hilfsmitteln. Für diese Arbeit ist von Bedeutung, welche Schritte eines Integrationsprojektes bereits zufriedenstellend unterstützt werden, sowie herauszuarbeiten, an welchen Stellen eine ausreichende Unterstützung fehlt. Die Abbildung 3.13 stellt die Häufigkeiten zu einzelnen Hilfsmitteln dar. Diese Angaben

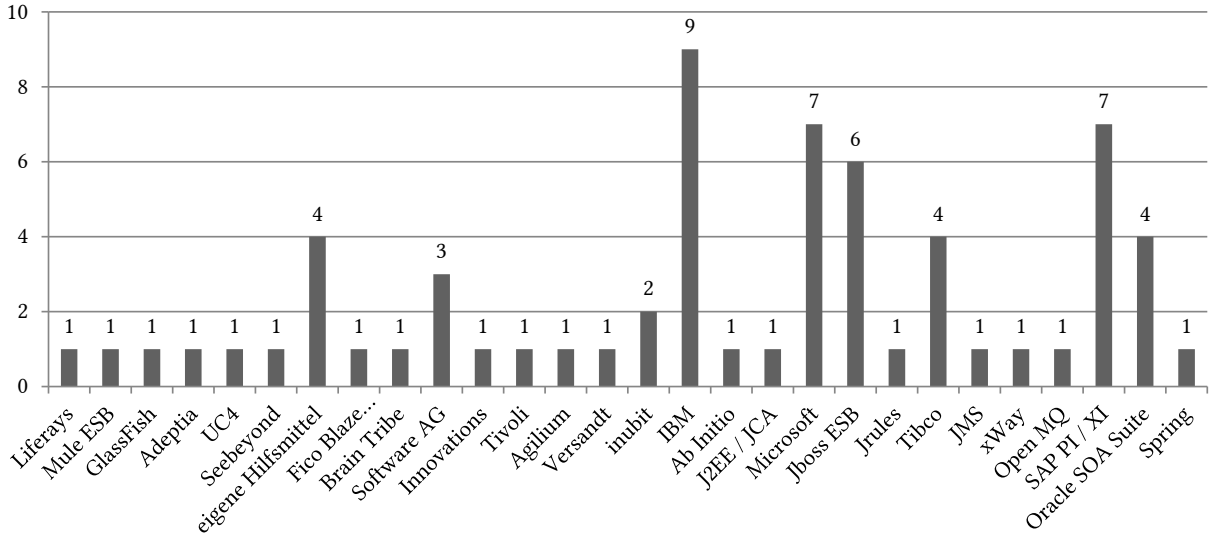


Abbildung 3.13: Häufigkeiten der Äußerungen zu Hilfsmitteln
Quelle:(Gebauer und Stefan, 2011b, S. 146)

ermöglichen es die Aussagen zu den Mustern und Hilfsmitteln welche den Einsatz bestimmter Muster erzwingen in Zusammenhang zu setzen. An dieser Stelle soll nicht weiter auf die Details der einzelnen Hilfsmittel eingegangen werden. Die Studie enthält keine Klassifikation der einzelnen Hilfsmittel in verschiedene Typen. Da es sich ohnehin bei den meisten Nennungen um Hybride handelt, welche sowohl Werkzeug- als auch Hilfsmitteleigenschaften vereinen.

Für einen Dienstleister von großem Interesse ist die Unterstützung des eingesetzten Personals mit Werkzeugen, welche die Bearbeitung der Aufgaben erleichtern. Lassen sich hier Effizienzgewinne erzielen, so resultiert daraus eine direkte Produktivitätssteigerung. Obwohl nach Auswertung der Ergebnisse der Interviews durchaus Informationen zu einzelnen Werkzeugen vorliegen, soll an dieser Stelle nur auf die Typen von Werkzeugen eingegangen werden. Die Abbildung 3.14 zeigt

[34] Vgl. (Gebauer und Stefan, 2011b).

die Häufigkeiten der einzelnen Werkzeugtypen. Mit Abstand am häufigsten sind Werkzeuge für

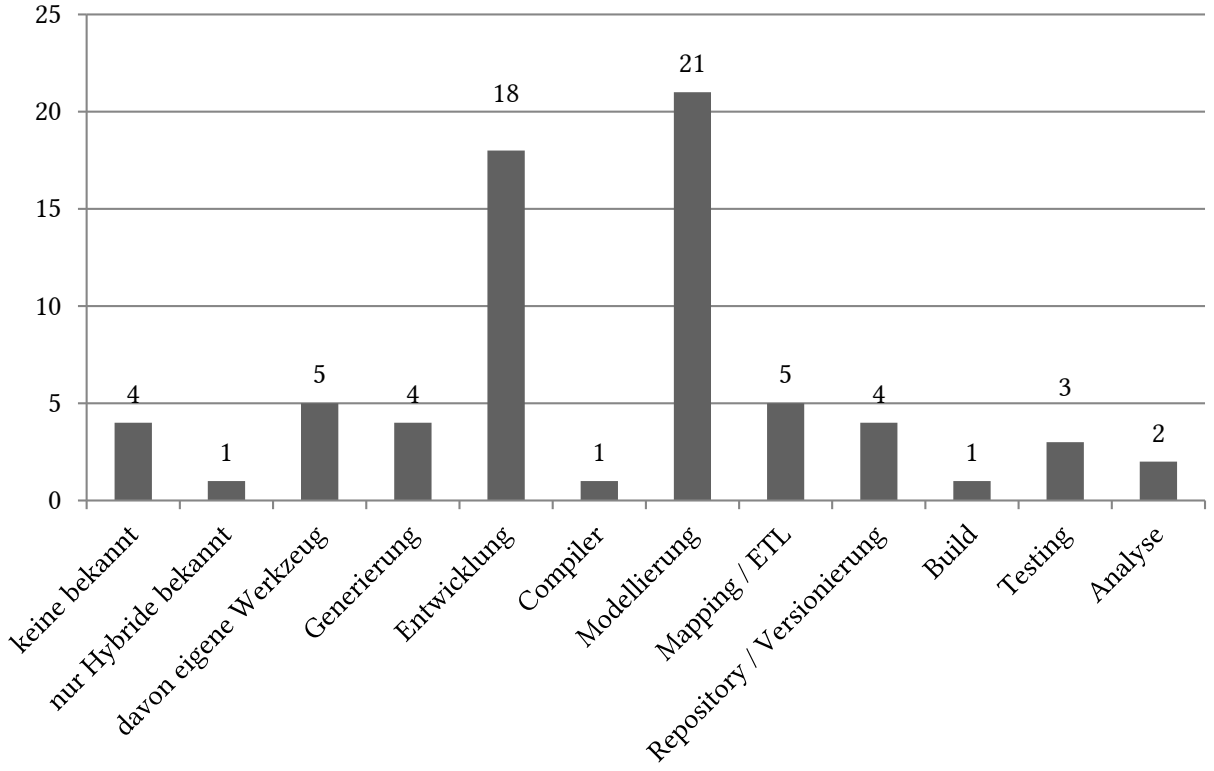


Abbildung 3.14: Häufigkeiten der Äußerungen zu Werkzeugtypen
Quelle:(Gebauer und Stefan, 2011b, S. 150)

die *Modellierung* und *Entwicklung* im Einsatz. Dabei bleibt offen, ob es sich dabei um spezielle Werkzeuge zur Integration handelt oder diese aus der Softwareentwicklung oder Geschäftsprozessmodellierung übernommen wurden und nur im Kontext der Integration eingesetzt werden. Unklar bleibt auch, inwieweit diese von den vorgenannten Gebieten abweichende, speziell auf die Integration zugeschnittene Funktionalität aufweisen. Eine Verbindung dieser Welten erscheint aus der Sicht der Integration geboten, da die Integration Ansätze aus allen Gebieten benötigt. Danach folgen – mit absteigender Häufigkeit der Nennung – *Mapping/ETL*, *Repository/Versionierung*, *Testing*[35] und die *Analyse*. Die ersten drei Balken in Abbildung 3.14 stellen keine Werkzeugtypen dar. Vier Experten sind schlicht keine Integrationswerkzeuge bekannt, wohingegen ein Experte angibt, nur hybride Werkzeuge zu kennen, die auch Hilfsmitteleigenschaften haben. Von den genannten Werkzeugtypen beziehen sich die Äußerungen der Experten in fünf Fällen auf eigene Werkzeuge.

Nachdem nun die Typen der Werkzeuge welche im Einsatz sind, dargelegt wurden folgt nun die Darstellung der Ergebnisse, welche sich mit fehlender Funktionalität bei diesen Werkzeugen auseinandersetzen. Zu erwarten wäre gewesen, dass die Werkzeugtypen – für die Analyse, das Testen und die Unterstützung der Build-Prozesse – welche am seltensten genannt wurden, zu den am häufigsten vermissten Funktionalitäten zählen. Am häufigsten wurde jedoch die fehlende Werkzeugunterstützung für die Abbildung der fachlichen Anforderungen auf die technische Lösung (*Business Technology Gap*) angeführt. Dies kann als Aufgabe der Analyse und des Entwurfes interpretiert werden, wodurch sich dann eine Übereinstimmung zu dem Bild der eingesetzten Werkzeugtypen ergibt. Interessanterweise wurde mit am zweit Häufigsten genannt, dass keinerlei

[35] Mit diesem Anglizismus ist das Testen gemeint.

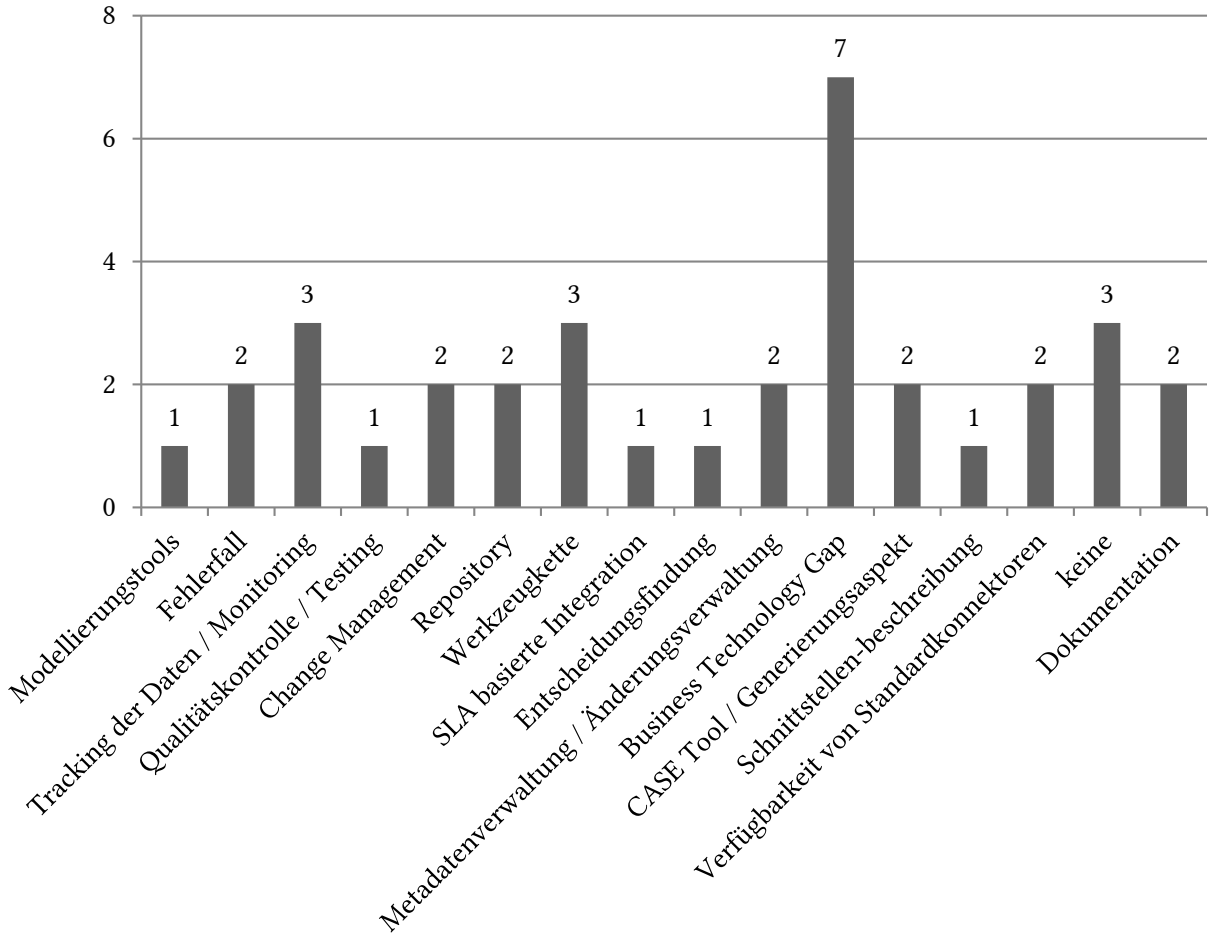


Abbildung 3.15: Genannte, fehlende Funktionalität und Häufigkeiten
Quelle:(Gebauer und Stefan, 2011b, S. 171)

Funktionalität fehlt. Ebenfalls in diese Gruppe gehören die Durchgängigkeit der Werkzeugkette und das Monitoring der Integrationslösung. Dabei bezieht sich die letztere Äußerung auf die Verknüpfung eines Werkzeugs mit einer laufenden Instanz einer Integrationslösung. Die restlichen Nennungen sind keineswegs weniger wichtig als die vorangegangenen. Insbesondere das Change-Management bestehender Integrationslösungen oder eine Metadatenverwaltung stellen herausfordernde Probleme dar, die zumeist deshalb nicht genannt werden, weil sie für zu visionär gehalten werden.

Für das Skizzieren der Funktionalität eines idealen Tools wurden die Interviewteilnehmer mit der in der folgenden Aufzählung enthaltenen Funktionalität konfrontiert. Diese sollte in einem idealen Tool enthalten sein und die Teilnehmer wurden gebeten zunächst einzuschätzen, inwieweit ein derartiges Tool bereist existiert und den Nutzen für ihre Art von Integrationsdienstleistung einschätzen.

- Erfassung IST und SOLL Unternehmensarchitektur,
- strukturierte Erfassung aller Integrationsinformationen,
- Hilfestellung bei der Auswahl der geeigneten Integrationsmethode (Entscheidungsunterstützung),
- Erhöhung des Automatisierungsgrades durch Generierung.

Prinzipiell als positiv und damit nützlich wurde die skizzierte Funktionalität von 13 Experten eingeschätzt. Dagegen sprachen sich nur 9 Experten aus. Die Abbildung 3.16 zeigt auf, mit welcher Häufigkeit zu den einzelnen Funktionalitäten Stellung bezogen wurde und kann damit einen Anhaltspunkt liefern, welche Bedeutung die einzelne Funktionalität hat. Die meisten

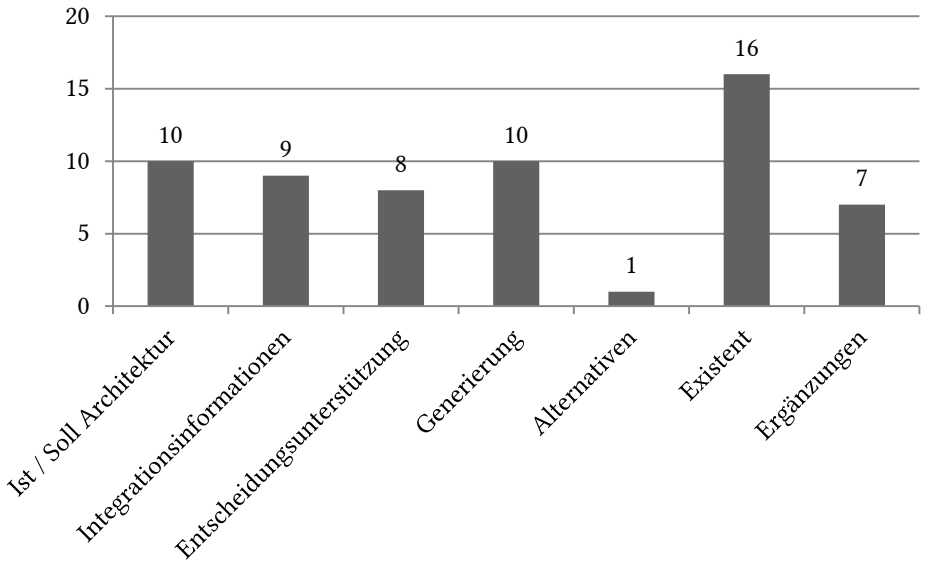


Abbildung 3.16: Wünschenswerte Funktionalitäten für ein ideales Tool und Existenz eines solchen

Quelle:(Gebauer und Stefan, 2011b, S. 176)

Äußerungen wurden getätigt um darauf hinzuweisen, dass es ein derartiges Werkzeug bereits gibt. Bei genauerem Nachfragen wurde jedoch deutlich, dass sich diese Aussage immer nur auf Teilfunktionen bezog. Ein Werkzeug, welches den vollständigen Funktionsumfang abdeckt, ist nicht bekannt. Die Aufnahme der IST/Soll Architektur wird von 10 Experten als wichtig beurteilt. Dabei wird aber darauf hingewiesen, dass man diese auf den relevanten Ausschnitt der Unternehmensarchitektur, welcher vom Integrationsprojekt betroffen ist, beschränken sollte. Bei großen Unternehmen würde die Erfassung ansonsten zu viel Zeit in Anspruch nehmen. Die Menge der erfassten Informationen bestimmt den Aufwand welcher für die Aktualisierung und das aktuell halten notwendig ist. Es wird teilweise billigend in Kauf genommen, dass die Informationsqualität an einigen Stellen suboptimal ist, damit der Aktualisierungsaufwand moderat bleibt. Weiterhin wäre eine automatische Aufnahme dieser Informationen wünschenswert. Dieser Schritt muss mit dem nächsten, der Erfassung aller Integrationsinformationen, Hand in Hand gehen.

Die Erfassung aller Integrationsinformationen wurde von neun Experten als fehlend beurteilt. Insbesondere die Granularität und die Menge der zu erhebenden Daten werden dabei als kritische Punkte gesehen. Es muss sichergestellt werden, dass eine umfangreiche Erfassung von Detailinformationen nicht praktikable Ergebnisse verhindert. Als Vorschlag wurde eine Abstufung je nach Zweck der Erfassung vorgeschlagen. Für die Auswahl eines Architekturmusters sind weniger Informationen notwendig als für eine Generierung. Darüber hinaus wäre es ebenfalls wünschenswert, wenn auch mit unvollständigen Informationen Ergebnisse erzielbar wären. Ein Experte betonte, dass damit die Bereiche Unternehmensarchitekturmanagement und Integration verbunden würden und dass aus seiner Sicht eine Kommunikation der Architekturfestlegungen an die mit der Integration betrauten Projektteams ausreichen würde.

Zu der Funktionalität der Entscheidungsunterstützung haben sich 8 Experten geäußert, womit dies die Funktionalität mir den wenigsten Nennungen ist. Hier wurden wiederum Bedenken

bezüglich der dafür notwendigen Datenmenge und Qualität geäußert. Würde es gelingen in 80 Prozent der Fälle eine sinnvolle Unterstützung zu liefern so käme man schon sehr weit. Die 20 Prozent der Spezialfälle sind, laut einem Experten, mit der Hand zu bewältigen. Auch hier wurde die bereits bei der Erfassung der Informationen erwähnte Abstufung der Datenmenge vorgeschlagen. Teilweise würde den Experten schon eine Beschreibung der Entscheidungspfade in Prosa ausreichen, um diese zu konservieren. Die strukturierte Präsentation wurde als noch wichtiger, als die Entscheidungsunterstützung selbst, eingeschätzt.

Die Generierung von Integrationslösungen haben 10 Experten kommentiert. Dieser Punkt wurde aufgrund des herrschenden Rationalisierungsdrucks hinzugefügt. Dabei wird dies als die am aufwendigsten zu realisierende Teilfunktionalität eingestuft. Da eine Zusammenführung aller Funktionen in einem Werkzeug zu anspruchsvoll erschien, wurde als Lösung ein Plug-in-Konzept für Generierungskomponenten vorgeschlagen. Mit der Zahl der verfügbaren Plug-in-Komponenten ließe sich die Zahl der unterstützten Endsysteme sukzessive erweitern. Darüber hinaus kann dieses Konzept nur in Bereichen funktionieren, die nicht Kernkompetenz der Kundenunternehmen und damit hochgradig individuell sind. In Kernbereichen ist die Individualität der Kunden zu groß, was die Gleichheit der Lösungen limitiert und einem Generierungsansatz die Grundlage entzieht.

3.5.5 Kundenfaktoren

Die Kundenfaktoren spielen für diese Arbeit nur insoweit eine Rolle, als dass sie die Auswahlkriterien enthalten, nach denen sich die Kunden der befragten Experten bei der Auswahl von Integrationslösungen richten. Diese Kriterien werden an dieser Stelle vorgestellt, da es von Interesse für die spätere Analyse bekannter Mechanismen der Wiederverwendung ist, inwieweit diese Kriterien direkt oder indirekt adressiert werden. Es ist davon auszugehen, dass der Einsatz von Konzepten für einen strukturierten Entwurf in der Praxis eine besonders große Akzeptanz findet, wenn die Auswirkungen auf diese Kriterien bekannt sind oder zumindest bestimmbar bleiben. Die Abbildung 3.17 stellt die fünf am häufigsten genannten Auswahlkriterien vor.

Die fünf häufigsten Auswahlkriterien stammen ausschließlich aus dem Bereich der nicht funktionalen Anforderungen. Dabei ist das am häufigsten genannte Kriterium das der Kosten. Die Abbildung differenziert dabei zwischen den *Projektkosten*, welche häufiger zur Auswahl herangezogen werden, und den *Total-Cost-of-Ownership*. Am zweit häufigsten steht die *Flexibilität, Änderbarkeit, Skalierbarkeit* einer Lösung als Kriterium im Vordergrund. Dies verdeutlicht die immensen Anforderungen an Integrationsdienstleister, welche nicht nur das aktuelle Integrationsproblem lösen müssen, sondern eine Lösung zur Verfügung stellen sollen, welche mit den Kunden und deren Anforderungen mit wächst. Die restlichen 3 Auswahlkriterien der TOP 5 weisen alle die gleiche Zahl an Nennungen auf. Neben der *Betreibbarkeit* und der *Stabilität* der Lösung wurde dabei auch die *Zeit bis zur Fertigstellung der Lösung* genannt. Diese steht in engem Zusammenhang mit dem Top 1 Kriterium den *Projektkosten* und verdeutlicht die in der Problemstellung skizzierte Problematik der Verkürzung der Projektlaufzeiten bei gleichzeitig immer weiter erodieren Budgets für Integrationslösungen[36]. Neben den TOP 5 wurden weitere Auswahlkriterien genannt, welche sich z. B. im Fall von *Performance*, nur durch eine Nennung von den Top 5 unterscheiden. Diese sind in Abbildung 3.18 enthalten. Bemerkenswert ist, dass erst an 10. Stelle – die Top 5 mit eingerechnet – mit lediglich 5 Nennungen die *Erfüllung der funktionalen und technischen Anforderungen* steht. Das lässt den Rückschluss zu, dass man sich als Integrationsdienstleister dadurch nicht all zu sehr am Markt differenzieren kann. Anders gesprochen kann man damit die Integration aus einer technisch-funktionalen Sichtweise als

[36] Vgl. Abschnitt 1.1

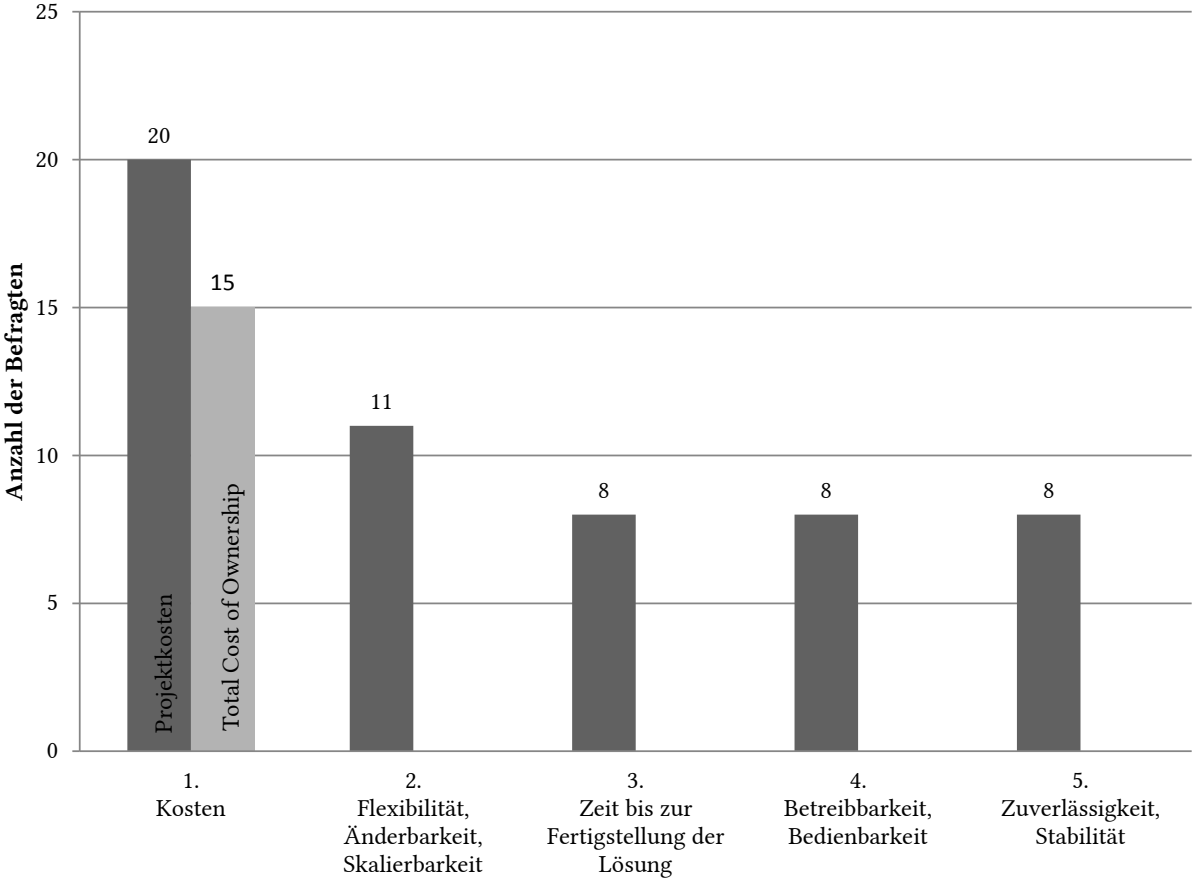


Abbildung 3.17: Top 5 Kriterien für die Auswahl einer Integrationslösung
Quelle: (Gebauer und Stefan, 2011b, S. 79)

austauschbares Handelsgut [37] bezeichnen. Für einen Integrationsdienstleister stellt sich also die Frage wie er die technisch-funktionalen Anforderungen bedienen und sich gleichzeitig in einem der anderen Kriterien – z. B. im Bereich Projektkosten – hervorheben kann. Darüber hinaus spielen politische Entscheidungen fast ebenso häufig – mit 4 Nennungen – eine Rolle, wie technisch-funktionale Anforderungen und legen den Einfluss Entscheidungen prinzipieller Natur dar. Ein Beispiel für politische Entscheidungen sind einmalige strategische Ausrichtungen der Unternehmen, welche die künftige Unternehmensentwicklung antizipieren. Laut der vorliegenden empirischen Erhebung ist der Einfluss dieser zukünftigen Unternehmensentwicklung auf die Ausrichtung der Integrationslösung abhängig vom jeweiligen Kunden. Das heißt, ob eine Ausrichtung stattfindet bzw. gefordert wird, hängt maßgeblich von dem Grad der IT-Durchdringung des Geschäftes des Kunden, der Größe des Kunden und auch der Größe der Geschäftspartner des Kunden ab. Vorzugsweise große Kunden und Kunden mit einer hohen IT-Durchdringung sind sich der Bedeutung einer mit wachsenden Integrationslösung bewusst und fordern eine entsprechende Planung ein. Die Größe der Geschäftspartner des Kunden spielt dann eine Rolle, wenn ein großer Partner eine entsprechende Ausrichtung – aufgrund der Bedeutung der Geschäftsbeziehung – diktieren kann. Die Umsetzung der strategischen Planung kann jedoch an den Strukturen und internen Verteilungskämpfen der Großunternehmen scheitern.

Sechs Experten vertraten die Erkenntnis, dass diese Planungen bei klein- und mittelständischen Unternehmen fehlen, im Gegensatz aber fachliche Bebauungspläne existieren. In diesen werden

[37] engl.: commodity

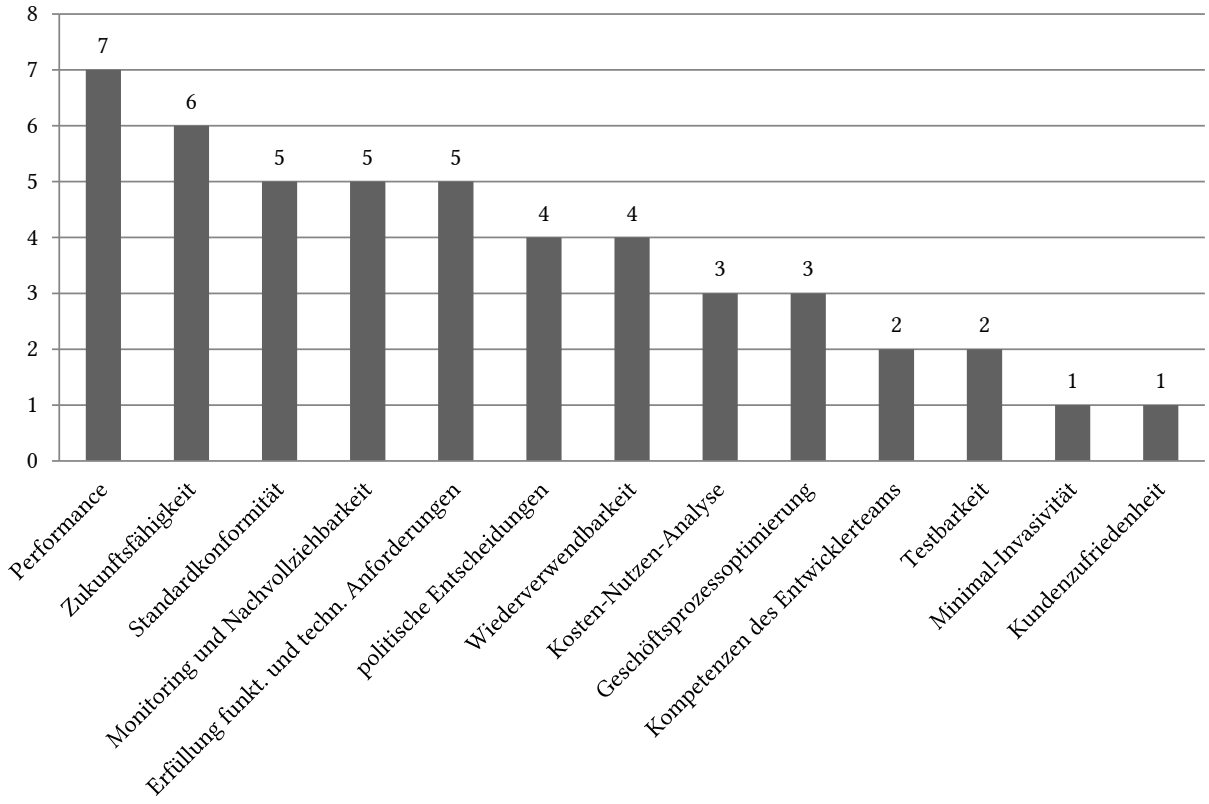


Abbildung 3.18: Weitere Kriterien für die Auswahl der Integrationslösung
Quelle: (Gebauer und Stefan, 2011b, S. 87)

keine IT-Systeme festgelegt, sondern die fachlichen Funktionen und Anforderungen definiert, welche zukünftig zu unterstützen sind. Darüber hinaus ist der Planungshorizont in diesen Fällen kürzer und ermöglicht es, flexibler auf Änderungen zu reagieren. Im Gegensatz zu diesen Äußerungen sind 10 Experten der Auffassung, dass diese Planungen generell – also unabhängig von den oben genannten Einflussgrößen – bei allen Unternehmen fehlen, mindestens nicht schriftlich festgehalten werden und meist nicht an die betreffenden Mitarbeiter kommuniziert sind. Alles in allem lässt sich festhalten, dass die langfristige Ausrichtung einer Integrationslösung mangelhaft ist und die Erstellung häufig durch operative Ziele getrieben wird.[38]

3.6 Zusammenfassung der empirischen Resultate

Die vorgestellte empirische Erhebung bildet die Grundlage für die in dieser Arbeit entwickelten Lösungskonzepte. Deshalb werden an dieser Stelle die wichtigsten Erkenntnisse noch einmal zusammengefasst. Diese dienen in den folgenden Kapiteln als Grundlage.

Die befragten Unternehmen sind ausgewiesene Integrationsdienstleister mit langer Präsenz am Markt. Alle – in die Auswertung eingeflossenen – Experten sind Spezialisten im Bereich der Integration und kennen ihre Unternehmen aufgrund einer längeren Zugehörigkeit. Sie bekleiden entsprechend hohe Positionen in der Hierarchie der Unternehmen, sodass sie die strategische Ausrichtung der Firmen voll überblicken.

[38] Vgl. (Gebauer und Stefan, 2011b, S. 92ff.).

Es existiert kein (Referenz-)Modell für Integrationsdienstleistungen, welches die notwendigen Ressourcen und Ergebnisse in Zusammenhang setzt. Dennoch ist es aus den Ergebnissen der Experteninterviews möglich, entsprechenden Input zu gewinnen. Rollen, Phasen und Arbeitsergebnisse wurden durch die Experten benannt und finden im weiteren Verlauf der Arbeit Verwendung.

Es sind wiederkehrende Tätigkeiten und Probleme der Integrierung bekannt und konnten durch die Experten benannt werden. Insbesondere Kommunikationsprobleme und das Beherrschen der Komplexität von Integrationsprojekten dominieren.

Diese werden zum Teil mit bekannten Lösungsansätzen adressiert, welche aber in unterschiedlicher Weise unter den Mitarbeitern verbreitet werden. Es dominieren Pull-Verfahren, weshalb diese Mechanismen nicht zwingend genutzt werden müssen. Sie werden teilweise bewusst, häufig unbewusst umgangen.

Muster sind den Mitarbeitern von Integrationsdienstleistern nur im Rahmen ihrer Ausbildung bekannt. Dabei handelt es sich um die Kenntnis von einzelnen Mustern auf verschiedenen Abstraktionsniveaus, ein konsistentes System ist nicht bekannt. Die bekannten Muster werden aber nicht bewusst für den Entwurf von Integrationslösungen eingesetzt. Die Verwendung liegt – wenn überhaupt – in der Initiative Einzelner begründet.

Nicht alle Schritte für die Erstellung einer Integrationslösung sind durch die aktuell verwendeten Werkzeuge oder Hilfsmittel abgedeckt. Es gibt fehlende Funktionalitäten, die von der Überwachung einzelner Lösungen bis hin zur Aufnahme und Analyse der Architektur der Kunden reichen. Die skizzierten Funktionalitäten eines idealen Tools werden als wünschenswert eingestuft und sind aus der Sicht der Dienstleister prinzipiell nützlich. Die Realisierung wird aufgrund der Komplexität und des Notwendigen Input an Daten, welcher von der Menge und Qualität her hoch ist, als schwer zu realisieren eingeschätzt. Die Empfehlungen der Experten laufen darauf hinaus, die Erfassung und Analyse zu unterstützen und die Entscheidungsunterstützung in einem späteren Schritt zu realisieren. Die Generierung sollte – wenn überhaupt – per Plug-in-Mechanismus angegangen werden.

Die Gesichtspunkte, nach denen die Kunden eine Integrationslösung auswählen, sind den Dienstleistern bekannt. Es handelt sich bei den 5 wichtigsten Kriterien jedoch ausnahmslos um nichtfunktionale Anforderungen. Aufgrund der wenig ausgeprägten Nutzung von Integrationsmustern sind die Eigenschaften der Muster in Bezug auf diese Nicht-funktionalen Anforderungen nicht bekannt und finden dementsprechend keine Verwendung.

4 Integration, eine Dienstleistung

Die Ziele dieser Arbeit sind die Rationalisierung und Qualitätsverbesserung der Integration. Dabei gilt es, dem Umstand der Erbringung von Integrationsleistungen Rechnung zu tragen. Aus diesem Grund wird in diesem Abschnitt zunächst bestimmt, was eine Dienstleistung ist und wie sich die Integration als Dienstleistung einordnen lässt. Darüber hinaus werden Unternehmen welche Integrationsleistungen offerieren als Integrationsdienstleister eingeordnet und im Anschluss daran – auf der Basis der Ergebnisse der Experteninterviews[1] – das Dienstleistungsbündel eines Integrationsdienstleisters bestimmt. Dafür wird ein Modell entworfen, für dessen Konstruktion existierende Abhandlungen zur Gestaltung von Integrationslösungen herangezogen und in diese neue Sichtweise auf die Integration eingearbeitet werden. Dies ist notwendig, um die Ergebnisse des nachfolgenden Kapitels – die hochstehenden Integrationsmuster – einordnen zu können und deren Auswirkungen auf die Erbringung von Integrationsdienstleistungen richtig einzuschätzen. Der Einsatz von Mustern – insbesondere auch der von abstrakten, hochstehenden Mustern – greift in die Zusammensetzung der Ressourcen für die Dienstleistung ein und verändert diese. Dadurch werden wesentliche Faktoren einer Dienstleistung wie Zeitdauer, Kosten und Ergebnisqualität beeinflusst. Diese Darstellung wird im Anschluss an die Entwicklung hochstehender Integrationsmuster[2] wieder aufgegriffen und der Einfluss derselben gezeigt.

Die Integration von Anwendungssystemen ist eine komplexe Aufgabe, die Unternehmen häufig nicht selbst erbringen. Erforderliche Leistungen werden bei Spezialisten am Markt eingekauft. Damit handelt es sich bei der Integration umgangssprachlich um eine Dienstleistung. In der Dienstleistungsforschung gibt es eine breite Diskussion des Dienstleistungsbegriffes. An dieser Stelle soll keinen Diskussionsbeitrag geleistet werden, die Arbeit stützt sich vielmehr auf entsprechende Definitionen. Im Einklang mit der systemtheoretischen Auffassung von Betrieblichen Informationssystemen, kommen Dienstleistungsdefinitionen in Betracht, welche ebenfalls von einem Sytembegriff ausgehen.[3] Dienstleistung wird"etwas vereinfachend und auf den Prozess- sowie den Kundennutzen ausgerichtet, als Leistung verstanden, die an einem Menschen oder am Objekt und ohne Transformation von Sachgütern erbracht wird."[4] Bei dieser Definition wird der maßgebliche Unterschied der Intangibilität einer Dienstleistung als Differenzierungsmerkmal von Sachgütern zu Grunde gelegt. Weiterhin kommt hinzu, dass der Erstellungs- und der Verbrauchszeitpunk einer Dienstleistung zusammen fallen. Im weiteren Verlauf wird die Definition einer Dienstleistung nach Böttcher und Klingner verwendet, welche Folgendes aussagt:

> *„Services are an offered functionality resulting from an interaction of a well-defined set of resources. This set can be called service system. The functionality leads to a change of the state of at least one resource of the service system. Furthermore at least one of the changed resources belongs to a customer to whom the change is of any value."*[5].

Eine detailliertere Erörterung der Aspekte des Dienstleistungsbegriffes kann bei Bieger[6] gefunden werden.

[1] Vgl. Abschnitt 3.5.
[2] Vgl. Kapitel 5.
[3] Vgl. (Spohrer u. a., 2007; Vargo und Lusch, 2004).
[4] (Bieger, 2007, S. 10), vgl. auch (Zarnekow, 2007, S. 9).
[5] (Böttcher und Klingner, 2011, S. 638)
[6] Vgl. (Bieger, 2007, S. 8ff.).

Integration ist vereinfacht ausgedrückt die Überbrückung von Heterogenität und die Sicherstellung der Kommunikation.[7] Integrationsleistungen in BIS oder in Systemen aus BIS[8] lassen sich auch als die Transformation von Unternehmensarchitekturen von einem Zustand in den nächsten auffassen. Die dazu notwendige Funktionalität wird durch die Leistungen eines Integrationsdienstleisters zur Verfügung gestellt. In dem Moment, wo eine Integrationslösung in Betrieb genommen wird, kann das Zusammenfallen des (finalen) Erstellungs- und Verbrauchszeitpunktes beobachtet werden. Der Integrationszustand der betroffenen Systeme ändert sich augenblicklich und die Systeme sind integriert. Das bestehende Integrationsszenario wird durch ein neues Szenario abgelöst.[9] Eine Transformation von Sachgütern in einem engen Sinne liegt ebenfalls nicht vor. Hardware, die in bestimmten Integrationssituationen für eine Integrationslösung neu in einem Unternehmen etabliert wird, wird streng genommen nicht transformiert. Insofern ist auch die Eigenschaft gegeben, dass keine Sachgüter transformiert werden. Zusätzlich lässt sich eine Einstufung von IT-Unternehmen als Dienstleister nachweisen.

Unternehmen lassen sich nach verschiedenen Gesichtspunkten gliedern. Gliederungsmöglichkeiten sind die Art der erstellten Leistung, der Art und Umfang der Leistungswiederholung und die Organisation des Fertigungsablaufes. Die Abbildung 4.1 zeigt die Einordnung der Unternehmen der Informationstechnologie (IT) nach Zarnekow.[10] Auf der Ersten Ebene wird das Gliederungs-

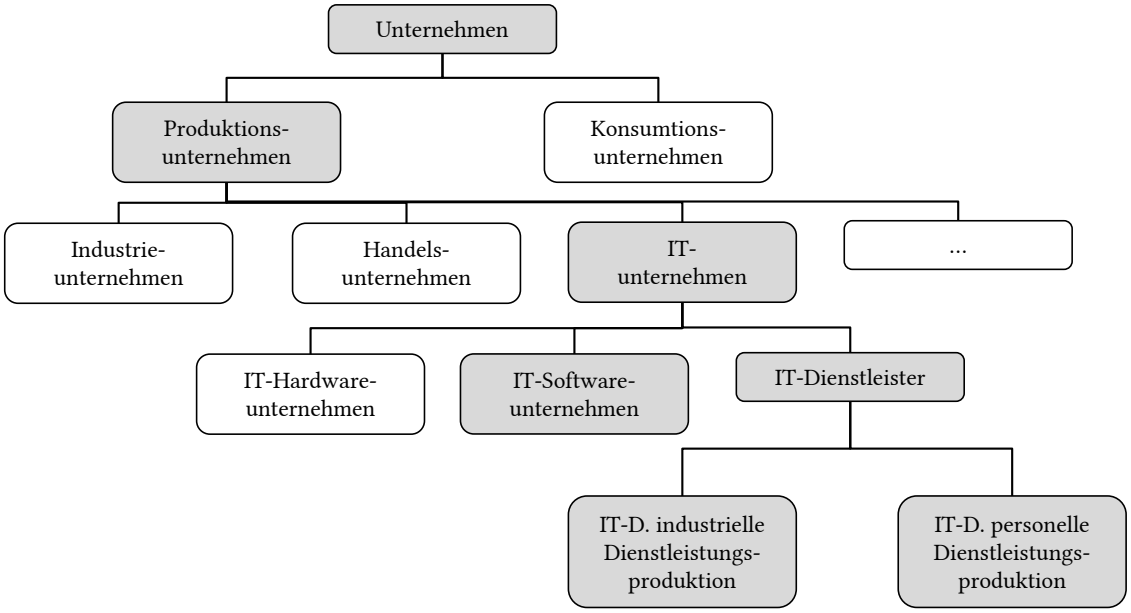


Abbildung 4.1: Einordnung von IT-Unternehmen nach Zarnekow
Quelle: Zarnekow, 2007, S. 10

merkmal des Wirtschaftszweiges angewandt. IT-Unternehmen ordnen sich demnach in den Zweig der Produktionsunternehmen und dort in den Unterzweig der IT-Unternehmen ein. In der zweiten Ebene werden die IT-Unternehmen nach der Art der erstellten Produkte weiter untergliedert,

[7] Vgl. Abschnitt 2.2.
[8] Dies deckt unternehmensübergreifende Aspekte ab.
[9] Vgl. Abschnitt 2.2 für die Definition eines Integrationsszenarios. Eine Ausnahme stellt eine Situation dar, bei der die Schnittstelle ein Integrationsobjektes durch eine neue abgelöst werden soll. Dazu kann das Integrationsobjekt um die neue Schnittstelle ergänzt werden und für eine bestimmte Zeit werden beide Schnittstellen parallel betrieben bevor die alte abgeschaltet wird.Hier gibt es temporär ein Zwischenintegrationsszenario. Dieser Fall tritt bei hoch verfügbaren Lösungen auf und erzeugt Anforderungen an die Schnittstellenverwaltung und -versionierung.
[10] Vgl. (Zarnekow, 2007).

wobei die Kategorien Hardware, Software und Dienstleistungen von einander unterschieden werden. In der letzten Ebene kann anhand der Unterscheidung nach der Art der Produktion, die personelle von der industriellen Dienstleistungsproduktion unterschieden werden. Legt man die allgemeine Annahme zu Grunde, dass IT-Unternehmen diejenige Gruppe von Unternehmen sind, deren Produkte in den Bereich der IT einzuordnen sind, so sind IT-Dienstleister als Unternehmen zu definieren, die Dienstleistungen im Bereich der Informationstechnologie erbringen.

Integrationsdienstleistungen mit dem Fokus auf Anwendungssystemintegration lassen sich der Gruppe der IT-Dienstleistungen zuordnen. Sie werden mit Hilfe von und an IT-Systemen erbracht. Aus der heutigen Sicht erfolgt jedoch die Erstellung von Integrationslösungen zu einem erheblichen Prozentsatz durch den Einsatz menschlicher Arbeitskraft.[11] Demnach handelt es sich um eine Dienstleistung mit personeller Dienstleistungsproduktion.

Ein Teilziel der Arbeit ist es die Produktion von Integrationsdienstleistungen effizienter zu gestalten, wozu zunächst einmal alle Details der Herstellung verstanden werden müssen. Um die Teile einer Integrationsdienstleistung detailliert und modellhaft zu erfassen, werden im folgenden Abschnitt die Methoden der Dienstleistungsmodellierung dargelegt.

4.1 Dienstleistungsmodellierung

Die Ausführungen in diesem Abschnitt haben zum Ziel die Grundlagen zu einem Modell für Integrationsdienstleistungen zu legen. Arbeiten wie eine Unified-Service-Description-Language (USDL)[12] müssen verworfen werden, da sie eine unzureichende Vereinfachung auf die Konsumption einer Dienstleistung darstellen. Für das Interesse dieser Arbeit spielt jedoch die Herstellungszeit die wichtigste Rolle. Da es eine Vielzahl an Arbeiten auf diesem Gebiet gibt, erfolgt eine Konzentration auf ein Werk, welches eine Synthese verschiedener Konzepte und eine Evaluation in der Praxis vorgenommen hat. Böttcher legt in seiner Arbeit[13] den Grundstein für die Theorie und Praxis der Dienstleistungsmodellierung. In die Analyse sind mehr als 20 Werke eingeflossen, welche sich mit den Konzepten und der Theorie der Dienstleistungsmodellierung auseinandersetzen. Darüber hinaus wurden 12 bereist existierende Modellierungsansätze geprüft und evaluiert und die verwendeten Konzepte extrahiert. Diese Analyse führte zu einer Konzeptmenge, welche mehr als 350 Einzelkonzepte enthielt, die dann weiter verdichtet wurden, in dem z. B. Synonyme und Homonyme aufgelöst und unterschiedliche Abstraktionsebenen der Konzepte berücksichtigt wurden. Das Ergebnis dieser Arbeit ist ein technologieunabhängiges Metamodell. Die praktische Einsetzbarkeit wurde von Böttcher mit einer Implementierung des Metamodells mithilfe des Eclipse-Modeling-Framework (EMF) und des Graphical-Modeling-Framework (GMF) und einer nachfolgenden Evaluation an Fallstudien nachgewiesen. Da es nicht Ziel dieser Arbeit ist, sich im Kern mit der Modellierung von Dienstleistungen auseinanderzusetzen, sondern vielmehr die Integration als Dienstleistung in Ihre Teile zu zerlegen, wird das bei Böttcher vorgestellte Metamodell verwendet, ohne auf spezifische Implementierungen einzugehen.

Die wesentlichen Teilmodelle eines Dienstleistungsmodells nach Böttcher sind das Ressourcenmodell, das Produktmodell, das Prozessmodell und das Modell der Dienstleistungskomponenten. Die Inhalte und Bedeutung der einzelnen Teilmodelle werden hier nur kurz vorgestellt, für die vollständigen Details sei auf die Arbeiten von Böttcher verwiesen.[14]

Es können vier Dimensionen unterschieden werden, welche einzeln verwendbar sind, jedoch so stark in Zusammenhang stehen, dass sich Änderungen übergreifend auswirken. Die Abbildung

[11] Diese Aussage wird durch das Ressourcenmodell Vgl. Abschnitt 4.2.2 bestätigt.
[12] Vgl. (Cardoso; Barros u. a., 2010).
[13] Vgl. (Böttcher, 2008).
[14] Vgl. (Böttcher, 2008; Böttcher und Klingner, 2011).

4.2 verdeutlicht die einzelnen Dimensionen. Zu unterscheiden ist zunächst das Komponenten

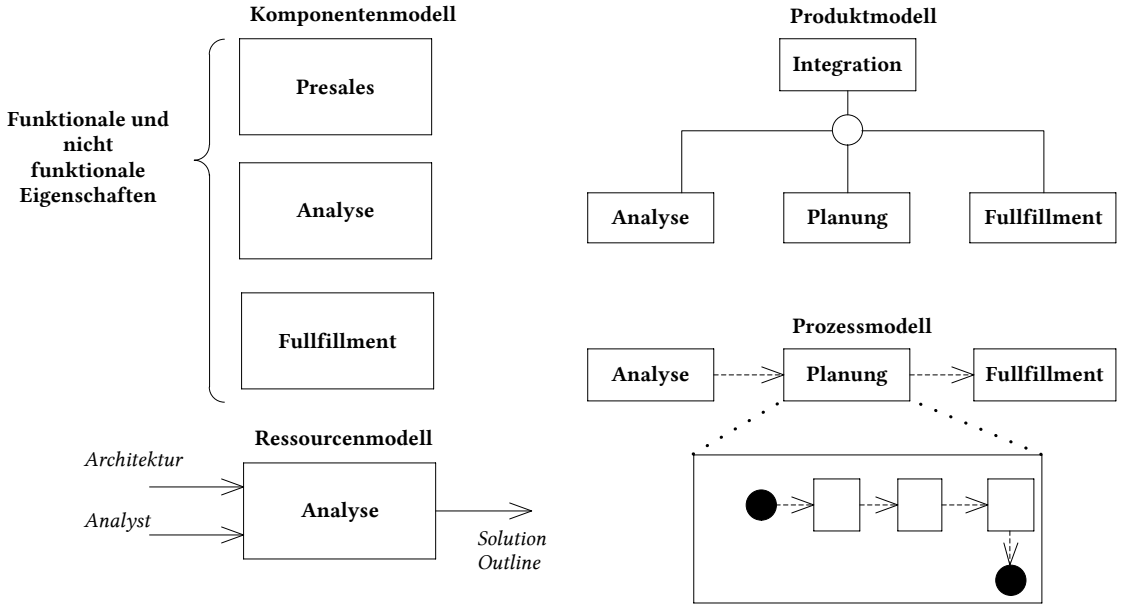


Abbildung 4.2: Dimensionen der Dienstleistungsmodellierung
In Anlehnung an: (Böttcher und Klingner, 2011, S. 656)

Modell, welches eine Dienstleistung in Teilleistungen aufspaltet. Die zweite Dimension stellt auf die Tatsache ab, dass eine Dienstleistung von Ressourcen abhängig ist, die miteinander interagieren. Diese Dimension wird Ressourcenmodell genannt. Böttcher und Klingner geben als Ziel der Modellierung von Dienstleistungen die Zusammenstellung von kundenspezifischen Dienstleistungen aus einer Menge an standardisierten Dienstleistungskomponenten an. Zu diesem Zweck verwenden Sie eine dritte Dimension, welche als Produktmodell oder hierarchisches Modell charakterisiert wird. Die vierte Dimension erfasst den zeitlichen Ablauf der Erbringung der Dienstleistung und damit die temporalen Abhängigkeiten der einzelnen Dienstleistungskomponenten. Diese Dimension wird Prozessmodell genannt.

4.2 Ein Modell für Integrationsdienstleistungen

In diesem Abschnitt wird ein Modell für Integrationsdienstleistungen vorgestellt, welches es ermöglichen soll die einzelnen Bestandteile einer Integrationsdienstleistung besser erfassen zu können. Entsprechend der Darlegungen in Abschnitt 4.1 sind für ein Dienstleistungsmodell die Servicekomponenten, Ressourcen, die zeitlichen Abhängigkeiten sowie die Kombinationsmöglichkeiten einzelner Servicekomponenten notwendig. Auf der Basis einer umfassenden Literaturanalyse und den Ergebnissen der in Kapitel 3 vorgestellten empirischen Untersuchungen wurde eine Synthese der verschiedenen Quellen erarbeitet. Diese werden in den folgenden Unterabschnitten vorgestellt. Die Darstellung bewegt sich auf dem Abstraktionsniveau eines Modells, welches ohne Anpassung im Sinne einer Instanziierung nicht direkt anwendbar ist. An einem Beispiel verdeutlicht, bedeutet dies, dass in dem Modell z. B. Ressourcen auf dem Abstraktionsniveau eines Integrationshilfsmittels vorgestellt werden. Dabei kann es sich um verschiedene Middleware-Plattformen handeln, welche für die Konfiguration einer konkreten Dienstleistung durch eine konkrete Middleware – z. B. einen JBoss ESB – zu belegen sind. In den folgenden Unterkapiteln werden die Servicekomponenten (4.2.1), die Ressourcen (4.2.2), das Produktmodell (4.2.3) und das Prozessmodell (4.2.4) vorgestellt. Dabei werden je Teilmodell zunächst die durch Böttcher

vorgeschlagenen Beschreibungskomponenten dahin gehend evaluiert, ob sie auf der Ebene der Analyse spezifizierbar und damit anwendbar sind. In den jeweiligen Tabellen kennzeichnet *wird verwendet*, dass es – im Rahmen der vorliegenden Arbeit – möglich ist dieses Merkmal näher zu charakterisieren. Ein mit *dienstleisterabhängig* ausgewiesenes Merkmal bedeutet keineswegs, dass dieses vernachlässigt werden kann. Es handelt sich um einen Punkt in dem Modell, der Gegenstand der Instanziierung durch den Dienstleister sein muss, da nur dieser dieses Merkmal endgültig spezifizieren kann. Das eingängigste Beispiel ist die Festlegung der Kosten. Im Anschluss an diese Evaluierung erfolgt die Darlegung des jeweiligen Teilmodells auf der Basis der in der Studie zur Systemintegration erhobenen Daten.

4.2.1 Die Servicekomponenten der Integration

Servicekomponenten sind zeitabhängig und repräsentieren Teilschritte eines Prozesses[15], im dargestellten Fall die Prozessschritte auf dem Weg zu einer Integrationslösung. Für die Erstellung der Servicekomponenten ist es möglich, sich auf die Ergebnisse der empirischen Untersuchung zu stützen.[16] Bisher wurden die Phasen der Integration nur auf einem hohen Abstraktionsniveau vorgestellt. Es wurden 7 Phasen unterschieden: Pre-Sales, Analyse, Planung, Fullfillment, Test, Transition und Betrieb.[17] Die Details der einzelnen Phasen, welche ebenfalls im Rahmen der Studie gewonnen wurden, haben bisher keinen Eingang in die vorliegende Arbeit gefunden. Auf dieser Basis werden nun sukzessive die Servicekomponenten beschrieben.[18] Eine Servicekomponente kann nach Böttcher und Klingner[19] als die Offerte einer wohldefinierten Funktionalität auf Basis präzise beschriebener Schnittstellen gesehen werden. Sie kann für die Komposition und die Dekomposition von Servicekomponenten feinerer oder gröberer Granularität genutzt werden. Gerade die selektive Komposition auf der Basis einer größeren Menge an Servicekomponenten erlaubt die Adaption der schlussendlich angebotenen Gesamtdienstleistung an den jeweiligen Kunden. Für die weitere Beschreibung werden verschiedene Eigenschaften einer Servicekomponente – auf der Basis des Metamodells von Böttcher – vorgestellt und in der Tabelle 4.1 für die Zwecke dieser Arbeit bewertet.

Nicht alle der vorgeschlagenen Eigenschaften lassen sich auf Integrationsdienstleistungen anwenden bzw. können spezifiziert werden. Insbesondere die mit *dienstleisterabhängig* gekennzeichneten Eigenschaften stellen Variationspunkte des Modells dar, die der einzelne Dienstleister zur Differenzierung nutzen kann. Eigenschaften wie die *Dauer* und das *Bezahlverfahren* oder die *Identität* der Servicekomponente hängen von den Gegebenheiten des jeweiligen Dienstleisters ab. Darüber hinaus lassen sich diese Eigenschaften für Servicekomponenten auf der gewählten Abstraktionsebene nicht angeben. Eine Besonderheit sind *lokale und temporale Verfügbarkeit* und *Evaluationskriterien*. In bestimmten Fällen können diese Merkmale objektiv bestimmt werden, was aber nicht immer der Fall ist, weshalb die Bewertung *wird eingeschränkt verwendet* zustande kommt.

Im Folgenden werden die einzelnen Servicekomponenten beschrieben. Die Beschreibungen beziehen sich dabei ausschließlich auf Fakten, welche in der Studie von den Befragten explizit geäußert worden. Am Beispiel der Planung und den genannten Zielen wird deutlich, dass sich schnell weitere Input- und Outputgrößen ableiten lassen. Die Annahmen wären jedoch spekulativ und ohne Nachweis, dass ein Einsatz in der Praxis wirklich stattfindet. So wurde auf diesen Schritt zunächst verzichtet. Dieser wird Gegenstand weiterer Forschungsbemühungen sein (siehe Kapitel 8).

[15] Vgl. (Böttcher und Klingner, 2011, S. 656).
[16] Vgl. Abschnitt 3.5, sowie (Gebauer und Stefan, 2011b).
[17] Vgl. Abbildung 3.6.
[18] Die zugehörigen Tabellen der Studie befinden sich der Vollständigkeit halber im Anhang A.
[19] Vgl. (Böttcher und Klingner, 2011, S. 657).

Eigenschaft	Beschreibung/Ausprägungen	Bewertung
Ziele	kann in Anbieter und Nutzerziele differenziert werden	wird verwendet
Klassifikation	implizite Funktionsbeschreibung durch Klassierung, Nutzung bestehender Standards wie eCl@ss oder UN/SPSC	für Integration sind keine Klassifikationsschematas verfügbar wird nicht verwendet
Ressourcen abhängige Funktionalität	implizite Spezifikation der Ressourcen; diese können Input oder Output sein	wird verwendet
Identität Anbieter lokale und temporale Verfügbarkeit	eindeutige Identifikation einer Dienstleistung Wer bietet die Dienstleistung an? Wann und Wo ist eine Dienstleistung verfügbar?	dienstleisterabhängig dienstleisterabhängig wird eingeschränkt verwendet
Dauer Bezahlverfahren Konsequenzen	Dauer der Dienstleistung Optionen der Bezahlung Was passiert wenn der Kontrakt der Dienstleistung verletzt wird? Diese Bedingungen werden häufig für Service Level Aggreements genutzt.	dienstleisterabhängig dienstleisterabhängig dienstleisterabhängig
Evaluationskriterien	Welche Kriterien eignen sich für die Evaluierung der Qualität der Dienstleistung?	wird eingeschränkt verwendet

Tabelle 4.1: Eigenschaften von Servicekomponenten
Quelle: In Anlehnung an (Böttcher und Klingner, 2011)

Servicekomponente	Ziele	Input	Output
Pre-Sales	A:Verständnis der strategischen und geschäftlichen Ziele des Kunden; N: Verständnis der anstehenden Aufgaben	*Vertrieb*, *Geschäftsleitung*	*Lösungsskizze(beschrieben)*
Analyse	Abstimmung der Anforderungen, klare Definition der Ziele	*UA*	*UA (teilw. dokumentiert)*
Planung	Relaisierungskonzept; Terminierung der Ressourcen		
Fullfillment	Entwicklung der Lösung		
Test	Test der Lösung	Integrationsartefakte (entwickelt), Hilfmittel (), Werkzeuge (), Integrationsobjekte ()	Integrationsartefakte (getestet)
Transition	Übergabe der Lösung an den Kunden und Inbetriebnahme		
Betrieb	Dauerhafter Betrieb der Lösung		
Ist Analyse	Erhebung des aktuellen Zustandes		*Unternehmensziele(festgelegt)*, *UA(ist)*
Soll Analyse	Festlegung des Zielzustandes	Unternehmensziele(festgelegt), UA(ist), Consultants	*UA(Soll)*
Fachliche Analyse	Analyse aus fachlicher Sicht		
Systemlandschaft aufnehmen	logische Anwendungskomponenten sind katalogisiert		Bebauungsplan
Prozessaufnahme	zu unterstützende Prozesse sind erfasst		Prozesslandkarte, Use-Cases, Anfoderungen
Prozess-System-Mapping	Systeme und Prozesse sind einander zugeordnet		Prozesslaufschema, Interaktionsmodell(grob)

Servicekomponenten der Integration – Fortsetzung auf der nächsten Seite

Service-komponente	**Ziele**	**Input**	**Output**
fachl. Nutzen-bewertung	Architektur ist nach fachlichen Gesichtspunkten bewertet		UA(ist, bewertet), Potential-analyse
Technische Analyse	technische Details der Architektur sind erhoben und bewertet		
Schnittstellen-identifikation	Zugriffspunkte sind identifiziert		Systemschnittstellen (dokumentiert)
Schnittstellen-klassifikation	Zugriffspunkte klassifiziert	Systemschnittstellen (dokumentiert)	Systemschnittstellen (dokumentiert,klassifiziert)
Kommunikations-analyse	Identifikation der Kommunikationstypen, -muster und -protokolle je System		*Kommunikations-dokumentation (beschrieben)*
technische Bewertung	Identifikation der Integrations-möglichkeiten, Implementierungstechnologien		Interaktionsmodell (detailliert), Systeme (bewertet)
Mappings	Identifikation von Heterogenität, Abbildungsvorschriften erarbeitet	Interaktionsmodell (detailliert), Schnittstellen, Anwendungsberater	Mappings(defined)
Konzeption	Lösungskonzept ist entwickelt	Referenzarchitekturen, Produktmanager, Anwendungs-spezialist, externe Berater	Blueprint, Architekturvorschlag, Relaisierungskonzept, Roadmap
Prototyping / Scope Selection	Pilotierung, Annäherung an die Lösung		Prototyp (fertig), Prozesse (selected)
Evaluierung	Auswahl und Bewertung von Technologien	Prototyp (fertig)	Technologie-/Toolmatrix (definiert)
Termin- und Ressourcenplanung			Lastenheft (definiert), Projektplan (erstellt), Personaleinsatzplan (fertig), Angebot (fertig)
Kundeninteraktion	Konzept und/oder Managementpräsentation, Abstimmung mit Kunden	Lösungskonzept (fertig)	Lösungskonzept (abgenommen)
Umgebung aufsetzen			Werkzeuge (installiert), Integrationsobjekte (installiert), Hilfsmittel (installiert)
Durchführung	entwickelte Integrationslösung		Werkzeuge , Hilfsmittel, Integrationsobjekte
Abnahme	intern abgenommene Lösung		Lösung (intern abgenommen)
Umsetzungs-konzept	geplanter Rollout	Rollout ()	Rollout (geplant)
Rollout		Rollout (geplant), Integrationsobjekte (), Hilfsmittel (), Integrationsartefakte	
Schulung			
Transition u. Parallelbetrieb			
Endabnahme			
Wartung			

Tabelle 4.2: Servicekomponenten der Integration

Einige der dargestellten Servicekomponenten sind sogenannte Obertypen, sodass bei diesen die Merkmale Input und Output ausgeblendet wurden. Diese ergeben sich als Aggregat der

Untertypen. Diese hierarchischen Beziehungen werden im nachfolgenden Kapitel anhand der Ressourcen erläutert.

4.2.2 Das Ressourcenmodell der Integration

Als Ressourcen der Integration kommen Menschen – im Sinne der Berater und Entwickler des Dienstleisters und des Kunden – sowie alle eingesetzten Werkzeuge und Hilfsmittel infrage. Um diese Ressourcen entsprechend des Metamodells von Böttcher näher zu spezifizieren, werden die Ergebnisse der empirischen Erhebung verwendet.[20] Für die Beschreibung eines Ressourcenmodells werden von Böttcher und Klinger die in Tabelle 4.3 beschriebenen Kriterien vorgeschlagen. Neben einer kurzen Vorstellung enthält die Tabelle Informationen, inwieweit die Eigenschaften hier verwendet werden können. Die Auswahlkriterien sind dabei analog der in Kapitel 4.2.1 verwendeten.

Eigenschaft	**Beschreibung/Ausprägungen**	**Bewertung**
Resource position	es kann Input und Output unterschieden werden	wird verwendet; Da diese Angabe jedoch abhängig von der Servicekomponente ist, erfolgt die Verwendung im Servicekomponenten-Modell.
Resource identity	eindeutige Identifikation	dienstleisterabhängig
Resource quantity	Quantität einer Ressource	dienstleisterabhängig
Capacity	es kann qualitative und quantitative unterschieden werden	diensteleisterabhängig
Flexibility	was ist notwendig um eine Ressource zu adaptieren	nicht verwendet
Resource relations	domänspezifische Relationen zwischen Ressourcen	wird verwendet
Resource hierarchy	spezielle Relation zum Ausdrücken hierarchischer Beziehungen	wird verwendet
Resource source type	{provider|customer|external}	wird verwendet
Tangibility	{tangible|human|intangible}	
Mobility	ist die Ressource mobil oder immobil	wird verwendet
Activity	{operand|consumtion-operator|usage operator}	wird verwendet
Type	domänspezifische Unterscheidung des Ressourcentyps {software|dokumentation}	wird verwendet; ergänzend definiert

Tabelle 4.3: Eigenschaften von Ressourcen
Quelle: In Anlehnung an (Böttcher und Klingner, 2011)

Bei der Darstellung der Ressourcen ist zu beachten, dass es sich nicht um die Ebene der Planung eines Dienstleistungsbündels eines konkreten Integrationsdienstleisters handelt, sondern ein Modell erstellt werden soll, welches ein Dienstleister instanziieren und verfeinern muss, um es anwenden zu können. Aus diesem Grund wurden Eigenschaften wie *Resource identity*, *Resource quantity* und *Capacity* mit *dienstleisterabhängig* bewertet und stellen Punkte dar, an denen eine individuelle Verfeinerung durch einen Dienstleister ansetzt.

Flexibility wird nicht verwendet, da auf dem hier spezifizierten abstrakten Niveau keine Aussagen über diese Eigenschaft getroffen werden können. Darüber hinaus werden die meisten intangiblen Ressourcen im Rahmen einer Integrationsdienstleistung erstellt. Damit ergeben sich die Informationen darüber, was zur Adaptierung der Ressource notwendig ist, aus der Servicekomponente in welcher die Ressource geschaffen wird.

Bis auf *Resource position* werden die übrigen Eigenschaften im Rahmen des Modells verwendet. *Resource position* stellt einen Sonderfall dar. Ob es sich um einen Input oder einen Output

[20] Vgl. Abschnitt 3.5.

handelt, hängt maßgeblich von der Servicekomponente ab, weshalb diese Informationen in das Modell der Servicekomponenten integriert wurden.[21]

Die restlichen Eigenschaften finden für das Modell in dieser Arbeit Verwendung, wobei *Resource relations* und *Resource hierarchie* der Übersichtlichkeit halber in ein separates Modell der Ressourcenhierarchie ausgelagert werden.[22] Dieses Modell dient dazu, die Ressourcen ebenfalls zu komponentisieren. Damit können Teile der Ressourcen getrennt voneinander bearbeitet werden. Darüber hinaus muss für Dienstleistungen die alle Teilressourcen erfassen nur die – hierarchisch gesehen – höchste Ressource angegeben werden, was die Darstellung vereinfacht.

Ein zusätzliches Konzept, welches in diesem Modell einführt wird, ist das Konzept einer abstrakten Ressource. Die Verwendung von abstrakten Ressourcen ist notwendig um den Charakter eines zu instanziierenden Modells aufrechtzuerhalten. Am Beispiel von Konfigurationsartefakten für eine Middleware wird schnell deutlich, dass die Art des Konfigurationsartefaktes stark von der gewählten Middleware abhängt und hier noch nicht vorgedacht werden kann. Die konkrete Ausprägung ist dann durch den Dienstleister zu spezifizieren, wenn dieser das Modell für sich ausprägt und instanziiert.

Für die Entwicklung von Integrationslösungen werden neben menschlicher Arbeit auch weitere Faktoren eingesetzt. In einer ersten Annäherung an ein Ressourcenmodell sollen im Folgenden die eingesetzten Ressourcentypen in das Modell der Produktionsfaktoren nach Gutenberg eingeordnet werden.[23] Entgegen anderer Betrachtung von IT als Produktionsfaktor soll hier nicht bestimmt werden, inwieweit IT in das Faktorensystem eines produzierenden Unternehmens eingeht, sondern welche Produktionsfaktoren benötigt werden, um Integrationslösungen herzustellen. Menschliche Arbeit – bei der Herstellung von Software in Form des Softwareentwicklers oder des Architekten – ist analog zu anderen Branchen, wie produzierenden Unternehmen – in die Kategorie der Potenzialfaktoren einzuordnen und stellt damit den unstrittigsten Punkt dar. Zur Softwareentwicklung sind Arbeitsplatzrechner (Hardware) zugehörige Betriebssysteme (Software) und verwendete Entwicklungsumgebungen (Software) notwendig. Diese lassen sich im Fall der Hardware den materiellen Betriebsmitteln und im Falle von Software den immateriellen Betriebsmitteln zuordnen. In die Kategorie der immateriellen Betriebsmittel fallen auch Transformatoren oder Generatoren im Sinne der Model-Driven-Architecture (MDA) und alle anderen im Sprachgebrauch der Wiederverwendung als Assets bezeichneten Artefakte. Besteht ein solches Asset aus Quellcode, so kann man vereinfachend annehmen, dass dieser – sofern keine Lizenzgebühren zu entrichten sind – kopiert wird und die Kopie in das Produkt eingeht. Das Potenzial eine neue Kopie zu erzeugen bleibt dabei erhalten. Anders verhält es sich mit Softwarekomponenten, die in das Produkt eingehen. Dies kann zum Beispiel eine Middleware sein, für die eine Lizenz gekauft werden muss. Der Verbrauch der Ressource liegt hier in den Lizenzgebühren begründet, die erneut zu entrichten sind, wenn ein neues Produkt erstellt wird.[24] Die Einordnung von Anwendungssystemen in das System der Produktionsfaktoren nach Gutenberg[25] fasst Tabelle 4.4 zusammen.

Das Ziel der Rationalisierung durch Automatisierung ist das substituieren menschlicher Arbeit durch Potenzialfaktoren in Form von Betriebsmitteln. Im Falle der Integration – das Vorhandensein des materiellen Betriebsmittels Hardware und des immateriellen Betriebsmittels Software vorausgesetzt – handelt es sich um die Schaffung immaterieller Betriebsmittel in Form von

[21] Vgl. Kapitel 4.2.1.
[22] Vgl. Abbildung 4.3(b)
[23] Vgl. (Gutenberg, 1983).
[24] Im Gegensatz zu dieser Argumentation steht Open-Source-Software bei der ein solcher Verbrauch in Form von Lizenzgebühren nicht angenommen werden kann. Diese wäre dann als Potenzialfaktor einzustufen, der aber in das Endprodukt eingeht, sprich an den Kunden ausgeliefert wird. Es muss immer wieder eine neue Kopie erzeugt werden.
[25] Vgl. (Gutenberg, 1983).

Komponente	ist ein	Beispiele, Erläuterung
Menschliche Arbeit	Potentialfaktor	Softwareentwickler, Consultant, IT-Architekt
genutzte Hardware	Betriebsmittel (materiell), Potentialfaktor	Hardware, die zur Erstellung der Software genutzt wird; eine physische Abnutzung ist gegeben Festplatten beinhalten bspw. bewegliche Teile die verschleißen. Auch die Anzahl der Lese-Schreibzyklen für Solid-State-Disks ist begrenzt.
genutzte Software	Betriebsmittel (immateriell), Potentialfaktor	Software, die zur Erstellung der Software genutzt wird z.B. Entwicklungsumgebungen,Generatoren
eingehende IT	Werkstoff, Rohstoff	Software und Hardware die in das Endprodukt eingeht z.B. Bibliotheken, eingekaufte Komponenten

Tabelle 4.4: Produktionsfaktoren der Integration

Modellen und Mustern. Weiterführend sind Metamodelle, geeignete Editoren und Generatoren im Sinne des Model-Driven Integration Engineering (MDIE) eine analoge Rationalisierung. Unter Verwendung der erzeugten Editoren können Modellinstanzen der Metamodelle erstellt werden, mit deren Hilfe Input für die Generatoren entsteht, die diesen als Konfigurationsinformationen verwenden, um den gewünschten Output zu erzeugen. Während Modelle und Muster durch Menschen in Anwendung gebracht werden erfolgt im MDIE nur die Modellierung durch den Menschen. Die so erhobenen Konfigurationsinformationen werden durch die Generatoren in Anwendung gebracht.

Ressource	Source Type	Tangibility	Mobility[26]	Activity
Unternehmensarchitektur (UA)	customer	intangible	mobile	operand/ usage operator
Umsetzungsplan	provider	intangible	mobile	operand/ usage operator
Bebauungsplan	customer	intangible	mobile	operand/ usage operator
Prozesslandkarte	customer	intangible	mobile	operand/ usage operator
Anwendungsfälle	customer	intangible	mobile	operand/ usage operator
Anforderungen	customer	intangible	mobile	operand/ usage operator
Prozesslaufschema	customer	intangible	mobile	operand/ usage operator
Interaktionsmodell	provider	intangible	mobile	operand/ usage operator
Potentialanalyse	provider	intangible	mobile	operand/ usage operator
Systemschnittstellen	customer	intangible	mobile	operand/ usage operator
Systeme	customer	teilw. tangible	teilw. immobile	operand/ usage operator
Kommunikations-dokumentation	customer	intangible	mobile	operand/ usage operator
Mappings	provider	intangible	mobile	operand/ usage operator
Referenzarchitekturen	provider	intangible	mobile	operand/ usage operator
Prototyp	provider	teilw. tangible	teilw. immobile	operand/ usage operator
Prozesse	customer	intangible	mobile	operand/ usage operator
Personaleinsatzplan	provider	intangible	mobile	operand/ usage operator
Projektplan	provider	intangible	mobile	operand/ usage operator
Angebot	provider	intangible	mobile	operand/ usage operator
Lösungskonzept	provider	intangible	mobile	operand/ usage operator
Lösung	provider	teilw. tangible	mobile	operand/ usage operator
Rollout	provider	intangible	mobile	operand/ usage operator

Tabelle 4.5: Übersicht Ressourcen der Integration vom Typ Dokument/Modell

Der *Source Type* einer Ressource lässt sich im vorliegenden Fall nicht immer eindeutig bestimmen. Insbesondere bei nicht-anfassbaren Ressourcen, die nur von Dienstleister und Kunde gemeinsam erstellt werden können, ist der Source Type nicht eindeutig feststellbar. Aus diesem Grund wurden alle Ressourcen, die bereits vor der Beauftragung eines Dienstleisters vorliegen könnten,

[26] Mobility bezieht sich grundsätzlich nur auf die Dokumentation nicht auf das Dokumentierte bspw. im Fall der Unternehmensarchitektur.

dem Kunden zugeordnet und alle Ressourcen die eindeutig als Ergebnis oder Zwischenergebnis entstehen dem Dienstleister zugeordnet. Beispiele sind die UA, welche auch vor der Beauftragung des Dienstleisters existiert und damit *Source Type = customer* erhält. Der *Umsetzungsplan* ist jedoch eindeutig als Zwischenergebnis zu werten und wird deshalb mit *provider* belegt.

Die Eigenschaft der *Tangibility* besitzt im Fall von Integrationsdienstleistungen nur wenig Aussagekraft. Nahezu alle Ressourcen sind *intangible*, womit sie sich anhand dieser Eigenschaft nur schwerlich unterscheiden lassen. Die mit *teilw. tangible* belegten Ressourcen verdeutlichen, dass für die Integration auch Hardware verwendet wird, welche natürlich anfassbar[27] ist. Die hierarchische Zusammensetzung zeigt, dass Hardware nicht der einzige Bestandteil der entsprechenden Ressourcen ist, was durch teilweise zum Ausdruck gebracht wird.

Mobility besitzt eine ähnliche Unterscheidungskraft wie *Tangibility*. Die Einstufung für fast alle Ressourcen der Integration weist diese als mobil aus. Es handelt sich bei vielen Ressourcen um Dokumente oder Modelle, was diese Eigenschaft erklärt. Als *teilweise immobile* werden Ressourcen gekennzeichnet, die eine Kombination von Hard- und Software darstellen. Der Hardwareteil ist nur begrenzt als *mobile* einzustufen.

Activity charakterisiert, wie eine Ressource im Rahmen einer Integration verwendet wird. Die Ausprägungen *operand*, *consumtion-operator* und *usage-operator* lassen sich begrenzt den Produktionsfaktoren nach Gutenberg zuordnen.[28] Ein *consumtion-operator* wird verbraucht und entspricht deshalb einem Werk- bzw. Rohstoff.

Die Einordnung der Ressourcen in den Tabellen 4.5, 4.6 und 4.7 zeigt, dass diese Eigenschaft bei der Integration nur auf wenige Ressourcen zutrifft. Dabei handelt es sich um alle Hardware und um die Integrationshilfsmittel, welche in die Lösung eingehen und beim Kunden installiert werden. Im Fall der Software drückt sich der Verbrauch in Form von zu entrichtenden Lizenzgebühren oder der Notwendigkeit Kopien zu erzeugen aus. Die meisten Ressourcen sind jedoch in die Gruppe usage-operator einzuordnen, da sich z. B. Dokumente oder Modelle nicht verbrauchen. Gleichzeitig sind diese aber auch Ressourcen der Gruppe operand, denn in jedem Schritt, in dem ein Dokument oder Modell ergänzt und weiter detailliert wird, ändert es auch seinen Zustand. Im Detail werden diese Veränderungen im Modell der Servicekomponenten deutlich, da diese Eigenschaften von der Servicekomponente abhängen.

Bei *Type* handelt es sich um ein zusätzlich eingeführtes Attribut. Alle Ressourcen der Integration lassen sich als mehr oder weniger *intangible* einstufen, weshalb von dieser Aussage wenig Aussagekraft ausgeht, außer dass es sich erwiesener Maßen bei der Integration um eine Dienstleistung handelt. Es lassen sich aber verschiedene Arten von intangiblen Ressourcen unterscheiden. Das Kriterium ist dabei der Grad der werkzeuggestützten und automatischen Weiterverarbeitbarkeit, welcher ein Indiz für den Grad der Automatisierung darstellt. Handelt es sich um ein Dokument, welches in keiner standardisierten Form vorliegt so ist diese von einem Modell welches einem wohldefinierten Metamodell genügt zu unterscheiden. Aus diesem Grund wurde zusätzlich *Type* eingeführt, um die Ressourcen weiter zu systematisieren. Diese Eigenschaft wird nicht durch eine separate Spalte ausgedrückt, vielmehr wurden die Ressourcen bereits in Gruppen der jeweiligen Typen zusammengefasst. Tabelle 4.5 zeigt die Gruppe der Dokumente und Modelle, die Tabelle 4.6 die Gruppe der Humanressourcen und die Tabelle 4.7 die Gruppe der Werkzeuge und Hilfsmittel, also der Software.

Integration ist eine personalintensive Dienstleistung. Aus den Ausführungen zu den Rollen auf Dienstleister und Kundenseite bei Gebauer und Stefan[29] lässt sich ein Modell der *Humanressourcen*

[27] engl. tangible
[28] Vgl. Tabelle 4.4.
[29] Vgl. (Gebauer und Stefan, 2011b).

für die Integration erstellen. Die Tabelle 4.6 listet die genannten Rollen auf. Dabei handelt es sich um Rollen, welche durch den Integrationsdienstleister mit Personen belegt werden müssen. Erst durch diese Zuordnung von Menschen zu den Rollen wird eine Abschätzung der in einem bestimmten Zeitraum bewältigbaren Dienstleistungen möglich.

Ressource	**Source Type**	**Tangibility**	**Mobility**	**Activity**
Vertrieb	provider	human	mobile	usage operator
Berater	provider	human	mobile	usage operator
Entwickler	provider u./o. customer	human	mobile	usage operator
Projektmanager	provider	human	mobile	usage operator
Projektleiter	provider u. customer	human	mobile	usage operator
Roll Out	provider	human	mobile	usage operator
Service und Support	provider	human	mobile	usage operator
Architekt	provider u./o. customer	human	mobile	usage operator
Produktmanager	provider	human	mobile	usage operator
Qualitätsmanagement, Tester	provider	human	mobile	usage operator
Change Management	provider	human	mobile	usage operator
Schulung	provider	human	mobile	usage operator
Bereichsleitung	provider	human	mobile	usage operator
Releasemanagement	provider	human	mobile	usage operator
Betrieb	provider	human	mobile	usage operator
Anwender	customer	human	mobile	usage operator
Prozessverantwortlicher	customer	human	mobile	usage operator
fachlicher Chefdesigner	customer	human	mobile	usage operator
technischer Projektleiter	customer	human	mobile	usage operator
Datenarchitekt	customer	human	mobile	usage operator
IT-Abteilung	customer	human	mobile	usage operator
Betriebsabteilung	customer	human	mobile	usage operator
Toolspezialist	customer	human	mobile	usage operator
Plattformverantwortlicher	customer	human	mobile	usage operator
Geschäftsleitung	customer	human	mobile	usage operator
Controller	customer	human	mobile	usage operator
Lenkungsausschuss	customer	human	mobile	usage operator
Business Architekt	customer	human	mobile	usage operator
Teamleiter	customer	human	mobile	usage operator
Betriebsorganisation	customer	human	mobile	usage operator
Compliance	customer	human	mobile	usage operator

Tabelle 4.6: Übersicht Ressourcen der Integration vom Typ Humanressource

Teilweise handelt es sich auch um die Nennung von ganzen Abteilungen, welche dann verschiedene Rollen subsummieren. Darüberhinaus ist anzumerken, dass je nach Größe des Dienstleisters oder Kunden die Ausübung der Rollen auch in Personalunion stattfinden kann. Die Darstellung der Rollen basiert auf den Erkenntnissen der empirischen Untersuchung und erhebt keinen Anspruch auf Vollständigkeit. Im Fall konkreter Dienstleister kann auch eine feinere Granularität der Rollen gegeben sein. Bei einigen Rollen kann es vorkommen dass sie sowohl durch den Dienstleister (provider), als auch den Kunden (customer) zu besetzen sind. Die Tangibility hat im Falle dieses Ressourcentyps eine geringe Aussagekraft, da es sich ausschließlich um Humanressourcen handelt. Gleiches gilt für Mobility, da Humanressourcen prinzipiell als mobil einzustufen sind. Weiterhin ist für Activity zum einen festzuhalten, dass alle Ressourcen mit usage-operator bewertet wurden, zum anderen dies aber die Bedeutung der Humanressourcen untermauert. Menschen arbeiten über einen längeren Zeitraum im Unternehmen und können nicht unbegrenzt viele Technologien, Hilfsmittel und Werkzeuge beherrschen. Daher sollten Investitionen in die Ausbildung mit Bedacht erfolgen um Lock-In-Effekte zu vermeiden. Diese Überlegung gilt natürlich für alle Potenzialfaktoren im Sinne Gutenbergs.

Bis auf die Ressource Hilfsmittel stellen alle anderen – in Tabelle 4.7 vorgestellten – Ressourcen solche vom Typ Werkzeug dar. Der Source Type von Hilfsmittel wurde mit allen drei möglichen Attributausprägungen bewertet, da es sich zum einen um einen neu zu etablierendes Hilfsmittel

Ressource	Source Type	Tangibility	Mobility	Activity
Hilfsmittel	provider/ customer/ external	intangible	mobile/ immobile	consumption-operator
Modellierungswerkzeug	provider/ external	intangible	mobile	usage-operator
Entwicklungsumgebung	provider/ external	intangible	mobile	usage-operator
Repository, Versionsverwaltung	provider/ external	intangible	mobile	usage-operator
Buildwerkzeug	provider/ external	intangible	mobile	usage-operator
Generator	provider/ external	intangible	mobile	usage-operator
Compiler	provider/ external	intangible	mobile	usage-operator
Mapping/ETL	provider/ external	intangible	mobile	usage-operator
Analyse / DSS	provider/ external	intangible	mobile	usage-operator
Testwerkzeug	provider/ external	intangible	mobile	usage-operator

Tabelle 4.7: Ressourcen der Integration vom Typ Hilfsmittel oder Werkzeug

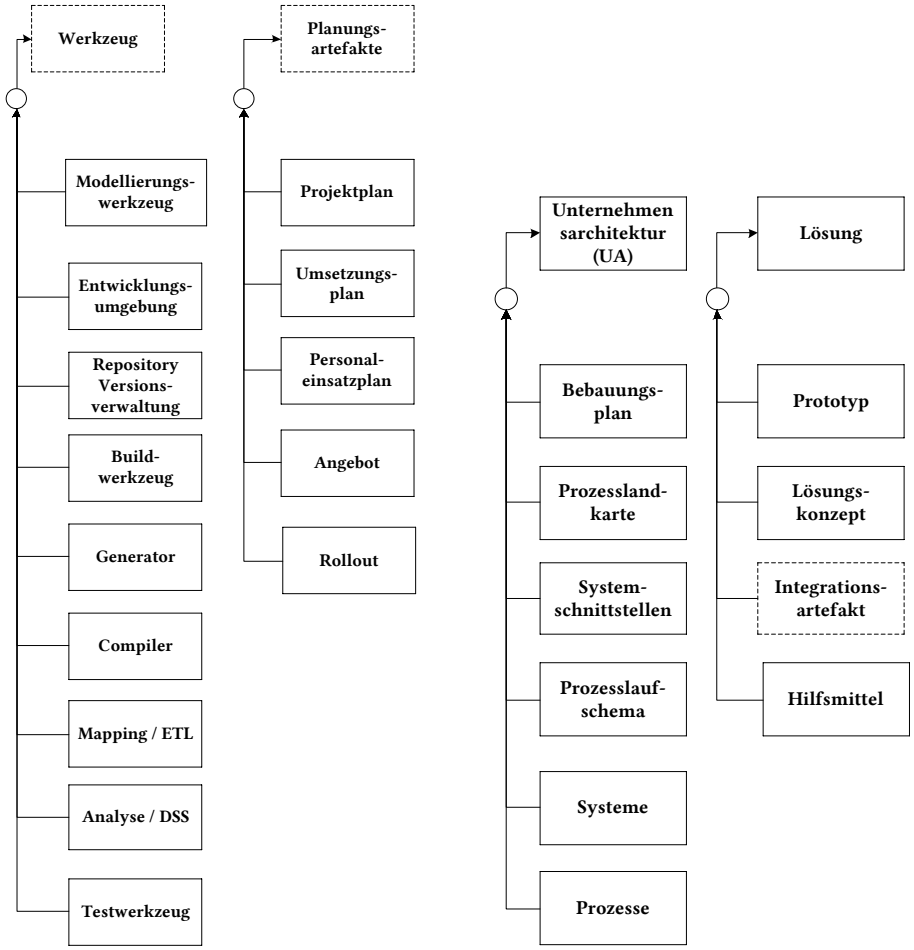
handeln kann (provider), zum anderen ein Hilfsmittel bereits beim Kunden im Einsatz sein kann (customer) oder es sich um eine extern eingekaufte Dienstleistung handeln kann. Der letztere Fall reflektiert Cloud-Angebote, welche in letzter Zeit an Bedeutung gewinnen. Dadurch erklärt auch die Bewertung von Mobility, welche auch den Fall immobile einschließt, was bei einem bereits beim Kunden etablierten Hilfsmittel der Fall wäre. Da es sich bei allen Ressourcen um Software handelt, folgt die Bewertung von Tangibility mit intangible. Für Activity ergibt sich ebenfalls eine Unterscheidung zwischen Hilfsmitteln und Werkzeugen. Für Hilfsmittel wird Activity mit consumption-operator bewertet. Wie bereits bei der Einordnung in das Produktionsfaktorensystem nach Gutenberg verdeutlicht wurde, verbrauchen sich Hilfsmittel, da bei jedem neuen Einsatz einer Software gegebenenfalls Lizenzkosten zu entrichten sind. Auch für den Fall von Open Source Hilfsmitteln wird dies angenommen, da jedes Mal eine neue Kopie zu erzeugen und zu installieren ist. Für Werkzeuge sind zwar auch Lizenzkosten zu entrichten, eine Mehrfachverwendung ist jedoch gegeben, was in der Bewertung usage-operator resultiert und damit einem Potenzialfaktor gleichkommt.

Hierarchie und abstrakte Ressourcen Im Gegensatz zu hierarchischen Beziehungen werden abstrakte Ressourcen bei Böttcher nicht diskutiert. An dieser Stelle wird dieses Konzept eingeführt, um zum einen Gruppen von Ressourcen fassen zu können und zum anderen mit Platzhaltern für konkrete Ressourcen arbeiten zu können. Dies ist notwendig um den Einsatz von Ressourcen –auf dem abstrakten Niveau eines Modells– beschreiben zu können, welche nur der Dienstleister in ihrer individuellen Ausprägung exakt beschreiben kann. Ein zweiter Aspekt ist die Vereinfachung der Beschreibung des Einsatzes von Ressourcen. Häufig sind Gruppen von Ressourcen gemeinsam von einem bestimmten Sachverhalt betroffen. Die Verwendung einer (abstrakten) Ressource – als Platzhalter für alle Ressourcen der Gruppe – vereinfacht die Darstellung im Modell und führt zu einer besseren Verständlichkeit.

Als Beispiel für hierarchische Beziehungen kann die Ressource *Unternehmensarchitektur* dienen. Es handelt sich dabei streng genommen um ein Modell, welches sich aus mehreren Bestandteilen zusammensetzt. Die Bestandteile repräsentieren die Ebenen einer Unternehmensarchitektur beziehungsweise die Bestandteile eines betrieblichen Informationssystems.[30] Die hierarchischen Beziehungen können analog zu dem Produktmodell gesehen werden. Sie stellen die Zusammensetzung einer Ressource aus Ressourcenkomponenten dar. Durch diese Zerlegung können Ressourcenkomponenten in separaten Schritten erarbeitet beziehungsweise eingesetzt werden. In der Abbildung 4.3(b) sind keine Kardinalitäten angetragen, welche kennzeichnen, in welcher Menge eine Ressourcenkomponente in eine andere eingeht. Dabei könnte es sich lediglich um Platzhalter handeln, da es sich bei Mengenangaben um dienstleisterspezifische Informationen

[30] Vgl. Abschnitt 2.1.

handelt. Damit liegt ein weiterer Punkt vor, welcher Gegenstand der Instanziierung des Modells ist.



(a) abstrakte Ressourcen (b) hierarchische Beziehungen

Abbildung 4.3: Abstrakte Ressourcen und hierarchische Beziehungen

Die Ressource *Integrationsartefakte* ist ein Beispiel für eine abstrakte Ressource. Integrationsartefakte können Codeteile oder Konfigurationsdateien sein, welche jedoch stark vom eingesetzten Hilfsmittel abhängen. Die genaue Zusammensetzung ist von der konkreten Integrationssituation abhängig. Verfügen alle Integrationsobjekte über entsprechende Schnittstellen (z. B. Web Services) und wird eine geeignete Middleware als Hilfsmittel verwendet zum Beispiel ein JBoss ESB, so besteht die Menge der Integrationsartefakte aus allen ESB-Komponenten, welche in dem Hilfsmittel eingerichtet werden müssen. Um im Rahmen dieses Modells dennoch den Einsatz der Ressource Integrationsartefakte beschreiben zu können, wurde das Konzept abstrakter Ressourcen gewählt, welche durch den Integrationsdienstleister mit einer konkreten Ausprägung hinterlegt werden müssen. Die weiteren abstrakten Ressourcen sind in Abbildung 4.3(a) dargestellt. Lediglich die abstrakte Ressource Humanressource wurde ausgespart, da für die Darstellung der verfügbare Platz nicht ausreicht.

Abhängigkeiten

Am Beispiel der *Integrationsartefakte* kann auch die wechselseitige Beeinflussung der Ressourcen dargelegt werden. Wird als Ausprägung für *Hilfsmittel* ein JBoss ESB gewählt, so beeinflusst dies a) die notwendigen Qualifikationen der Entwickler, b) die konkrete Ausprägung der *Inte-*

grationsartefakte und c) die realisierbaren *Lösungen*. Diese Abhängigkeiten sind durchaus als wechselseitig oder auch bijektiv zu sehen. Die Richtung hängt vom Integrationsdienstleister und der als gesetzt geltenden Ressource ab. In Abbildung 4.4 werden diese Zusammenhänge näher dargestellt, dabei werden nur Ressourcentypen verwendet, um die Abbildung übersichtlicher zu gestalten. Die Abbildung zeigt, dass Humanressourcen eine zentrale Stellung einnehmen. Sie

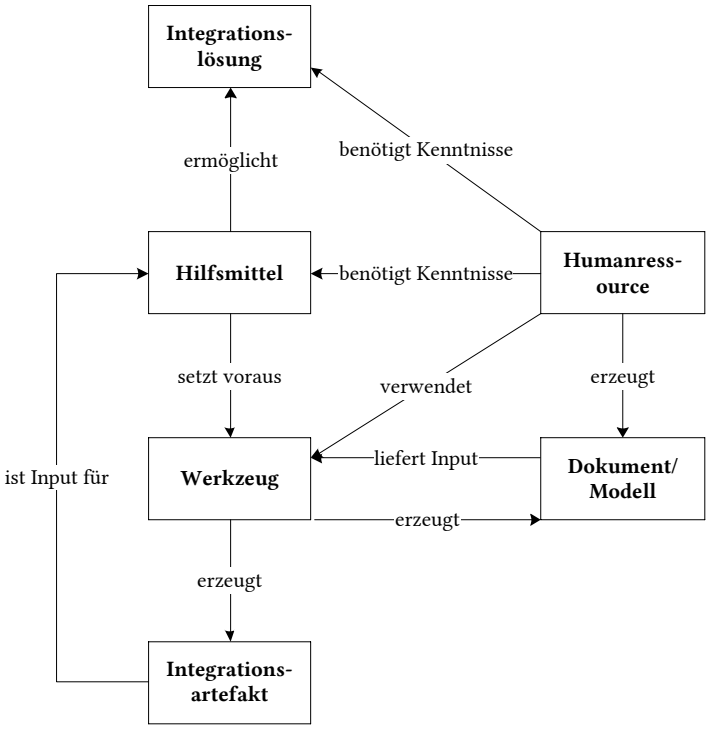


Abbildung 4.4: Abhängigkeiten zwischen Ressourcentypen der Integration

benötigen Kenntnisse über die zu erstellende Integrationslösung, die dafür geplanten Hilfsmittel sowie Kenntnisse über die bei der Erstellung verwendeten Werkzeuge. Des weiteren erzeugen sie – gegebenenfalls unter Verwendung entsprechender Werkzeuge – Dokumente beziehungsweise Modelle, welche wiederum Input für Werkzeuge sein können. Mithilfe von Werkzeugen werden Integrationsartefakte erstellt, die Input für Hilfsmittel darstellen. Damit sind insbesondere Code- oder Konfigurationsartefakte gemeint, welche notwendig sind, um mit einem Hilfsmittel eine vollständige Lösung zu realisieren.

Aus diesen unmittelbaren Abhängigkeiten resultiert die Tatsache, dass für einen Integrationsdienstleister Hilfsmittel und Werkzeuge nur in Kombination mit entsprechenden Humanressourcen einsetzbar sind und dies die realisierbaren Integrationslösungen bestimmt. Es handelt sich um gekoppelte Ressourcen, deren Wert – alleine eingesetzt – geringer ist, als in Kombination. Aus den realisierbaren Integrationslösungen leitet sich wiederum zu einem gewissen Grad die Attraktivität des Dienstleisters am Markt ab.

Je nach Knappheit der Ressource oder der Höhe bereits getätigter Investitionen, kann ein Hilfsmittel oder einer Humanressource als gesetzt eingestuft werden. Sind zum Beispiel Entwickler mit bestimmten Kenntnissen am Markt besonders preiswert verfügbar, so kann sich die Investition in ein neues Hilfsmittel lohnen, da die Personalkosten diese Investition wieder aufwiegen. Auf der anderen Seite kann sich die Ausbildung neuer Kenntnisse rechnen, wenn entsprechende Entwickler sehr teuer sind, die Kenntnisse aber zwingend benötigt werden. Als Beispiel kann der Mainframe dienen, welcher in den meisten Unternehmen nicht ohne Weiteres ersetzt werden

kann. Deshalb sind Unternehmen dazu übergegangen, Entwickler mit diesen Qualifikationen neu auszubilden. Dieses Beispiel verdeutlicht die Komplexität der Ressourcenoptimierung aufseiten eines Integrationsdienstleisters. Im Fall von Humanressourcen ist der Optimierungsspielraum durch geltende Arbeitszeitregelungen im Bereich der qualitativen Kapazität der Ressource zu sehen, da die Obergrenzen der Gesamtverfügbarkeit weitgehend als fest einzustufen sind. Im Kern geht es um die Gestaltung der Produktionsfunktion[31] der Integrationsdienstleistung, welche beschreibt, welcher Produktionsfaktor in welcher Menge einzusetzen ist, um eine Outputmenge X zu erzeugen.

4.2.3 Das Produktmodell der Integration

Das Produktmodell (Abbildung 4.5) stellt dar, inwieweit sich eine Dienstleistung aus Subkomponenten zusammensetzt und in welcher Kardinalität die Subkomponenten benötigt werden. Ziel dieser Darstellung ist es, die Anpassung von Dienstleistungen an den jeweiligen Kunden durch Konfiguration zu ermöglichen. Neben Verbindungsknoten werden zusätzlich noch Kardinalitäten verwendet. Die Verbindungsknoten können mit einer Aussage (UND, Oder, Exclusives Oder) belegt werden.[32] Im Gegensatz zu der Darstellung der Kardinalitäten an den Verbindungsknoten wird eine alternative Darstellung verwendet, welche bereits früher durch Böttcher diskutiert wurde.[33]

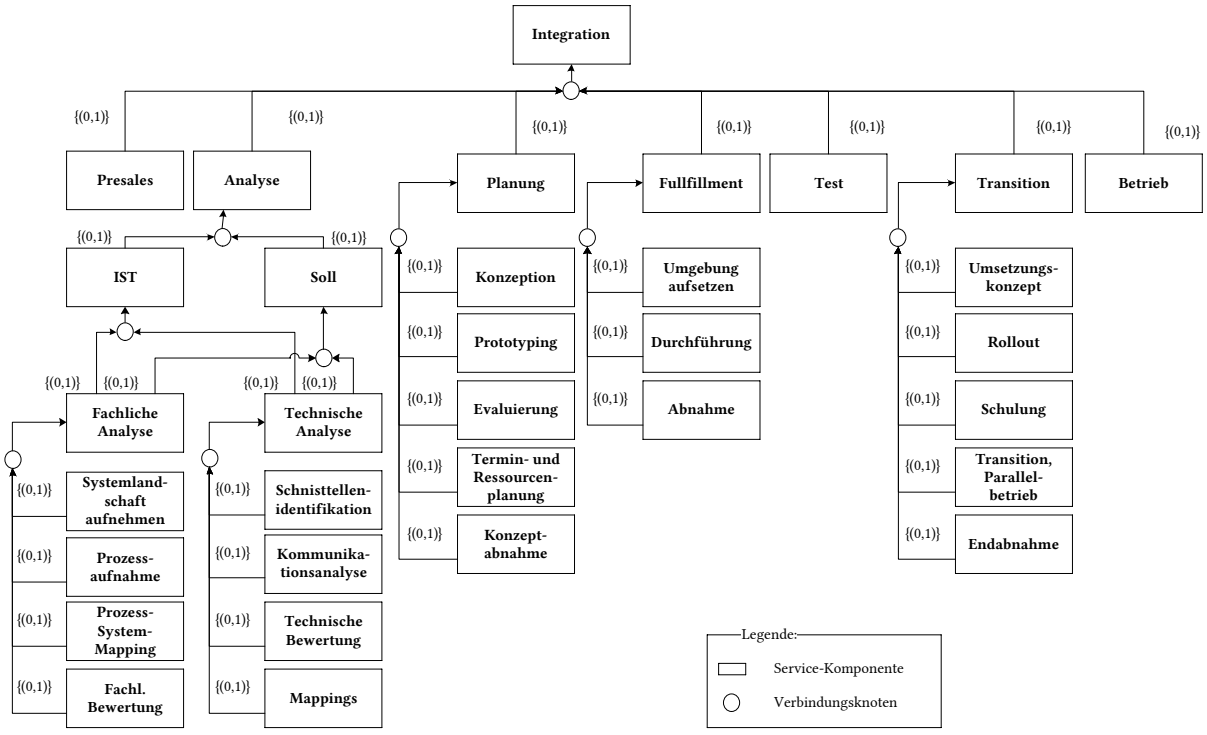


Abbildung 4.5: Produktmodell der Integration

Auffällig ist die nicht vorhandene Detaillierung der Test- und der Betriebskomponente. Die Ursachen dafür ergeben sich aus der empirischen Untersuchung und bestätigen das Testen als

[31] Vgl. (Hennies, 2003, S. 90).
[32] Vgl. (Böttcher und Klingner, 2011).
[33] Vgl. (ebenda).

Problempunkt der Integration[34]. Es handelt sich um eine Unterspezifizierung des Modells und einen Punkt, der zukünftige Forschung erfordert. Die Betriebsphase ist ebenfalls unterspezifiziert, was im Geschäftsmodell der befragten Dienstleister begründet liegt. Die Lösungen werden zumeist durch den Kunden selbst betrieben und nicht durch den Dienstleister, weshalb Letzterer zu dieser Komponente kaum Aussagen machen kann. Das vermehrte Angebot von Cloud-Computing-Diensten wird dazu führen, dass auch die Integration in und durch die Cloud erfolgen muss. Diese Tendenz wird weiteren Forschungsbedarf im Bereich der Betriebskomponente nach sich ziehen, da diese im Fall einer Cloud-Integrationsleistung durch den Dienstleister zu erbringen ist.

4.2.4 Das Prozessmodell der Integration

Das Prozessmodell der Integration stellt die temporalen Abhängigkeiten der einzelnen Servicekomponenten voneinander dar. Auf der Basis der Ergebnisse der empirischen Untersuchung lässt sich kein eindeutiges Bild des Prozessmodells konstruieren. Es wurden verschiedene temporale Abhängigkeiten zwischen den Servicekomponenten geäußert. Auf der obersten Abstraktionsebene gab es neben einer linearen Verkettung der Phasen auch Ansichten, welche Rückschritte zwischen den Servicekomponenten vorsehen. In der Softwareentwicklung sind diese Arten der Verkettung im Rahmen von Vorgehensmodellen diskutiert worden. Eine weitere Möglichkeit, die sich aus der empirischen Untersuchung ableiten lässt, ist eine lineare Verkettung der Komponenten Pre-Sales, Analyse, und Planung und eine iterative Verkettung der Unterkomponenten von Fullfillment und Test. Dabei wird das Gesamtproblem in einzelne Teile zerlegt und die Servicekomponenten wiederholt – für jedes Teilproblem erneut – erbracht. Diese Art der temporalen Verbindung der Servicekomponenten findet sich auch in dem Vorgehensmodell zum Integration Engineering nach Thränert.[35] Je nach Dienstleister, Kunde und Projekt können die Komponenten Transition und Betrieb erst nach Abschluss aller Iterationen durchgeführt werden oder sich ebenfalls in die iterative Ausführung einreihen. Dies ist dann der Fall, wenn eine Lösung auch schrittweise in Betrieb genommen wird. Für ein Teilproblem gilt jedoch, dass Transition und Betrieb erst nach Abschluss der Testkomponente ausgeführt werden können. Dieser Teil des Integrationsdienstleistungsmodells kann auf der Basis der vorliegenden Ergebnisse nur rudimentär dargestellt werden. Eine weitere Detaillierung ist Gegenstand weiterer Forschungsbemühungen.[36]

4.3 Das GAP-Modell der Integration

Obwohl es im Bereich der Systemintegration bisher eine Vielzahl von Ansätzen gegeben hat, die sich in unterschiedlichem Ausmaß der Problemstellung heterogener betrieblicher Informationssysteme und ihrem Zusammenspiel widmeten, sind Integrationsprojekte nicht risikofrei. Integrationsprobleme sind für einen wesentlichen Kostenanteil an IT-Projekten verantwortlich. Wesentliche Probleme scheinen immer noch ungelöst. In diesem Abschnitt werden auf der Basis vorhandener Literatur die Probleme der Integration untersucht und daraus systematisch ein GAP-Modell erstellt. Auf Basis dieses GAP-Modells lässt sich der Beitrag von Lösungen zu aktuellen Forschungsproblemen bemessen, des weiteren ist ein Vergleich mit der Studie möglich.[37] Tabelle 4.8 listet die analysierte Literatur auf und gibt eine Übersicht über die Methode, das Ziel und die Ebene der enthaltenen Untersuchung. Obwohl analysiert, wurde die Arbeit von Cardoso; Aalst u. a. nicht in die Übersicht aufgenommen und somit nicht in das GAP-Modell integriert. Die

[34] Vgl. die geringen Äußerungen zur Häufigkeit von Werkzeugtypen in Abbildung 3.14, sowie zur fehlenden Funktionalität in Abbildung 3.15 und die Auswahlkriterien für eine Integrationslösung in Abbildung 3.18.
[35] Vgl. (Thränert, 2008).
[36] Vgl. Kapitel 8
[37] Vgl. Kapitel 3.

Arbeit gibt eine Zusammenfassung eines Expertenpanels, welches im Rahmen einer Konferenz durchgeführt wurde. Dabei sind vor allem Trends enthalten, welche eine Neuabstimmung von Forschungsagenden animieren sollen und weniger Probleme der Integrationspraxis.[38]

Quelle	Methode; Ziel	Abstraktionsebene
(Sumner, 2000)	sieben Fallstudien; Risikofaktoren	Enterprise wide Information Management Systems
(Scott und Vessey, 2002)	zwei Fallstudien; Risikofaktoren	Enterprise Systems
(Lublinsky und Farell, 2002)	n.a.; Ursachen für das Scheitern	EAI
(Trotta, 2003)	Webinar Session mit Experten; Fallen	EAI
(Lam und Shankararaman, 2004)		EAI
(Lam, 2005)	Literaturanalyse, Fallstudie; kritische Erfolgsfaktoren	EAI
(Vogler, 2006)	literaturanalyse, Praxisprojekte; Probleme der Integration	Prozess- und Systemintegration
(Schwinn und Winter, 2007)	Literaturanalyse, Erfolgsfaktoren, Performanzkriterien	EAI
(Purao; Paul und Smith, 2007)	Fokusgruppeninterviews; Risikofaktoren	Enterprise Integration
(Molina u. a., 2007)	Expertenzirkel [39]	Enterprise Integration and Networking
(Panetto und Molina, 2008)	n.a. ; collection of trends and issues	Manufacturing Systems
(Moll und Ammerlaan, 2008)	63 Fallstudien; identifizieren von Fallstricken	technische und administrative Integration
(Bajgoric und Moon, 2009)	Literaturquellen;Fortführung der geschäftlichen Tätigkeit bei Teilsystemausfall	Informationssystem
(Schmidt; Otto und Österle, 2010)	Literaturanalyse, Fallstudie; Integrationsproblem Klassifikation auf Basis von Integrationsobjekten	Information Systems

Tabelle 4.8: Quellen für das GAP Modell

Auf der Basis der analysierten Arbeiten lassen sich Gruppen von Problemen identifizieren, welche eingehend beschrieben und im Anschluss den Lücken des GAP Modells zugeordnet werden. Neben der Gruppe Soft Skill sind dies Architektur, Business-IT-Alignment, die Projektorganisation, der Grad des Management Support, Kommunikation, Strategie, die Lösungsentwicklung sowie die Anforderungen.

Die Äußerungen zu den **Soft Skills** lassen sich zu vier wesentlichen Punkten zusammenfassen. Einer davon ist, dass für die Herausbildung dieser Qualifikationen Training und systematische Vermittlung von Wissen und Fähigkeiten zu kurz kommen[40]insbesondere auch auf der Seite der späteren Nutzer einer Integrationslösung[41]. Diese Ansichten finden sich auch im Zusammenhang mit ERP-Systemen bei anderen Autoren.[42] Grundsätzlich wird fehlendes Wissen und fehlende Fähigkeiten (EAI Skills) beklagt.[43] Dabei sind insbesondere neue Technologien unbekannt[44], auch wird die Wichtigkeit von applikationsspezifischem Wissen unterschätzt[45]. Die Zusammenstellung des Teams aus internen und externen Beteiligten stellt besonders dann ein Problem dar, wenn

[38] Vgl. (Cardoso; Aalst u. a., 2008).

[39] Mit Expertenzirkel ist bei dieser Quelle das Technical Committee 5.3 – Enterprise Integration and Networking – der International Federation of Automatic Control (IFAC) gemeint. Vgl. (IFAC TC 5.3 International Federation of Automatic Control, 2012).

[40] (Sumner, 2000, S. 184), (Scott und Vessey, 2002, S. 75)

[41] Vgl. (Sumner, 2000).

[42] Vgl. (Al-Mashari; Al-Mudimigh und Zairi, 2003, S. 359), (Brown, 2007, S. 15ff.), (Dey; Clegg und Bennett, 2010, S. 569ff.).

[43] Vgl. (Scott und Vessey, 2002, S. 75), (Trotta, 2003), sowie (Umapathy; Purao und Barton, 2008, S. 519) und (Moll und Ammerlaan, 2008, S. 334).

[44] Vgl. (Scott und Vessey, 2002, S. S. 75).

[45] Vgl. (Sumner, 2000, S. 186).

die interne Expertise unzureichend ist.[46] Dabei kommt auch der frühzeitige Einsatz von Beratern häufig zu kurz.[47] Dadurch entsteht Unkenntnis der potenziellen Lösungsvarianten und der Konsequenzen der Integrationsentscheidungen.[48]

In der Gruppe der Probleme oder Risiken der **Architektur** wird zunächst eine allgemein unpassende Architektur bemängelt.[49] Häufig ist die bestehende Architektur – wobei hier die gesamte Unternehmensarchitektur gemeint ist – schlecht dokumentiert oder das Dokumentenmanagement ist unzureichend, was fehlende Reviews erstellter Dokumente einschließt.[50] Problematisch sind fehlende Datenintegration und -standardisierung, sowie das Ausmaß der Architektur, welche im Zweifel bei der Datenintegration auf das ganze Unternehmen ausgerichtet werden sollte.[51] Weitere Probleme auf der Architekturebene sind eine zu feingranulare Planung der Kommunikation, welche zu einer schlechten Gesamtperformance führen kann[52], Ineffizienzen, Redundanzen und Lücken in der Architektur.[53] Ebenfalls Redundanzen, sowie einen suboptimalen Integrationsgrad und fehlende Kenntnis der Integrationsbeziehungen merkt Vogler an.[54] Letzteres lässt sich auf die eingangs erwähnte schlechte Dokumentation zurückführen.

Die Abstimmung zwischen der geschäftlichen Tätigkeit und der IT eines Unternehmens macht sich in Form der Problemgruppe **Business-IT-Alignment** bemerkbar. Scott und Vessy verweisen auf eine schwache Organisationsstruktur und ein schlechtes Verständnis der Unternehmenskultur als Ursachen.[55] Neben nicht zur Organisationsstruktur passenden technischen Berechtigungen, welche mehrere Anwendungssysteme überspannen[56], ist häufig die Notwendigkeit der Integration aus geschäftlicher Sicht nicht klar[57], weswegen eine Wirtschaftlichkeitsberechnung empfohlen wird.[58] Eine fehlerhafte Ausrichtung auf die geschäftliche Tätigkeit kommt dadurch zum Ausdruck, dass die Prozessintegration die Geschäftsprozesse nicht reflektiert.[59] Strittig ist, ob die Prozesse angepasst werden sollten[60] oder müssen die Anwendungen – die Integrationsobjekte – zu den geschäftlichen Erfordernissen passend ausgewählt werden, was nicht immer der Fall ist.[61] Im schlechtesten Fall entsteht eine Entkopplung des organisationalen Systems vom technischen Teil des Informationssystems, wodurch die Koordinationsnotwendigkeit in Form des Business-IT-Alignments begründet ist.[62] Weiterhin nützt es nichts nur die IT-Systeme zu verbinden, wenn die Interoperabilität auf der Ebene der Geschäftsprozesse mit den Geschäftspartnern (B2B Integration) nicht gegeben ist.[63]

Da Integrationsprojekte Beteiligte verschiedener Bereiche und häufig auch verschiedener Unternehmen umfassen, kommt der **Projektorganisation** eine besondere Bedeutung zu. Obwohl sich ähnliche Probleme auch bei anderen Projekten finden, werden hier nur die, in der analysierten Literatur geäußerten, Probleme wiedergegeben, um auch diese in das Gap-Modell

[46] Vgl. (ebenda, S. 184f.).
[47] Vgl. (Scott und Vessey, 2002, S. 75).
[48] Vgl. (Vogler, 2006, S. 24).
[49] Vgl. (Lam und Shankararaman, 2004, S. 41).
[50] Vgl. (Moll und Ammerlaan, 2008, S. 334).
[51] Vgl. (Sumner, 2000, S. 185f.).
[52] Vgl. (Lublinsky und Farell, 2002, S. 42).
[53] Vgl. (Lam und Shankararaman, 2004, S. 41).
[54] Vgl. (Vogler, 2006, S. 24).
[55] Vgl. (Scott und Vessey, 2002, S. 75).
[56] Vgl. (Lublinsky und Farell, 2002, S. 41).
[57] Vgl. (Lam und Shankararaman, 2004, S. 41).
[58] Vgl. (Lam, 2005, S. 181).
[59] Vgl. (Lam und Shankararaman, 2004).
[60] Vgl. (Sumner, 2000, S. 184), (Scott und Vessey, 2002, S. 75)
[61] Vgl. (ebenda, S. 75).
[62] Vgl. (Bajgoric und Moon, 2009, S. 79).
[63] Vgl. (Lam, 2005, S. 177).

integrieren zu können. Das schwerwiegendste Problem in dieser Gruppe ist das Fehlen einer Integrationsmethodik[64], da sich im Vergleich zur Anwendungsimplementierung das Gewicht einzelner Phasen verschiebt, wiegt dies besonders schwer. Die Implementierung hat im Vergleich zur Systementwicklung eine geringere Gewichtung, wohingegen die Bedeutung von Entwurf- und Architekturphasen wesentlich stärker ist.[65] Die Folge einer fehlenden Methodik kann eine chaotische Projektdurchführung sein, weil teilweise unvorhergesehene Arbeiten auftauchen [66], was für eine unbekannte Komplexität von Integrationsprojekten spricht.[67] Zu den wegen Unkenntnis vernachlässigten Tätigkeiten zählt ein Konfigurationsmanagement was – ob der involvierten Beteiligten – unternehmensübergreifend sein muss und ein Störungsmanagement, welches Ursache und Lösung für spätere Effizienzverbesserungen dokumentiert.[68] Dadurch kann ein ineffizientes Ressourcenmanagement entstehen, ebenso wie ein grundsätzlich fehlerhaft gesteckter Aufgabenbereich des Projektes.[69] Insbesondere im Hinblick auf Zeit und Kosten ist die – übliche – Ausweitung des Aufgabenbereiches während des Projektes kritisch.[70] Einen zu hohen Zeitdruck auf Integrationsprojekte – unabhängig von einer Aufgabenerweiterung – attestiert auch Vogler als Problem.[71]

Ein weiteres Problem ist ein fehlender Gesamtverantwortlicher für ein Integrationsprojekt.[72] Fehlt dieser leidet auch die Kontrolle des Projektes[73] dabei wird häufig die Berichterstattung nicht eingefordert[74] und es fehlen Eskalationsmechanismen[75]. Trotz Kontrolle müssen eigene Entscheidungen des Projektteams möglich sein.[76]

Vernachlässigt wird auch die Evaluation im Anschluss[77], welche z. B. in Form von Tests durchgeführt werden könnte. Das Gesamtsystem ist nicht getestet, nur weil es integriert wurde, was eine weitverbreitete, falsche Annahme ist.[78] Aufgrund der Vielzahl an Subsystemen und Arbeitsergebnissen, welche für einen Integrationstest verfügbar sein müssen, handelt es sich dabei um eine besonders kritische Projektphase.[79]

Aufseiten des Kunden ist die **Unterstützung des Managements** von außerordentlicher Bedeutung. Das diese grundsätzlich vorhanden sein muss merken einige Autoren an.[80] Die Ursache für die Bedeutung dieser Unterstützung liegt darin, dass Integration eine gesamtunternehmerische Aufgabe darstellt und deshalb nicht einer einzelnen Abteilung zugerechnet werden kann.[81] Die Integration in das Organisationsmanagement des Unternehmens ist insbesondere auch deshalb wichtig, um die Angst von Managern ohne tief greifende IT-Kenntnisse vor dem IT-System zu überwinden.[82]

[64] Vgl. (Vogler, 2006, S. 24).
[65] Vgl. (Lublinsky und Farell, 2002, S. 42).
[66] Vgl. (Lam und Shankararaman, 2004, S. 41).
[67] Vgl. (Vogler, 2006, S. 24).
[68] Vgl. (Moll und Ammerlaan, 2008, S. 336).
[69] Vgl. (Scott und Vessey, 2002, S. 75).
[70] Vgl. (Sumner, 2000, S. 185).
[71] Vgl. (Vogler, 2006, S. 24).
[72] Vgl. (Sumner, 2000, S. 184), (Scott und Vessey, 2002, S. 75) und (Moll und Ammerlaan, 2008, S. 333).
[73] Vgl. (Scott und Vessey, 2002, S. 75).
[74] Vgl. (Sumner, 2000, S. 186).
[75] Vgl. (Moll und Ammerlaan, 2008, S. 336).
[76] Vgl. (Scott und Vessey, 2002, S. 75).
[77] Vgl. (ebenda, S. 75).
[78] Vgl. (Moll und Ammerlaan, 2008, S. 333f.).
[79] Vgl. (ebenda, S. 334).
[80] Vgl. (Sumner, 2000, S. 184), (Scott und Vessey, 2002, S. 75), (Lam, 2005, S. 181).
[81] Vgl. (Trotta, 2003).
[82] Vgl. (Bajgoric und Moon, 2009, S. 79).

Durch die Vielzahl verschiedener Beteiligter und Betroffener von Integrationsprojekten können **Kommunikationsprobleme** entstehen. Neben der allgemeinen Feststellung dieser Probleme kann eine Verhinderung auch absichtlich passieren.[83] Die Beteiligten wollen nicht miteinander kommunizieren.[84] Die Folge kann ein Bruch zwischen dem Management und dem IT-Personal sein, welcher auf unterschiedliche Fachsprachen zurückgeführt werden kann.[85]

Einer **Strategie** und strategischen Vision kommt demnach eine große Bedeutung zu.[86] Da Integration auch Veränderung bedeutet – im Sinne der Transformation der Unternehmensarchitektur in einen neuen Zustand – wirkt sich auch eine Veränderungsmanagement-Strategie positiv aus[87], insbesondere deshalb, weil diese Veränderungen regelmäßig auftreten.[88] Gleiches trifft für eine Technologiestrategie zu[89], obwohl betont wird dass Integration kein Werkzeug, sondern ein komplexes System ist.[90] Der Erfolg kann durch eine Vollzeitunterstützung der Kunden in Bezug auf Projektaktivitäten und das Projektmanagement signifikant gesteigert werden.[91]

Bei der **Lösungsumsetzung** treten verschiedene Probleme auf, welche in **Integrationsanforderungen** begründet liegen. Problematisch ist die Nichtnutzung von Standards, sofern die Integrationsobjekte, -hilfsmittel und -werkzeuge die Möglichkeiten dazu bieten, ebenso wie die Nutzung verfügbarer Zusatzmodule zu bestehenden Systemen.[92] Dennoch können auch Standards fehlerhaft verwendet werden[93] oder gar fehlen[94]. Die Verteilung und Komplexität der Systeme verkompliziert sowohl die Implementierung von Transaktionen als auch das End-to-End Monitoring.[95] Grundsätzlich lassen sich viele Probleme und Anforderungen auf die Heterogenität der Integrationsobjekte zurückführen. Diese Probleme stellen nur eine verkürzte Darstellung dar und spiegeln nicht alle dokumentierten Erfahrungen wieder.

Die vorgestellten Problemgruppen lassen sich unterschiedlichen Lücken in einem GAP-Modell zuordnen. Die Abbildung 4.6 verdeutlicht die verschiedenen Lücken. Zunächst muss gesagt werden, dass davon ausgegangen wird, dass zuerst eine umfassende Erfassung der *Integrationsanforderungen* stattfindet, aus denen dann die *Integrationsarchitektur* abgeleitet wird. Bei der Integrationsarchitektur handelt es sich um eine logische Architektur, welche zwar Lösungsprinzipien in Anwendung bringt, aber noch von konkreten Integrationshilfsmitteln abstrahiert. In der Terminologie der modellgetriebenen Softwareentwicklung handelt es sich um ein plattformunabhängiges Modell. Im Anschluss daran erfolgt die Auswahl der *Integrationshilfsmittel*, es wird also festgelegt, welche Produkte zur Realisierung verwendet werden. Damit ist aber noch keine vollständige *Integrationslösung* umgesetzt, denn unter anderem sind für die Integrationshilfsmittel Konfigurationsartefakte zu erstellen.

Neben dieser groben Strukturierung des Ablaufes eines Integrationsprojektes ist von der Trennung der beteiligten Personen in Integrationsdienstleister und Kunde auszugehen. Weiterhin teilen sich beide Personengruppen in jeweils zwei Untergruppen. Zum einen in *technisch orientierte Beteiligte* und zum anderen in *geschäftlich orientierte Beteiligte*. Im Vergleich zu dem – auf den empirischen Resultaten basierenden – Ressourcenmodell in Kapitel 4.2.2 stellt diese Unterteilung

[83] Vgl. (Scott und Vessey, 2002, S. 75).
[84] Vgl. (Lublinsky und Farell, 2002, S. 41).
[85] Vgl. (Bajgoric und Moon, 2009, S. 79).
[86] Vgl. (Lam, 2005, S. 182), (Scott und Vessey, 2002, S. 75)
[87] Vgl. (Sumner, 2000, S. 186).
[88] Vgl. (Trotta, 2003).
[89] Vgl. (Sumner, 2000, S. 185).
[90] Vgl. (Trotta, 2003).
[91] Vgl. (Sumner, 2000, S. 185).
[92] Vgl. (ebenda, S. 186).
[93] Vgl. (Trotta, 2003).
[94] Vgl. (Vogler, 2006, S. 24).
[95] Vgl. (Lublinsky und Farell, 2002, S. 42).

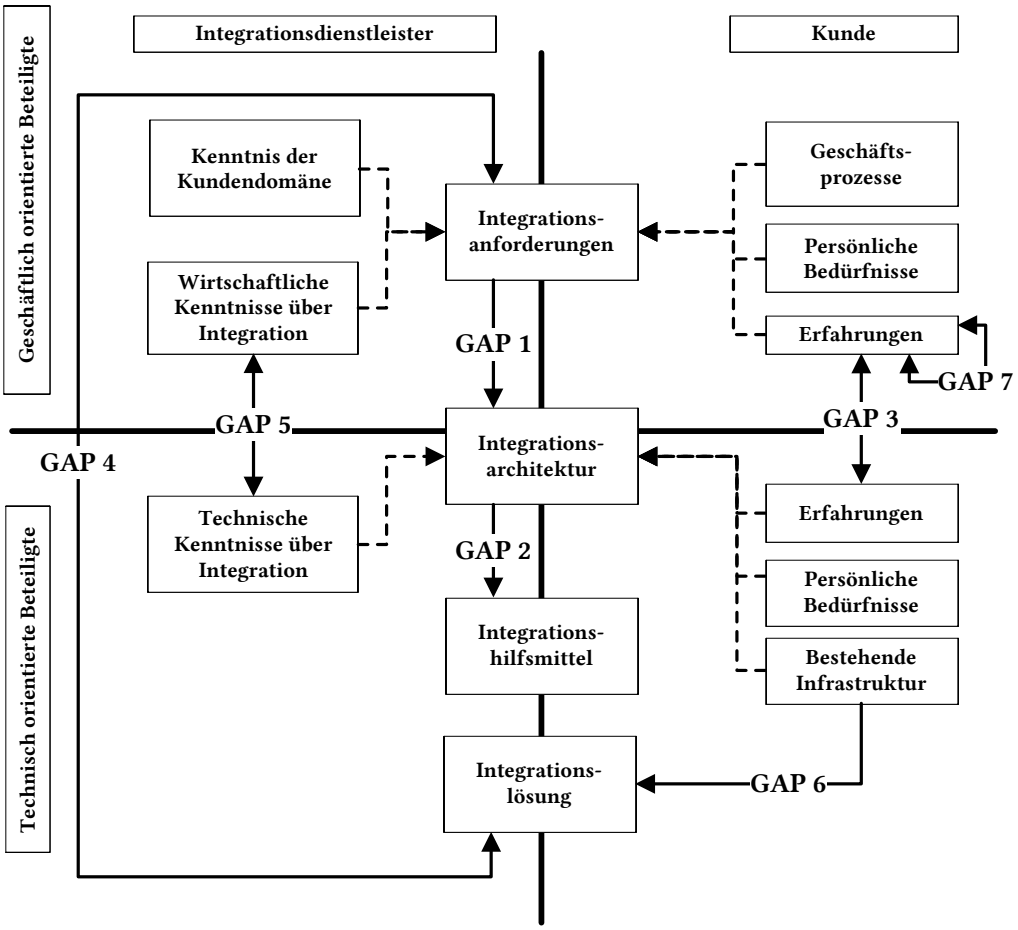


Abbildung 4.6: Gap-Modell der Integration

eine Verkürzung dar, welche aber grundsätzlich ausreicht, um die Kommunikationsprobleme und die daraus resultierenden Lücken darzustellen.

Grundsätzlich kann eine Problemgruppe zu mehreren Lücken führen und an unterschiedlichen Stellen Auswirkungen entfalten. Es bleibt festzuhalten, dass Probleme und Kombinationen von Problemen zu den Lücken des Modells führen. Zunächst werden die Lücken des Gap-Modells vorgestellt und im Anschluss dargelegt, welche Problemgruppen zu dieser Lücke führen können.

Aus der idealisierten Vorstellung des Ablaufes lassen sich verschiedene prinzipielle Lücken ableiten, die sich wie folgt darstellen:

Gap1: Die Ableitung der Integrationsarchitektur erfolgt nicht systematisch anhand der Requirements.

Gap2: Die Auswahl der Integrationslösung erfolgt nicht systematisch anhand der Integrationsarchitektur und den Integrationsanforderungen.

Gap3: Es existieren massive Kommunikationsprobleme zwischen fachlich- und technisch orientierten Projektbeteiligten.

Gap4: Die Integrationslösung erfüllt nicht alle Anforderungen, die an sie gestellt wurden.

Gap5: siehe Gap3.

Gap6: Infrastructural-Gap - die angestrebte Integrationsarchitektur steht nicht in Übereinstimmung mit der existierenden technischen Infrastruktur. Um zum Beispiel auf der Applikationsebene eine Beziehung zwischen zwei Integrationsobjekten herstellen zu können, muss es eine physische Verbindung der entsprechenden Knoten geben. Weitere Determinanten entstehen unter anderem durch maximal mögliche Netzwerklasten, welche unter Umständen eine Realisierung verhindern und die Skalierung der Lösung beeinflussen.

Gap7: Diese Lücke stellt exemplarisch dar, dass es auch innerhalb einer Gruppe von Beteiligten zu Kommunikationsproblemen (siehe Gap3) kommen kann. Diese Lücke findet sich bei allen anderen Gruppen im Modell, wird aber nur hier exemplarisch dargestellt.

Die Problemgruppe der Soft Skills wirkt ausnahmslos auf alle Gaps ein, weshalb diese in den nachfolgenden Erläuterungen nicht noch einmal mit aufgeführt wird. Gleiches gilt für die Problemgruppe der Projektorganisation.

Zusätzlich zu diesen Problemgruppen finden sich bei Gap1 Soft Skill Probleme und Probleme der Unterstützung des Managements sowie Probleme der Integrationsanforderungen. Hier wirken sich auch insbesondere die Probleme des Business-IT-Alignments sowie die Probleme der Strategie aus. Diese führen zu einer schlechten Qualität der Integrationsarchitektur. Bei Gap2 wirken die Problemgruppen der Lösungsumsetzung und der Architektur. Gap3, Gap5 und Gap7 sind maßgeblich durch Kommunikationsprobleme bestimmt und verdeutlichen so den enormen Einfluss dieser Problemgruppe, da sie sich an entsprechend vielen Stellen bei der Integration auswirkt. Auf Gap6 wirkt die Problemgruppe der Integrationsanforderungen. Dabei handelt es sich um jene Teile dieser Gruppe, welche z. B. durch bestehende Infrastruktur bedingt werden.

Das vorgestellte Modell verdeutlicht die Auswirkungen verschiedener Probleme auf die Erstellung einer Integrationslösung. Es stellt einen ersten Ansatz dar, diese Probleme systematisch zu erfassen. Weitere Arbeiten sowie bereits jetzt identifizierte Verbesserungsmöglichkeiten werden in Kapitel 8 diskutiert.

Das, sich Anschließende, Kapitel 5 befasst sich mit der Entwicklung von Integrationsmustern und der Anpassung des Abstraktionsniveaus dieser, um eine verbesserte Einsetzbarkeit bei der Integration zu erreichen.

5 Entwicklung hochstehender Integrationsmuster

Dieses Kapitel bildet neben dem Dienstleistungsmodell der Integration[1] einen der Hauptbestandteil der Arbeit und legt die Entwicklung hochstehender Integrationsmuster dar. Ausgangspunkt ist die in Kapitel 3 herausgearbeitete Erkenntnis, dass Integrationsmuster keinen Einsatz in der Praxis finden.[2] Dies wird darauf zurückgeführt, dass der Zugang zu Mustern zu schwierig ist und eine umfangreiche Auseinandersetzung mit dieser Thematik erfordert. Dieser Initialaufwand kann im Rahmen kurz zyklischer Projektarbeit – wie sie für Integrationsdienstleister typisch ist – regelmäßig nicht geleistet werden. Zusätzlich müssen die Muster dann auf die jeweilige Problemstellung übertragen werden, wobei der direkte Beitrag zu den Zielen der Integration – vor allem zu den Gesichtspunkten unter denen ein Kunde eine Lösung auswählt[3] – nicht direkt sichtbar ist. Dieser Erkenntnis folgend sollen hier Mechanismen entwickelt werden bestehende Muster derart zu abstrahieren, dass die Einsetzbarkeit insbesondere in den Phasen Analyse und Planung verbessert wird.[4]

In einem ersten Schritt wird dazu in Kapitel 5.1 eine Definition für Integrationsmuster entwickelt. Es wird eine Definition auf der Basis charakteristischer Eigenschaften angestrebt, die es auch Muster als Integrationsmuster zu erkennen, wenn sie nicht explizit als solche ausgezeichnet wurden. Dies sichert die Erweiterbarkeit und Anwendbarkeit der in dieser Arbeit entwickelten Ergebnisse.

Da das primäre Ziel der Arbeit nicht die Entwicklung grundlegend neuer Muster, sondern eine Abstraktion bestehender ist, wird die entwickelte Definition eines Integrationsmusters in Kapitel 5.2 angewandt, um die Basis für die dann folgenden Abstraktionsschritte zu schaffen. Zu diesem Zweck wird die existierende Literatur gemäß der Definition gesichtet und eine Grundmenge an Integrationsmustern extrahiert.

Bevor mit der Entwicklung abstrakterer Muster begonnen werden kann, werden in Kapitel 5.3.1 bestehende Ansätze zur Abstraktion in der Informatik und Wirtschaftsinformatik untersucht, diese werden dann für die Entwicklung hochstehender Integrationsmuster in Kapitel 5.4 verwendet. Dabei werden verschiedene Abstraktionsarten zum Einsatz gebracht, wie die Generalisierung, die Klassifikation bestehender Muster.

Gemäß der Abhängigkeiten der Abstraktionsarten werden zunächst Generalisierungen entwickelt, aus denen dann Klassen abgeleitet werden können. Die ausgewählten Muster werden dann in einem Schritt der Klassierung in diese Klassen eingeordnet. Auf dieser Basis entsteht eine Klassifikation bestehender Integrationsmuster. Im Anschluss daran erfolgt auf der Basis der Klassifikation eine erneute Generalisierung.

[1] Vgl. Kapitel 4.
[2] Vgl. (Gebauer und Stefan, 2011b).
[3] Vgl. Abbildung 3.17 und Abbildung 3.18, sowie Kapitel 3.5.5 im Allgemeinen für die Auswahlkriterien.
[4] Vgl. für die Phasen Kapitel 3.

5.1 Integrationsmuster Annäherung und Definition

Nachdem in der wissenschaftlichen Einordnung die Begriffe Entwurfsmuster und Architekturmuster dargelegt wurden[5], folgt nun eine Analyse existierender Definitionen des Begriffes Integrationsmuster.[6] Hintergrund dieser Bemühungen ist es, die charakteristischen Merkmale eines Integrationsmusters zu extrahieren, um auch nicht explizit als Integrationsmuster gekennzeichnete Muster als solche identifizieren zu können. Diese Merkmale liefern die Grundlage für die Auswahl einer Grundgesamtheit an Mustern, die Gegenstand der weiteren Abstraktion werden.

Um dieses Ziel zu erreichen, wurden zu erst explizite Definitionen von Integrationsmustern untersucht und die beschriebenen Eigenschaften extrahiert (Abschnitt 5.1.1). Da in mehreren Publikationen explizite Verweise darauf gefunden worden, dass es sich bei Integrationsmustern um Architekturmuster handelt, welche Integrationsarchitekturen beschreiben, wird in Abschnitt 5.1.2 analysiert, welche Merkmale für Architekturstile und Architekturmuster gelten.[7] Diese Untersuchung liefert ergänzende Erkenntnisse für die Definition von Integrationsmustern. Diese Eigenschaften aufgreifend untersucht der Abschnitt 5.1.3 Definitionen von Integrationsarchitekturen. Den Abschluss dieser Untersuchungen bildet der Abschnitt.5.1.4 Dieser fasst die identifizierten charakteristischen Merkmale zusammen und versucht eine Definition eines Integrationsmusters zu geben, auf deren Basis die Muster identifiziert werden können, welche – zumindest im Rahmen dieser Arbeit – die Grundgesamtheit der weiteren Betrachtungen darstellen.

5.1.1 Explizite Definitionen von Integrationsmustern

Der erste und einfachste Ansatz, charakteristische Merkmale von Integrationsmustern zu extrahieren, ist es bestehende Definitionen zu analysieren. Bei der Sichtung der Literatur fällt auf, dass selbst Publikationen welche den Begriff Integration Patterns[8] im Titel verwenden, keine Definition eines Integrationsmusters enthalten.[9] Damit lässt sich der Vorwurf einer willkürlichen Auswahl der vorgestellten Muster erheben. Weiterhin kann festgestellt werden, dass viele Publikationen Hohpe und Woolf[10] zitieren und stillschweigend exakt die dort publizierten Muster als Integrationsmuster übernehmen, ohne eine Definition zu geben oder die Auswahl zu begründen. Als Beispiel für ein solches Vorgehen kann die Arbeit von de Daniel und Steinrötter[11] genannt werden. Ein weiteres Phänomen, welches anhand dieses Beispiels deutlich wird, ist die Verknüpfung von Mustern mit Integrationshilfsmitteln. De Daniel und Steinrötter befassen sich explizit mit der Umsetzung von Mustern auf Basis des Produktes SAP NetWeaver PI. Andere Publikationen beziehen sich auf Adams u. a.[12] und übernehmen die dort vorgeschlagene Trennung von Integrations-, Applikations- und Geschäftsmustern.[13] Dabei erfolgt keine Definition oder Abgrenzung gegenüber anderen Musterarten. Im Folgenden werden die existierenden Definitionen vorgestellt und die charakteristischen Merkmale extrahiert um diese im Anschluss zu verwenden. Die Tabelle 5.1 listet die Definitionen auf, welche Integrationsmuster explizit definieren.

Der Gegenstandsbereich von Integrationsmustern sind Komponentenarchitekturen.[14] Als Strukturelemente werden dabei 2 Arten von Komponenten unterschieden domänenspezifische (Geschäfts-

[5] Vgl. Kapitel 2.3.
[6] engl. integration pattern.
[7] In dieser Arbeit werden Architekturstil und Architekturmuster als synonym betrachtet.
[8] engl. für Integrationsmuster
[9] Vgl. (Trowbridge u. a., 2004).
[10] Vgl. (Hohpe und Woolf, 2003).
[11] Vgl. (Daniel und Steinrötter, 2008).
[12] Vgl. (Adams u. a., 2001).
[13] Vgl. (Zwanziger und Herden, 2004).
[14] Vgl. (Mularz, 1995, S. 443).

Quelle	Definition
(Mularz, 1995, S. 443)	„An integration pattern specifies an integration problem and a corresponding integration model that resolves the problem. The model consists of structural entities that can potentially be dispersed throughout an architecture, either as separate integration components or as adjuncts to the functional components. An integration pattern is therfore a well-formed arrangement of entities (that is, their behavior and relative placement is explicitly specified as part of the architecture).“
(Adams u. a., 2001)	„Integration patterns (both Front-End and Back-End) connect other Business patterns together to create solutions with advanced functionality. Integration patterns are used to combine Business patterns in advanced e-business solutions.“ „Integration Patterns address more complex e-business applications that combine multiple Business Patterns at the front-end via Access Integration or at the back-end via Application Integration. Integration Patterns are differentiated from Business Patterns in that they themselves do not automate specific business problems. Rather, they combine Business Patterns to support more advanced functions or to make Composite Patterns feasible.“
(Juric u. a., 2007, S. 52)	„Integration patterns will help us to understand the different solutions and methods for integration and allow us to choose the best solution for our problem. They allow us to look at integration problems from a certain abstraction level. Integration patterns describe proven methods and processes used for integration within the enterprise or beyond. They emerge from classifications of common integration solutions. Each integration pattern defines a common integration problem and a sound solution.“
(Engels und Voß, 2008)[22]	„Integrationsmuster sind in bestimmten Kontexten erprobte Standard-Integrationsarchitekturen“
(Zhao u. a., 2008, S. 1273)	„They are orthogonal to business patterns and are used to support business patterns by providing integrated solutions to business functions. There are two general integration patterns: Access Integration and Application Integration.“

Tabelle 5.1: Übersicht über Definitionen des Begriffs Integrationsmuster

komponenten) und Integrationskomponenten.[15] Wesentlich für die Integrationskomponenten ist, dass sie keine Geschäftsfunktionalität automatisieren.[16] Mularz bezeichnet Integrationsmuster als Architekturmuster.[17] Im Gegensatz dazu steht die Aussage von Engels und Voß, die Integrationsmuster als Standard-Integrationsarchitekturen sehen.[18] Diese Definition ist widersprüchlich zu der in Abschnitt 5.1.3 herausgearbeiteten Definition einer Integrationsarchitektur. Die Muster beschreiben explizit das Verhalten der Strukturkomponenten sowie deren Beziehungen und die relative Positionierung in der Architektur.[19] Sie stellen erprobte Lösungen für bestimmte Integrationsprobleme[20] dar, die von Mularz auch als Integrationskonflikte bezeichnet werden.[21]

Durch die Analyse der Definitionen wurde klar, dass es sich bei Integrationsmustern um Architekturmuster handelt. Aus diesem Grund analysiert das nächste Kapitel die Eigenschaften von Architekturmustern, um die Sammlung an charakteristischen Merkmalen weiter zu ergänzen.

5.1.2 Eigenschaften von Architekturmustern und Architekturstilen

Im vorangegangenen Abschnitt wurden Definitionen von Integrationsmustern untersucht. In der wissenschaftlichen Einordnung wurde bereits darauf verwiesen, dass es unterschiedliche Musterarten gibt. Dabei wurden Entwurfsmuster und Architekturmuster unterschieden. Integrati-

[22] Vgl. dazu auch (Engels; Hess u. a., 2008).
[15] Vgl.(ebenda, S. 443), (Adams u. a., 2001).
[16] Vgl. (ebenda), (Zhao u. a., 2008, S. 1273).
[17] Vgl. (Mularz, 1995, S. S. 443).
[18] Vgl. (Engels und Voß, 2008), sowie (Engels; Hess u. a., 2008).
[19] Vgl. (Mularz, 1995, S. 443).
[20] Vgl. (Juric u. a., 2007, S. 52).
[21] Vgl. (Mularz, 1995, S. 443).

onsmuster wurde den Definitionen nach mehrfach als Architekturmuster dargestellt. Aus diesem Grund werden hier Definitionen von Architekturmustern analysiert, um weitere charakteristische Eigenschaften abzuleiten. Da in einigen Publikationen der Begriff Architekturstil synonym zu dem eines Architekturmusters gebraucht wird, werden auch Definitionen, die sich auf Architekturstile beziehen analysiert. Im Folgenden wird jedoch nur noch von Architekturmuster gesprochen, um einen einheitlichen Begriff zu verwenden.

Quelle	**Definition**[23]
(Garlan und Shaw, 1993, S. 2)	„[. . .] to treat an architecture of a specific system as a collection of computational components-or simply components-together with a description of the interactions between these components-the connectors. [. . .] As we will see, connectors can represent interactions as varied as procedure call, event broadcast, database queries, and pipes. An architectural style, then, defines a family of such systems in terms of a pattern of structural organization. More specifically, an architectural style determines the vocabulary of components and connectors that can be used in instances of that style, together with a set of constraints on how they can be combined.“
(Garlan; Allen und Ockerbloom, 1995, S. 180)	„An architectural style is a recurring pattern of system organization that provides an abstract framework for some family of applications.“
(Fielding, 2000, S. 13)	„An architectural style is a coordinated set of architectural constraints that restricts the roles/features of architectural elements and the allowed relationships among those elements within any architecture that conforms to that style.“
(GI Fachgruppe Software Architektur, 2009)	„Architekturstile sind prinzipielle Lösungsstrukturen, die für ein Element einer Architektur durchgängig und mit weitgehendem Verzicht auf Ausnahmen angewandt werden.“
(Heutschi, 2007, S. 8)	Gibt keine eigene Definition, sondern zitiert die Definition von Fielding.
(Buschmann; Meunier u. a., 1996)	„An Architectural Pattern expresses a fundamental structural organization schema for software systems. It provides a set of predefined subsystems, specifies their responsibilities,and includes rules and guidelines for organizing the relationships between them.“
(Lutz, 2000)	verwendet den Begriff des Architectural Patterns als Synonym für EAI Architecture Pattern.
(Hasselbring, 2006, S. 48)	„Architekturmuster hingegen beschreiben den Stil der Gesamtarchitektur eines Systems.“
(Heutschi, 2007, S. 8)	„Architekturmuster bieten auf heuristischem Modellierungswissen basierende generische Lösungsansätze für wiederkehrende Designprobleme.“ Dabei werden (Alexander u. a., 1977) und (Hasselbring, 2006) zitiert.
(Winter; Brocke u. a., 2009, S. 536)	„Schrittweise hat sich der Gegenstandsbereich von Mustern erweitert: Entwurfsmuster (Gamma u. a., 1995) leisten Unterstützung beim Entwurf einzelner objektorientierter Softwarekomponenten, Architekturmuster (Buschmann; Meunier u. a., 1996) bei der Komposition von Softwarekomponenten zu Anwendungen, Architekturmuster für Unternehmensanwendungen (Fowler, 2003) schließlich erfassen Gestaltungsprinzipien für die Vernetzung von Anwendungen zu IT-Landschaften“.

Tabelle 5.2: Übersicht über Definitionen der Begriffe Architekturstil und Architekturmuster

us diesen Definitionen lassen sich einige Eigenschaften für Architekturmuster ableiten. Die Elemente eines Architekturmusters können als Architekturelemente[24] bezeichnet werden und stellen Rechenkomponenten oder einfach Komponenten[25] dar. Zu diesen werden Komponenten und Konnektoren[26], sowie vordefinierte Subsysteme[27] gezählt. Das Wesen eines Architekturmusters ist es, die strukturelle Organisation, das Schema eines Systems festzuhalten.[28]

[23] A

[24] Vgl. (Fielding, 2000, S. 13).

[25] Vgl. (Garlan und Shaw, 1993, S. 2).

[26] Vgl. (ebenda, S. 2).

[27] Vgl. (Buschmann; Meunier u. a., 1996).

[28] Vgl. (Garlan und Shaw, 1993, S. 2), (Garlan; Allen und Ockerbloom, 1995, S. 180), (Buschmann; Meunier u. a., 1996), (Hasselbring, 2006, S. 48).

Neben den Elementen wird in einem Architekturmuster auch beschrieben, wie die Elemente kombiniert werden können. In Form von Restriktionen werden Rollen, Eigenschaften und die zwischen den Elementen erlaubten Beziehungen festgelegt.[29]

Eine wesentliche Eigenschaft ist die Abstraktion von einzelnen Ausprägungen hin zur Beschreibung der Lösung für eine Familie von Applikationen.[30] Damit wird ein prinzipieller Lösungsansatz [31] für wiederkehrende Probleme[32] unterstrichen. Soll eine Architektur ausschließlich nach einem Muster gestaltet werden, so ist dieses weitgehend ohne Ausnahme anzuwenden[33].

Einen Zusammenhang zu anderen Mustern stellen Winter u. a. her. Sie ordnen Architekturmuster in einer Hierarchie ein, in der Entwurfsmuster eine einzelne objektorientierte Anwendung beschreiben, Architekturmuster die Komposition von Komponenten zu Anwendungen und wiederum Architekturmuster für Unternehmensanwendungen die Vernetzung von Anwendungen beschreiben.[34]

5.1.3 Herleitung von Eigenschaften auf der Basis von Integrationsarchitekturdefinitionen

Bisher wurden explizite Definitionen von Integrationsmustern und die von Architekturmustern bzw. Architekturstilen analysiert. Als Gestaltungsgegenstand von Architekturmustern können Architekturen angenommen werden. Aus diesem Zusammenhang kann geschlussfolgert werden, dass es sich beim Gestaltungsgegenstand von Integrationsmustern – welche Architekturmuster sind – um Integrationsarchitekturen handelt. Demzufolge werden nun Definitionen von Integrationsarchitekturen analysiert, um über den Gestaltungsgegenstand zu weiteren Aussagen über Eigenschaften von Integrationsmustern zu gelangen.

Quelle	Definition
(Keshav und Gamble, 1998, S. 2)	„[. . .] define integration elements that are at the same abstraction level as component architecture descriptions. These definitions stem from architecture modeling [. . .] and case study analysis [. . .]. Each element is a minimal model of integration functionality that is based on some model of incompatibility. Composing multiple elements forms an integration architecture.“
(Dous, 2007, S. 176)	„Die Integrationsarchitektur beschreibt (Middleware-) Dienste, die auf Basis standardisierter Schnittstellen für eine transparente Kommunikation verteilter Anwendungen sorgen. Beispiele hierfür sind Portlets, EAI-Systeme oder WebServices [. . .].“
(Heutschi, 2007, S. 10)	„Die Integrationsarchitektur beschreibt die Kopplung verteilter Funktionen und Daten über Applikationen hinweg [Vgl. (Schwinn, 2006)]. Sie bietet Middlewarekomponenten mit standardisierten Schnittstellen und Protokollen und unterstützt eine transparente Kommunikation zwischen Applikationen, indem sie die Heterogenität von Systemplattformen überbrückt [. . .].“
(Hagen und Schwinn, 2006, S. 276)	„Die Integrationsarchitektur ist auf verschiedenen Ebenen definiert. Etwas vereinfacht besteht sie aus (a) Infrastruktur (Middleware) für die Integration von Applikationen und Systemen, (b) Methoden und Prozessen für die Nutzung der Integrationsmittel und für die Qualitätssicherung der Integrationslösungen und (c) Steuerungsmassnahmen zur Verbesserung der Applikationslandschaft in Bezug auf die Integration.“

Übersicht über Definitionen des Begriffs Integrationsarchitektur – Fortsetzung auf der nächsten Seite

[29] Vgl. (Garlan und Shaw, 1993, S. 2), (Fielding, 2000, S. 13), (Buschmann; Meunier u. a., 1996).
[30] Vgl. (Garlan; Allen und Ockerbloom, 1995, S. 180).
[31] Vgl. (GI Fachgruppe Software Architektur, 2009).
[32] Vgl. (Heutschi, 2007, S. 8).
[33] Vgl. (GI Fachgruppe Software Architektur, 2009), Dies steht jedoch im Widerspruch zu dem in der Praxis vorzufindenden Betrieb von Mischarchitekturen, die nach den Prinzipien mehrerer Muster gestaltet wurden. Siehe dazu auch Kapitel 3.
[34] Vgl. (Winter; Brocke u. a., 2009, S. S. 536).

Quelle	Definition
(Geib, 2006, S. 122)	„Die Integrationsarchitektur beschreibt die Integration von Applikationen und Datenbanken auf der Basis von standardisierten Schnittstellen und Protokollen. Elemente der Integrationsarchitektur sind Systeme für die Prasentationsintegration, die Datenintegration und die Funktionsintegration.“
(Vogel u. a., 2009, S. 55)	„Integrationsarchitektur beschäftigt sich mit der Planung und Realisierung von integrativen Lösungen mit dem Ziel, mehrere Anwendungen oder Systeme eines oder mehrerer Unternehmen miteinander zu verbinden. Dabei müssen meist heterogene Plattformen, Technologien, Organisationen und Daten in Einklang gebracht werden. Eine wichtige Aufgabe ist hierbei oftmals auch die Berücksichtigung von Legacy- Systemen und deren Anschluss an E-Business-Lösungen.“
(Erl, 2009, S. 28)	„Integration Architecture - The technology architecture of two or more connected applications or systems including whatever technologies, resources, or extensions were added to enable their integration. Many integration architectures include middleware platforms and associated adapter or bridging extensions.“
http://hubpages.com/hub/Integration-Architecture-Explained	Lässt sich unterteilen in Point-to-Point Integrationsarchitektur; Hub-and-Spoke Integrationsarchitektur; Distributed Integrationsarchitektur; Service Orientierte Architektur.
(Davis und Gamble, 2002, S. 70)	„An integration architecture, then, is defined to be the software architecture description of a solution to interoperability problems between at least two interacting component systems.“
(Payton u. a., 2000, S. 4)	„We use the term integration architecture to describe those strategies and patterns that make up a complete integration solution for a particular application. The integration architecture is different from the middleware product because it does not include extraneous functions, i.e., functionality that is not directly related to the known interoperability problems. Describing an integration architecture at the same level of abstraction as the software architecture of the component system requires expressing minimal functional units to resolve low-level conflicts. Different compositions of these building blocks (and their variations) form integration architectures [. . .], which in turn provides a comparative analysis of frameworks. We call these minimal functional units integration elements [. . .].“
(Lutz, 2000, S. 66)	„The purpose of enterprise integration architecture is to bring order and consistency to the many EAI applications across the enterprise.“
(Hepner u. a., 2006, S. 538)	„This integration architecture is an abstraction of the core functionality needed in the integration strategy, down to single functional elements.“
(Engels; Hess u. a., 2008, S. 207)	„Das Ergebnis der Arbeit eines Architekten ist die Integrationsarchitektur – eine Verfeinerung der [. . .] eingeführten Kopplungsarchitektur auf der physischen Ebene“[35]. „Unter der Integrationsarchitektur verstehen wir die Festlegung, welche physischen Schnittstellen welcher AL-Komponenten unter Verwendung welcher technischen Services integriert werden und von welcher Art sie sind. Dies schließt die Entscheidung ein, welche logischen AL-Komponenten durch Orchestrierung realisiert werden“.

Tabelle 5.3: Übersicht über Definitionen des Begriffs Integrationsarchitektur

Aus den in der Tabelle 5.3 angeführten Definitionen lassen sich die nachfolgenden Eigenschaften einer Integrationsarchitektur ableiten. Die beschriebenen Elemente bewegen sich auf der Abstraktionsebene von Softwarekomponenten.[36] Dabei können Systeme für die Präsentations-, Daten- und Funktionsintegration[37], Services[38] als auch jedwede Technologie, Ressource oder Erweiterung beschrieben werden, die für die Erreichung der Integration notwendig sind[39].

Obwohl einige Definitionen darauf verweisen, dass die beschriebenen Elemente Middleware-Komponenten oder -dienste auf Basis standardisierter Schnittstellen und Protokolle sind[40], die

[35] Vgl. auch (Engels; Hess u. a., 2008, S. 195)
[36] Vgl. (Keshav und Gamble, 1998, S. 2),(Davis und Gamble, 2002, S. 70),(Payton u. a., 2000, S. 4).
[37] Vgl. (Geib, 2006, S. 122).
[38] Vgl. (Engels; Hess u. a., 2008, S. 195).
[39] Vgl. (Erl, 2009, S. 28).
[40] (Dous, 2007, S. 176), (Heutschi, 2007, S. 10), (Hagen und Schwinn, 2006, S. 276), (Geib, 2006, S. 122), (Erl, 2009, S. 28), (Engels; Hess u. a., 2008, S. 207)

durch die Architektur geordnet werden[41], gibt es auch Aussagen, die dazu im Widerspruch stehen. Payton zur Folge unterscheidet sich eine Integrationsarchitektur von einer Middleware dahin gehend, dass sie keine Funktionen beschreibt, welche über die Lösung des Konflikts hinaus gehen. Diese Abstraktion wird – im Vergleich zu einer konkreten Realisierung – dadurch untermauert, dass die Elemente als Strategien und Muster bezeichnet werden.[42] Diesen Ansatz der Abstraktion unterstützten auch Hepner u. a. nach deren Auffassung eine Integrationsarchitektur eine Abstraktion hinsichtlich der Kernfunktionalität ist, welche für eine Integrationsstrategie benötigt wird, heruntergebrochen auf funktionale Elemente.[43] Einen zusätzlichen Blickwinkel eröffnen Hagen und Schwinn nach denen Integrationsarchitekturen auf den Ebenen a) Infrastruktur, b) Methoden, und Prozesse c) Steuerungsmaßnahmen zur Verbesserung der Applikationslandschaft definiert werden können. Dies unterstützt zum einen den Abstraktionsgedanken, geht jedoch über den Gestaltungsbereich der anderen Definitionen hinaus. Eine zusätzliche Erweiterung möglicher Abstraktion tragen Engels u. a. al. bei, für die eine Integrationsarchitektur eine Verfeinerung – also Konkretisierung – einer Kopplungsarchitektur darstellt.[44]

Kompositionen der Integrationselemente formen die Integrationsarchitektur.[45] Eine der Hauptfunktionen von Integrationselementen ist das Überbrücken von Inkompatibilität[46] bzw. von Heterogenität, welche in Form von Systemplattformen[47], Technologien und Organisationen [48] auftreten kann. Zusammenfassend dienen Integrationselemente der Lösung von Interoperabilitätsproblemen.[49]

Die zweite Funktion der Integrationselemente ist die Organisation von transparenter Kommunikation.[50] Dabei steht die Kopplung – im Sinne der Umsetzung einer Kopplungsarchitektur[51] – verteilter Funktionen und Daten [52], Datenbanken[53], Anwendungen[54] bzw. Systeme eines Unternehmens[55] im Fokus. Ziel ist es eine integrierte Lösung bzw. einen Megakonnektor zu schaffen.[56]

5.1.4 Zusammenfassung der Merkmale von Integrationsmustern

Für die Analyse der Eigenschaften von Integrationsmustern wurden Definitionen von Integrationsmustern, Architekturmustern und -stilen sowie Definitionen von Integrationsarchitekturen untersucht. Hintergrund der Analyse war es die Eigenschaften von Integrationsmustern so herauszuarbeiten, dass auch Muster anhand ihrer Eigenschaften als Integrationsmuster eingestuft werden können, auch dann, wenn sie nicht explizit als solche ausgewiesen sind. In diesem Abschnitt werden die Analyseergebnisse zusammengefasst und ein Schema entwickelt, mit dessen Hilfe im nachfolgenden Abschnitt die Grundgesamtheit der Integrationsmuster für diese Arbeit erstellt wird.

[41] Vgl. (Lutz, 2000, S. 66).
[42] Vgl. (Payton u. a., 2000, S. 4).
[43] Vgl. (Hepner u. a., 2006, S. 538).
[44] Vgl. (Engels; Hess u. a., 2008, S. 195).
[45] Vgl. (Keshav und Gamble, 1998, S. 2), (Payton u. a., 2000, S. 4).
[46] Vgl. (Keshav und Gamble, 1998, S. 2).
[47] Vgl. (Heutschi, 2007, S. 10).
[48] Vgl. (Vogel u. a., 2009, S. 55).
[49] Vgl. (Davis und Gamble, 2002, S. 70), (Payton u. a., 2000, S. 4).
[50] Vgl. (Dous, 2007, S. 176).
[51] Vgl. (Engels; Hess u. a., 2008, S. 195).
[52] Vgl. (Heutschi, 2007, S. 10).
[53] Vgl. (Geib, 2006, S. 122).
[54] Vgl. (Dous, 2007, S. 176).
[55] Vgl. (Vogel u. a., 2009, S. 55), (Erl, 2009, S. 28).
[56] Vgl. (Payton u. a., 2000, S. 4).

Strukturelemente welche in Integrationsmustern vorkommen sind: Businesskomponenten, Integrationskomponenten (funktionale Elemente, welche Integrationsfunktionen realisieren), Rechenkomponenten, Konnektoren, Subsysteme, Softwarekomponenten, Services, kurz alle für eine Integration notwendigen Komponenten. Wesentliches Merkmal der Integrationskomponenten selbst ist, dass sie keine Geschäftsfunktionalität automatisieren.

Beschrieben werden muss das Verhalten, die Rollen, Eigenschaften und die Beziehungen der Komponenten sowie deren relative Positionierung in der Integrationsarchitektur. Zusammengefasst beschreibt ein Integrationsmuster die strukturelle Organisation, das Schema, eines Teils des Integrationssystems.

Das Integrationssystem ist dabei querschnittlich zu den bisher besprochenen Systemen und Subsystemen des BIS, da es sowohl aus Software und Hardware als auch – im Fall von manuellen Teilschritten – Menschen bestehen kann.

Es muss eine Lösung für einen Integrationskonflikt präsentiert werden, welche der Art abstrahiert ist, dass sie für eine Familie von Applikationen gilt oder einen prinzipiellen Ansatz für wiederkehrende Probleme liefert.

Die Probleme stammen aus dem Bereich der Überbrückung von Heterogenität beziehungsweise der Herstellung von Interoperabilität und der Organisation der Kommunikation der Strukturelemente untereinander. Die Organisation der Kommunikation schließt demnach Zugriffspunkte, Kommunikationsprotokolle, die Adressierung der Kommunikationsteilnehmer, dass Routing der Kommunikation und die Verteilung der Funktionsbausteine, welche für die Realisierung der Kommunikation notwendig sind, mit ein. Integrationsmuster sind wiederum Bestandteile von Integrationsarchitekturen, welche wiederum Bestandteile von BIS sind.[57] Zusammengefasst lassen sich diese Eigenschaften in der folgenden Definition ausdrücken:

> *Eine Integrationsarchitektur besteht aus Integrationsmustern, welche das Zusammenspiel von Strukturelementen – die je nach Abstraktionsgrad Softwarekomponenten, Applikationen oder Subsysteme eines Systems sein können – organisiert. Diese Muster beschreiben hauptsächlich die Organisation der Kommunikation sowie die Überbrückung von Heterogenitäts- bzw. Interoperabilitätsproblemen zwischen den Strukturelementen. Dabei ist die Überbrückung von Interoperabilitätsproblemen im Sinne korrespondierender Systemelemente zu verstehen. Eine Integrationsarchitektur entspricht der Komposition von Integrationsmustern bezüglich einer gegebenen Problemstellung.*
>
> *Eine Integrationsarchitektur ist dann vollständig, wenn sie alle Integrationsprobleme einer oder mehrerer vorliegender Problemdefinitionen in der geforderten Qualität löst. Die Problemdefinition entspricht den gewünschten Veränderungen der restlichen Bestandteile der Unternehmensarchitektur.*

Diese Definition wird im weiteren Verlauf der Arbeit als maßgeblich verwendet. Das sich anschließende Kapitel befasst sich mit der Identifikation von Integrationsmustern in Form von explizit ausgewiesenen Integrationsmustern.

5.2 Übersicht der Integrationsmuster

In diesem Kapitel wird die Grundgesamtheit der Integrationsmuster zusammengestellt, die zum Gegenstand der Abstraktionsbemühungen werden wird. Für die Auswahl werden zum einen die extrahierten Merkmale aus Kapitel 5.1 verwendet als auch Publikationen analysiert, die bestimmte Muster explizit als Integrationsmuster ausweisen.

[57] Vgl. Abschnitt 2.1.

Zunächst werden die expliziten Nennungen von Mustern als Integrationsmuster betrachtet. Davis und Gamble[58] bezeichnen verschiedene Muster aus unterschiedlichen Quellen als Integrationsmuster. Zu diesen Mustern zählen Proxy[59], Broker[60], sowie die Muster Wrapper Facade, Component Configurator, Interceptor aus Schmidt; Stal u. a.[61] und der Integration Mediator[62].

Weiterhin finden sich bei Mularz neben dem Broker die Muster (Legacy) Wrapper, Work Flow Manager und Shared Repository, welche ebenfalls als Integrationsmuster ausgewiesen werden [63]. Die Muster nach Hohpe und Woolf[64] werden von diesen als Enterprise Integration Pattern bezeichnet und gelten damit nach dem hier vertretenen Verständnis ebenfalls als explizit ausgewiesene Integrationsmuster. In Tabelle 5.4 werden die Muster mithilfe der in Abschnitt 5.1.4 zusammengefassten Merkmale charakterisiert. Dies dient der Überprüfung, ob die extrahierten Merkmale in der Lage sind explizit gekennzeichnete Integrationsmuster auch anhand ihrer Eigenschaften als solche zu identifizieren. Die Muster nach Hope und Woolf[65] wurden aus Platzgründen weggelassen, da es sich um eine Summe von 65 Mustern handelt.

Pattern, Quelle	**Strukturelemente**	**Verhalten, Rollen,Eigenschaften, Beziehungen**	**gelöster Integrationskonflikt**	**Überbr. Heterogenität**	**Org. Kommunikation**
Proxy, (Buschmann; Meunier u. a., 1996)	IO:OO-Komponenten IC:Proxy	der Proxy agiert als Stellvertreter und stellt die selbe Schnittstelle wie das Original zur Verfügung	Kommunikation	-	X
Broker, (Mularz, 1995)	IO:heterogene Komponenten, multiple Plattformen und Sprachen; IC: Broker (enthält Locator, Wrapper)	der Broker greift auf die Services des einen IO zu und stellt das Ergebnis einem anderen IO zur Verfügung (Stellvertreter) bspw. RPC, ORB	Kommunikation, Ortstransparenz, Funktionszentralisierung	-	X
Wrapper Facade, (Schmidt; Stal u. a., 2000)	IO:nicht-objektorientierte API IC:objektorientierte API	OO-API kapselt die Funktionalität	Kapselung, Entkopplung	(X)	X
Component Configurator, (ebenda)	IO:OO-Komponenten; IC:Component Repository, Component Configurator	Interface entkoppelt die Implementierung, dynamische Registrierung von Komponenten zur Laufzeit	Entkopplung, Funktionszentralisierung	(X)	X

Evaluation der Eigenschaften explizit benannter Integrationsmuster – Fortsetzung auf der nächsten Seite

[58] Vgl. (Davis und Gamble, 2002).
[59] Vgl. (Buschmann; Meunier u. a., 1996).
[60] Vgl. (Mularz, 1995).
[61] Vgl. (Schmidt; Stal u. a., 2000).
[62] Vgl. (Lutz, 2000).
[63] Vgl. (Mularz, 1995, S. 444ff.), Darüber hinaus entsteht bei diesem Muster eine Diskrepanz. So wird der Work Flow Manager nach (ebenda) als Integrationsmuster bezeichnet. Auf einer entsprechenden Abstraktionsebene kann man aber dennoch davon sprechen, dass durch dieses Muster Geschäftsfunktionalität automatisiert wird. Die Trennschärfe dieses Unterscheidungskriteriums steht und fällt mit dem Grad in dem – über das zusammenschalten von Services hinaus – Logik in den Workflow Manager integriert wird.
[64] Vgl. (Hohpe und Woolf, 2003).
[65] Vgl. (ebenda).

Pattern, Quelle	Strukturelemente	Verhalten, Rollen,Eigenschaften, Beziehungen	gelöster Integrationskonflikt	Überbr. Heterogenität	Org. Kommunikation
Interceptor, (Schmidt; Stal u. a., 2000)	IO:Framework, IC: Interceptor Callback Interface, Dispatcher	Ereignis gesteuerte Erweiterung von Frameworks zur Laufzeit	Entkopplung, dynamisches Einbinden	(X)	X
Integration Mediator, (Lutz, 2000)	IC:Integration Mediator, IO: applications	stateless/ statefull, Zentralisierung der Interaktionslogik	Kapselung, Entkopplung, Funktionszentralisierung	(X)	X
(Legacy) Wrapper, (Mularz, 1995)	IO:Applikationen, IC:Wrapper	stellt ein Interface zur Verfügung, welches sich wenig bis gar nicht ändert	Kapselung, Indirektion der Kommunikation, Ablösbarkeit der Applikationen	X	(X)
Work Flow Manager, (ebenda)	IO:Applikationen IC:Task Manager, Task Specifier, Locator, Functional Component Profile	Integrationsprobleme die als diskrete Schritte beschreibbar sind werden durch den Aufruf der Applikationen mit hilfe des Task Manager gelöst	Zentralisierung der Kontrollflusslogik	-	X
Shared Repository, (ebenda)	IC: Shared Repository, Importer, Exporter IO: Komponenten	über Importer und Exporter können Informationen aus dem Repository gelesen/geschrieben werden	Komponenten bearbeiten stark überlappende Informationen, min. Interfacezahl, max. Unabhängigkeit	X	X

IC: Integration Component (Integrationskomponente); IO: Integration Object (Integrationsobjekt);
X: Eigenschafft erfüllt; (X): Eigenschaft teilweise erfüllt bzw. Sekundäreigenschaft; -: nicht erfüllt

Tabelle 5.4: Evaluation der Eigenschaften explizit benannter Integrationsmuster

Alle der vorgestellten Muster beschreiben ausschließlich Komponenten einer Integrationsarchitektur und keinerlei Geschäftsfunktionalität. Die relative Positionierung in der Architektur wird durch die jeweiligen Integrationsobjekte möglich, da immer die Beziehung zu diesen beschrieben wird. Alle Muster befassen sich mit Integrationskonflikten. Es handelt sich immer um entweder ein Kommunikations- oder Heterogenitätsproblem oder um eine Mischform. Es überwiegen die reinen Kommunikationsprobleme und die Mischformen. Reine Heterogenitätsprobleme werden selten adressiert. Mithilfe der extrahierten Eigenschaften ließen sich alle Muster als Integrationsmuster identifizieren. Dies kann als positives Evaluationsergebnis der extrahierten Kriterien gewertet werden, mit denen im weiteren versucht werden wird, weitere Muster zu identifizieren.

Die folgende Tabelle 5.5 zeigt die Auswahl an Literatur, welche gesichtet wurde, um die dort enthaltenen Muster zu extrahieren und mit den herausgearbeiteten Merkmalen von Integrationsmustern zu vergleichen. Auf dieser Basis soll die Auswahl an Mustern für die weiteren Betrachtungen entstehen. Dabei kann kein objektives Vollständigkeitskriterium für die Menge der Muster angegeben werden. Es werden fortlaufend neue Publikationen veröffentlicht, welche Muster enthalten, von denen nicht jede Publikation gesichtet werden konnte.

Literaturquelle	Kurzbeschreibung, ggf. Anzahl	Auswahl
(Gamma u. a., 1993)	Entwurfsmuster	Nein
(Mularz, 1995)	Integrationsmuster, 4	Ja
(Coplien und Schmidt, 1995) PLOP1	verschiedene Konferenzbeiträge	(Ja), E
(Gamma u. a., 1995), GOF	Entwurfmuster, 23	Ja
(Buschmann; Meunier u. a., 1996), POSA1	Mustersystem für Software Architektur ,17	Ja

Übersicht über relevante Publikationen zu Mustern – Fortsetzung auf der nächsten Seite

Literaturquelle	Kurzbeschreibung, ggf. Anzahl	Auswahl
(Vlissides; Coplien und Kerth, 1996), PLOP2	verschiedene Konferenzbeiträge	(Ja), E
(Rohnert, 1996)	Entwurfsmuster	E
(Martin; Riehle und Buschmann, 1998), PLOP3	verschiedene Konferenzbeiträge	(Ja), E
(Keshav und Gamble, 1998)	Zuordnung von Mustern zu Strategien	Nein
(Harrison; Foote und Rohnert, 2000), PLOP4	verschiedene Konferenzbeiträge	(Ja), E
(Lutz, 2000)	Musterbeschreibungen für EAI, 5	Ja
(Schmidt; Stal u. a., 2000),POSA2	Muster für Middlewarekonstruktion	E
(Andersson und Johnson, 2001)	Integrationsstile für große verteilte Anwendungen, 6	Ja
(Dyson, 2001), EuroPlop	verschiedene Konferenzbeiträge	(Ja), E
(Adams u. a., 2001)	E-Business Pattern	(Ja), E
(Völter; Schmid und Wolff, 2002)	Serverkomponenten-Muster	Nein
(Hohpe und Woolf, 2003)	Patternsystem für Integrationsmuster, nachrichtenorientiert, 65	Ja
(Fowler, 2003)	Muster für Unternehmensapplikationen	Nein
(Trowbridge, 2003)	Integrationsmuster .Net	Ja
(Dustdar; Gall und Hauswirth, 2003)	Architekturmuster im wesentlichen aus der POSA Reihe	(Ja), E
(Trowbridge u. a., 2004)	Integrationsmuster, nachrichtenorientiert, 18	Ja
(Zdun, 2004)	Muster für Komponenten und Sprachintegration, starker POSA und Gamma Bezug	(Ja), E
(Hasselbring u. a., 2004)	Das Dublo Architekturmuster	(Ja), E
(Kircher und Jain, 2004)(POSA3)	Mustersystem für Ressourcenmanagement	Nein
(Kircher; Voelter u. a., 2004)	Neubetrachtung des Broker Muster	Ja
(Schwinn und Schelp, 2005)	abstrakte Muster zur Datenintegration	(Ja), E
(Völter; Kircher und Zdun, 2005)	Mustersystem für verteilte Kommunikation	(Ja), E
(Avgeriou und Zdun, 2005)	Mustersprache, Klassifikationsansatz	(Ja), E
(Marquardt, 2005), EuroPlop	verschiedene Konferenzbeiträge	(Ja), E
(Hentrich und Zdun, 2006)	SOA Muster und Beziehungen zueinanden	(Ja), E
(Longshaw und Zdun, 2006), EuroPlop	verschiedene Konferenzbeiträge	(Ja), E
(Manolescu; Voelter und Noble, 2006), PLOP5	verschiedene Konferenzbeiträge	(Ja), E
(Jung, 2006)	Architekturen zur Datenintegration	Nein
(Schwinn, 2006)	Musterverleich aus (Gamma u. a., 1995), (Lutz, 2000) und (Adams u. a., 2001)	(Ja), E
(Herden u. a., 2006)	Musterkatalog, ausschließlich Muster aus (Gamma u. a., 1995),(Fowler, 2003),(Buschmann; Meunier u. a., 1996) und (Hohpe und Woolf, 2003)	Nein
(Conrad u. a., 2006)	EAI Muster mit hauptsächlichem Bezug auf (Hohpe und Woolf, 2003), (Adams u. a., 2001)	Nein
(Hruby, 2006)	Geschäftsmuster vergleichbar mit den Mustern für Unternehmensanwendungen	(Ja), E
(Harrison und Avgeriou, 2007)	Beschreibung von Qualitätseigenschaften von Mustern	Nein
(Zdun und Hvatum, 2007), EuroPlop	verschiedene Konferenzbeiträge	(Ja), E
(Malich, 2008)	Patternsysteme, Qualitätseigenschaften von Mustern	(Ja), E
(Khomh und Guéhéneuc, 2008b)	Untersuchung von Qualitaätseigenschaften	(Ja)
(Khomh und Guéhéneuc, 2008a)	Untersuchung von Qualitaätseigenschaften	(Ja)
(Daniel und Steinrötter, 2008)	Muster für SAP PI die alle (Hohpe und Woolf, 2003) entstammen	Nein
(Reussner und Hasselbring, 2009)	in Kapitel 16 Architekturmuster, Verweise auf bekannte Kataloge	Nein
(Mandl, 2009)	keine Musterbeschreibung nur Architekturkonzeption	Nein
(Erl, 2009)	SOA Architekturmuster	Ja
(Vogel u. a., 2009)	Bezug auf Architekturstile, keine einzelnen Muster	(Ja), E
(Majidi; Alemi und Rashidi, 2010)	nur qualitative Analyse ohne Musterbeschreibung	Nein
(Eilebrecht und Starke, 2010)	Hauptsächlicher Bezug zu (Fowler, 2003) und (Gamma u. a., 1995)	(Ja), E
(Schatten u. a., 2010)	Vorrangiger Bezug zu (Hohpe und Woolf, 2003)	Nein

Ja - Publikation wurde ausgewählt, Nein - Publikation wurde nicht ausgewählt, (Ja) Publikation kommt nur teilweise in Frage, E - Publikation kommt in Frage kann aber nicht berücksichtigt werden

Tabelle 5.5: Übersicht über relevante Publikationen

Die relevanten Publikationen wurden nun nach Mustern durchsucht, welche mit den identifizierten Kriterien als Integrationsmuster einordenbar sind.

Die folgende Tabelle stellt Musternamе, Quelle sowie die Erfüllung der Eigenschaften vor und weist damit exemplarisch den Prozess aus, der für jedes Muster angewandt wird, bevor es in die Grundgesamtheit aufgenommen wird. Die explizit benannten Integrationsmuster sind natürlich in dieser Auswahl enthalten. Der Übersichtlichkeit halber sind die Muster nach den Publikationen organisiert.

Muster, Quelle	**Struktur-elemente**	**Verhalten, Rollen, Eigenschaften, Beziehungen beschrieben**	**eindeutiger Integrations-konflikt**	**Überbr. Heterogenität**	**Org. Kommunikation**
		(Lutz, 2000)			
Integration Adapter	X	X	X	X	(X)
Integration Messenger	X	X	X	-	X
Integration Facade	X	X	X	X	(X)
Integration Mediator	X	X	X	X	X
Process Automator	X	X	X	(X)	X
		(Andersson und Johnson, 2001)			
Adapter	X	X	X	X	(X)
Point-to-Point	X	-	-	-	X
Mediator	X	X	X	-	X
Gateway	X	X	X	X(thick)	X(thin)
Message Router	X	X	X	-	X
Database Federation	X	X	X	X	(X)
Desktop Integration	X	X	X	(X)	X
		(Trowbridge, 2003)			
		Web Presentation Patterns			
Model-View-Controller	(X)	(X)	-	-	(X)
Page Controller	(X)	(X)	-	-	(X)
Front Controller	(X)	(X)	-	-	(X)
Intercepting Filter	(X)	(X)	-	-	(X)
Page Cache	(X)	(X)	-	-	(X)
Observer	(X)	(X)	(X)	-	(X)
		Deployment Patterns			
Layered Application	-	-	-	-	(X)
Three Layered Services Application	-	-	-	-	(X)
Tiered Distribution	-	-	-	-	(X)
Three Tiered Distribution	-	-	-	-	(X)
Deployment Plan	-	-	-	-	-
		Distributed Systems Patterns			
Broker	X	X	X	-	X
Data Transfer Object	X	X	X	-	X
Singleton	(X)	(X)	(X)	-	(X)
		Services Patterns			
Service Interface	(X)	X	(X)	(X)	X
Service Gateway	X	X	X	X	X
		Performance and Reliability Patterns			
Server Clustering	-	-	-	-	X
Load-Balanced Cluster	-	-	-	-	X
Failover Cluster	-	-	-	-	X

IC: Integration Component (Integrationskomponente); IO: Integration Object (Integrationsobjekt);
X: Eigenschafft erfüllt; (X): Eigenschaft teilweise erfüllt bzw. Sekundäreigenschaft; -: nicht erfüllt

Tabelle 5.6: Prüfung der Integrationmustereigenschaften ausgewählter Quellen

Gateway, Desktop Integration, Message Router und die Database Federation werden als Sub-Stile bzw. Muster des generellen Mediator-Stiles bezeichnet.[66] Das Point-to-Point-Muster kann nicht ausgewählt werden, da es keine Integrationsarchitektur mit Komponenten gestaltet, die eigens für diese Funktionen benötigt werden, sondern die Integrationsfunktionalität auf die Integrationsobjekte verteilt.

Die Muster von Trowbridge[67] werden mit einem speziellen Fokus auf Active-Server-Pages für .Net (ASP.Net)[68] beschrieben. Einige der Muster sind auf der Ebene von Entwurfsmustern angesiedelt. Sie entsprechen dabei nicht exakt dem Begriff eines Integrationsmusters, liefern jedoch teilweise den Architekturmustern zugrunde liegende Konzepte. Sie werden deshalb nicht direkt ausgewählt, später aber für Vergleiche herangezogen. Deployment Plan ist ein Beispiel für ein Pattern auf der organisatorischen Ebene, es schlägt ein Meeting inclusive der zu besprechenden Inhalte zwischen verschiedenen an einer Entwicklung und einem Rollout beteiligten Gruppen vor.

Die Tabelle 5.5 stellt eine Sammlung an infrage kommenden Literaturquellen dar, welche prinzipiell in Frage kommen und auch durchgearbeitet wurden. Bereits die reine Menge an Mustern ist zu umfangreich, um in dieser Arbeit dargestellt werden zu können. Hinzu kommt, dass viele Muster wie z. B. der Integration Messenger in unterschiedlichen Ausprägungen beschrieben werden, welche für den Untersuchungszweck als einzelne Muster behandelt werden sollten. Aus diesem Grund war es notwendig, die Menge der zu untersuchenden Muster zu beschränken. Die Auswahl geschah teilweise im Hinblick auf Vorarbeiten[69], um die dort durchgeführten Arbeiten als Ausgangspunkt verwenden zu können. Die Auswahl beansprucht in keiner Weise, vollständig oder abschließend zu sein. Ein gewisser Grad an Willkürlichkeit wird bewusst in Kauf genommen. Die Tabelle 5.7 stellt die Menge der so erhaltenen Muster dar.

Pattern	**Quelle**	**IC/IO**	**B**	**IK**	**H**	**K**
Adapter Style	(Andersson und Johnson, 2001, S. 229)	X	X	X	X	(X)
Adapter static	(ebenda, S. 229)	X	X	X	X	(X)
Adapter dynamic	(ebenda, S. 229)	X	X	X	X	(X)
Adapter thin	(ebenda, S. 229)	X	X	X	X	(X)
Adapter thick	(ebenda, S. 229)	X	X	X	X	(X)
Adapter centralized	(ebenda, S. 229)	X	X	X	X	(X)
Adapter distributed	(ebenda, S. 229)	X	X	X	X	(X)
Adapter	(Khomh und Guéhéneuc, 2008a)	X	X	X	X	(X)
(Integration) Adapter	(Lutz, 2000)	X	X	X	X	(X)
Component Wrapper	(Goedicke und Zdun, 2002, S. 7ff.)	X	X	X	X	(X)
Wrapper Facade	(Schmidt; Stal u. a., 2000, S. 47ff.)	X	X	X	(X)	(X)
(Legacy) Wrapper	(Mularz, 1995)	X	X	X	X	(X)
Facade	(Khomh und Guéhéneuc, 2008a)	X	X	X	(X)	(X)
Integration Facade	(Lutz, 2000)	X	X	X	(X)	(X)
Service Interface	(Trowbridge, 2003)	X	X	X	-	X
Integration Messenger (broker)	(Lutz, 2000)	X	X	X	-	X
Integration Messenger (message queuing)	(ebenda)	X	X	X	-	X
Integration Messenger (publish subscribe)	(ebenda)	X	X	X	-	X
Proxy	(Buschmann; Meunier u. a., 1996, S. 263ff.)	X	X	X	-	X
Proxy	(Khomh und Guéhéneuc, 2008a)	n.a.	n.a.	n.a.	n.a.	n.a.
Mediator	(Gamma u. a., 1995, S. 277)	X	X	X	-	X
Integration Mediator	(Lutz, 2000, S. 71)	X	X	X	-	X

Grundgesamtheit der zu untersuchenden Muster – Fortsetzung auf der nächsten Seite

[66] Vgl. (Andersson und Johnson, 2001, S. 228).
[67] Vgl. (Trowbridge, 2003).
[68] Eine Technologie von Microsoft zum Erstellen von Webseiten, Webanwendungen und WebServices.
[69] Vgl. (Gebauer und Schmidt, 2012).

Pattern	Quelle	IC/IO	B	IK	H	K
Mediator	(Buschmann; Henney und Schmidt, 2007a, S. 411)	X	X	X	-	X
Mediator	(Khomh und Guéhéneuc, 2008a)	n.a.	n.a.	n.a.	n.a.	n.a.
Mediator Style	(Andersson und Johnson, 2001, S. 230)	X	X	X	-	X
Message Router	(ebenda, S. 231)	X	X	X	-	X
Message Router	(Hohpe und Woolf, 2003, S. 81)	X	X	X	-	X
Message Router	(Buschmann; Henney und Schmidt, 2007a, S. 232)	X	X	X	-	X
Gateway	(Andersson und Johnson, 2001)	X	X	X	-	X
Service Gateway	(Trowbridge, 2003, S. 293ff.)	X	X	X	-	X
Broker	(ebenda, S. 207f.)	X	X	X	-	X
Direct Broker	(Trowbridge u. a., 2004, S. 155,215ff.)	X	X	X	-	X
Indirect Broker	(ebenda, S. 221)	X	X	X	-	X
Message Broker	(ebenda, S. 240f.)	X	X	X	-	X
Message Broker	(Hohpe und Woolf, 2003, S. 322ff.)	X	X	X	-	X
Broker	(Buschmann; Meunier u. a., 1996, S. 119ff.)	X	X	X	-	X
Broker Revisited	(Kircher; Voelter u. a., 2004, S. 581)	X	X	X	-	X
Broker	(Harrison und Avgeriou, 2007, S. 267f.)	X	X	X	-	X
Pipes and Filter	(Hohpe und Woolf, 2003, S. 70ff., 322ff.)	X	X	X	-	X
Pipes and Filter	(Buschmann; Meunier u. a., 1996, S. 67ff.)	X	X	X	-	X
Pipes and Filter	(Trowbridge u. a., 2004, S. 292f.)	X	X	X	-	X
Pipes and Filters	(Garlan und Shaw, 1993, S. 7f.)	X	X	X	-	X
Pipes and Filters	(Meunier, 1995, S. 437)	X	X	X	-	X
Pipes and Filters	(Harrison und Avgeriou, 2007, S. 267f.)	X	X	X	-	X
Observer	(Khomh und Guéhéneuc, 2008a)	n.a.	n.a.	n.a.	n.a.	n.a.
Observer	(Trowbridge, 2003, S. 129f.)	X	X	X	-	X
Publish and Subscribe	(Trowbridge u. a., 2004, S. 280f.)	X	X	X	-	X
Process Automator	(Lutz, 2000)	X	X	X	-	X
Database Federation	(Andersson und Johnson, 2001)	X	X	X	X	-
Desktop Integration	(ebenda)	X	X	X	-	X
Data Transfer Object	(Trowbridge, 2003, S. 229)	X	X	X	-	X
Singleton	(ebenda, S. 257)	X	X	X	-	X
Singleton	(Khomh und Guéhéneuc, 2008a)	n.a.	n.a.	n.a.	n.a.	n.a.
File Gateway	(Erl, 2009, S. 457)	X	X	X	-	X
Process Centralizazion	(ebenda, S. 193)	X	X	X	-	X
Protocol Bridging	(ebenda, S. 687)	X	X	X	X	-
Legacy Wrapper	(ebenda, S. 441)	X	X	X	(X)	X
Service Broker (COMP)	(ebenda, S. 707)	X	X	X	X	X
Enterprise Service Bus (COMP)	(ebenda, S. 704)	X	X	X	X	X
Proxy Capability	(ebenda, S. 497)	X	X	X	-	X
Service Facade	(ebenda, S. 333)	X	X	X	(X)	(X)
UI Mediator	(ebenda, S. 366)	X	X	X	-	X
Data Model Transformation	(ebenda, S. 671)	X	X	X	X	-
Data Format Transformation	(ebenda, S. 681)	X	X	X	X	-
Asynchronous Queuing	(ebenda, S. 582)	X	X	X	-	X
Intermediate Routing	(ebenda, S. 549)	X	X	X	-	X

IC: Integration Component; IO: Integration Object; B:Beschreibung; IK: Integrationskonflikt; H:Heterogenität; K: Kommunikation; Eigenschaft ist X=erfüllt; - nicht erfüllt; (X) partiell erfüllt bzw. Seiteneffekt; n.a.: nicht angebbar

Tabelle 5.7: Grundgesamtheit der zu untersuchenden Muster

Auffällig in der Zusammenfassung sind die Muster, für welche nicht angebbar bei allen Eigenschaften ausgewiesen wird. Dieser Umstand entsteht durch die Herkunft der Muster aus Quellen, die sich nur mit nicht funktionalen Eigenschaften befassen. Sie sind der Vollständigkeit halber enthalten, um die gesamte Grundgesamtheit zu präsentieren. Diese Zusammenfassung bildet die Grundgesamtheit für die nachfolgenden Bemühungen, den Abstraktionsgrad und damit den Zugang, zu dem so konservierten Lösungswissen, zu verbessern. In Kapitel 8 werden die Möglichkeiten einer Ausdehnung dieser Menge und die Untersuchung weiterer Publikationen diskutiert.

5.3 Muster und Abstraktion

Der Zusammenhang zwischen Mustern und Abstraktion, sowie weitere Möglichkeiten der Abstraktion sind Gegenstand dieses Kapitels. Der eingangs gestellten These dieser Arbeit folgend, dass wiederverwendbares Wissen zur Integration zwar existiert aber unzureichend abstrahiert vorliegt, untersucht dieser Abschnitt die grundsätzlichen Mechanismen der Abstraktion. Welche Mechanismen sind bereits in dem Konzept eines Musters enthalten und welche eignen sich um die Einsetzbarkeit der Muster weiter zu verbessern.

5.3.1 Abstraktionsarten

Die Abstraktion ist eine grundlegende Technik des menschlichen Denkens zur Organisation und Generierung von Wissen[70] und dient dem Hervorheben bzw. Filtern des in einer Situation Wesentlichen.[71] Bereits das Übertragen einer Lösung von einer Situation in eine Andere setzt diese Fähigkeit voraus. Unwesentliche Eigenschaften der Situationen müssen weggelassen und die Gemeinsamkeiten, welche die Anwendung der Lösung erlauben, erkannt werden. Neben der Mathematik, welche nach Matros 8 Abstraktionsarten kennt[72], wird die Abstraktion auch in der Informatik und Wirtschaftsinformatik diskutiert, vor allem im Bereich der Modellbildung und Modellierung. Beispiele sind die Entwicklung einer Sprache für die Anforderungsspezifikation, bei der Abstraktion eine Rolle im Rahmen der Strukturierung von Anforderungsmodellen spielt[73], und die Entwicklung eines stochastischen Risikomodells für komplexe Projekte[74]. Häufig ist der Zweck der Abstraktion die Vereinfachung und die übersichtliche Organisation eines Systems als Modell.[75] Dabei muss die Gültigkeit bezüglich des betrachteten Aspekts erhalten bleiben, um gültige Aussagen anhand des Modells zu ermöglichen.[76] Leppänen konstatiert jedoch, dass die Begriffe, Abstraktionsprinzipien in verschiedenen Teilgebieten – unter anderem im Software Engineering, im Bereich Datenbanken – derart unterschiedlich gebraucht werden, dass viel Konfusion entsteht.[77] Im Folgenden werden auf der Basis der Semiotik verschiedene Abstraktionsprinzipien vorgestellt.

Die Semiotik ist die Lehre der Verwendung von Zeichen.[78] In der Abbildung 5.1 wird das semiotische Dreieck nach Odger[79] dargestellt, welches in der Tradition von Peirce[80], Saussure[81] und Frege[82] steht[83]. Das semiotische Dreieck erklärt die Beziehungen zwischen einem Gegenstand (einem Ding der realen Welt), einem Symbol und einem Konzept (Begriff). Konzept und Begriff sind mentale Dinge und synonym für Zeichen zu sehen. Sie sind Erklärungsgegenstand der Semiotik.

Ein einfaches Beispiel sind Stühle an einem Tisch. Jeder ist ein Gegenstand der realen Welt mit unterschiedlicher Form, Farbe und Größe. Das Konzept ist es, eine Sitzfläche mit einer Lehne und vier Beinen zu verbinden, sodass ein Mensch darauf sitzen kann. Losgelöst von einem konkreten Gegenstand wird unter dem Wort Stuhl eine Repräsentation des Konzeptes verstanden.

[70] Vgl. (Kurpjuweit, 2009, S. 12).
[71] Vgl. (Frantz, 1995), (Kapici, 2005, S. 33).
[72] Vgl. (Matros, 2006).
[73] Vgl. (Joos, 2000).
[74] Vgl. (Kapici, 2005).
[75] Vgl. (Lee und Fishwick, 1996, S. 764).
[76] Vgl. (Frantz, 1995, S. 1413).
[77] Vgl. (Leppänen, 2007, S. 166).
[78] Vgl. (Eco, 1994).
[79] Vgl. (Ogden und Richards, 1974).
[80] Vgl. (Eco, 1994).
[81] Vgl. (Amsterdamska, 1987).
[82] Vgl. (Biro und Kotatko, 1995).
[83] Vgl. (Klettke u. a., 2001, S. 15).

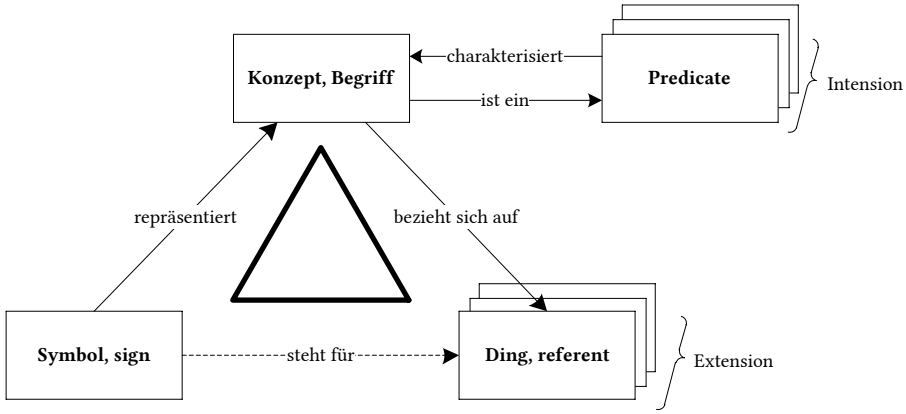


Abbildung 5.1: Semiotisches Dreieck

Quelle: In Anlehung an (Ogden und Richards, 1974) und (Leppänen, 2007, S. 168f.)

Die *Intension* eines Konzeptes ist nach Leppänen die Summe der Prädikate bzw. Eigenschaften, welche das ursprüngliche Konzept charakterisieren. Prädikate stellen selbst Konzepte dar. Die Intension eines Konzeptes (Summe aller Prädikate) kann als Idee verstanden werden, bei der die Entfernung auch nur eines Teils unweigerlich die Zerstörung der Idee zur Folge hat.[84]

Als *Extension* eines Konzeptes wird die Menge aller referenzierten Dinge bezeichnet, auf die sich das Konzept bezieht. Existieren diese Dinge aktuell in der Realität, so spricht man von der Population des Konzeptes.[85] Konzepte, welche nichts referenzieren, sich momentan auf nichts beziehen, sind abstrakte Konzepte. Leppänen bezeichnet Konzepte, die viele Dinge referenzieren, also eine große Population haben, als generische Konzepte, welche auch als Typen bezeichnet werden. Umgekehrt sind die Elemente der Extension eines solchen Konzeptes Instanzen.[86]

Abstraktion beschrieben in den Termini des semiotischen Dreiecks lässt sich in zwei Kategorien unterteilen, je nach dem, an welcher Ecke Sie ansetzt. Die Abstraktion erster Ordnung (first-order abstraction) setzt an den Konzepten an und bildet Typen, Subtypen, Ganzheiten und Gruppen.[87] Die Abstraktion von einzelnen Personen hin zu einer Gruppe von Personen ist ein Beispiel für eine derartige Abstraktion. Die Zweite Kategorie der Abstraktion ist die der zweiten Ordnung (second-order abstraction). Diese abstrahiert nicht von dem eigentlichen Konzept, sondern von den Prädikaten. Die Verwendung von Perspektiven, Views und Viewpoints steht exemplarisch für diese Kategorie der Abstraktion.[88]

Im Weiteren werden einzelne Abstraktionsmechanismen vorgestellt und es wird dargelegt, wie diese Mechanismen funktionieren. Zu diesem Zweck werden die so eben eingeführten Termini verwendet. Die Tabelle 5.8 stellt eine Übersicht der Abstraktionsmechanismen zusammen und ordnet die ausgewerteten Publikationen zu.

Einige Arbeiten wurden nicht in die Übersicht aufgenommen, da die dort beschriebenen Abstraktionsmechanismen nach anderen Gesichtspunkten gebildet wurden. So legt Kapici[89] als Abstraktionsmechanismen die Reduktion und die Idealisierung dar. Die Reduktion kann der

[84] Vgl. (Leppänen, 2007, S. 168).
[85] Vgl. (ebenda, S. 168).
[86] Vgl. (ebenda, S. 168).
[87] Vgl. (ebenda, S. 169).
[88] Vgl. (ebenda, S. 169).
[89] Vgl. (Kapici, 2005).

Publikation	Generalisierung	Klassifikation	Komposition	Gruppierung	Benutzung	Typisierung	Aggreation	ordnende	analogisierende	formalisierende	objektivierende	algorithmisierende	interpretierende
(Glinz, 2005)	X		X		X								
(Matros, 2006)	X	X						X	X	X	X	X	X
(Kühne, 2006)	X	X											
(Leppänen, 2007)	X	X	X	X									
(Kurpjuweit, 2009)	X			X		X	X						
(Uzquiano, 2010)	X	X	X		X								

Tabelle 5.8: Übersicht der Abstraktionsmechanismen

Abstraktion zweiter Ordnung zugerechnet werden, wenn sie Prädikate – die momentan nicht von Interesse sind – ausblendet, was als Vermutung gewertet werden kann. Die Idealisierung wird als eine vereinfachte Darstellung der Beschreibungselemente charakterisiert. An dieser Stelle kann nur vermutet werden, dass es sich um die Konzeptabstraktion, die Abstraktion erster Ordnung handelt.

Thalheim[90] stellt das Was – von dem abstrahiert wird – und nicht das Wie in den Vordergrund. Die vorgestellten Abstraktionsmechanismen sind die Komponentenabstraktion, die Lokalisierungsabstraktion und die Implementationsabstraktion. Ein ähnliches Bild ergibt sich bei Frantz[91], hier taucht z. B. die Komposition bzw. Aggregation als State-Aggregation und als Entity-Aggregation auf. Analog ist die Arbeit von Lee und Fishwick zu sehen[92], welche zwischen Daten-, Struktur- und Verhaltensabstraktion unterscheiden. Diese Systematisierungen lassen es nicht zu, dass Wie der Abstraktion – was Ziel dieses Abschnitts ist – zu kategorisieren, weshalb diese Arbeiten ausgeblendet werden.

Obwohl in der Übersicht enthalten, geht die Arbeit von Uzquiano[93] nicht über eine Nennung der Abstraktionsmechanismen hinaus und kann so keine näheren Details zu den Abstraktionsmechanismen beitragen. In der weiteren Darstellung werden die Abstraktionsmechanismen aus Tabelle 5.8 weiter erläutert.

Die Abstraktion durch **Generalisierung** geht von einem gemeinsamen Merkmal[94] aus und blendet die Verschiedenartigkeit anderer Merkmale aus. Ziel ist die Zusammenfassung von Elementen durch ein allgemeineres Element.[95] Aus einer Generalisierung resultiert eine ist-ein Beziehung zwischen den ursprünglichen Elementen und dem generalisierten Element. Die inverse Operation ist die Spezialisierung.[96] Die ist-ein Beziehung ist reflexiv, transitiv und antisymmetrisch.[97] Die generalisierende Abstraktion ist die häufigste Art der Abstraktion. Neben der abduktiven und induktiven Regelerfassung gehört auch die deduktive Regelerfassung zu dieser Form der Abstraktion. Weiterhin wird angemerkt, dass die generalisierende Abstraktion so

[90] Vgl. (Thalheim, 2003).
[91] Vgl. (Frantz, 1995).
[92] Vgl. (Lee und Fishwick, 1996).
[93] Vgl. (Uzquiano, 2010).
[94] Vgl. (Glinz, 2005, S. 96).
[95] Vgl. (Kühne, 2006, S. 374), sowie (Kurpjuweit, 2009, S. 15).
[96] Vgl. (ebenda, S. 15) und (Matros, 2006, S. 1) Dieser Umstand wird teilweise hinterfragt. Zur kotra-intuitiven Semantik von Generalisierung und Spezialisierung siehe (Frank, 2003).
[97] Vgl. (Kurpjuweit, 2009, S. 15f.).

grundlegend ist, dass alle anderen Arten der Abstraktion als Facetten angesehen werden können.[98] Die Generalisierung bildet eine Grundvoraussetzung für die Klassifikation[99], da sich diese immer auf ein gemeinsames, abstrahiertes Merkmal aller – in die Klasse eingeordneten – Elemente bezieht. Diese Abstraktionsoperation kann iterativ angewendet werden, was zur Bildung von Typhierarchien genutzt wird.[100] In der vorliegenden Arbeit wird die Generalisierung erneut im Anschluss an die Klassifikation angewendet, was einer Typhierarchie entspricht und die zweifache Anwendung der Generalisierung bedeutet, da die Klassifikation eine Generalisierung voraussetzt.

Die klassifizierende Abstraktion (**Klassifikation**) stellt auf ein bereits generalisiertes Merkmal ab und ordnet alle Elemente mit gleicher Ausprägung des Merkmals einer Klasse zu.[101] Andere Merkmale, welche die Elemente nicht gemeinsam haben, werden ausgeblendet. Das Ergebnis der Klassifikation ist ein Typ, denn: "Classification is the principle of abstraction by which the concept c^{ty}, called the *type*, is generated from other concepts c_i^{in}, called instances"[102]. Kurpjuweit verwendet den Begriff Typisierung anstelle von Klassifikation[103], welcher als Synonym gelten kann. Hesse spricht in diesem Zusammenhang von Stellvertreterprojektion.[104] Die entstehende Beziehung zwischen den Elementen und den gebildeten Typen ist eine *ist-Instanz-von* Beziehung, die inverse Operation ist die Instanziierung. Die Beziehung *ist-Instanz-von* ist nicht reflexiv, nicht transitiv und antisymmetrisch.[105] Eine iterative Anwendung ist möglich[106], wobei dann Metatypen entstehen, deren Instanzen Typen sind.[107]

Abstraktion durch **Komposition** – synonym auch als Aggregation bezeichnet – ist das Prinzip ein Ganzes aus Teilen zu bilden.[108] Die Zusammenfassung erfolgt dabei unter Weglassung von Details, mit dem Ziel in sich möglichst geschlossene Komponenten mit einer hohen Kohäsion zu erhalten.[109] Entscheidendes Merkmal der Komposition ist, dass die einzelnen Elemente die ein Ganzes bilden miteinander in Beziehung stehen.[110] Dieses Prinzip kann sowohl auf der Ebene von Typen – wenn der Aggregation eine Klassifikation vorausgegangen ist – als auch auf der Ebene von Instanzen angewendet werden.[111] Damit ist es unabhängig von einer vorausgegangenen Abstraktion. In Abhängigkeit ob Typen oder Instanzen komponiert werden, bestimmen sich jedoch einige Eigenschaften der resultierenden *partOf*-Beziehung, welche zwischen dem Aggregat und seinen Elementen besteht. Die *partOf*-Beziehung ist auf der Ebene der Instanzen nicht-reflexiv und sowohl auf der Ebene der Instanzen als auch auf der Ebene der Typen antisymmetrisch.[112] Die inverse Operation zur Komposition ist die Dekomposition[113], bei der das Aggregat in seine – miteinander in Beziehung stehenden – Teile zerlegt wird. Weiterhin kann die *partOf*-Beziehung in Abhängigkeit des Grades der Teilung und der Abhängigkeit des Ganzen von den Teilen gegliedert werden. Diese Betrachtungen spielen für die vorliegende Arbeit jedoch keine Rolle, sodass auf

[98] Vgl. (Matros, 2006, S. 1f.).
[99] Vgl. (ebenda).
[100] Vgl. (Kurpjuweit, 2009, S. 16).
[101] Vgl. (Matros, 2006, S. 2).
[102] (Leppänen, 2007, S. 170)
[103] Vgl. (Kurpjuweit, 2009, S. 14).
[104] Vgl. (Hesse, 2006, S. 389).
[105] Vgl. (Kurpjuweit, 2009, S. 14), (Leppänen, 2007, S. 170).
[106] Vgl. (Kurpjuweit, 2009, S. 14).
[107] Vgl. (Leppänen, 2007, S. 170).
[108] Vgl. (Kurpjuweit, 2009), (Henderson-Sellers und Barbier, 1999).
[109] Vgl. (Glinz, 2005, S. 94).
[110] Vgl. (Leppänen, 2007, S. 174).
[111] Vgl. (ebenda, S. 169).
[112] Vgl. (Henderson-Sellers und Barbier, 1999, S. 22), (Leppänen, 2007, S. 174f.), (Kurpjuweit, 2009, S. 17).
[113] Vgl. (Leppänen, 2007, S. 174).

Leppänen[114] und Kurpjuweit[115] verwiesen wird. Die Abstraktion durch Komposition kann iterativ angewendet werden, wodurch Kompositionshierarchien entstehen. Die *partOf*-Beziehung in einer solchen Hierarchie muss nicht transitiv sein.[116]

Bei der **Gruppierung** werden mehrere Elemente zu einer Gruppe zusammengefasst. Im Gegensatz zur Komposition stehen die Elemente, welche in einer Gruppe sind (*member-of*), nicht miteinander in Beziehung und sind in der Regel gleichartig.[117] Diese Unterscheidung zwischen *partOf* und *member-of*– also Komposition und Gruppierung – wurde bereits von Motschnig-Pitrik und Kaasbøll angeregt.[118] Die Gruppe als Ganzes ist Gegenstand des Interesses und wird anstelle der einzelnen Mitglieder untersucht.[119] Die inverse Operation ist die Individualisierung. Die Relation *member-of* ist nicht-reflexiv, antisymmetrisch und nicht-transitiv. Eine iterative Anwendung ist möglich.[120] Eine weitere Unterscheidung verschiedener Arten der Gruppierung ist anhand der Kardinalität der *member-of* Beziehung möglich. Leppänen unterscheidet dabei die Paare *Group type/Group instance* und *Member type/Member instance*, welche jeweils durch eine *instance-of* Beziehung verbunden sind.[121] Damit ist die Klassifikation – und indirekt auch die Generalisierung – die Voraussetzung für diese Abstraktionsart.

Bis zu diesem Punkt sind alle vorgenannte Abstraktionsarten der Abstraktion erster Ordnung zuzuordnen. Obwohl die Abstraktion zweiter Ordnung sich auf die Prädikate der Konzepte bezieht, finden dort dieselben Abstraktionsmechanismen Anwendung.[122]

Die Abstraktion durch **Benutzung** wird nur von Glinz dargelegt.[123] Sie baut auf der Komposition auf und stellt auf Hierarchien von Komponenten ab. Für die Benutzung einer Komponente ist die Kenntnis des Aufbaus – genauer die Zusammensetzung – der Komponente nicht von Bedeutung. X *uses* Y entspricht dabei der Aussage, dass $X = Y$ *komponiert mit* Z. Dabei ist Z genau der Teil von X, der nicht in Y enthalten ist. Die Benutzung wird vor allem eingesetzt um voneinander relativ unabhängige Schichten zu bilden.[124] Wendet man diese Abstraktion auf der Basis einer Komposition an, welche auf Typen beruht, so ist mittelbar eine Generalisierung Voraussetzung für diese Abstraktionsart.

Die **ordnende Abstraktion** setzt die generalisierende und klassifizierende Abstraktion voraus. Es wird ein Merkmal betrachtet, jedoch dessen unterschiedliche Ausprägungen in den Fokus gerückt. Es können Reihenfolgen oder Dinge einander unter- bzw. übergeordnet werden.[125]

Im Rahmen der **analogisierenden Abstraktion** wird ebenfalls ein Merkmal in den Fokus gerückt, dieses wird jedoch in verschiedenen Gegenstandsbereichen betrachtet und von einem Bereich auf den anderen übertragen. Dabei kann es notwendig sein, das Merkmal im Zielgegenstandsbereich gegebenen Falls neu zu erzeugen.[126]

[114] Vgl. (ebenda, S. 175).
[115] Vgl. (Kurpjuweit, 2009, S. 18).
[116] Vgl. (Barbier und Henderson-Sellers, 2001, S. 22).
[117] Vgl. (Leppänen, 2007, S. 176).
[118] Vgl. (Motschnig-Pitrik und Kaasbøll, 1999, S. 794).
[119] Vgl. (Leppänen, 2007, S. 176).
[120] Vgl. (ebenda, S. 177).
[121] Vgl. (ebenda, S. 177).
[122] Vgl. (ebenda, S. 181).
[123] Vgl. (Glinz, 2005).
[124] Vgl. (ebenda, S. 95).
[125] Vgl. (Matros, 2006, S. 2).
[126] Vgl. (ebenda, S. 2).

Die **formalisierende Abstraktion** Abstraktion bildet den Kernbereich der Mathematik und kann auch als idealisierende Abstraktion aufgefasst werden. So werden Symbole als Platzhalter für tatsächliche Operationen verwendet.[127]

Der Vorgang der **objektivierenden Abstraktion** bezieht sich auf den Versuch, die Subjektivität aus dem Abstraktionsprozess heraus zu streichen. Aus ich habe x gemacht, deshalb folgt y wird wenn x dann y. Der Subjektbezug in Form des Handlungsträgers wird eliminiert.[128]

Als **algorithmisierende Abstraktion** bezeichnet man eine Abstraktionsart, welche für die Informatik – aber nicht nur – herausragende Bedeutung hat, stellt doch die Bildung von Algorithmen einen wesentlichen Teil dieser Wissenschaft dar. Im Fokus steht die Bildung einer Kette von Einzelhandlungen zur Erreichung eines Handlungsziels. Die Abstraktionsleistung besteht darin, notwendige von nicht notwendigen Bedingungen und sinnvolle von nicht sinnvollen Verfahrensweisen zu trennen.[129]

Der Einsatz der **interpretierenden Abstraktion** erfolgt bei der Signifikation. Immer dann, wenn ein Zeichen als Symbol verwendet wird, erhält es Bedeutung für ein anderes.[130]

Die vorangegangenen Ausführungen haben verschiedene in der Wissenschaft bekannte Abstraktionsarten dargelegt, wobei das Wie der Abstraktion – also den Abstraktionsmechanismus – in den Vordergrund gerückt wurde. Im nächsten Kapitel wird nun analysiert, welche Abstraktionsmechanismen das Konzept eines Musters bereits impliziert und welche Mechanismen darüber hinaus im Zusammenhang mit Mustern Anwendung finden.

5.3.2 Implizite und in der Literatur verwendete Abstraktionsarten

Die Darstellungen in diesem Kapitel lassen sich in zwei Teile gliedern. Zum einen stellen Muster bereits eine implizite Abstraktion dar, was im Abschnitt 5.3.2 dargelegt wird. Zusätzlich dazu existieren Bemühungen, Muster an sich weiter zu systematisieren und zu abstrahieren. Einen wesentlichen Anteil an dieser Abstraktion haben Klassifikationen von Mustern, welche in Abschnitt 5.3.2 analysiert werden.

Implizite Abstraktionseigenschaften von Mustern

Ein Muster selbst stellt bereits eine Abstraktion dar. Für die weitere Analyse der Abstraktionseigenschaften eines Musters erfolgt eine Orientierung an der von Alexander eingeführten Beschreibungsform[131], welche auch alexandrinische Form genannt wird[132]. Die Tabelle 5.9 listet die Elemente dieser Beschreibung auf und ordnet diesen die Elemente des semiotischen Dreiecks[133] zu.

Der Name eines Musters kann als Symbol gewertet werden, denn in der Praxis steht er stellvertretend für das – durch das Muster verkörperte – Lösungsprinzip. Das Lösungsprinzip (*Solution*) ist in der Sprache der Semiotik das Konzept. Die Elemente *Problem*, *Context*, *Forces*, *Examples*, *Resulting-Context*, *Rationale* und *Related-Patterns* lassen sich als Prädikate des (Lösungs-)Konzeptes eines Muster interpretieren, die das Konzept näher charakterisieren. Die Extension wird durch *Known use* verkörpert, da es sich um die Anwendung des Lösungskonzeptes bzw.

[127] Vgl. (Matros, 2006, S. 3f.).
[128] Vgl. (ebenda, S. 4f.).
[129] Vgl. (ebenda, S. 5).
[130] Vgl. (ebenda, S. 6ff.).
[131] Vgl. (Alexander, 1979).
[132] Vgl. (Tešanović, 2004, S. 6).
[133] Vgl. Abbildung 5.1.

Beschreibungs-element	Erklärung	Element des semiotischen Dreiecks
Name	sprechender Bezeichner	Symbol
Problem	Fragestellung, Problem welches das Muster lösen will	Predicate
Context	Die Situation welche das Problem hervorruft	Predicate
Forces	Kräfte die in dem Szenario wirken	Predicate
Solution	Lösung des Problems	Concept
Examples	Eine Beispielanwendung der Lösung	Predicate
Resulting context	Resualtat nach Anwendung des Musters	Predicate
Rationale	Beschreibung von Schritten oder Regeln des Musters	Predicate
Related patterns	statische und dynamische	Predicate
Known use	Verwendung der Muster in existierenden Systemen	Extension

Tabelle 5.9: Alexandrinische Normalform
Quelle: In Anlehnung an (Alexander, 1979)

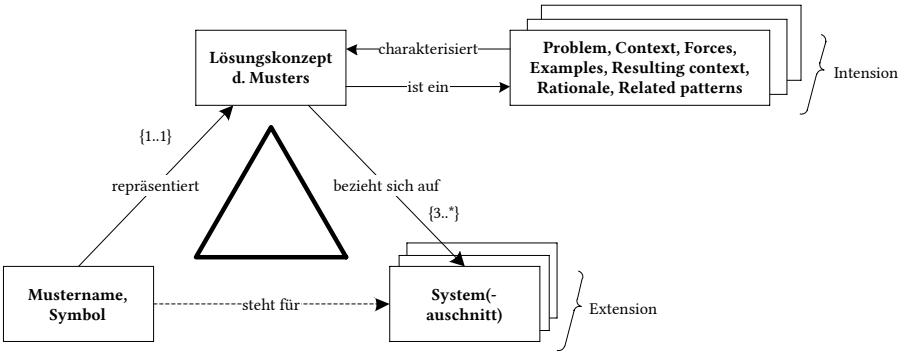


Abbildung 5.2: Semiotisches Dreieck für Muster

Musters in existierenden Systemen handelt. Mit dem Bezug zu einem existierenden System – dem Original – erfüllen Muster das Abbildungsmerkmal, welches eines der konstituierenden Merkmale von Modellen ist.[134] Muster müssen aber ein System nicht vollständig erfassen, sie stellen also eine Reduktion des betrachteten Realweltausschnittes dar. Sie erfüllen damit die Eigenschaft des Verkürzungsmerkmals, welches für Modelle gelten muss.[135] Das letzte konstituierende Merkmal eines Modells ist das pragmatische Merkmal[136], welches besagt, dass ein Modell immer von einem Ersteller zu einem bestimmten Zweck geschaffen wurde und deshalb einem pragmatischen Zweck folgt. Diese kann bei Mustern als gegeben angesehen werden, obwohl durch die dreimalige Existenz des Lösungskonzeptes in der Realität und den typischen Prozess der Patternentstehung eine gewisse Intersubjektivität ergibt. In diesem Fall bezieht sich das pragmatische Merkmal auf eine Gruppe von Personen. Die Abbildung 5.2 gibt die Einordnung der Konzepte eines Musters in das semiotische Dreieck wieder.

Mit dem Nachweis der Modelleigenschaft eines Musters und der Einordnung der Konzepte in das semiotische Dreieck können nun einige Formalismen einführt werden, die von Kühne[137] stammen. Diese werden um Beschreibungen von Leppänen[138] und eigene Ergänzungen erweitert. Sei S das betrachtete System bzw. Systemausschnitt, so ist M das (Lösungskonzept) bzw. Muster, ein

[134] Vgl. (Stachowiak, 1973).
[135] Vgl. (ebenda).
[136] Vgl. (ebenda).
[137] Vgl. (Kühne, 2006).
[138] Vgl. (Leppänen, 2007).

Modell des Systems. Kühne verwendet dafür $\triangleleft$ als Symbol für die Relation zwischen System und Modell.

$$\begin{aligned} S &\triangleleft M \\ \rho\{S, M\} &\rightarrow S \triangleleft M \\ M &= \alpha(S) \end{aligned} \tag{5.1}$$

Die Relation ρ ist eine Untermenge von $\triangleleft$ und enthält nur diejenigen Repräsentationen eines Systemausschnittes, welche nicht zufällig entstanden sind. $\alpha(S)$ drückt aus, dass mit einem Systemausschnitt etwas geschieht, wenn er in Form eines Musters bzw. Modells abgebildet wird.

$$\alpha = \tau \circ \alpha' \circ \pi^{139} \tag{5.2}$$

π ist die Projektion vom Original auf eine Repräsentation und τ stellt die Übersetzung der Bestandteile des Systems in z. B. Elemente einer grafischen Repräsentation dar. α' ist dabei eine mögliche Abstraktion, die verwendet werden kann.

In Abbildung 5.2 sind Kardinalitäten an zwei Relationen angetragen. Ein Mustername sollte immer nur genau ein Muster bzw. Lösungskonzept repräsentieren, was mit *{1..1}* veranschaulicht wird.

An der Relation *bezieht sich auf* ist die Kardinalität *{3..*}* vermerkt, welche eine konstituierende Eigenschaft von Mustern – bekannt unter dem Namen *The rule of three*[140] – repräsentiert. Diese Regel fordert, dass ein Muster 3 voneinander unabhängige Verwendungen in existierenden Systemen nachweisen muss.[141] Daraus kann geschlussfolgert werden, dass:

$$\alpha' = \Gamma \circ \Lambda^{142} \tag{5.3}$$

Γ ist eine Generalisierungsfunktion mit einer Äquivalenzrelation $\sim_\Gamma$. Seien s_1 und s_2 Elemente von S so bedeutet die Projektion auf dasselbe Modellelement – die Generalisierung von s_1 und s_2 – nicht, dass diese beiden Elemente auch gleich sein müssen: $\pi(s_1) = \pi(s_2) \not\rightarrow s_1 = s_2$. Es gilt aber: $s_1 \sim_\Gamma s_2 \rightarrow \pi(s_1) = \pi(s_2)$.[143] Λ hingegen ist eine Klassifikationsfunktion mit einer Equivalenzrelation $\sim_\Lambda$. Unter der Voraussetzung, dass ein Muster M aus mehreren Elementen m_i bestehen kann, gilt: wenn $\pi(s_1) = \pi(s_2)$ und $s_1 \sim_\Lambda s_2 \leftrightarrow \Lambda(s_1) = \Lambda(s_2) = m_1$.

Damit können Muster zusammenfassend in die Kategorie der Typmodelle eingeordnet werden, da sie den Ausprägungen der Realität (Extension) einen Typ (Muster) zuordnen.[144] Es müssen demnach eine Projektion, eine Generalisierung und eine Klassifizierung stattgefunden haben, um zu einem Muster zu gelangen. Anders formuliert implizieren Muster bereits die Abstraktionsarten Generalisierung und Klassifikation. Diese finden streng genommen eine zweimalige Anwendung bei der Konstruktion eines Musters, wie die Abbildung 5.3 verdeutlicht.

Die in den Systemen enthaltenen Lösungskonzepte werden von den einzelnen Systemen – welche zu der Extension eines Musters $\epsilon(M)$ gehören – abstrahiert zu einem Muster M also einem Lösungstyp. Die Intension des Musters $\iota(M)$ stellt einen Problemtypen dar, der aus der Abstraktion der Einzelprobleme – welche zu den Systemen der Extension gehören von $\epsilon(M)$ – herrührt. Da es

[139] (Kühne, 2006, S. 2)
[140] Vgl. (Tešanović, 2004, S. 2).
[141] Vgl. auch (ebenda, S. 7), sowie (Winn und Calder, 2002, S.64).
[142] (Kühne, 2006, S. 6)
[143] Vgl. (ebenda, S. 5).
[144] Im Gegensatz zu Tokenmodellen, welche nur eine Projektion und Generalisierung erfordern. Vgl. dazu (ebenda).

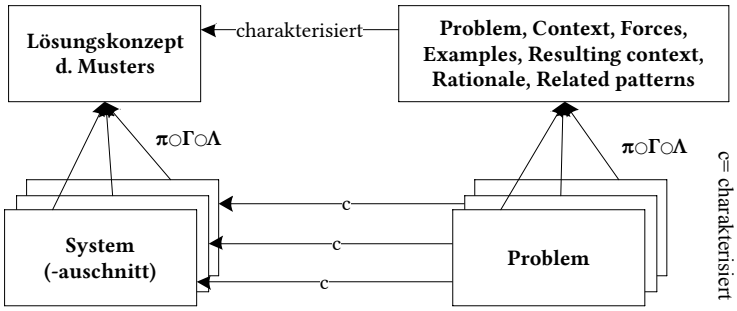


Abbildung 5.3: Implikation einer doppelten Abstraktion durch Muster

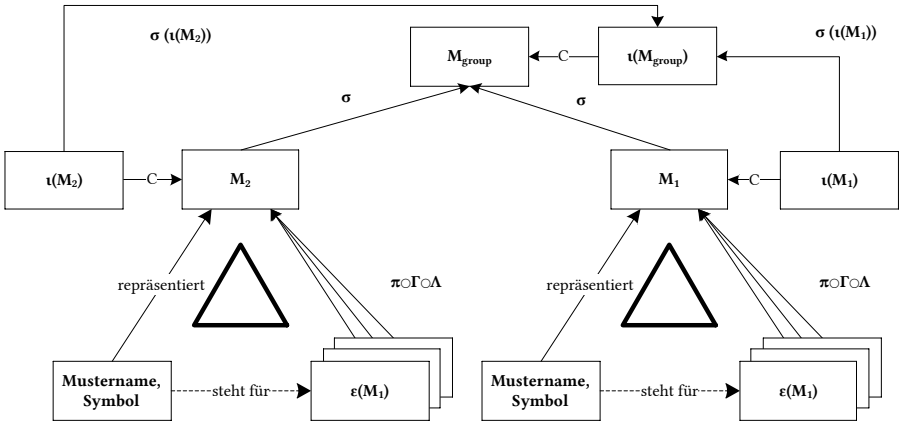


Abbildung 5.4: Patternabstraktion durch Gruppierung

für die weiteren Untersuchungen nicht möglich ist die – jeder Extension eines Musters zugrunde liegenden – Einzelprobleme mit zu untersuchen, muss dieser Blickwinkel beiseitegelassen werden.

Bestehende Abstraktionen von Mustern

Ausgehend von den vorgestellten impliziten Abstraktionsarten wird hier dargelegt, inwieweit bisher weitere Abstraktionen von Mustern vorgenommen wurden. Im Wesentlichen handelt es sich dabei um eine erneute Anwendung der Klassifikation und die Gruppierung von Mustern.

Obwohl auf den ersten Blick nicht zu vermuten, stellt die Zusammenfassung von Mustern in einem Katalog eine Form der Abstraktion dar, nämlich eine **Gruppierung**. Ein Katalog ist meist nicht zufälliger Natur, sondern folgt einer Intention. Als Beispiel seien zwei Kataloge angeführt, welche beide Architekturmuster umfassen. Der Katalog von Erl[145] beinhaltet das Gruppierungsmerkmal der Einsetzbarkeit eines Musters in serviceorientierten Architekturen. Der Katalog „Enterprise Integration Patterns“[146] umfasst nur Muster, welche in nachrichtenbasierten Architekturen einsetzbar sind; also die Unterstützung des Messaging-Prinzips als Gruppierungsmerkmal haben. Nicht jedes Muster verfügt dabei zwingend über eine Beziehung zu allen anderen Mustern des Katalogs, weshalb die Komposition als Abstraktionsart für die Katalogbildung ausscheidet.[147]

[145] Vgl. (Erl, 2009).
[146] Vgl. (Hohpe und Woolf, 2003).
[147] Vgl. Abschnitt 5.3.1.

Formal kann Gruppierung spezifiziert werden als eine Funktion:

$$\begin{aligned}\sigma(M_i) : memberOf(M_i, M_{group})\ genau\ dann\ wenn\\ \iota(M_{group}) \cap \iota(M_i) \neq \oslash\\ und\ g_{K_i} \in (\iota(M_{group}) \cap \iota(M_i))\\ mit\ G_K\{g_{K_1}, \ldots, g_{K_n}\}\ \forall\ g_{K_i} \in G_K\end{aligned} \tag{5.4}$$

Damit wird ein Muster der Gruppe zugeordnet, wenn in der Extension des Musters das Gruppierungskriterium enthalten ist, z. B. die Eignung für serviceorientierte Architekturen. Dieses Merkmal findet sich entsprechend in der Extension der Gruppe, also dem Musterkatalog wieder.

Die folgenden Ausführungen stellen die Nutzung von **Klassifikationen** vor, welche als zweite Abstraktionsart bisher im Zusammenhang mit Mustern verwendet wurden. Es können katalogin-terne – sich nur auf eine Gruppe beziehende – und katalogübergreifende – Gruppen übergreifende – Klassifikationen unterschieden werden. Damit wird klar, dass die beiden Abstraktionsmechanismen unabhängig voneinander und parallel anwendbar sind im Gegensatz zur Generalisierung, welche zwingende Voraussetzung einer Klassifikation ist. Formal kann eine Klassifikation wie folgt beschrieben werden:

$$\Lambda(M_i) : instanceOf(M_i, M_{Klass}) := \iota(M_{Klass}) \subset \iota(M_i) \rightarrow |\epsilon(M_{Klass})| \succ |\epsilon(M_i)| \tag{5.5}$$

Diese formale Beschreibung setzt voraus, dass die Instanziierung eines Prädikates von der Inklusion des Prädikates in der Intension eines Musters unterschieden wird. Leppänen schreibt dazu: „The concepts of Type and Instance are defined by their intensions, which are composed of predicates, TypePredicates and InstPredicates, respectively. All the predicates in the TypeIntension (e.g. hasName, hasAddress, hasTwoLegs) are included in the InstIntension, some of them being instantiated (e.g. [hasName]:John).“[148] Nur unter dieser Annahme kann der Vergleich der Intensionen gelingen. Ein Prädikat, welches eine Instanz eines Typs haben kann, ist demnach zwingend in der Intension der Instanz enthalten aber nur instanziiert, wenn dieser auch ein Wert zu gewiesen wird. Damit entspricht die Intension dem Genotyp eines Musters und die Instanziierung repräsentiert den Phänotyp des Musters in Anlehnung an die Genetik. Ein in der Intension enthaltenes Prädikat, welches nicht instanziiert ist kann man als nicht ausgeprägt bezeichnen. Sehr wohl kann die Intension der Instanz $\iota(M_i)$ Prädikate enthalten, welche nicht in der Intension des Typs $\iota(M_{Klass})$ enthalten sind.[149] Diese Formalisierung wird durch die Abbildung 5.5 verdeutlicht. Teile der Intensionen von M_1 und M_2 wurden durch Generalisierung und Klassifikation auf dieselben Prädikate der Intension von M_{Klass} abgebildet. Aus den Musternamen Adapter und Wrapper wird z. B. hasName.

Für eine **katalogin terne Klassifikation** mit M_K als Menge der Typen, welche synonym auch Klassen genannt werden $\{M_{K_1}, \ldots, M_{K_n}\}$, muss gelten, dass alle Muster M_i Elemente einer Gruppe M_{group} sind.

Die Anwendung dieser Mechanismen ist vielfältig. Die von Gamma u. a.[150] genutzte Klassifikation der vorgestellten Entwurfsmuster stellt auf zwei Dimensionen – ab. Dabei unterscheiden sie purpose und scope. Scope bezieht sich auf den Gültigkeitsbereich und kann entweder klassen- oder objektbasiert sein. Weiterhin unterscheidet purpose als Aufgabe zwischen Erzeugungs-,

[148] (Leppänen, 2007, S. 170)
[149] Vgl. (ebenda, S. 171).
[150] Vgl. (Gamma u. a., 1995).

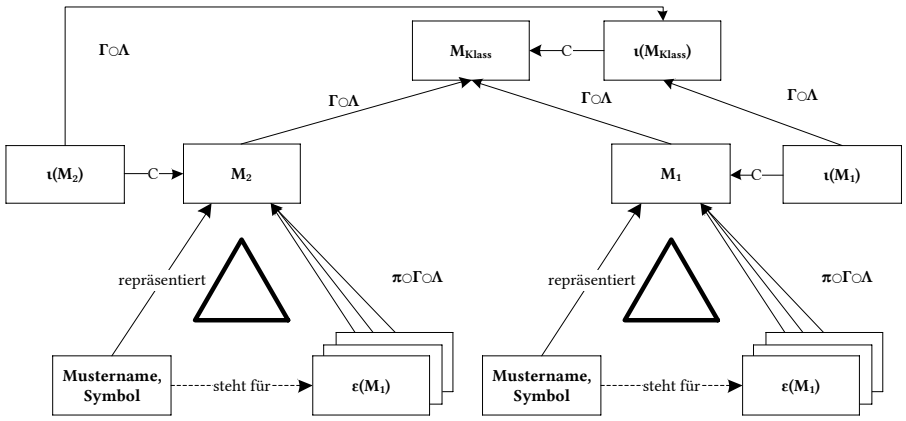


Abbildung 5.5: Patternabstraktion durch Klassifikation

Struktur und Verhaltensmustern. Malich bezeichnet diese Klassifikation als zwei-dimensional[151], Fettke und Loos sprechen hingegen von einer Facettenklassifikation[152]. Beide Autoren gehen nicht auf die Unterschiede zu der Klassifikation ein, die in einer früheren Arbeit von Gamma u. a.[153] zu finden ist. Diese unterscheidet zusätzlich beim Gültigkeitsbereich eine dritte Ausprägung, genannt Compund. Darüber hinaus ergaben sich bei einigen Mustern auch Umbenennungen.

Zimmer reklassifiziert die Muster nach Gamma, sodass für diese Menge an Pattern drei Klassifikationen verfügbar sind. Buschmann; Meunier u. a.[154] verwenden ebenfalls eine Facettenklassifikation[155], welche 2 Facetten bzw. Dimensionen hat[156]. Buschmann; Meunier u. a. unterscheiden hinsichtlich des Abstraktionsniveaus und der Problembereiche der Muster. Ausgehend von diesen beiden Klassifikationen und der Eigenschaft der Erweiterbarkeit hinsichtlich der Facetten einer Facettenklassifikation erarbeitet Malich[157] eine integrierte Klassifikation, die beide vorgenannten zusammenfasst.

Bei einer **Katalog übergreifenden Klassifikation** müssen mindestens zwei verschiedene Gruppen M_{group_1}, M_{group_2} beteiligt sein.

Insbesondere der Pattern Almanach von Linda Rising[158] und die frühe Arbeit von Tichy[159] stellen auf Katalogübergreifende Klassifikationsschemata ab. Tichy klassifiziert mehr als 100 Muster nach den Problemklassen, die sie lösen und verwendet dafür 9 Dimensionen: Decoupling, Variant Management, State Handling, Control, Virtual Machine, Convenience Patterns, Compound Patterns, Concurrency und Distribution. Die Kategorien sollen disjunkt sein und jede Subkategorie stellt eine echte Teilmenge ihrer Oberkategorie dar. Ähnlichkeiten bei der Implementierung der Muster werden von Tichy ausgeblendet.[160] Das Ziel von Tichy ist die Klassifikation von allgemeinen Mustern der Softwareentwicklung. Insbesondere die Arbeit von Tichy eignet sich für die Erarbeitung eines eigenen Klassifikationsschemas, sodass nachfolgend die Facetten vorgestellt werden und auf deren Verwendbarkeit für eine Klassifikation von Integrationsmustern eingegangen wird.

[151] Vgl. (Malich, 2008, S. 116).
[152] Vgl. (Fettke und Loos, 2001, S. 8).
[153] Vgl. (Gamma u. a., 1993).
[154] Vgl. (Buschmann; Meunier u. a., 1996).
[155] Vgl. (Fettke und Loos, 2001).
[156] Vgl. (Malich, 2008, S. 117).
[157] Vgl. (ebenda).
[158] Vgl. (Rising, 2000).
[159] Vgl. (Tichy, 1997).
[160] Vgl. (ebenda, S. 2).

Unter **Decoupling**, der Entkopplung, versteht Tichy die Strukturierung von Software in unabhängige Bestandteile, sodass diese unabhängig voneinander erzeugt, verändert, ersetzt, wiederverwendet und rekombiniert werden können.[161] Diese Eigenschaft ist ein wesentlicher Grund, weshalb Integrationsarchitekturen überhaupt notwendig sind. Weiterhin obliegt es der Gestaltung von Integrationsarchitekturen diese Eigenschaft zu erhalten und ist damit eine geeignete Kategorie für die Klassifikation von Integrationsmustern.

Das Management von Varianten (**Variant Management**) ist nicht vordringlich relevant für die Klassifikation von Integrationsmustern.

State Handling ist analog zu Varaint Management zu sehen. Es hat keine vordringliche Relevanz für Integrationsmuster.

Die Organisation eines Integrationsobjekt übergreifenden Kontrollflusses (**Control**) ist eine wesentliche Aufgabe von Integrationsarchitekturen und damit ein Gestaltungsgegenstand von Integrationsmustern.

Convenience Pattern helfen den geschriebenen Code zu vereinfachen, was im Fall von Integrationsmustern von untergeordneter Bedeutung ist und als Kategorie vernachlässigt werden kann.

Compound Pattern verwendet Tichy, um Kompositionen aus mehreren Mustern zu erfassen. Die Erfassung dieser Muster würde zu einer Einordnung in mehrere der von Tichy verwendeten Kategorien führen, weshalb derartige Muster in der Compound Kategorie erfasst werden. Bei Integrationsmustern ist dieser Fall nicht auszuschließen, weshalb eine derartige Kategorie einbezogen wird.

Concurrency bezeichnet eine Kategorie, in der sich alle Muster finden, die sich mit der Problematik der Nebenläufigkeit befassen. Bei der Gestaltung des Integrationsobjekt-übergreifenden Kontrollflusses kann es zu Nebenläufigkeit kommen, weshalb diese Kategorie übernommen wird.

Distribution spricht den eigentlichen Grund für Integrationsbemühungen an, und adressiert Probleme der Verteilung. Beim Einsatz mehrerer Aufgabenträger entstehen diese Probleme zwangsläufig. Diese Kategorie wird ebenfalls in die Erarbeitung der Klassifikation einbezogen.

Die Analyse der Abstraktionsarten ist nun abgeschlossen. Neben der allgemeinen Analyse, welche Abstraktionsmechanismen bekannt sind, wurde deren Einsatz im Gebiet von Mustern untersucht. Das sich anschließende Kapitel wendet diese Erkenntnisse nun auf die in Kapitel 5.2 identifizierte Menge an Mustern an.

5.4 Höhere Integrationsmuster durch Klassifikation

Eines der Ergebnisse der empirischen Untersuchung, welche in Abschnitt 3 vorgestellt wurde, ist, dass zwar Kenntnisse über Integrationsmuster bei Integrationsdienstleistern vorhanden sind, diese aber nicht aktiv für die Lösungsgestaltung eingesetzt werden. Es findet keine Zuordnung des zu lösenden Problems P zu einem Muster M_i oder einem Typen M_{Klass} statt, auf dessen Basis dann die Lösungsentwicklung aufsetzt und eine neue Extension des Musters geschaffen wird.

In diesem Abschnitt wird die Klassifikation der ausgewählten Muster erarbeitet. Der Übersichtlichkeit halber wird jede Facette einzeln beschrieben, wobei ein standardisierter Aufbau eingehalten wird. Dieser besteht aus vier Teilen. Zuerst wird eine allgemeine Motivation der jeweiligen Facette gegeben. Im Anschluss daran wird das generalisierte Merkmal dieser Facette näher erläutert.

[161] Vgl. (Tichy, 1997, S. 2).

Dies geschieht vor allem, um den Zusammenhang und die Untrennbarkeit der Abstraktionsarten Klassifikation und Generalisierung deutlich herauszuarbeiten. Daran schließt sich ein weiterer Unterabschnitt an, der die Fokusse der Facette - also die möglichen Ausprägungen – beschreibt. Als Abschluss werden dann die Muster der Grundgesamtheit bezüglich der beschriebenen Facette klassiert, wodurch jeweils ein Teil der Gesamtklassifikation entsteht.

5.4.1 Auswahl des Klassifikationsverfahrens

Das Ziel dieser Arbeit ist es, den Zugang zu Integrationsmustern zu verbessern. Es handelt sich demnach um die Gestaltung künstlicher Systeme mit Hilfe von Lösungsmustern, bei denen mit Erweiterungen zu rechnen ist. Die Verwendung von Mustern kann als ein Ansatz der Wiederverwendung betrachtet werden, weshalb auf Erfahrungen mit Klassifikationen im Bereich der der Softwarewiederverwendung zurückgegriffen werden kann.[162] Bereits in Gamma u. a.[163] sind erste Überlegungen zur Ordnung der Muster enthalten, Fettke und Loos vergleichen verschiedene Ordnungsvarianten miteinander.[164]

Da diese in verschiedenen Publikationen mit unterschiedlichen Systematiken veröffentlicht wurden, kommt nur eine katalogübergreifende Klassifikation in Betracht. Aus diesem Grund bieten sich die bestehenden katalogübergreifenden Systematiken als Orientierung an.

Die Arbeit von Rising ist für die Zwecke der hier vorgenommenen Untersuchung zu allumfassend und nicht speziell auf Integrationsprobleme abgestimmt, weshalb es schwerfällt, diesen Ansatz in geeigneter Weise zu übertragen. Die Arbeit von Tichy beinhaltet gute Ansätze und kann als Basis für die Erarbeitung einer umfassenderen Klassifikation von Integrationsmustern verwendet werden. Kritisch zu sehen ist jedoch, dass Muster nur in eine Kategorie eingeordnet werden durften. Nimmt man ein Muster wie den Broker, so wird dieser von Tichy in die Kategorie Distribution eingeordnet, was dem Hauptanliegen des Broker-Musters entspricht. Es ist jedoch nicht von der Hand zu weisen, dass der Einsatz eines Brokers einen entkoppelnden Effekt auf jede Kommunikation hat, die nun durch diesen durchgeführt wird. Für die problembezogene Auswahl von Integrationsmustern ist die Beurteilung zusätzlicher Auswirkungen ebenfalls von Bedeutung, weshalb ein Ansatz der den Anforderungen genügt diese mit erfassen und ausweisen muss.

Bei einer Facettenklassifikation – auch als analytisch-synthetische Klassifikation bezeichnet – handelt es sich um eine polyhierarchische Klassifikation. Mit dieser Eigenschaft kann die eben gestellte Forderung abgedeckt werden, da nur innerhalb einer Facette eine disjunkte Einordnung gegeben sein muss.

Die verwendeten Facetten der katalogübergreifenden Klassifikationsschemas sind in weiten Teilen ungeeignet, um einem Nutzer eines Musterkataloges einen problembezogenen Zugang zu Integrationsmustern – und damit den dokumentierten Lösungen – zu gewähren. Außerdem bewegen sie sich weitestgehend auf der gleichen Abstraktionsebene. Damit wird die Abstraktionsmöglichkeit von komponierten Mustern nicht explizit adressiert.

[162] Vgl. (Prieto-Diaz, 1991).
[163] Vgl. (Gamma u. a., 1993).
[164] Vgl. (Fettke und Loos, 2001).

Aus diesem Grund wird im Nachfolgenden Abschnitt 5.4 ein neues Klassifikationsschema mit Facetten und Fokussen[165] entwickelt und die ausgewählte Grundgesamtheit an Mustern entsprechend dieses Schemas klassifiziert.[166]

5.4.2 Facette Subsystem des BIS

Muster finden auf unterschiedlichen Gestaltungsebenen eines BIS Anwendung. Mit dieser Facette soll ermöglicht werden, dass alle Muster zur Gestaltung eines BIS in einer Systematik erfasst werden können. Darüber hinaus abstrahieren die wissenschaftlichen Arbeiten zur Integration, welche die Grundlage dieser Arbeit bilden, durch die Verwendung des Begriffes Integrationsobjekt von der konkreten Ausprägung der Integrationsobjekte. Dies lässt darauf schließen, dass es gemeinsame, grundlegende Prinzipien der Integration gibt. Um später Gemeinsamkeiten zwischen Integrationsmustern aufzeigen zu können, die sich auf unterschiedliche Abstraktionsebenen – und damit (Sub-) Subsysteme des BIS beziehen – wird diese Facette eingeführt.

Das generalisierte Merkmal dieser Facette ist das gestaltete Subsystem eines BIS und dadurch implizit auch der Abstraktionsgrad des Musters. Beide Aspekte korrespondieren sehr stark miteinander. In Kapitel 2.1 wurden die systemtheoretischen Grundlagen von betrieblichen Informationssystemen erläutert, die an dieser Stelle aufgegriffen werden, um die Einordnung verschiedener Muster in ein betriebliches Informationssystem aufzuzeigen.

Betrachtet man ein einzelnes Unternehmen, so handelt es sich um ein Informationssystem[167], welches Menschen, Prozesse, IT-Systeme und die gesamte Infrastruktur umfasst. Die Elemente des Unternehmensinformationssystems wurden zu zwei Subsystemen zusammengefasst. Dem technischen Subsystem und dem sozio-ökonomischen Subsystem. Das technische Subsystem besteht wiederum aus einem Softwaresubsystem und einem Hardwaresubsystem. Bestandteile des Softwaresubsystems sind die Applikationen, welche in einem Unternehmen eingesetzt werden. Die verschiedenen bekannten Musterarten strukturieren nun in unterschiedlicher Weise die vorhandenen Subsysteme eines BIS. Abbildung 5.6 verdeutlicht das zugrunde liegende Schema.

Auf der rechten Seite der Abbildung sind die eingehenden Elemente in die jeweilige Abstraktionsebene abgebildet und auf der linken Seite die strukturierende Musterart. Dabei formen die eingehenden Elemente – strukturiert durch die Muster – jeweils ein Element der darüber angeordneten Ebene, angedeutet durch die Kanten, welche die Ebenen verbinden.

Auf der untersten Ebene befinden sich Entwurfsmuster, welche den Aufbau von Bestandteilen eines Softwareprogramms beschreiben, sofern diese in einer objektorientierten Programmiersprache[168] geschrieben wird. Mithilfe dieser Muster lässt sich der Aufbau einzelner Softwarekomponenten des Softwaresubsystems eines BIS beschreiben.

In Abgrenzung von Entwurfsmustern werden darüber hinaus so genannte *Architekturmuster* betrachtet.[169] Ein Architekturmuster beschreibt nicht den Aufbau einer einzelnen Applikation, sondern die Anordnung der Elemente einer Softwarekomponente und ihre Beziehungen zueinander. Architekturmuster werden meist implementierungsunabhängig beschrieben in dem Sinn, dass sie – anders als Entwurfsmuster – nicht den Einsatz einer objektorientierten Sprache festlegen.

[165] Obwohl in vielen Publikationen die Kombination Focus und Fokusse aus dem Lateinischen gebräuchlich ist, wird die deutsche Schreibweise Fokus und Fokusse verwendet.

[166] Bei einer Klassifikation spricht man auch von klassieren, wenn das Einordnen von Klassifikationsobjekten bezüglich des Schemas erfolgt und von einer Klassifikation bzw. einem klassifizierten Objekt nach erfolgter Einordnung. Vereinfachend wird jedoch einheitlich klassifizieren verwendet werden.

[167] Häufig wird das Informationssystem eines Unternehmens auch als Unternehmensinformationssystem bezeichnet.

[168] Die Einschränkung auf objektorientierte Sprachen wird an dieser Stelle getroffen, da für andere Sprachen (z.B. funktionale) nur wenige Publikationen über Muster existieren.

[169] Vgl. (Buschmann; Meunier u. a., 1996).

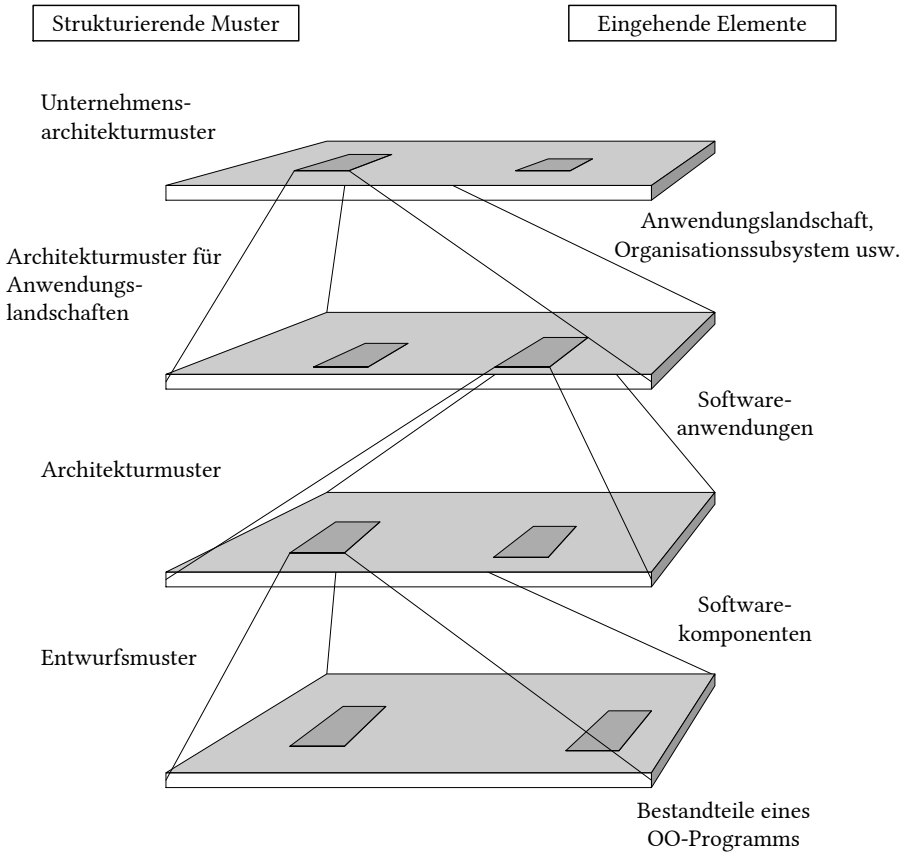


Abbildung 5.6: Ebenen von Integrationsmustern

Damit beschreiben sie nur einen Teil eines Anwendungssystems, da dieses – der Definition in Kapitel 2.1 nach – auch die zugehörige Hardware umfasst.

Softwareanwendungen bilden also Elemente des Softwaresubsystems und formen in ihrer Summe eine Anwendungslandschaft. Wie diese geeignet zu strukturieren ist, beschreiben *Architekturmuster für Anwendungslandschaften.*[170]

Wie eine Softwarelandschaft mit den weiteren Elementen und Subsystemen eines BIS zusammen strukturiert wird, um die Unternehmensarchitektur zu gestalten, beschreiben *Unternehmensarchitekturmuster.*[171] Diese sind nicht zu verwechseln mit Enterprise-Architecture-Management-Mustern (EAM-Mustern)[172], welche Muster im Sinne von Best-Practice für das Management der Unternehmensarchitektur darstellen, nicht jedoch für den strukturellen und funktionalen Aufbau derselben. Sie helfen also bei der Gestaltung des Gestaltungsprozesses.

Da sich für alle anderen Subsysteme eine zweifelsfreie Zuordnung zu den jeweiligen Abstraktionsebenen der Muster ergibt, muss diese Eigenschaft nicht separat erfasst werden und wird implizit mit codiert. Im Folgenden werden nun die Fokusse für das generalisierte Merkmal Abstraktionsgrad beschrieben. Sowohl Buschmann [173] als auch Malich [174] liefern dafür Ansatzpunkte. Sie

[170] In diesem Bereich existieren bisher kaum Veröffentlichungen. Dieser Umstand wird in Kapitel 8 weiter diskutiert.

[171] Auch für diese Musterart existieren bisher wenige Publikationen, wenngleich davon ausgegangen werden kann, dass spezialisierte Beratungsdienstleister derartiges Wissen aufgebaut haben z.B. in Form einer Business-Process-Masterlist Vgl. (Gebauer und Stefan, 2011b, S. 44).

[172] Vgl. (Ernst, 2008).

[173] Vgl. (Buschmann; Meunier u. a., 1996, S. 363ff.).

[174] Vgl. (Malich, 2008, S. 109).

beschreiben die Abstraktion ausgehend von Quellcode, der die konkrete Ausprägung der Lösung eines konkreten Problems in einer Programmiersprache darstellt. Idiome liefern prinzipielle Lösungsansätze für eine spezielle Sprache und bilden die zweite Abstraktionsstufe und den ersten Fokus dieser Facette. Entwurfsmuster sind der zweite Fokus. Sie abstrahieren von der konkreten Sprache und beziehen sich auf die Familie der objektorientierten Sprachen. Architekturmuster stellen den dritten Fokus dar. Diese erfassen Lösungsprinzipien für Mengen von Applikationen und stellen den dritten Fokus.

Organisationsmuster sind in den Bereich der Betriebswirtschaft einzuordnen und werden hier angeführt, um die Tragweite der angestrebten Klassifikation zu verdeutlichen. Im Rahmen der Organisationstheorie[175] und -gestaltung[176] sind zwar Überlegungen zur Strukturierung bekannt, diese greifen jedoch nicht die Beschreibungsform im Sinne der in dieser Arbeit behandelten Muster auf. Ein Nutzen dieser Konzepte in diesem Bereich wird in Kapitel 8 diskutiert. Gleiches trifft auf Hardwaremuster zu, deren Name unzureichend vereinfacht. Bei dieser Musterkategorie geht es vielmehr darum, alle Muster unterhalb der Schicht sechs im OSI-Referenzmodell inclusive aller Hardware wie z. B. spezifische Verkabelungen etc. zu erfassen. Alle Muster der Netzwerkarchitektur[177] fallen in diese Kategorie. Für weitere Ausführungen sei auf das Kapitel 8 verwiesen.

Die weiteren Abstraktionsniveaus werden zunächst ausgeblendet. Diese sind Architektur-Taktiken[178] und darüber die fundamentalen Prinzipien der Softwareentwicklung[179]. Nicht alle Fokusse bilden notwendigerweise eine Hierarchie. Dies ist zwar im Fall von Idiom, Entwurfsmuster und Architekturmuster der Fall, trifft aber auf Organisationsmuster und Hardwaremuster nicht zu. Für die nachfolgende Klassifikation werden die Fokusse: Idiom, Entwurfsmuster, Architekturmuster, Organisationsmuster und Hardwaremuster verwendet. In Tabelle 5.10 ist die Klassifikation der exemplarisch ausgewählten Muster verzeichnet.

Pattern	**Quelle**	**Fokus**
Adapter Style	(Andersson und Johnson, 2001, S. 229)	Architekturmuster
Adapter static	(ebenda, S. 229)	Architekturmuster
Adapter dynamic	(ebenda, S. 229)	Architekturmuster
Adapter thin	(ebenda, S. 229)	Architekturmuster
Adapter thick	(ebenda, S. 229)	Architekturmuster
Adapter centralized	(ebenda, S. 229)	Architekturmuster
Adapter distributed	(ebenda, S. 229)	Architekturmuster
Adapter	(Khomh und Guéhéneuc, 2008a)	Entwurfsmuster
(Integration) Adapter	(Lutz, 2000)	Architekturmuster
Component Wrapper	(Goedicke und Zdun, 2002, S. 7ff.)	Entwurfsmuster
Wrapper Facade	(Schmidt; Stal u. a., 2000, S. 47ff.)	Entwurfsmuster
(Legacy) Wrapper	(Mularz, 1995)	Architekturmuster
Facade	(Khomh und Guéhéneuc, 2008a)	
Integration Facade	(Lutz, 2000)	Architekturmuster
Service Interface	(Trowbridge, 2003)	Architekturmuster
Integration Messenger (a, broker)	(Lutz, 2000)	Architekturmuster
Integration Messenger (b, message queuing)	(ebenda)	Architekturmuster
Integration Messenger (c, publish subscribe)	(ebenda)	Architekturmuster
Proxy	(Buschmann; Meunier u. a., 1996, S. 263ff.)	Entwurfsmuster
Proxy	(Khomh und Guéhéneuc, 2008a)	
Mediator	(Gamma u. a., 1995, S. 277)	Entwurfsmuster
Integration Mediator	(Lutz, 2000, S. 71)	Architekturmuster

Klassifikation bezüglich des Abstraktionsgrades – Fortsetzung auf der nächsten Seite

[175] Vgl. (Kieser, 1999).
[176] Vgl. (Weinert, 2002).
[177] Vgl. (Day, 2008).
[178] Vgl. (Bass; Clements und Kazman, 2003).
[179] Vgl. (Parnas, 1972), sowie (Malich, 2008, S. 108ff.).

Pattern	Quelle	Fokus
Mediator	(Buschmann; Henney und Schmidt, 2007a, S. 411)	Entwurfsmuster
Mediator	(Khomh und Guéhéneuc, 2008a)	
Mediator Style	(Andersson und Johnson, 2001, S. 230)	Architekturmuster
Message Router	(ebenda, S. 231)	Architekturmuster
Message Router	(Hohpe und Woolf, 2003, S. 81)	Architekturmuster
Message Router	(Buschmann; Henney und Schmidt, 2007a, S. 232)	Architekturmuster
Gateway	(Andersson und Johnson, 2001)	Architekturmuster
Service Gateway	(Trowbridge, 2003, S. 293ff.)	Architekturmuster
Broker	(ebenda, S. 207f.)	Architekturmuster
Direct Broker	(Trowbridge u. a., 2004, S. 155,215ff.)	Architekturmuster
Indirect Broker	(ebenda, S. 221)	Architekturmuster
Message Broker	(ebenda, S. 240f.)	Architekturmuster
Message Broker	(Hohpe und Woolf, 2003, S. 322ff.)	Architekturmuster
Broker	(Buschmann; Meunier u. a., 1996, S. 119ff.)	Architekturmuser
Broker Revisited	(Kircher; Voelter u. a., 2004, S. 581)	Architekturmuster
Broker	(Harrison und Avgeriou, 2007, S. 267f.), bezieht sich auf Buschmann	
Pipes and Filter	(Hohpe und Woolf, 2003, S. 322ff.) , (ebenda, S. 70ff.)	Architekturmuster
Pipes and Filter	(Buschmann; Meunier u. a., 1996, S. 67ff.)	Architekturmuster
Pipes and Filter	(Trowbridge u. a., 2004, S. 292f.)	Architekturmuster
Pipes and Filters	(Garlan und Shaw, 1993, S. 7f.)	Architekturmuster
Pipes and Filters	(Meunier, 1995, S. 437)	Architekturmuster
Pipes and Filters	(Harrison und Avgeriou, 2007, S. 267f.) Bezieht sich auf Buschmann	
Observer	(Khomh und Guéhéneuc, 2008a)	
Observer	(Trowbridge, 2003, S. 129f.)	Entwurfsmuster
Publish and Subscribe	(Trowbridge u. a., 2004, S. 280f.)	Architekturmuster
Process Automator	(Lutz, 2000)	Architekturmuster
Database Federation	(Andersson und Johnson, 2001)	Architekturmuster
Desktop Integration	(ebenda)	Architekturmuster
Data Transfer Object	(Trowbridge, 2003, S. 229)	Entwurfsmuster
Singleton	(ebenda, S. 257)	Entwurfsmuster
Singleton	(Khomh und Guéhéneuc, 2008a)	
File Gateway (457)	(Erl, 2009)	Architekturmuster
Process Centralizazion (193)	(ebenda)	Architekturmuster
Protocol Bridging (687)	(ebenda)	Architekturmuster
Legacy Wrapper (441)	(ebenda)	Architekturmuster
Service Broker (707) (compund Pattern)	(ebenda)	Architekturmuster
Enterprise Service Bus (704) (compund Pattern)	(ebenda)	Architekturmuster
Proxy Capability (497)	(ebenda)	Architekturmuster
Service Facade (333)	(ebenda)	Architekturmuster
UI Mediator (366)	(ebenda)	Architekturmuster
Data Model Transformation (671)	(ebenda)	Architekturmuster
Data Format Transformation (681)	(ebenda)	Architekturmuster
Asynchronous Queuing (582)	(ebenda)	Architekturmuster
Intermediate Routing (549)	(ebenda)	Architekturmuster

Tabelle 5.10: Klassifikation bezüglich des Abstraktionsgrades

Muster, für die kein Fokus angegeben wurde, sind weder explizit noch implizit einem der Fokusse der Facette zuordenbar, wenn man sich auf die zugehörige Quelle beschränkt. Dies kann z. B. der Fall sein, wenn es sich wie im vorliegenden Fall um eine funktionale Facette handelt und die Quelle sich auf nicht funktionale Eigenschaften beschränkt. Sie werden der Vollständigkeit halber mit aufgelistet, um zu zeigen, dass die exemplarische Klassifikation immer mit derselben Grundgesamtheit an Mustern durchgeführt wurde. Diese Vorgehensweise wird für alle nachfolgenden Facetten beibehalten. Diese Eigenschaft wird von dem gewählten Klassifikationsmechanismus – der Facettenklassifikation – unterstützt. Nicht jedes zu klassifizierende Objekt muss Ausprägungen in allen Facetten aufweisen.

Wie nicht anders zu erwarten finden sich in der Auflistung nur Muster, welche dem Softwaresubsystem zugehören. Für die weitere Ausarbeitung der Klassifikation sei auf Kapitel 8 verwiesen. Die nachfolgende Facette hat die verschiedenen Operationen an einem Integrationsobjekt zum Gegenstand die auf einem abstrakten Niveau der Integration für alle Integrationsobjekte gültig sind.

5.4.3 Facette Integrationsoperation

Die Integrationsoperationen dienen im Wesentlichen dazu, zwischen der ex ante und der ex post Integration zu unterscheiden. Die ex ante Integration kann als die Herstellung von Integrationsfähigkeit verstanden werden, ohne dabei bereits zwei oder mehr Integrationsobjekte miteinander zu verbinden. Die ex post Integration geht hingegen von der Existenz der Integrationsobjekte mit gegebenen Eigenschaften aus und versucht diese miteinander zu integrieren. Diese Facette dient der Zuordnung von Mustern zu entsprechenden Integrationsoperationen und kann damit die Auswahl entsprechender Muster erleichtern. Weiterhin kann geprüft werden, ob bestimmte Operationen überhaupt durch Muster abgedeckt sind, oder ob diese erst noch identifiziert werden müssen. Thränert diskutiert in seiner Dissertation verschiedene Integrationsoperationen[180], die in Tabelle 5.11 aufgelistet sind. Dies ermöglicht die Extraktion der Fokusse.

Zeitpunkt	**Typ**	**Subtyp**
ex-ante Integrierung	Erstellung	-
	Vernichtung	-
	Anpassung	Restrukturierung
		Schnittstellenergänzung
		Schnittstellenentfernung
		Adaption
		Desintegrierung
ex-post Integrierung	direkte Kopplung	-
	indirekte Kopplung	-
	Verschmelzung	-
	Entkopplung	-

Tabelle 5.11: Übersicht über die Integrationsoperationen nach Thränert
Quelle: (Thränert, 2008)

Für die Facette Integrationsoperation entstehen die folgenden Fokusse: Erstellung, Vernichtung, Restrukturierung, Schnittstellenergänzung, Schnittstellenentfernung, Adaption, Desintegrierung, direkte Kopplung, indirekte Kopplung, Verschmelzung, Entkopplung. Es wird erwartet, dass die Mehrzahl, wenn nicht alle, der Muster auf die Fokusse *direkte Kopplung*, *indirekte Kopplung* und *Adaption* entfallen. Nachdem nun die Fokusse festgelegt sind, gilt es die Musterauswahl entsprechend zu klassifizieren.

Pattern	**Quelle**	**Fokus**
Adapter Style	(Andersson und Johnson, 2001, S. 229)	Adaption
Adapter static	(ebenda, S. 229)	Adaption
Adapter dynamic	(ebenda, S. 229)	Adaption
Adapter thin	(ebenda, S. 229)	Adaption
Adapter thick	(ebenda, S. 229)	Adaption
Adapter centralized	(ebenda, S. 229)	Adaption
Adapter distributed	(ebenda, S. 229)	Adaption
Adapter	(Khomh und Guéhéneuc, 2008a)	
(Integration) Adapter	(Lutz, 2000)	Adaption

Klassifikation bezüglich der Integrationsoperation – Fortsetzung auf der nächsten Seite

[180] Vgl. (Thränert, 2008, S. 29ff.).

Pattern	**Quelle**	**Fokus**
Component Wrapper	(Goedicke und Zdun, 2002, S. 7ff.)	Adaption
Wrapper Facade	(Schmidt; Stal u. a., 2000, S. 47ff.)	Adaption
(Legacy) Wrapper	(Mularz, 1995)	Adaption
Facade	(Khomh und Guéhéneuc, 2008a)	
Integration Facade	(Lutz, 2000)	indirekte Kopplung
Service Interface	(Trowbridge, 2003)	Adaption
Integration Messenger (a, broker)	(Lutz, 2000)	indirekte Kopplung
Integration Messenger (b, message queuing)	(ebenda)	indirekte Kopplung
Integration Messenger (c, publish subscribe)	(ebenda)	indirekte Kopplung
Proxy	(Buschmann; Meunier u. a., 1996, S. 263ff.)	
Proxy	(Khomh und Guéhéneuc, 2008a)	
Mediator	(Gamma u. a., 1995, S. 277)	indirekte Kopplung
Integration Mediator	(Lutz, 2000, S. 71)	indirekte Kopplung
Mediator	(Buschmann; Henney und Schmidt, 2007a, S. 411)	indirekte Kopplung
Mediator	(Khomh und Guéhéneuc, 2008a)	
Mediator Style	(Andersson und Johnson, 2001, S. 230)	indirekte Kopplung
Message Router	(ebenda, S. 231)	indirekte Kopplung
Message Router	(Hohpe und Woolf, 2003, S. 81)	indirekte Kopplung
Message Router	(Buschmann; Henney und Schmidt, 2007a, S. 232)	indirekte Kopplung
Gateway	(Andersson und Johnson, 2001)	direkte Kopplung
Service Gateway	(Trowbridge, 2003, S. 293ff.)	indirekte Kopplung
Broker	(ebenda, S. 207f.)	indirekte Kopplung
Direct Broker	(Trowbridge u. a., 2004, S. 155,215ff.)	direkte Kopplung
Indirect Broker	(ebenda, S. 221)	indirekte Kopplung
Message Broker	(ebenda, S. 240f.)	indirekte Kopplung
Message Broker	(Hohpe und Woolf, 2003, S. 322ff.)	indirekte Kopplung
Broker	(Buschmann; Meunier u. a., 1996, S. 119ff.)	indirekte Kopplung
Broker Revisited	(Kircher; Voelter u. a., 2004, S. 581)	indirekte Kopplung
Broker	(Harrison und Avgeriou, 2007, S. 267f.), bezieht sich auf Buschmann	
Pipes and Filter	(Hohpe und Woolf, 2003, S. 322ff.) , (ebenda, S. 70ff.)	indirekte Kopplung
Pipes and Filter	(Buschmann; Meunier u. a., 1996, S. 67ff.)	indirekte Kopplung
Pipes and Filter	(Trowbridge u. a., 2004, S. 292f.)	indirekte Kopplung
Pipes and Filters	(Garlan und Shaw, 1993, S. 7f.)	indirekte Kopplung
Pipes and Filters	(Meunier, 1995, S. 437)	indirekte Kopplung
Pipes and Filters	(Harrison und Avgeriou, 2007, S. 267f.) Bezieht sich auf Buschmann	
Observer	(Khomh und Guéhéneuc, 2008a)	
Observer	(Trowbridge, 2003, S. 129f.)	Schnittstellenergänzung
Publish and Subscribe	(Trowbridge u. a., 2004, S. 280f.)	indirekte Kopplung
Process Automator	(Lutz, 2000)	indirekte Kopplung
Database Federation	(Andersson und Johnson, 2001)	indirekte Kopplung
Desktop Integration	(ebenda)	direkte Kopplung
Data Transfer Object	(Trowbridge, 2003, S. 229)	direkte Kopplung
Singleton	(ebenda, S. 257)	direkte Kopplung
Singleton	(Khomh und Guéhéneuc, 2008a)	
File Gateway	(Erl, 2009, S. 457)	indirekte Kopplung
Process Centralizazion	(ebenda, S. 193)	indirekte Kopplung
Protocol Bridging	(ebenda, S. 687)	indirekte Kopplung
Legacy Wrapper	(ebenda, S. 441)	Adaption
Service Broker (COMP)	(ebenda, S. 707)	Adaption
Enterprise Service Bus (COMP)	(ebenda, S. 704)	indirekte Kopplung
Proxy Capability	(ebenda, S. 497)	indirekte Kopplung
Service Facade	(ebenda, S. 333)	indirekte Kopplung
UI Mediator	(ebenda, S. 366)	indirekte Kopplung
Data Model Transformation	(ebenda, S. 671)	indirekte Kopplung
Data Format Transformation	(ebenda, S. 681)	indirekte Kopplung
Asynchronous Queuing	(ebenda, S. 582)	indirekte Kopplung
Intermediate Routing	(ebenda, S. 549)	indirekte Kopplung

Tabelle 5.12: Klassifikation bezüglich der Integrationsoperation

Die meisten Muster dienen der Kopplung von Integrationsobjekten, bevorzugt der indirekten. Wenige lassen sich der Adaption zuordnen und ein Muster weist die Klassifizierung Schnittstellenergänzung auf.

5.4.4 Facette Integrationsarchetyp

In neueren Arbeiten werden die phänomenologischen Untersuchungen zur Integration kritisiert. Dabei werden mehrere Kritikpunkte angeführt. Die auf Basis dieser Arbeiten entstehenden Ergebnisse sind nicht nachweisbar vollständig noch sind sie überschneidungsfrei. Stattdessen propagieren die Autoren eine auf dem Method-Engineering basierende Vorgehensweise, welche ausgehend von – aus einem Metamodell abgeleiteten – Integrationsarchetypen flexibel kombinierbare Methodenfragmente zu entwickeln sucht. In diesem Abschnitt werden die Integrationsarchetypen verwendet, um bestehende Muster nach diesen zu klassifizieren. Ziel ist es damit nachzuweisen, ob sich alle als Integrationsmuster identifizierten Muster diesen Archetypen zuordnen lassen. Damit wären Anhaltspunkte für eine noch stärkere Generalisierung von Mustern gegeben und zum anderen kann der vorgeschlagene Ansatz geprüft werden.

Die Fokusse der Facette Archetyp entsprechen den Archetypen. Diese wurden von Winter und Aier vorgeschlagen. Das vollständige Bild des Ansatzes lässt sich nur über mehrere Arbeiten hinweg konstruieren. Der grundsätzliche Ansatz entstammt einem Diskussionsbeitrag und regt an, die Ableitung von Archetypen auf Basis eines Metamodells einer Unternehmensarchitektur zu machen.[181] Aus diesem wurden dann die Archetypen abgeleitet, bei denen es sich um die folgenden handelt:

- Alignment,
- Ableitung,
- Bindung,
- Vereinigung.[182]

Mit *Alignment* wird das aneinander Ausrichten heterogener Artefakte beschrieben, die sich unabhängig voneinander ändern können.[183] Der Archetyp *Ableitung* hat seinen Ursprung in der Identifikation der Archetypen anhand eines Metamodells für Unternehmensarchitekturen. Als Beispiele der Ableitung eines neuen Typs aus einem vorhandenen Elementtyp werden die Ableitung der Aufbau- aus Ablauforganisation und die Ableitung der Software- aus der Datenarchitektur genannt. Der Archetyp *Bindung* bezieht sich auf die Verbindung gleichartiger Artefakttypen. Für diesen Archetyp werden explizit Verbindungen mit Hilfe von EAI-Plattformen u.ä. angeführt. Der letzte Artefakttyp, die *Vereinigung* wird so beschreiben, dass nach dessen Anwendung das Metamodell weniger Artefakttypen aufweist. An dieser Stelle sei angemerkt, dass es zweifelhaft ist, ob sich dies nicht eigentlich auf das Modell bezieht, denn auf das Metamodell. Die Entfernung von Metamodellelementen würde den Beschreibungsraum eines Metamodells verändern und darüber hinaus zu Inkonsistenzen vormals erzeugter Modellinstanzen führen. Es ist also davon auszugehen, dass damit Elemente innerhalb eines Modells gemeint sind. Als Erwartung im Hinblick auf die Klassifizierung der Muster bezüglich der Integrationsarchetypen ist zu formulieren, dass es sich bei den meisten Mustern um solche vom Typ *Bindung* handeln wird. Die Tabelle 5.13 klassifiziert die Musterauswahl anhand der Archetypen.

[181] Vgl. (Winter, 2008).
[182] Vgl. (Winter, 2009a).
[183] Vgl. (ebenda, S. 25).

Pattern	**Quelle**	**Fokus**
Adapter Style	(Andersson und Johnson, 2001, S. 229)	Bindung
Adapter static	(ebenda, S. 229)	Bindung
Adapter dynamic	(ebenda, S. 229)	Bindung
Adapter thin	(ebenda, S. 229)	Bindung
Adapter thick	(ebenda, S. 229)	Bindung
Adapter centralized	(ebenda, S. 229)	Bindung
Adapter distributed	(ebenda, S. 229)	Bindung
Adapter	(Khomh und Guéhéneuc, 2008a)	
(Integration) Adapter	(Lutz, 2000)	Bindung
Component Wrapper	(Goedicke und Zdun, 2002, S. 7ff.)	Bindung
Wrapper Facade	(Schmidt; Stal u. a., 2000, S. 47ff.)	
(Legacy) Wrapper	(Mularz, 1995)	Bindung
Facade	(Khomh und Guéhéneuc, 2008a)	
Integration Facade	(Lutz, 2000)	Bindung
Service Interface	(Trowbridge, 2003)	Bindung
Integration Messenger (a, broker)	(Lutz, 2000)	Bindung
Integration Messenger (b, message queuing)	(ebenda)	Bindung
Integration Messenger (c, publish subscribe)	(ebenda)	Bindung
Proxy	(Buschmann; Meunier u. a., 1996, S. 263ff.)	Bindung
Proxy	(Khomh und Guéhéneuc, 2008a)	
Mediator	(Gamma u. a., 1995, S. 277)	Bindung
Integration Mediator	(Lutz, 2000, S. 71)	Bindung
Mediator	(Buschmann; Henney und Schmidt, 2007a, S. 411)	Bindung
Mediator	(Khomh und Guéhéneuc, 2008a)	
Mediator Style	(Andersson und Johnson, 2001, S. 230)	Bindung
Message Router	(ebenda, S. 231)	Bindung
Message Router	(Hohpe und Woolf, 2003, S. 81)	Bindung
Message Router	(Buschmann; Henney und Schmidt, 2007a, S. 232)	Bindung
Gateway	(Andersson und Johnson, 2001)	Bindung
Service Gateway	(Trowbridge, 2003, S. 293ff.)	Bindung
Broker	(ebenda, S. 207f.)	Bindung
Direct Broker	(Trowbridge u. a., 2004, S. 155,215ff.)	Bindung
Indirect Broker	(ebenda, S. 221)	
Message Broker	(ebenda, S. 240f.)	Bindung
Message Broker	(Hohpe und Woolf, 2003, S. 322ff.)	Bindung
Broker	(Buschmann; Meunier u. a., 1996, S. 119ff.)	Bindung
Broker Revisited	(Kircher; Voelter u. a., 2004, S. 581)	Bindung
Broker	(Harrison und Avgeriou, 2007, S. 267f.), bezieht sich auf Buschmann	
Pipes and Filter	(Hohpe und Woolf, 2003, S. 322ff.) , (ebenda, S. 70ff.)	Bindung
Pipes and Filter	(Buschmann; Meunier u. a., 1996, S. 67ff.)	Bindung
Pipes and Filter	(Trowbridge u. a., 2004, S. 292f.)	Bindung
Pipes and Filters	(Garlan und Shaw, 1993, S. 7f.)	Bindung
Pipes and Filters	(Meunier, 1995, S. 437)	Bindung
Pipes and Filters	(Harrison und Avgeriou, 2007, S. 267f.) Bezieht sich auf Buschmann	
Observer	(Khomh und Guéhéneuc, 2008a)	
Observer	(Trowbridge, 2003, S. 129f.)	Bindung
Publish and Subscribe	(Trowbridge u. a., 2004, S. 280f.)	Bindung
Process Automator	(Lutz, 2000)	Bindung
Database Federation	(Andersson und Johnson, 2001)	Bindung
Desktop Integration	(ebenda)	Bindung
Data Transfer Object	(Trowbridge, 2003, S. 229)	Bindung
Singleton	(ebenda, S. 257)	Bindung
Singleton	(Khomh und Guéhéneuc, 2008a)	
File Gateway	(Erl, 2009, S. 457)	Bindung
Process Centralizazion	(ebenda, S. 193)	Bindung
Protocol Bridging	(ebenda, S. 687)	Bindung

Klassifikation bezüglich des Integrationsarchetyp – Fortsetzung auf der nächsten Seite

Pattern	Quelle	Fokus
Legacy Wrapper	(Erl, 2009, S. 441)	Bindung
Service Broker (compund Pattern)	(ebenda, S. 707)	Bindung
Enterprise Service Bus (compund Pattern)	(ebenda, S. 704)	Bindung
Proxy Capability	(ebenda, S. 497)	Bindung
Service Facade	(ebenda, S. 333)	Bindung
UI Mediator	(ebenda, S. 366)	Bindung
Data Model Transformation	(ebenda, S. 671)	Bindung
Data Format Transformation	(ebenda, S. 681)	Bindung
Asynchronous Queuing	(ebenda, S. 582)	Bindung
Intermediate Routing	(ebenda, S. 549)	Bindung

Tabelle 5.13: Klassifikation bezüglich des Integrationsarchetyp

Diese Facette kann erst ihre vollständige Aussagekraft entfalten, wenn Integrationsmuster aus anderen Subsystemen als dem Softwaresubsystem in die Klassifikation integriert werden.[184] Unterschiedliche Softwaretypen, welche aus der Innensicht des Softwaresubsystems heterogen sind, werden – da es sich bei allen Komponenten um Software handelt – als homogen interpretiert, was dazu führt, dass die Muster zum Typ Bindung zählen. Insofern erfüllt sich zunächst die Erwartung, dass sich alle Muster auf homogene Artefakte vom Typ Software beziehen.

5.4.5 Facette Komponente der Referenzarchitektur

Ein Mechanismus der Wiederverwendung ist die Beschreibung von Referenzarchitekturen. Im Fall dieser Arbeit sind Integrationsreferenzarchitekturen von Interesse. Da eine Architektur aus einer Menge von Mustern besteht, ist es sinnvoll diese beiden Mechanismen der Wiederverwendung zusammenzuführen.[185] Dazu wird mithilfe dieser Facette herausgearbeitet, auf welches Element einer Referenzarchitektur sich ein Muster bezieht. Auf dieser Basis können die korrespondierenden Elemente der Muster- und Referenzarchitekturbeschreibungen im Zusammenhang betrachtet werden. Es kann entschieden werden, welche Detailausprägungen – in Form von Mustern – innerhalb eines Referenzarchitekturelementes notwendig sind. Weiterhin kann geprüft werden ob die Summe an Mustern, welche in einem Referenzarchitekturelement umgesetzt werden sollen, die Gültigkeit der Instanz der Architektur bezüglich der Referenzarchitektur beeinflussen und zu guter Letzt können konkrete Integrationshilfsmittel daraufhin geprüft werden, inwieweit sie Referenzarchitekturelemente und die gewünschten Muster umsetzen. Zur Erarbeitung der Fokusse dieser Facette werden bestehende Referenzarchitekturen analysiert und Gemeinsamkeiten bzw. Unterschiede herausgearbeitet. Grundvoraussetzung ist, dass die Referenzarchitekturen explizit die Thematik der Integration adressieren müssen. Die nachfolgende Tabelle gibt einen Überblick über die analysierten Quellen.

Die erste zu analysierende Referenzarchitektur ist der **Integration Architecture Blueprint**: Leitfaden zur Konstruktion von Integrationslösungen.[186] Die Autoren schreiben alle aus dem Kontext eines am Markt aktiven Integrationsdienstleisters, was eine hohe Praxisorientierung verspricht.

Ein Blueprint wird als „ein umfangreicher Bauplan zur effektiven Implementierung einer bestimmten Applikationsform, wie beispielsweise einer Web-Anwendung, einer Business Intelligence Lösung, eines Output Management Systems oder einer Integrationsplattform in einer definierten

[184] Vgl. Ausblick in Kapitel 8.
[185] Vgl. Abschnitt 5.1.3.
[186] Vgl. (Liebhart u. a., 2008).

Referenzarchitektur	Quellen	Beschreibung
Quasar	(Engels; Hess u. a., 2008), (Engels und Voß, 2008)	Trennung in Präsentations-, Logik- und Datenintegration, Komponentenarchitektur mit Lauf- und Entwicklungszeit
Integration Architecture Blueprint	(Liebhart u. a., 2008)	Schichtenarchitektur inclusive querschnittlicher Komponentenarchitektur, nur Laufzeit
Architektur nach Hofmeister	(Hofmeister, 2008)	Schichtenarchitektur
Architektur nach Keller und Erweiterung nach Vogel	(Keller, 2002), (Vogel u. a., 2009)	Schichtenarchitektur

Tabelle 5.14: Übersicht der Integrationsreferenzarchitekturen

Umgebung“[187] bezeichnet.[188] Dabei betonen die Autoren, dass ein Blueprint auf erprobten Techniken, Komponenten und Mustern aufbaut und diese entsprechend den Anforderungen der Zieldomäne in einer gültigen Struktur anordnet. Weiterhin sollen die Beschreibungen eines Blueprint abstrakt gehalten werden, um sie im konkreten Projekt zu verfeinern und zu vervollständigen. Der vorgestellte Blueprint wird hersteller-, produkt- und technologieneutral gehalten.[189] Die Tabelle 5.15 stellt den Aufbau des Blueprint bestehend aus den beiden Sichten, den zugehörigen Level und Layern sowie Rollen und Bausteinen dar.

Grundsätzlich gliedert sich der Blueprint in 2 Sichten. Die *Application and Information View* und die *Integration View*. In der *Application and Information View* werden alle Systeme zusammengefasst, welche nicht Bestandteil der Integrationsarchitektur und damit des Gestaltungsbereiches der Referenzarchitektur sind. In der Terminologie dieser Arbeit gesprochen, sind in dieser Sicht die Integrationsobjekte zu finden. Es werden Typen von Systemen (Integrationsobjekten) unterschieden. Die erste Gruppe sind transaktionale Informationsspeicher (klassische, relationale Datenbanken und Nachrichtensysteme) mit dem Schwerpunkt auf Datenintegration. Die zweite Gruppe bilden nicht transaktionale Informationsspeicher (Dateisysteme) welche ebenfalls schwerpunktmäßig bei der Datenintegration eine Rolle spielen. Die dritte Systemgruppe sind Applikationen, die sowohl transaktional als auch nicht transaktional sein können, auf die aber über ein standardisiertes Application Programming Interface (API) zugegriffen werden kann. Bei diesen Systemen liegt der Fokus auf einer Applikations- oder Prozessintegration. Mit dieser Systematisierung werden Legacy-Systeme zu einem bestimmten Grad ausgeblendet, bei denen in einem ersten Schritt die Integrationsfähigkeit – zum Beispiel durch die Schaffung von Schnittstellen – erst hergestellt werden muss. In der Realität sind aber gerade diese Systeme Gegenstand besonders intensiver Integrationsbemühungen, da hier oft Lock-In-Effekte zum Tragen kommen, welche eine Ablösung verhindern.

Die zweite Sicht ist der *Integration View*, welcher die Gestaltung der Integrationsarchitektur erfasst und den Hauptteil des Blueprint bildet. Diese gliedert sich in drei Schichten (Level) mit entsprechenden Unterschichten: Transport-Level, Integration Domain und Application. Jedes Level erfasst unterschiedliche Aspekte einer Integrationsarchitektur. Das Transport-Level die Aufgabe Informationen von einem Integrationsobjekt zum Anderen zu transportieren. Die Integration Domain löst vorwiegend Probleme des Zugriffs und der Verteilung, sowie die Vermittlung (*Mediation*) und Transformation, sofern Heterogenität besteht. Zusätzlich dazubildet der Process-

[187] (ebenda, S. 89)
[188] Vgl. auch (Kruchten, 1995) zum Begriff des Architectural Blueprints.
[189] Vgl. (Liebhart u. a., 2008, S. 89).

View	Level	Layer	Rolle/ Beschreibung	Bausteine
Application u. Information			Quelle	Applications, transactional/ non transactional Information Storage
			Ziel	
Integration	Transport	Communication	Transporter, Kapselung der technischen Details	Transportprotokolle Transportformate
	Integration Domain	Collection/ Distribution	Distributor / Collector, verbinden von Komponenten	Adapter / Connector Gateway Mapper Data Mapper DAO
		Mediation	Weiterleiten der Informationen, Transformation, Filterung	Canonical Data Model Message Endpoint (Event Driven Consumer, Polling Consumer) Message Construction (Channel) Message Routing (Content-Based, Dynamic, Splitter/ Aggregator) Message Transformation (Filter, Enricher)
	Application	Process	Orchestrating, Steuerung der Integrationsprozesse durch Steuerung der Komponenten im Mediation Layer	Job-Scheduler Portal Workflow(Sequence, Parallel Split, Synchronization, Merge, Multiple Choice) Event Processing (Filter, Aggr. over Window, Correlation Join)
Querschnittsfunktionen (von mehreren Layern genutzt)	Assembly/Deployment		Konfigurationen der Komponenten und Dienste	-
	Transaction		Transaktionsinfrastruktur	-
	Security/Management		Security- und Betriebsinfrastruktur	-
	Monitoring, BAM, Qos		Betriebsüberwachung	-
	Governance		Artefakte sind Basis für SLA's und QoS (fachliche Regularien)	-

Tabelle 5.15: Integration View

Quelle: In Anlehnung an (Liebhart u. a., 2008, S. 90ff.)

Layer die Zentralisierung der Interaktionen zwischen den Integrationsobjekten ab. Den Abschluss bilden Querschnittsfunktionen wie Assembly / Deployment, Transaction, Security, Monitoring und Governance. Diese müssen zwar Bestandteil einer Integrationsarchitektur sein, werden aber von allen anderen Schichten gleichermaßen genutzt. Darüber hinaus sind diese Funktionen nicht immer integrationsspezifisch und finden sich ebenfalls in Referenzarchitekturen für Anwendungen.

In den Ausführungen wird bereits eine 1:1-Zuordnung von bekannten Mustern zu Elementen der Referenzarchitektur vorgenommen.[190] Diese Zuordnung kann als Anhaltspunkt für die spätere Klassifikation verwendet werden.

Da durch den Layer das Level und die Sicht eindeutig bestimmt sind, bietet es sich an dieses Merkmal für die Fokusse der diskutierten Facette zu verwenden. Dazu ist es notwendig zunächst *Application & Information* als Layer zu deuten und entsprechend zu verwenden, da sonst diese Informationen wegfallen würden. Aus dem Integration View verbleiben dann *Communication*, *Collection/Distribution*, *Mediation* und *Process*. Hinzukommen sämtliche Querschnittsfunktionen.

Die **Qualitätssoftwarearchitektur** (Quasar Enterprise) ist eine weitere praxisorientierte Referenzarchitektur, welche ebenfalls aus dem Umfeld eines Integrationsdienstleisters stammt. Da

[190] Vgl. Tabelle 5.15 in der Spalte Bausteine.

Quasar neben Services der Laufzeit auch jene der Entwicklungszeit umfasst ist sie – im Sinne einer Integrationsdienstleistung – die vollständigste der untersuchten Referenzarchitekturen. Im Vergleich zu den anderen Referenzarchitekturen ist damit ein signifikantes Alleinstellungsmerkmal gegeben. Für die in dieser Arbeit bezweckte Untersuchung von Integrationsmustern sind die Services der Entwicklungszeit nur von untergeordneter Bedeutung, weshalb sie für die Extraktion der Fokusse der hier beschriebenen Facette weitgehend ausgeblendet werden.

Quasar liefert streng genommen getrennte Referenzarchitekturen für die grundsätzlichen Integrationsstrategien der Präsentationsintegration (P), der Logikintegration (L) und der Datenintegration (D) und unterscheidet entsprechende Sichten. Die Tabelle 5.16 führt die einzelnen Komponenten nach Quasar auf und kennzeichnet Ihr Vorkommen in der Präsentations-, Logik- oder Datenintegration.

Um die einzelnen Architekturen für die drei Integrationsstrategien in einer Architektur zusammenzuführen und eine Vergleichbarkeit mit den anderen Referenzarchitekturen herzustellen, werden zunächst die Gemeinsamkeiten und Unterschiede der einzelnen Sichten analysiert. Ziel ist es, zu einer konsolidierten Gesamtsicht unabhängig von der Integrationsstrategie zu gelangen. Erst nach dieser Konsolidierung ist einen Vergleich mit den Referenzarchitekturen möglich, welche diese Unterscheidung nicht zugrunde legen. Es wird bewusst von diesem Quasar-Merkmal abstrahiert.

Darüber hinaus ist in Quasar an einigen Stellen eine Vermischung von was und wie zu beobachten. Teilweise werden die Services nicht nach dem Was gegliedert, sondern das Wie – also die Art und Weise der Serviceerbringung – wird anstelle des Was genannt oder mit diesem vermischt. Auch hier wird eine Vereinheitlichung im Sinne des Was angestrebt.

Die Darlegung beginnt zunächst mit den Services der Entwicklungszeit, da deren Zahl geringer ist und diese damit überschaubarer sind. Für die Problemstellung der Transformation[191] fällt auf, dass für die Logikintegration von Transformationsmodellierung, bei der Präsentationsintegration hingegen von Transformationsentwicklung gesprochen wird.[192] Bei der Verwendung eines Modells im Sinne der modellgetriebenen Softwareentwicklung – welche mit Transformationsmodellierung – angesprochen sein könnte – handelt es sich um eine spezielle Form der Entwicklung. Deshalb können Transformationsmodellierung und -entwicklung zusammengefasst und nur noch mit *Transformationsentwicklung* bezeichnet werden. Diese Bezeichnung subsumiert alle Arten der Transformationsentwicklung.

Der Begriff Adapterentwicklung[193] verwirrt in den Ausführungen zu Quasar. In den Erläuterungen zu den korrespondierenden Services der Laufzeit wird bei der technischen Transformation[194] explizit der Adapter angeführt. Wenn es sich bei einem Adapter um ein Element handelt, welches dem Bereich Transformation zugeordnet werden kann, so ist es fraglich, inwieweit der Service der Entwicklungszeit allein stehen kann oder ob dieser nicht unter die Transformationsentwicklung fällt. Einzig, die Argumentation, dass für die Entwicklung eines Adapters eine Programmiersprache verwendet wird, welche nicht zwingend Bestandteil der Plattform sein muss, vermag die Trennung zu begründen. Es wurde sich gegen eine Trennung entschieden und dieser Service als Bestandteil der Transformationsentwicklung betrachtet, was zu einem Wegfall dieses Services der Entwicklungszeit führt.

In Tabelle 5.17 nehmen die Komponenten *Prozessmodellierung*, *Rollen, Rechte- & Benutzerpflege* und *Kompositionsentwicklung* eine Sonderstellung ein. In der ursprünglichen Darstellung im Rahmen von Quasar handelt es sich um Komponenten, die der Laufzeit zugeordnet wer-

[191] Im Allgemeinen sind Transformationen notwendig, wenn Heterogenitätskonflikte vorliegen. Vgl. Abschnitt 5.4.6.
[192] Vgl. (Engels; Hess u. a., 2008, S. 237 u. S. 242).
[193] Vgl. (ebenda, S. 241).
[194] Vgl. (ebenda, S. 237f.).

Zeit	Gruppe	Komponente	P	L	D
Laufzeit	Laufzeit-management	Monitoring		X	
		Ausfallsicherheit		X	
		Lastverteilung		X	
		Fehlerbehandlung		X	
	Transfor-mation	fachliche	X	X	X
		technische	X	X	X
		Cleansing			X
		Matching, Split&Merge			X
	Kommunika-tion	Protokollierung		X	
		Lieferung	X	X	
		dyn.&stat. Adressierung		X	
		Stubs domänspez. Services			X
		Massendatenzugriff			X
		Einzeldatenzugriff			X
	Sicherheit	Identifizierung& Authentifizierung		X	
		Zugriffsschutz		X	
		Single Sign-on	X		
	Inhalte-steuerung	Prozess& Workflow	X		
		Personalisierung	X		
		Autorisierung	X		
		Komposition	X		
	Quer-schnittliche Portaldienste	Suche	X		
		Collaboration& Community	X		
		Content Management	X		
		Tracking	X		
	Inhaltspflege	Prozessmodellierung	X		
		Rollen, Rechte-,& Benutzerpflege	X		
		Kompositionsentwicklung	X		
	Benutzer-schnittstelle	Benutzerkanalsicherung	X		
		Multikanalunterstützung	X		
		Darstellung	X		
		Dialogsteuerung	X		
	Prozess-steuerung	Process Repository			X
		Process Engine		X	X
		Business Activity Monitoring		X	X
		Transaktionen		X	
		Ereignisverwaltung		X	
	Datenpflege	Reporting			X
		Suchen & Ändern			X
		Manuelle Dubletten Bereinigung			X
	Datenhaltung	Datenmodell			X
		Hierarchien, Taxonomien, Relationen			X
		Datenbasis			X
		Historische Daten			X
	Workflow	Postkorb		X	
		Dialogeinsprung		X	
	Repository	Organisationsrepository		X	
		Prozessrepository		X	
		Servicerepository		X	
Entwicklungs-zeit	Entwicklung	Prozessmodellierung		X	X
		Transformationsmodellierung		X	X
		Adapterentwicklung		X	
		Konfigurationsmanagement	X	X	X
		Oberflächenentwicklung	X		
		Kommunikationsentwicklung	X		
		Transformationsentwicklung	X		
		Stammdatenmodellierung			X
		API Entwicklung			X
		Zugriffsberechtigungsmodell			X

P: Präsentationsintegration; L: Logiktintegration; D: Datenintegration

Tabelle 5.16: Komponententypen und Untertypen nach Quasar
Quelle: In Anlehnung an (Engels; Hess u. a., 2008)

Komponente	P	L	D
Transformationsentwicklung (Adapter)	X	X	X
Konfigurationsmanagement	X	X	X
Oberflächenentwicklung	X		
Kommunikationsentwicklung	X		
Stammdatenmodellierung			X
API Entwicklung			X
Zugriffsberechtigungsmodell			X
Prozessmodellierung	X	X	X
Rollen, Rechte-,& Benutzerpflege	X		
Kompositionsentwicklung	X		

P: Präsentationsintegration; L: Logiktintegration; D: Datenintegration

Tabelle 5.17: Bereinigte Quasar Komponenten der Entwicklungszeit

den. Insbesondere im Fall der Prozessmodellierung ist auffallend, dass dies nur in der Sicht der Präsentationsintegration der Fall ist. In den anderen Sichten wird diese Komponente der Entwicklungszeit zugeordnet. Darüber hinaus ist als Kritik an Quasar anzumerken, dass eine Präsentationsintegration ohne eine initiale Prozessmodellierung nicht möglich bzw. nicht effizient ist. Wird ein solcher Service zur Laufzeit offeriert, so ist anzunehmen, dass er – für die gleiche Aufgabe – auch zur Entwicklungszeit genutzt wurde. Diese Komponenten lassen sich also je nach Sichtweise sowohl der Laufzeit als auch der Entwicklungszeit zuordnen und nehmen damit eine hybride Stellung ein. Ziel der Verfügbarmachung dieser Komponenten der Entwicklungszeit zur Laufzeit ist es, dem Nutzer einer Integrationsarchitektur die Möglichkeit zu gegeben diese – in einem gewissen Rahmen – zu verändern. Dennoch können diese Services aus den Betrachtungen ausgeblendet und eindeutig der Entwicklungszeit zugeordnet werden. Durch die Veränderung der bestehenden Konfigurationsartefakte einer Plattform – die als Umsetzung einer Integrationsarchitektur gesehen werden kann – werden neue Integrationsbeziehungen entwickelt oder bestehende maßgeblich verändert. Die Veränderung einer bestehenden Integrationsbeziehung kann als die Etablierung einer neuen Beziehung mit weitgehend ähnlichen Parametern aufgefasst werden. Erst wenn diese aktiviert wird, ist es möglich, dass auf der Basis der Integrationsbeziehung Interaktionen stattfinden. Deshalb werden diese Services ebenfalls der Entwicklungszeit zugeordnet, da im Fokus der Betrachtungen die Integrationsarchitektur als solche steht und die Entwicklung von z. B. Konfigurationsartefakten außen vor gelassen wird.

Als ein erster Konsolidierungsschritt für die Komponenten der Laufzeit werden Dopplungen entfernt, welche durch die Einordnung eigentlich gleicher Komponenten in unterschiedliche Komponentengruppen entstehen. Da die Bildung der Fokusse der Facette und der Vergleich mit anderen Referenzarchitekturen nur auf der Ebene der Komponenten durchgeführt wird, wären diese bei Dopplungen nicht disjunkt, was jedoch eine zwingende Eigenschaft der Fokusse einer Facette ist. So wird das *Prozessrepository* in der Sicht Logikintegration dem Untertyp *Repository* zugeordnet.[195] In der Datenintegrationssicht hingegen erfolgt eine Zuordnung zu *Prozessteuerung*[196], außerdem wird der englische Bezeichner *Process Repository* verwendet. Dass es sich um eine eindeutige Dopplung handelt, legen die Beschreibungen dar. In der Logikintegrationssicht handelt es sich um einen Baustein, welcher als Speicher der Prozessdefinitionen dienen soll, in dem diese versioniert abgelegt werden.[197] In der Datenintegration wird nur die Gruppe der *Prozessteuerung* näher erläutert, welche die internen Prozesse der Datenaktualisierung und

[195] Vgl. (Engels; Hess u. a., 2008, S. 238).
[196] Vgl. (ebenda, S. 247).
[197] Vgl. (ebenda, S. 238).

-verteilung automatisieren soll.[198] Dazu ist eine Speicherung der Prozesse notwendig, welche geschlussfolgert in dem *Process Respository* erfolgt. In beiden Fällen handelt es sich um die identische Aufgabe, lediglich die gewählten Prozesse divergieren. In der Gruppe der Inhaltesteuerung findet sich ein Service *Prozess & Workflow*, welcher sehr ähnlich zu den Services der Gruppe *Prozesssteuerung* zu sein scheint. Dies bestätigt sich, da die Autoren diese Gruppen dahin gehend voneinander abgrenzen, dass es sich bei *Prozess & Workflow* um die Steuerung von Arbeitsschrittfolgen in einem Portal handelt und nicht um die Orchestrierung von Services auf der Ebene der Logikintegration. Dieser Service wird entsprechend des Was in *Steuerung von Portalarbeitsschrittfolgen* umbenannt.

Bei der *Autorisierung* – obwohl im Zusammenhang mit der Inhaltesteuerung im Rahmen der Präsentationsintegration verwendet – handelt es sich eindeutig um einen Sicherheitsdienst. Dementsprechend wird dieser Service in die Gruppe der Sicherheit umgeordnet. Gleiches trifft auf die Benutzerkanalsicherung zu. Die Einordnung in die Gruppe Benutzerschnittstelle orientiert sich zwar an dem Was, führt jedoch dazu, dass man sich fragen muss, warum es eine eigene Gruppe Sicherheit in der Referenzarchitektur geben muss, wenn Bestandteile der Sicherheitsgruppe in anderen Gruppen einsortiert werden. Dies kann mit der Gewichtung der Einordnungsmerkmale zusammenhängen und der Tatsache geschuldet sein, dass die Quasar Autoren die Zusammenfassung aller Benutzerschnittstellenservices höher gewichtet haben. Für die Bildung von disjunkten Fokussen einer Facette wäre dies jedoch hinderlich, sodass die Benutzerkanalsicherung der Gruppe Sicherheit zugeordnet wird. Darüber hinaus kann sie mit dem Service Zugriffsschutz zusammengefasst werden, da dieser unter anderem die Vertraulichkeit der Nachrichtenübermittlung sicherstellt.[199] Im Wesentlichen werden dabei Verschlüsselungsverfahren eingesetzt, welche sowohl bei der Kommunikation zwischen Applikationen auf der Ebene der Logikintegration, als auch bei einem Zugriff über einen Client oder den Browser verwendet werden. Als Beispiel ist Secure-Sockets-Layer (SSL) zu nennen.

In der Gruppe *Kommunikation* wird ein Service *Protokollierung* aufgeführt, welcher alle Kommunikationsinteraktionen erfassen soll. Zusätzlich dazu gibt es in der Gruppe *Laufzeitmanagement* einen Service *Monitoring*. „Neben der reinen Erfassung von Messwerten sind anschließende Funktionen wie Logging und Alarmierung in diesem Service vorhanden“[200]. Logging kann mit Protokollierung übersetzt werden, was Anlass war, die *Protokollierung* und das *Monitoring* zu *Monitoring & Protokollierung* zusammenzufassen. Gleichzeitig kann der Service Business Activity Monitoring – aus der Gruppe *Prozessteuerung* – als ein Spezialfall angenommen werden, weshalb er mit dem *Monitoring* zusammengefasst wird.

Komponente	**Beschreibung**	**P**	**L**	**D**
Monitoring & Protokollierung (Business Activity Monitoring)	*Nachweis aller Kommunikationsinteraktionen, Überwachung des Plattformverhaltens, Sammeln der Laufzeitdaten der (Geschäfts-)Prozesse*	X	X	X
Ausfallsicherheit	*Verwaltung bewusst realisierter Redundanzen*		X	
Lastverteilung	*Lastverbund, Einhaltung der Performancezusagen*		X	
Fehlerbehandlung	*Behandlung technischer Fehler (Retry, Eskalation, Ignore)*		X	
Reporting	*Auskuft über Datennutzung, -umfang, -qualität und Aktualität*			X
fachliche Transformation	*(Mapping) Umsetzung fachlich definierter Datendarstellungen(Strukturabbildung, Werteabbildung), Abbildung der Geschäftsobjekte aus der Anwendungslandschaft auf das Format der Datenbasis*	X	X	X
technische Transformation	*(Adapter) Umsetzung technischer Protokolle und technischer Darstellungen, einheitliche Nutzung von Geschäftsobjekten*	X	X	X

Bereinigte Quasar Komponenten der Laufzeit – Fortsetzung auf der nächsten Seite

[198] Vgl. (Engels; Hess u. a., 2008, S. 248).
[199] Vgl. (ebenda, S. 239).
[200] (ebenda, S. 240)

Komponente	Beschreibung	P	L	D
Cleansing	*Verbesserung der Datenqualität (automatisch)*			X
Matching, Split&Merge	*automatische Erkennung identischer Geschäftsobjekte, Markierung als Doubletten, halb- o. vollautomatische Zusammenführung*			X
Lieferung	*reine Datenübertragung, Unterstützung verschiedener Kommunikationsstile*	X	X	
dyn.&stat. Adressierung	*auffinden geeigneter Kommunikationspartner*		X	
Stubs f. domänspez. Services	*besondere Funktionen für sepzielle Geschäftsobjekte z.B. Adressprüfung, Konsistenzprüfung v. Kontonummern*			X
Massendatenzugriff	*effiziente Übertragung großer Datenmengen*			X
Einzeldatenzugriff	*Zugriff auf einzelne Geschäftsobjekte*			X
Identifizierung& Authentifizierung	*Verwaltung und Sicherstellung Benutzerauthentizität*		X	
Zugriffsschutz & Benutzerkanalsicherung	*Zusammenfassung aller weiteren sicherheitsrelevanten Funktionenm, Wahrung Geheimhaltung vom Endgerät bis zur Plattform*	X	X	
Single Sign-on	*Realisierung der einmaligen Anmeldung*	X		
Autorisierung	*Berechtigungsprüfung, -regeln*	X		
Steuerung von Portalarbeitsschrittfolgen	*Schrittfolgen bei der Benutzung portalbasierter Anwendungen*	X		
Personalisierung	*Verwaltung benutzerspezifischer Einstellungen*	X		
Komposition	*Definition von Navigationspfaden, Zusammenstellung von Inhalten*	X		
Collaboration& Community	*Benachrichtigung,Chat, gemeinsame Dokumentenerstellung etc.*	X		
Content Management	*Steuerung d. Lebenszyklus der Inhalte*	X		
Tracking	*Sammlung von Informationen über das Benutzerverhalten*	X		
Multikanalunterstützung	*automatische Anpassung der Dargestellten Inhalte z.B. PC, Mobiltelefon etc.*	X		
Darstellung	*Kodierung der Inhalte*	X		
Dialogsteuerung	*reagiert auf technische Navigationsbefehle, weiterverarbeitung ohne Endgerätespezifika als logische Navigationsbefehle*	X		
Process Engine	*Automatisierung interner Prozesse zur Datenaktualisierung und -verteilung, Laufzeitumgebung für die Prozesse aus dem Prozessrepository*		X	X
Transaktionen	*Transaktionsmanager für verschiedene Transaktionssemantiken*		X	
Ereignisverwaltung	*Übersetzung eintreffender Nachrichten und Aufrufe in Ereignisse, sowie Pufferung und Verteilung*		X	
Suchen & Ändern	*elementare Operationen auf Stammdaten, Suche nach Inhalten und Dialogen*	X		X
Manuelle Dubletten Bereinigung	*manuelle, dialoggestüzte Zusammenführung von Geschäftsobjekten*[201]			X
Datenmodell	*Ablage und Nutzung von Metadaten*			X
Hierarchien, Taxonomien, Relationen	*Strukturierung der Daten*			X
Datenbasis	*zentrale Datenhaltung*			X
Historische Daten	*separate Haltung historischer Daten, früherer Versionen der Geschäftsobjekte*			X
Postkorb	*Zuordnung anstehender Aufgaben an menschlichen Benutzer*		X	
Dialogeinsprung	*Aufruf von Komponenten mit Dialog für Benutzerinteraktion*		X	
Organisationsrepository	*Informationen über Zuständigkeiten, Rollen, rollenbasierte Berechtigungen*		X	
Prozessrepository	*Speicher der Prozessdefinitionen, Geschäftsprozessmodelle*		X	X
Servicerepository	*Angaben zu physischen Komponenten der Anwendungslandschaft, Schnittstellen, Operationen*		X	

P: Präsentationsintegration; L: Logiktintegration; D: Datenintegration

Tabelle 5.18: Bereinigte Quasar Komponenten der Laufzeit

Hofmeister[202] hat in seiner Arbeit eine Referenzarchitektur für Composite-Applications entwickelt, wobei die Referenzarchitektur ausschließlich für die Anwendung von serviceorientierten Architekturen auf heterogene Anwendungslandschaften gültig ist. Damit handelt es sich um eine Architektur, welche einen größeren Bereich adressiert als reine Integration, sich jedoch auf ein grundlegendes Architekturparadigma beschränkt. Deshalb kann sie zwar Hinweise für diese Arbeit liefern, ist jedoch nicht in der Lage den vollen Betrachtungsgegenstand der Arbeit abzudecken, da dieser gezielt auf gemischten Architekturen liegt. Die alleinige Anwendung eines singulären Architekturparadigmas kann als Praxis fern bezeichnet werden, da die Regel Architekturen mit mehreren Paradigmen sind.

Gruppe	**Layer**	**Beschreibung**
Composite Application	Service Orchestration	Aggregierung der Services von tieferen Ebenen
	Service Coordination	Orchestierte Entitäten des Service Coordination Layers können in Form von Aggregationen der Funktionalität der Applikationslandschaft beschrieben werden.
	Eventing System	Konsistenzmanagement, Entkopplung der Prozesse und Prozessschritte
	Data Repository	Kontext Verwaltung, verteilte Transaktionen
	Data Exchange & Transformation	Interaktionen mit Back-End-Systemen, Abstraktion von der Kommunikationssemantik
	Connectivity	
Application Landscape	Applikations	Back-End-Systeme

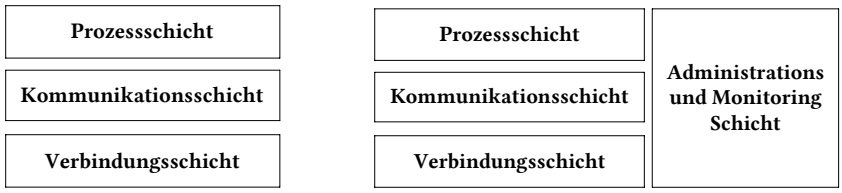
Tabelle 5.19: Bestandteile der Referenzarchitektur nach Hofmeister
Quelle: In Anlehnung an (Hofmeister, 2008)

Es handelt sich bei der Referenzarchitektur nach Hofmeister um eine Schichtenarchitektur, welche eine gröbere Granularität der Schichten aufweist als die beiden vorangegangenen Referenzarchitekturen. Die Integrationsobjekte werden hier als expliziter Bestandteil (*Application Landscape*) der Architektur gesehen. Streng genommen kann mit dieser Referenzarchitektur teilweise auch eine ereignisgesteuerte Architektur[203] realisiert werden, was in dem *Eventing System* begründet liegt.

Die Referenzarchitektur nach Keller[204] gehört zu den frühen Publikationen auf diesem Gebiet. Sie hat im Vergleich zu Quasar relativ wenige Bestandteile, wobei auch hier ein klares Schichtenprinzip verwendet wird. Die Architektur von Keller wird von Vogel erweitert[205], weshalb neben der Variante von Keller (Abbildung 5.7(a)) die Erweiterung nach Vogel (Abbildung 5.7(b)) dargestellt ist.

Vogel u.a. erweitern die Architektur von Keller um eine Administrations- und Monitoringschicht. Damit sind nunmehr vier Schichten voneinander zu unterscheiden. Was den relativ grob granularen Aufbau im Vergleich zu z. B. Quasar mit 41 Komponenten – in der konsolidierten Darstellung,

[201] Der Begriff Geschäftsobjekt wird im Rahmen von Quasar anders gebraucht als im Rest der Arbeit, in dem ein Geschäftsobjekt ein implementierungsunabhängiges, logisches Objekt der fachlichen (z. B. betriebswirtschaftlichen) Ebene bezeichnet. Bei den hier als Geschäftsobjekten bezeichneten Objekten handelt es sich streng genommen um Datenobjekte. Im Sinne einer getreuen Wiedergabe der Quelle wurde die missverständliche Bezeichnung jedoch beibehalten.
[202] Vgl. (Hofmeister, 2008).
[203] engl. Event Driven Architecture (EDA)
[204] Vgl. (Keller, 2002).
[205] Vgl. (Vogel u. a., 2009, S. 440).



(a) Referenzarchitektur nach Keller

(b) Erweiterung nach Vogel

Abbildung 5.7: Referenzarchitektur nach Keller und Erweiterung nach Vogel
Quelle: (Keller, 2002) und (Vogel u. a., 2009, S.)

beschränkt auf die Komponenten der Laufzeit – zeigt.[206] Weiterhin fällt auf, dass die zu verbindenden Integrationsobjekte z.B. Applikationen nicht Bestandteil der Architekturdarstellung sind. Dies steht im Gegensatz zu den anderen drei Referenzarchitekturen, welche in die Betrachtungen einfließen. Diese weisen die zu integrierenden Applikationen explizit aus.

Bisher wurden die verschiedenen Referenzarchitekturen nebeneinanderstehend vorgestellt. Nun nehmen wird eine Synthese vorgenommen, und so die Fokusse der Facette erarbeitet. Dies wird damit begonnen, dass untersucht wird, ob sich die Referenzarchitekturen aufeinander abbilden lassen. Die nachfolgende Tabelle 5.20 zeigt die Zuordnungen.

Quasar	**Blueprint**	**Hofmeister**	**Keller, Vogel**
Monitoring & Protokollierung (Business Activity Monitoring)	Monitoring, BAM, QoS	-	Administrations und Monitoring Schicht
Reporting		-	-
Ausfallsicherheit	-	-	-
Lastverteilung	-	-	-
Fehlerbehandlung	-	-	-
fachliche Transformation	Mediation	Data Exchange & Transformation	-
technische Transformation			-
Cleansing	-	-	-
Lieferung	Communication, (Collection/Distribution)	Connectivity, Data Exchange & Transformation	Verbindungs- / Kommunikationsschicht
dyn.&stat. Adressierung	Mediation		
Massendatenzugriff	Collection Distribution		
Einzeldatenzugriff			
Stubs f. domänspez. Services	-	-	-
-	Collection/Distribution	-	-
Identifizierung& Authentifizierung	Security/Management	-	-
Zugriffsschutz & Benutzerkanalsicherung		-	-
Single Sign-on		-	-
Steuerung von Portalarbeitsschrittfolgen	Process	Service Orchestration, Service Coordination	Prozessschicht
Komposition			
Personalisierung	-	-	-
Collaboration& Community	-	-	-
Content Management	-	-	-
Tracking	-	-	-
Multikanalunterstützung	-	-	-

Synthese der Referenzarchitekturen – Fortsetzung auf der nächsten Seite

[206] Vgl. Tabelle 5.18.

Quasar	**Blueprint**	**Hofmeister**	**Keller, Vogel**
Darstellung	-	-	-
Dialogsteuerung	-	-	-
Process Engine	Process	Service Orchestration / Service Coordination	-
Transaktionen	Transaction	Data Repository	-
Ereignisverwaltung	-	Eventing System	-
Suchen & Ändern	-	-	-
Manuelle Dubletten Bereinigung	-	-	-
Datenmodell	-	-	-
Hierarchien, Taxonomien, Relationen	-	-	-
Datenbasis	-	-	-
Historische Daten	-	-	-
Postkorb	-	-	-
Dialogeinsprung	-	-	-
Organisationsrepository	-		-
Prozessrepository	-	Data Repository	-
Servicerepository	-		-
-	Governance	-	-

Tabelle 5.20: Synthese der Referenzarchitekturen

Die vorangegangene Tabelle zeigt die Abbildung der einzelnen Elemente der Referenzarchitekturen aufeinander. Es wird deutlich, dass die Granularität der Elemente sehr unterschiedlich gewählt ist, weshalb häufig keine 1:1- sondern 1:n-Abbildungen realisierbar sind. Als Fokusse werden alle Komponenten von Quasar sowie *Collection / Distribution* und *Governance* des Integration Architecture Blueprints übernommen.

Pattern	**Quelle**	**Fokus**
Adapter Style	(Andersson und Johnson, 2001, S. 229)	Collection/Distribution
Adapter static	(ebenda, S. 229)	Collection/Distribution
Adapter dynamic	(ebenda, S. 229)	Collection/Distribution
Adapter thin	(ebenda, S. 229)	Collection/Distribution
Adapter thick	(ebenda, S. 229)	Collection/Distribution
Adapter centralized	(ebenda, S. 229)	Collection/Distribution
Adapter distributed	(ebenda, S. 229)	Collection/Distribution
Adapter	(Khomh und Guéhéneuc, 2008a)	
(Integration) Adapter	(Lutz, 2000)	Collection/Distribution
Component Wrapper	(Goedicke und Zdun, 2002, S. 7ff.)	Collection/Distribution
Wrapper Facade	(Schmidt; Stal u. a., 2000, S. S. 47ff.)	
(Legacy) Wrapper Facade	(Mularz, 1995) (Khomh und Guéhéneuc, 2008a)	Collection/Distribution
Integration Facade	(Lutz, 2000)	Collection/Distribution
Service Interface	(Trowbridge, 2003)	Collection/Distribution
Integration Messenger (broker)	(Lutz, 2000)	Lieferung
Integration Messenger (message queuing)	(ebenda)	Lieferung
Integration Messenger (publish subscribe)	(ebenda)	Lieferung
Proxy	(Buschmann; Meunier u. a., 1996, S. 263ff.)	Collection/Distribution
Proxy	(Khomh und Guéhéneuc, 2008a)	
Mediator	(Gamma u. a., 1995, S. 277)	Lieferung
Integration Mediator	(Lutz, 2000, S. 71)	Lieferung
Mediator	(Buschmann; Henney und Schmidt, 2007a, S. 411)	Lieferung
Mediator	(Khomh und Guéhéneuc, 2008a)	
Mediator Style	(Andersson und Johnson, 2001, S. 230)	Lieferung

Klassifikation bezüglich des Elementes der Referenzarchitektur – Fortsetzung auf der nächsten Seite

Pattern	Quelle	Fokus
Message Router	(ebenda, S. 231)	Lieferung
Message Router	(Hohpe und Woolf, 2003, S. 81)	Lieferung
Message Router	(Buschmann; Henney und Schmidt, 2007a, S. 232)	Lieferung
Gateway	(Andersson und Johnson, 2001)	Collection/Distribution
Service Gateway	(Trowbridge, 2003, S. 293ff.)	Collection/Distribution
Broker	(ebenda, S. 207f.)	Lieferung
Direct Broker	(Trowbridge u. a., 2004, S. 155,215ff.)	Lieferung
Indirect Broker	(ebenda, S. 221)	
Message Broker	(ebenda, S. 240f.)	Lieferung
Message Broker	(Hohpe und Woolf, 2003, S. 322ff.)	Lieferung
Broker	(Buschmann; Meunier u. a., 1996, S. 119ff.)	Lieferung
Broker Revisited	(Kircher; Voelter u. a., 2004, S. 581)	Lieferung
Broker	(Harrison und Avgeriou, 2007, S. 267f.), bezieht sich auf Buschmann	
Pipes and Filter	(Hohpe und Woolf, 2003, S. 322ff., 70ff.)	technische Transformation
Pipes and Filter	(Buschmann; Meunier u. a., 1996, S. 67ff.)	technische Transformation
Pipes and Filter	(Trowbridge u. a., 2004, S. 292f.)	technische Transformation
Pipes and Filters	(Garlan und Shaw, 1993, S. 7f.)	technische Transformation
Pipes and Filters	(Meunier, 1995, S. 437)	technische Transformation
Pipes and Filters	(Harrison und Avgeriou, 2007, S. 267f.) Bezieht sich auf Buschmann	
Observer	(Khomh und Guéhéneuc, 2008a)	
Observer	(Trowbridge, 2003, S. 129f.)	Collection/Distribution
Publish and Subscribe	(Trowbridge u. a., 2004, S. 280f.)	Lieferung
Process Automator	(Lutz, 2000)	Process Engine
Database Federation	(Andersson und Johnson, 2001)	Massendatenzugriff
Desktop Integration	(ebenda)	Steuerung von Portalarbeitsschritten
Data Transfer Object	(Trowbridge, 2003, S. 229)	Lieferung
Singleton	(ebenda, S. 257)	Collection/Distribution
Singleton	(Khomh und Guéhéneuc, 2008a)	
File Gateway	(Erl, 2009, S. 457)	Collection/Distribution
Process Centralizazion	(ebenda, S. 193)	Process Engine
Protocol Bridging	(ebenda, S. 687)	technische Transformation
Legacy Wrapper	(ebenda, S. 441)	Collection/Distribution
Service Broker (comp.)	(ebenda, S. 707)	technische Transformation
Enterprise Service Bus (comp.)	(ebenda, S. 704)	Lieferung
Proxy Capability	(ebenda, S. 497)	Lieferung
Service Facade	(ebenda, S. 333)	Collection/Distribution
UI Mediator	(ebenda, S. 366)	Collection/Distribution
Data Model Transformation	(ebenda, S. 671)	fachliche Transformation
Data Format Transformation	(ebenda, S. 681)	technische Transformation
Asynchronous Queuing	(ebenda, S. 582)	Lieferung
Intermediate Routing	(ebenda, S. 549)	Lieferung

Tabelle 5.21: Klassifikation bezüglich des Elementes der Referenzarchitektur

Im Fall des (Integration-) Adapter und der Database Federation ist die gewählte Klassifizierung diskussionswürdig. Der (Integration) Adapter kann auch für Transformationen verwendet werden, ebenso wie im Fall der Database Federation diskutiert werden kann, ob ein Einzeldatenzugriff oder ein Massendatenzugriff die wesentliche Funktion ist. Es wurde sich für die Einordnung in *Collection/Distribution* für den Adapter und *Massendatenzugriff* für die Database Ferderation entschieden. Die folgende Facette untersucht die Eignung der Muster, Interoperabilitätskonflikte bzw. Heterogenitäten zu überbrücken.

5.4.6 Facette Interoperabilitätskonflikt, Heterogenität

Eines der wesentlichen Merkmale von Integrationsmustern ist die Überbrückung von Heterogenität.[207] In der Literatur sind verschiedene Heterogenitäts- bzw. Interoperabilitätskonflikte bekannt, welche in Integrationsprojekten – ebenso wie Muster – keine explizite Anwendung finden. Eine Klassifizierung von Integrationsmustern nach der Art der Heterogenität würde eine Auswahl der Muster anhand des – im Einzelfall – vorliegenden Konfliktes ermöglichen. So können alle Muster, welche eine Lösung für einen spezifischen Konflikt offerieren, gesichtet und anhand weiterer Kriterien verglichen werden.[208] Für die Erarbeitung dieser Facette werden auf der Basis einer umfassenden Literaturauswahl verschiedene Konflikte studiert und eine Zusammenfassung erarbeitet. Im Anschluss werden die identifizierten Integrationsmuster anhand dieser Konflikte klassifiziert. Die nachfolgende Tabelle 5.22 gibt Aufschluss über die analysierten Publikationen. Das Auswahlkriterium – für die zu untersuchenden Publikationen – war eine explizite Auseinandersetzung mit und Systematisierung von verschiedenen Interoperabilitätskonflikten.

Quelle	**Beschreibung**	Ebene
(Kim und Seo, 1991)	Klassifikationsansatz für Heterogenität in Multidatenbanksystemen	Daten
(Kim, 1995)	Interoperabilität moderner Datenbanksysteme	Daten
(Kashyap und Sheth, 1996a; Kashyap und Sheth, 1996b)	semantische und schematische Ähnlichkeiten zwischen Objekten in Datenbanken	Daten
(Goh, 1997)	semantische Konflikte in heterogenen Informationssystemen	Daten
(Sauter, 1998)	Interoperabilität von Datenbanksystemen bei struktureller Heterogenität	Daten
(Sheth, 1999)	Interoperabilität in Informationssystemen	Daten
(Ouksel und Sheth, 1999)	semantische Interoperabilität in globalen Informationssystemen	Informationssysteme
(Davis und Gamble, 2000)	Muster von Interoperabilitätskonflikten	Softwaresysteme
(Conrad, 2002)	Schemaintegration	Daten
(Davis; Flagg u. a., 2003)	Klassifikation von Interoperabilitätskonflikten	Softwaresysteme
(Wache, 2003)	semantische Mediation, also Überbrückung der Konflikte	Daten
(Hergula, 2003)	Föderierte Datenbanksysteme, Daten- und Funktionsintegration	Daten & Funktionen
(Leicher, 2005)	Kompositionskonflikte in Komponentensystemen	Softwaresysteme
(Hepner u. a., 2006)	Konfliktmuster zwischen Softwarekomponenten	Softwaresysteme
(Vogler, 2006)	Prozess- und Systemintegration	Softwaresysteme
(Jung, 2006)	Architekturen zur Datenintegration	Daten
(Agt u. a., 2009)	Modellbasierte semantische Konfliktanalyse für Software- und Datenintegration	Softwaresysteme

Tabelle 5.22: Analysierte Publikationen zur Heterogenität

Durch die Einordnung der Ebene wird deutlich, dass Konflikte bisher besonders aus der Sichtweise von Datenbanken untersucht worden sind. Die folgenden Ausführungen stellen die einzelnen Klassifikationsansätze vor und erarbeiten eine Synthese aus diesen, um einen geeigneten Rahmen für die Klassifikation der Muster nach der Art der Heterogenität zu schaffen.

Die Arbeit von Wache[209] kann man dem Bereich der Datenintegration zuordnen. In diesem Zusammenhang klassifiziert er sehr ausführlich Integrationskonflikte. Dabei bezieht er die Arbeiten von Kim und Seo[210] sowie deren Überarbeitung[211] ein. Weitere Arbeiten die Wache zugrunde

[207] Vgl. Abschnitt 5.1.
[208] Dieses Vorgehen lässt sich auf alle Facetten der hier vorgestellten Klassifikation anwenden.
[209] Vgl. (Wache, 2003, S. 2).
[210] Vgl. (Kim und Seo, 1991).
[211] Vgl. (Kim, 1995).

Art	Typ	Subtyp	Konflikt
Strukturelle Heterogenitätskonflikte	Konflikte bei bilateralen Korrespondenzen		Bezeichnerkonflikte Datentypkonflikte
		Integritätskonflikte	Defaultwertkonflikte Schlüsselkonflikte Integritätsconstraintkonflikte
	Konflikte bei multilateralen Korrespondenzen		multilaterale Attributkorrespondenzen multilaterale Entitätkorrespondenzen fehlende Informationen
	Metaebenen Diskrepanzen		Daten-Attribute-Korrespondenzen Daten-Entitäten-Korrespondenzen Attribute-Entitäten-Korrespondenzen
Semantische Heterogenitätskonflikte	Datenheterogenitäten		Skalierungs und Einheitenkoflikte Repräsentationskonflikte surjektive Abbildungskonflikte
	Domänenheterogenitäten		Subsumptionskonflikte Überlappungskonflikte Inkompatibilitäten Aggregationskonflikte
Inkonsistenz- und Redundanzprobleme			Datenungenauigkeitskonflikte temporale Inkonsistenzkonflikte Redundanzprobleme Konflikte durch fehlende Daten

Tabelle 5.23: Übersicht über die Klassifikation von Integrationskonflikten
Quelle: (Wache, 2003)

legt sind Kayshap und Seth[212] sowie Goh[213]. Die von Wache erstellte Systematisierung integriert die vorgenannten Arbeiten, weshalb diese hier nicht noch einmal gesondert aufgeführt werden.

Auf der obersten Ebene lassen sich strukturelle Heterogenitätskonflikte, semantische Heterogenitätskonflikte sowie Inkonsistenz- und Redundanzprobleme unterscheiden. Die Tabelle 5.23 gibt Aufschluss über die Systematisierung nach Wache.

Die Arbeit von Hergula[214] ist im Gebiet der föderierten Datenbanksysteme angesiedelt und untersucht entsprechende Integrationsszenarien. Hergula unterscheidet zwischen Datenintegration und Funktionsintegration und geht darüber hinaus auch auf Kombinationen beider Integrationsarten ein. Die Beschäftigung mit Funktionsintegration ist ein hervorstechendes Merkmal der Arbeit, da es nur wenige andere Arbeiten gibt, die sich mit einer Systematisierung von Heterogenitätskonflikten der Funktionsintegration beschäftigen. Die Zielstellung ist dabei ähnlich der Bildung eines Föderierten Schemas, die Abbildung einer globalen Funktion auf eine Sequenz von lokalen Funktionen. Bei der Integration von Daten werden semantische und strukturelle Heterogenitäten unterschieden. Für die semantischen Konflikte wird dabei die Arbeit von Sauter[215] wiedergegeben, weshalb die Tabelle 5.24 hier für beide Arbeiten spricht.

Die strukturellen Konflikte werden in Form einer Aufzählung dargestellt. Diese bleibt in der Strukturierung und Systematisierung hinter der Arbeit von Wache zurück.[216] Die folgende Tabelle 5.25 listet die strukturellen Heterogenitätskonflikte der Datenintegration nach Hergula auf und ordnet diesen bereits die korrespondierenden Konflikte nach Wache zu. Dies geschieht in Vorwegnahme weiterer späterer Synthesen der Konflikte der einzelnen Publikationen.

[212] Vgl. (Kashyap und Sheth, 1996a; Kashyap und Sheth, 1996b).
[213] Vgl. (Goh, 1997).
[214] Vgl. (Hergula, 2003).
[215] Vgl. (Sauter, 1998).
[216] Vgl. Tabelle 5.23.

Art	Typ	Subtyp	Konflikt
Unschärfe komplexer Gebilde		unklare Elemente fehlender Kontext Komplexität	
Begriffsungenauigkeit	ausdrucksschwache Bezeichnung		Sprache Abkürzungen undefiniert
	ausdrucksstarke Bezeichnung	Homonyme	lexikalische Definition Umfang Abkürzungen
		Synonyme	

Tabelle 5.24: Semantische Konflikte der Datenintegration nach Hergula und Sauter
Quelle: (Hergula, 2003; Sauter, 1998)

Für die Funktionsintegration konstatiert Hergula, dass alle bei der Datenintegration identifizierten Konflikte weiterhin bestand haben und zu diesen weitere hinzukommen können. Diese werden in der Tabelle 5.26 aufgelistet. Dabei werden diese nicht in Form von einzelnen Konflikten vorgestellt, sondern in Form von aufeinander aufbauenden Fällen. Der komplexere Fall beinhaltet dabei alle Konflikte der weniger komplexen Fälle. Ziel der Funktionsintegration ist es eine globale oder auch föderierte Funktion auf eine Menge lokaler Funktionen abzubilden. Die globale Funktion leitet demnach die Ausführung an die lokalen Funktionen weiter.

Aufgrund der Tatsache, dass mit der Arbeit von Hergula bisher die einzige Arbeit vorliegt, welche sich explizit der Heterogenitätskonflikte bei der Funktionsintegration annimmt, kann kein Vergleich zu den anderen Arbeiten erstellt werden. Die Ausführungen zur Funktionsintegration fließen demnach vollständig in die Synthese der Konflikte zur Definition der Fokusse der hier beschriebenen Facette ein.

Die Arbeit von Leicher[217] ist im Bereich komponentenbasierter Softwaresysteme angesiedelt. Es wird ein modellbasierter Ansatz verwendet, der von jeder Komponente eine Abstraktion in Form eines Strukturmodells (welches ein Typmodell enthält), eines Verhaltensmodells und eines Eigenschaftsmodells erzeugt.[218] Die Analyse der Konflikte erfolgt dann auf der Basis der Modelle. Grundsätzlich unterscheidet Leicher zwischen zwei Zielen. Das erste Ziel ist die Komposition zweier Komponenten (Bindung) und das zweite Ziel der Austausch bzw. Ersatz einer Komponente durch eine Andere. Das letztere Ziel spielt im Rahmen der Integration eine untergeordnete Bedeutung. Da die Arbeiten von Leicher unterschiedlichen Umfang haben und sich die Konflikte sukzessive geschärft haben, findet nur das Modell aus der letzten Quelle Verwendung.[219] Ein früherer technischer Bericht[220] und die Arbeit von Leicher, Busse und Süß[221] werden nur ergänzend zurate gezogen. Die Tabelle 5.27 liefert eine Übersicht über die Konflikte. Die Integritätskonflikte können in der Betrachtung ausgeblendet werden, da sie sich auf die zugrunde liegenden Modelle beziehen und auftreten, wenn ein zu verarbeitendes Modell einer Komponente den Bedingungen des Frameworks nicht genügt und deshalb nicht weiter verarbeitet werden kann.[222]

Die Protokollverletzungen, welche durch das Prüfen ob zwischen zwei Komponenten eine Bisimulation Beziehung gegeben ist, identifiziert werden, beziehen sich auf einen Test bezüglich der

[217] Vgl. (Leicher, 2005).
[218] Vgl. (Leicher, 2004, S. 15f.).
[219] Vgl. (Leicher, 2005).
[220] Vgl. (Leicher, 2004).
[221] Vgl. (Leicher; Busse und Süß, 2005).
[222] Vgl. (Leicher, 2005, S. 86f.).

Konflikt	Korrespondierender Konflikt nach Wache
semi- oder unstrukturierte Daten des Quellsystems, kein formal beschreibenes Schema	n.a.
Schemata in heterogenen Datenmodellen relational vs. objektorientiert	n.a.
persistente Verwaltung der Daten im Zielsystem	temporale Inkonsistenzkonflikte und/oder Redundanzprobleme
1:n Abbildung von Ziel- auf Quellschemata	multilaterale Korrespondenzen
bidirektionale Verbindung der Systeme	Redundanzprobleme und/oder temporale Inkonsistenzkonflikte
Berücksichtigung statischer und Dynamischer Aspekte, es existieren Funktionen, Trigger usw.	n.a.
Überlappende, disjunkte Weltausschnitte	kein struktureller Konflikt, semantische Domänenheterogenität
partielle Replikation, Zielsystem beinhaltet nur eine Untermenge der Ausprägungen	kein struktureller Konflikt, semantische Domänenheterogenität
unterschiedliche Bezeichner, Synonyme und Homonyme	Bezeichnerkonflikte
Datentypkonflikte	Datentypkonflike
n:m Abbildung der Attribute	multilaterale Attributkorrespondenzen
Objekte stehen auch in einem n:m Verhältnis	Multilaterale Entitätkorrespondenzen
vernetzte Objekte durch Referenzen, Primärschlüssel/Fremdschlüsselbeziehungen	Schlüsselkonflikte
Zielobjekte sind auch vernetzt, referentielle Integrität wird unterstützt	Schlüsselkonflikte

n.a.: nicht angebbar

Tabelle 5.25: Strukturelle Heterogenitätskonflikte der Datenintegration nach Hergula
Quelle: (Hergula, 2003)

Fall	Konflikte/Beschreibung
Trivialer Fall	1:1 Abbildung von 2 Funktionen, identische Funktionssemantik, identische Signaturen, Funktions- und Parameternamen differieren
Einfacher Fall	1:1 Parameterabbildung 1:n Parameterabbildung n:1 Parameterabbildung
Unabhängiger Fall	
Abhängiger Fall	lineare Abhängigkeit 1:n Abhängigkeit n:1 Abhängigkeit
Allgemeiner Fall	Iterative Abhäbgigkeit Kombination aller Konflikte möglich

Tabelle 5.26: Heterogenitätskonflikte der Funktionsintegration
Quelle: (Hergula, 2003, S. 24 ff.)

Typ	Subtyp	Konflikt	Erkennung
Structural/ Type Conflicts	mismatched Interface types		subtype relationship
Behavior Conflicts	constraint violations	pre-conditions post-conditions	n.a. n.a.
	protocol violations	name conflicts Kompatibilität Austauschbarkeit	n.a. simulation bisimulation
	property violations	deadlock	n.a.
Property conflicts	communication mechanism technological characteristics		taxonomy based feature model checking
n.a.: nicht angebbar			

Tabelle 5.27: Konflikte nach Leicher
Quelle: in Anlehnung an (Leicher, 2005)

Austauschbarkeit der Komponenten. Für die Zwecke der Arbeit reicht daher ein Test auf eine Simulation-Beziehung.[223]

Die Arbeiten von Keshav und Gamble[224], Davis u.a.[225] sowie Hepner u.a.[226] sind im Zusammenhang zu sehen, da sie erst gemeinsam betrachtet eine vollständige Methodik ergeben. Alle Arbeiten basieren auf einem Grundverständnis von Komponententypen, welche für eine Integrationsarchitektur benötigt werden. Dabei handelt es sich um eine Systematik welche von Keshav und Gamble eingeführt wurde.[227] Es wird zwischen *Controller*, *Translator* und *Extender* als Basiskategorien unterschieden. Darauf aufbauend werden auch Mischformen *Translator-Controller*, *Translator-Controller-Extender*, *Translator-Extender* und *Controller-Extender* beschrieben. In der Arbeit von Davis und Gamble[228] wird eine Mustersprache für Konflikte motiviert und ein Beispielmuster dargelegt. Diese Arbeit ist auch die Einzige – aus den hier vorgestellten – welche einen Abschnitt (*Satisfycing Pattern*) enthält, in dem bekannte Integrationsmuster als Lösungen für diesen Konflikt angeführt werden. Weitere Arbeiten widmen sich der Systematisierung und Identifikation von möglichen Konflikten.[229] Den Arbeiten liegt die Motivation zugrunde, eine umfassende Konfliktbewertung vor der eigentlichen Integrierung vornehmen zu können. Zu diesem Zweck müssen die betreffenden Eigenschaften der Integrationsobjekte miteinander verglichen werden. Davis u.a. legen Ihrer Arbeit eine umfangreiche Analyse bestehender Literatur zugrunde und konnten in dieser ca. 74 verschiedene Charakteristika finden, welche für Softwarearchitekturen gelten. Den Abschluss bildet die Arbeit von Hepner u.a.[230] in der eine Vorstellung der Konflikte in der üblichen Darstellungsform von Mustern erfolgt. Die Tabelle 5.28 listet die Konflikte der verschiedenen Arbeiten auf und ermöglicht einen Vergleich dieser. Die – im Zeitverlauf gesehen – mittlere Arbeit enthält die umfangreichste Zusammenstellung an Konflikten. Warum diese Zusammenstellung in der späteren Arbeit – welche diese in Musterform aufbereitet – nicht vollständig eingeflossen ist, sondern nur 6 Konflikte von ursprünglich 13 übernommen wurden, bleibt unklar. Es werden zunächst die Konflikte verwendet, um die hier beschriebene Facette aufzubauen. Die Taxonomie der Integrationskomponenten (*Controller*, *Translator*, *Extender* usw.)

[223] Vgl. (Leicher, 2005, S. 100).
[224] Vgl. (Keshav und Gamble, 1998).
[225] Vgl. (Davis und Gamble, 2000; Davis und Gamble, 2002; Davis; Flagg u. a., 2003).
[226] Vgl. (Hepner u. a., 2006).
[227] Vgl. (Keshav und Gamble, 1998).
[228] Vgl. (Davis und Gamble, 2000).
[229] Vgl. (Davis und Gamble, 2002; Davis; Flagg u. a., 2003).
[230] Vgl. (Hepner u. a., 2006).

Davis und Gamble,2000	**Davis u.a.,2003**	**Hepner u.a.,2006**
Inconsistent Data	Inconsistent Data	Inconsistent Data
-	Restricted Points of Control Transfer	Restricted Control Transfer
-	Unspecified Control Destination	Unspecified Control Destination
-	Inhibited Rendezvous	Inhibited Rendezvous Conflict
-	Multiple Unsequenced Control Transfers	-
-	Restricted Points of Data Transfer	-
-	Unspecified Data Destination	-
-	Unspecified Data Location	-
-	Invalid Data	Invalid Data
-	Multiple, unsequenced Data Transfers	-
-	Mismatched Data Transfer Assumtions	-
-	Uninitialized Control Transfer	Uninitialized control transfer
-	Uninitialized Data Transfer	-

Tabelle 5.28: Konflikte nach Davis,Gamble und Hepner

Quelle: in Anlehnung an (Davis und Gamble, 2000; Davis; Flagg u. a., 2003; Hepner u. a., 2006)

kann als Basis für eine weitere Facette verwendet werden deren Aufbau in Kapitel 8 diskutiert wird.

Die Arbeit von Agt u.a.[231] basiert auf einem Modell, auf dessen Basis die Konfliktanalyse durchgeführt wird. Es werden die Modellebenen der modellgetriebenen Softwareentwicklung übernommen und zwischen Computation-Independent-Model (CIM), Platform-Independent-Model (PIM) und Platform-Specific-Model (PSM) unterschieden. Die Konfliktanalyse wird dabei auf der Ebene eines PIM angesiedelt. In der Arbeit werden Konflikttypen erläutert, aber keine feingranularen Konflikte. Es wird zwischen den folgenden Typen unterschieden: a) Semantic conflicts, b) Behavior conflicts, c) Property conflicts, d) Structure conflicts und e) Communication conflicts. Agt u.a. unterscheiden orthogonal zu dieser Einteilung in automatisch lösbare, nicht automatisch lösbare und in nicht lösbare Konflikte. Die Typen können zur Einordnung detaillierterer Konflikte verwendet werden.

Vogler[232] beschäftigt sich in ihrer Arbeit mit der Prozess- und Systemintegration und trifft dabei auch Aussagen zu der Heterogenität von Anwendungen. Deshalb kann diese Arbeit in die Ebene der Softwaresysteme eingeordnet werden. Dies, obwohl Teile des umliegenden Informationssystems als Rahmen – im Sinne einer Anforderungsanalyse – betrachtet werden. Der explizite Gestaltungsgegenstand ist jedoch das Softwaresystem. Das Informationssystem wird in der Arbeit von Vogler implizit über die Gestaltung des Softwaresystems verändert, jedoch nicht explizit. Für die Unterscheidung der Heterogenität von Anwendungen verwendet Vogler einen modellbasierten Ansatz. Dieser orientiert sich im Vergleich zu Agt jedoch nicht an den Prinzipien der modellgetriebenen Softwareentwicklung, sondern dient der strukturierten und einheitlichen Darstellung der Applikationen sowie der Partitionierung dieser in Teilmodelle. Es wird zwischen einer logischen und einer physischen Ebene der Anwendungen unterschieden, wobei die logische Ebene die Struktur der Anwendungen erfasst und die physische Ebene die Art der Implementierung.[233] Dieser Einteilung folgend kann zwischen logischer und physischer Heterogenität unterschieden werden, was die grundsätzliche Möglichkeit gleicher logischer Strukturen aber einer unterschiedlichen Umsetzung im physischen Sinne einschließt. Die logische Heterogenität wird – der Intention folgend, dass Anwendungen aus Methoden und Daten bestehen – mit Hilfe der Applikations- und Datenmodelle unterschieden.[234] Dabei wird ein – die Funktionen des

[231] Vgl. (Agt u. a., 2009).
[232] Vgl. (Vogler, 2006).
[233] Vgl. (ebenda, S. 45).
[234] Vgl. (ebenda, S. 46).

Art der Heterogenität	Ausprägung	Beispiel
keine	homogene Modelle	
Prozessmodell	homogene Applikationsmodelle & unterschiedliche Anwendermodelle	integration mehrerer SAP R/3 Instanzen mit unterschiedlichen Anwendern
logische	heterogenes Applikationsmodell & homogenes Datenmodell	
logische	homogenes Applikationsmodell & heterogenes Datenmodell	
logische	heterogenes Applikationsmodell & heterogenes Datenmodell	
physische	homogene Modelle & physische Heterogenität	mehrere Instanzen
physische	homogene Modelle & physische Heterogenität	Verteilung auf mehrere Rechner
physische	Unterschiedliche Technologische Plattformen	
physische	Verteilung auf mehreren Rechnern	
physische	mehrere Instanzen	
physische	Varianten einer Anwendung	gesetzliche Anforderungen, Lokalisierung der Anwendung

Tabelle 5.29: Heterogenität nach Vogler
Quelle: in Anlehnung an (Vogler, 2006)

Informationssystems beschreibendes – Prozessmodell als diesen Modellen zu Grunde liegend dargestellt, wobei jede Applikation einen Teilausschnitt der gesamten Prozesse realisiert.[235]

Streng genommen lassen sich alle Konflikte miteinander kombinieren. Neben den von Vogler dargestellten Kombinationen ist z. B. auch der Fall von prozessualer Heterogenität bei heterogenem Datenmodell und heterogenem Applikationsmodell und gleichzeitiger physischer Heterogenität denkbar. Darüber hinaus können die Heterogenitätsarten nach Vogler auf der Ebene von Typen eingestuft werden. Im Vergleich zu anderen hier untersuchten Arbeiten ist die Ausarbeitung nicht sehr feingranular, kann aber zur Einordnung detaillierterer Konflikte verwendet werden.

Jung[236] betrachtet Architekturen zur Datenintegration, weshalb sich diese Arbeit auf der Ebene von Daten einordnen lässt. Im engen Zusammenhang steht die Arbeit von Conrad[237], welche für Jung eine Grundlage bildet. Diese lässt sich ebenfalls der Ebene der Daten zuordnen. Bei beiden Arbeiten steht die Schemaintegration im Vordergrund, welche die Schemata lokaler Datenbanken in einem gemeinsamen, globalen, integrierten Schema verfügbar macht. Jung vertritt jedoch die Ansicht einer Top-Down-Vorgehensweise – im Gegensatz zu der üblichen Bottom-up-Vorgehensweise – bei der Schemaintegration, um das globale Schema anforderungsgerecht zu gestalten und gegebenenfalls auch Informationen der lokalen Schemata zu vernachlässigen, sofern diese nicht benötigt werden. Die nachfolgende Tabelle stellt den Vergleich der Konflikte nach Conrad und Jung dar. Dabei werden – neben den eigentlichen Konflikten – die Ebenen der Konflikte im Sinne der Unterscheidung zwischen Datenmodell, Datenschema und Instanzen verwendet und der Konflikttyp dargestellt.

Beide Autoren beschreiben im Wesentlichen die gleichen Konflikte. Der Begriff Heterogenitätskonflikt wird anders – als in dieser Arbeit üblich – verwendet. Bei Conrad und Jung beziehen sich

[235] Vgl. Abbildung 3.13 in (Vogler, 2006, S. 47).
[236] Vgl. (Jung, 2006).
[237] Vgl. (Conrad, 2002).
[238] Beispielsweise Modellierung durch ein Entity-Relationship-Modell im Vergleich zu einer objektorientierten Darstellung.
[239] Die semantischen Wiedersprüche, setzen voraus, dass es mindestens eine oder mehrere Entsprechungen je Attribut des globalen Schemas verglichen mit den lokalen Schemata gibt.
[240] Disjunktheit stellt für die Schemaintegration kein Problem dar, da die Tupelmengen einfach vereinigt werden Vgl. (Jung, 2006, S. 161).

Ebene	Jung	Conrad	Typ
Datenmodelle[238]	Heterogenität der Datenmodelle	Heterogenitätskonflikte	heterogene Datenmodelle
	-	strukturelle Konflikte	strukturelle Konflikte
Datenschemata[239]	synonyme Bezeichner homonyme Bezeichner Wertebereichskonflikt Skalierungskonflikt Genauigkeitskonflikt Vagheitskonflikt Integritätskonflikte	synonyme Bezeichner homonyme Bezeichner Wertebereichskonflikt Skalierungskonflikt Genauigkeitskonflikt Vagheitskonflikt Integritätskonflikte	Beschreibungskonflikte
Instanzen	Extensionale Äquivalenz Einschluss/Teilmenge Überlappung Disjunktheit[240]	Extensionale Äquivalenz Einschluss/Teilmenge Überlappung Disjunktheit	Extensionale Konflikte
	Attributwertkonflikte starke Attributäquivalenz schwache Attributäquivalenz disjunkte Attributäquivalenz	- - - -	Datenwertkonflikte

Tabelle 5.30: Vergleich der Konflikte nach Jung und Conrad
Quelle: in Anlehnung an (Jung, 2006) und (Conrad, 2002)

Heterogenitätskonflikte auf die Verwendung unterschiedlicher Datenmodelle. Als Beispiel wird eine objektorientierte im Vergleich zu einer relationalen Modellierung angeführt. Als Folgeerscheinung können semantische Anreicherungen oder die Vermeidung von Verlusten bei der Herstellung einer Vergleichsbasis die Folge sein.[241] Conrad blendet im Gegensatz zu Jung einen Teil der Konflikte auf der Ebene der Instanzen aus – sofern diese nicht die Folge von Beschreibungskonflikten sind.[242] Jung hingegen beschreibt keine strukturellen Konflikte, welche dadurch entstehen, dass sich derselbe Sachverhalt auch bei gleichem Datenmodell unterschiedlich ausdrücken lässt. So kann ein wertbasiertes Attribut auch durch eine Referenz ersetzt werden, ohne dass sich der Informationsgehalt zwingend ändert.[243]

Ouksel und Sheth[244] und weitere Arbeiten von Sheth[245] enthalten identische Konfliktsystematiken. Im Rahmen der hier vorgestellten Arbeiten stellen sie eine Ausnahme dar, da explizit Informationssysteme adressiert werden. Im Gegensatz zu anderen Arbeiten wird hier der Informationssystembegriff in Übereinstimmung mit der in dieser Arbeit vertretenen Auffassung verwendet und ein Informationssystem explizit als Gestaltungsgegenstand adressiert.[246] Vogler beispielsweise betrachtet das Informationssystem als Rahmen gebenden Einflussfaktor, stellt jedoch nur auf die Gestaltung des Softwaresystems ab.[247]

Obwohl die Arbeit explizit Informationssysteme adressiert, erscheint die Menge der benannten Informationssysteme – im Vergleich zur Realität – gering. Auffällig ist, dass der Bezeichnung nach diese lediglich auf die Funktion eines Datenspeichers reduziert werden. Dabei unterscheiden Ouksel und Sheth zwischen Datenbankmanagementsystemen und Digital-Media-Repositories. Eigentlich ist eine Systematisierung anhand der Systemeigenschaften – die zur Heterogenität

[241] Vgl. (Conrad, 2002, S. 104).
[242] Vgl. (ebenda, S. 105).
[243] Vgl. (ebenda, S. 104).
[244] Vgl. (Ouksel und Sheth, 1999).
[245] Vgl. (Sheth, 1999).
[246] Vgl. Kapitel 2.1.
[247] Vgl. (Vogler, 2006, S. 158).

Ebene	Konflikt	Typ	Interoperabilitätsart
Information	Semantic Heterogeneity	Semantic	Semantic
	Representational Schematic Heterogeneity	Structural	Structural
	Format Heterogeneity	Syntactic	Syntactic
System	Digital Media Repository Systems DBMS Data models DBMS concurrency control DBMS recovery	Information System	System
Platform	File System Naming File Types Operation transaction support IPC	Operating Systems	
	instruction set data representation/coding	Hardware	

Tabelle 5.31: Heterogenität und Interopterabilitätsarten nach Ouksel und Sheth
Quelle: in Anlehnung an (Ouksel und Sheth, 1999)

führen – zielführender. Für Datenbankmanagementsysteme wird diese Unterscheidung (Data models, concurrency control, recovery) angedeutet.

Konsolidierung der Heterogenitätsarten und Fokusse der Facette: Die Konflikte nach Ouksel und Shet wurden dafür als Ausgangspunkt verwendet, da sie das Informationssystem als Gestaltungsgegenstand adressieren. Da alle anderen Klassifikationen sich auf Subsysteme eines Informationssystems beziehen, lassen sich alle anderen Arbeiten in diese Systematik einordnen bzw. konfligieren nicht mit dieser. Die erste Arbeit, welche in diese Systematik eingeordnet wird, ist die von Wache. Die Einordnung stellt kein Problem dar, da sich alle Typen von Heterogenitäten finden und eine Zuordnung über die Typen gewährleistet ist. Alle strukturellen Konflikte erweitern den entsprechenden Typ nach Ouksel und Sheth. Mit den anderen Heterogenitätstypen wird analog verfahren. Die Struktur der Systematik soll es ermöglichen, gleichartige Konflikte zu Klassen bzw. Typen zusammenzufassen. Gleichzeitig soll der Bezug zu den Subsystemen eines betrieblichen Informationssystems hergestellt werden, da diese der Gestaltungsgegenstand dieser Arbeit sind.

Umfangreichere Konsolidierungen ergeben sich bei der Einordnung der Konflikte nach Hergula. Da diese Arbeit eine hohe Überschneidung mit der Arbeit von Wache aufweist, findet sich in Tabelle 5.25 bereits eine Gegenüberstellung der sich entsprechenden Konflikte. Dieser Darstellung können die direkten Abbildungen aufeinander entnommen werden, weshalb an dieser Stelle nur auf ausgewählte Fälle eingegangen wird. Bei den Zuordnungen in der Tabelle kann es sich, sowohl um eine Subsumption als auch eine Gleichheit handeln.

Für alle im Folgenden dargelegten Konsolidierungen der Konflikte wird wie folgt verfahren: handelt es sich nicht um einen neu aufzunehmenden Konflikt, bezogen auf die initiale Auswahl der Konflikte von Ouksel und Sheth, sondern wird dieser bereits durch die bestehenden Konflikte abgedeckt, so wird die neue Quelle in der entsprechenden Spalte der Tabelle ergänzt.

Die *persistente Verwaltung der Daten im Zielsystem* kann sowohl auf *temporale Inkonsistenzkonflikte* als auch auf *Redundanzprobleme* abgebildet werden, ebenso wie *bidirektionale Verbindung der Systeme*. Dabei ist die persistente Verwaltung der Daten auch im Zielsystem –es wird vermutlich davon ausgegangen, dass dieser Fakt für das Quellsystem gegeben ist – Grundvoraussetzung dafür, dass Redundanzprobleme als auch temporale Inkonsistenzen überhaupt auftreten können.

Eine bidirektionale Verbindung verschärft dieses Problem, da die Inkonsistenzen nicht mehr nur von der Übertragung geänderter Daten in das Zielsystem abhängen, sondern in diesem auch geändert werden können und der umgekehrte Übertragungs- und Aktualisierungsweg ebenfalls eine Rolle spielt. Eine *1:n-Abbildung von Ziel- auf Quellschemata* entspricht dem Konflikt *multilaterale Korrespondenzen. Überlappende, disjunkte Weltausschnitte* wird als struktureller Konflikt klassifiziert. Nach Wache handelt es sich dabei jedoch um semantische Domänenheterogenität. Als ein gleichartiger Konflikt lässt sich das Problem partielle Replikation auffassen, es handelt sich ebenfalls um *semantische Domänenheterogenität*, da das Zielsystem nur eine Untermenge der Ausprägungen beinhaltet. Unterschiedliche *Bezeichner, Synonyme und Homonyme* finden sich bei Wache als *Bezeichnerkonflikte*. Der Konflikttyp *Datentypkonflikte* ist in beiden Klassifikationen identisch benannt, woraus sich die Zuordnung automatisch ergibt. *N:M-Abbildung der Attribute* sind nach Wache als *multilaterale Attributkorrespondenzen* benannt. Die letztere Bezeichnung wird übernommen. Auf gleich Weise wird bei *Objekte stehen in einem n:m-Verhältnis* verfahren. Hier wird die Bezeichnung *Multilaterale Entitätkorrespondenzen* übernommen. Sowohl *vernetzte Objekte durch Referenzen* als auch *Zielobjekte sind auch vernetzt* finden sich in der Systematik von Wache durch *Schlüsselkonflikte* repräsentiert. *Schemata in heterogenen Datenmodellen relational vs. objektorientiert* lässt sich zwar nicht auf die Konflikte nach Wache abbilden, findet sich jedoch in den Ausführungen von Ouksel und Sheth. Der Konflikt entspricht den Datenmodell Konflikten nach Ouksel und Sheth. Da sich für alle der vorgenannten Konflikte eine Entsprechung in den bisher konsolidierten Arbeiten findet, werden nur die Quellvermerke der Konflikte ergänzt.

Für die nachfolgenden Konflikte findet sich keine solche Entsprechung, weshalb diese in der Systematik ergänzt werden. Der Konflikt *semi- oder unstrukturierte Daten des Quellsystems, kein formal beschriebenes Schema* lässt sich nicht auf die Konflikte nach Wache abbilden. Obwohl man bei weiter Auslegung von Datenmodellheterogenität auch die Abwesenheit eines Datenmodells unter diesen Konflikten subsumieren könnte, wird der Konflikt separat aufgeführt, da die Lösungsmechanismen beider Phänomene sich unterscheiden und die gezieltere Auswahl von Lösungsmustern Gegenstand dieser Arbeit ist. *Berücksichtigung statischer und dynamischer Aspekte* in Form von Funktionen und Triggern, findet ebenfalls keine Entsprechung.

Eine Sonderstellung nehmen die Konflikte der Funktionsintegration ein, die durch Hergula beschrieben wurden, da diese Sicht in keiner anderen Klassifikation enthalten ist. Die Sichtweise, welche diesen Konflikten zugrunde liegt, ist analog zur Datenintegration. Die Rolle des globalen Datenschemas übernimmt eine globale, föderierte Funktion, die auf eine Menge lokaler Funktionen abgebildet wird. Unter Funktionsintegration wird zuweilen auch die Integration von Softwaresystemen über die Nutzung von Schnittstellen auf der Ebene des Funktionskerns verstanden. Diese Integration wird von der Datenintegration und der Präsentationsintegration unterschieden, ist jedoch im Zusammenhang mit der Arbeit von Hergula nicht gemeint. Demzufolge beziehen sich die Heterogenitätskonflikte vornehmlich auf die Abbildung der globalen auf die lokalen Funktionen. Als Beispiel sei der *triviale Fall* genannt, welcher eine 1:1-Abbildung einer globalen Funktion auf genau eine lokale Funktion umfasst. Durch die Einzigartigkeit der beschriebenen Konflikte können diese direkt übernommen und in die Systematik integriert werden. Vor dem Hintergrund der Abbildung eines Geschäftsprozesses – der als globale Funktion gesehen werden kann – helfen diese Konflikte mit der wiederkehrenden Tätigkeit des Business-Technology-Mapping umzugehen[248]. Die Konflikte bauen dabei aufeinander auf, so kann der Fall *Einfacher Fall* alle Konflikte des Falls *Trivialer Fall* beinhalten[249], die Situation *Unabhängiger Fall* erweitert *Einfacher Fall* [250] und so weiter.

[248] vlg. Abschnitt 3.5.3, insbesondere Abbildung 3.10
[249] Vgl. (Hergula, 2003, S. 27).
[250] Vgl. (ebenda, S. 29).

Die Thematik der Funktionsintegration ist damit jedoch nicht vollständig abgedeckt. Für eine vollständige Untersuchung der Funktionsintegration muss der Aspekt der Interaktion der lokalen Funktionen über die Bestimmung der Abhängigkeiten hinaus und sich daraus ergebende Konflikte berücksichtigt werden. Bauen die lokalen Funktionen aufeinander auf und ist die Weitergabe z. B. eines Ergebnisses unter den lokalen Funktionen notwendig, so müssen dafür entweder zentrale koordinierende Mechanismen geschaffen werden oder die Softwaresysteme müssen untereinander – im Sinne korrespondierender benötigter und zur Verfügung gestellter Schnittstellen – kommunizieren. Diese Sicht und die resultierenden Konflikte werden von Hergula nicht erfasst.

Bei der Kosolidierung bzw. der Integration der Arbeiten von Davis und Gamble[251], Davis u.a.[252] und Hepner[253] fällt auf, dass es für einen Teil der Konflikte unerheblich ist, auf welcher Ebene diese auftreten. Obwohl diese Arbeiten alle auf der Ebene von Softwarekomponenten angesiedelt sind, enthalten sie Konflikte wie *Inconsistent Data* und *Invalid Data*, welche sich auch auf der Ebene der Informationen finden. *Inconsistent Data* lässt sich auf *temporale Inkonsistenzkonflikte* abbilden. Damit wird deutlich, dass es für die Art der Informationsheterogenität unerheblich ist, ob sie auf der Ebene der Datenhaltung – im Sinne eines Datenbanksystems – auftritt oder auf der Ebene einer funktionalen Schnittstelle in Form von Rückgabewerten bzw. Datenobjekten. Aus diesem Grund kann der Teil der Informationsheterogenität als wiederverwendbare Einheit gesehen werden, welche sowohl bei der Datenintegration, als auch bei der Funktionsintegration eine Rolle spielt. Aus Gründen der Redundanzvermeidung werden die Konflikte nur unter der Ebene *Informationen* aufgeführt. Das Gleiche gilt für *Invalid Data*. Bei diesem Konflikt handelt es sich um einen von gröberer Granularität, als die bereits enthaltenen Konflikte, da er sich mit allen Konflikten vom Subtyp *Integritätskonflikte* gleichsetzen lässt. Deshalb findet keine Abbildung statt, da bereits ein Satz detaillierterer Konflikte in der Systematik enthalten ist. Die restlichen Konflikte sind in der bestehenden Systematik noch nicht enthalten und werden in dieser entsprechend ergänzt.

Da die Arbeit von Vogler[254] bereits auf der Ebene von Typen eingeordnet wurde, ist für die Integration in die konsolidierte Systematik zu überprüfen, ob bereits Detailkonflikte zu den benannten Typen existieren. Ist dies der Fall, müssen die Typen nicht aufgenommen werden. Sind keine Detailkonflikte enthalten, welche sich auf einen der Typen nach Vogler abbilden lassen, so muss über die Aufnahme des Typen entschieden werden. Vogler betrachtet alle Fälle von Heterogenität, auch den Fall *homogene Modelle*, also die Abwesenheit von Heterogenität. Aus diesem Fall lassen sich keine Rückschlüsse für die Integration ziehen, da diese zwingend Heterogenität voraussetzen, andernfalls ist kein Integrationsaufwand notwendig.[255]

Der Fall eines heterogenen *Prozessmodells* nimmt eine Sonderstellung in der Systematik ein, da er in keiner anderen Klassifikation zu finden ist. Er bezieht sich auf ansonsten homogene Applikationen, welche jedoch von unterschiedlichen Anwendergruppen genutzt werden. Das Business-Technology-Mapping bildet die zu unterstützenden Prozesse auf unterschiedliche Instanzen der gleichen Anwendung ab. Dieser Konflikttyp wurde in die konsolidierte Systematik aufgenommen, in dem die Ebene Prozessmodell hinzugefügt wurde. Der Vollständigkeit halber sei darauf hingewiesen, dass sich ein Teil der bei Vogler beschriebenen Anwendermodelle auch in den Domänenheterogenitäten wieder finden lassen. Überlappungen und Subsumptionen von Geschäftsobjekten werden von diesen erfasst. Unter *heterogenes Applikationsmodell* können alle

[251] Vgl. (Davis und Gamble, 2000).
[252] Vgl. (Davis; Flagg u. a., 2003).
[253] Vgl. (Hepner u. a., 2006).
[254] Vgl. (Vogler, 2006).
[255] Die Überbrückung von Heterogenität ist eines der Hauptmerkmale von Integrationsmustern. Vgl. Abschnitt 5.1.4.

Konflikte verstanden werden, welche sich auf der Ebene Softwaresystem befinden. Diese sind bereits in größerem Umfang Bestandteil der Systematik, sodass keine Ergänzungen notwendig sind. Gleiches gilt für den Fall *heterogenes Datenmodell*. Diese Heterogenitäten werden auf der Ebene Informationen. Die *physische Heterogenität* wird in der Systematik durch die Ebene *Plattform* repräsentiert. Dabei umfasst diese Ebene mehr als die bei Volger angedeuteten Konflikte, da das Betriebssystem auch auf dieser Ebene erfasst wird. Teile der physischen Konflikte sind die Einzigen, welche im Rahmen der Konsolidierung ergänzt werden. Ergänzt werden müssen *mehrere Instanzen*, *mehrere Rechner* und *Varianten einer Anwendung*. Unterschiedliche technologische Plattformen ist zum einen durch die bereits bestehenden Konflikte der Art *Operating System* abgedeckt, kann sich zum anderen aber auch auf verwendete Frameworks beziehen. Häufig gibt es im Rahmen des Softwaresystems eine Schichtenarchitektur und im Rahmen eines Integrationsprojektes wird nur die eigentliche Anwendung betrachtet und nicht die verwendeten Frameworks, welche jedoch auch Konflikte verursachen können. Je nachdem, ob verwendete Frameworks der Ebene des Softwaresystems oder der Ebene der Plattform zugeordnet werden, müsste dieser Konflikt dort integriert werden. Es wird die letztere Variante favorisiert und versucht die Ebene Softwaresystem von unverändert genutzten Komponenten frei zu halten. Deshalb wird auch dieser Konflikt auf der Ebene der Plattform ergänzt.

Die Letzten in die Systematik aufzunehmenden Arbeiten sind die von Jung und Conrad, welche bereits einander gegenübergestellt wurden.[256] Da die Klassifikation nach Jung die umfangreichere von beiden ist, werden im Folgenden für die Konfliktbezeichnungen die Bezeichner nach Jung verwendet. Die *Heterogenität der Datenmodelle* ist in der Systematik auf der Ebene *Softwaresystem* mit dem Subtyp *Datenmodelle* bereits enthalten. Für den Konflikt *strukturelle Konflikte* – welcher sich auf die Darstellung des identischen Sachverhaltes mit unterschiedlichen strukturellen Mitteln bezieht – sind bereits durch die Typen *bilaterale Korrespondenzen* und *multilaterale Korrespondenzen* repräsentiert. *Wertebereichskonflikte*, finden ihre Abbildung in Form von *Datentyp*, ebenso findet sich der *Skalierungskonflikt* direkt in Form der *Skalierungs- und Einheitenkonflikte* wieder. Ein *Genauigkeitskonflikt* ist bisher noch nicht enthalten, weshalb dieser ergänzt werden muss. Gleiches gilt für den *Vagheitskonflikt*. Die *Integritätskonflikte* sind jedoch bereits in verschiedenen, detaillierteren Ausprägungen enthalten. Sämtliche *Extensionalen Konflikte* sind bisher nicht in der Systematik enthalten und werden hinzugefügt. Ebenso wird mit den Konflikten vom Typ *Datenwertkonflikt* verfahren. An dieser Stelle muss ausdrücklich darauf hingewiesen werden, dass in der Systematik der Konflikte zum jetzigen Zeitpunkt keine wechselseitigen Abhängigkeiten der Konflikte berücksichtigt sind. Ob ein Konflikt einen Anderen zwingend bedingt, zwei Konflikte immer gemeinsam auftreten und so von der Existenz des Einen auf die Existenz des Anderen geschlossen werden kann, verbleibt Gegenstand weitergehender Forschungsbemühungen und wird im Rahmen des Ausblicks in Kapitel 8 diskutiert. Eine Untersuchung dieser Zusammenhänge erscheint aber insofern lohnenswert, als das von Konfliktabhängigkeiten im Problemraum auf Musterabhängigkeiten im Lösungsraum geschlossen werden könnte.

Ebene	**Art**	**Typ**	**Konflikt**
Prozessmodell			
	semantisch	Datenheterogenitäten	Skalierungs- und Einheitenkonflikte Repräsentationskonflikte surjektive Abbildungskonflikte
		Domänenheterogenitäten	Subsumptionskonflikte Überlappungskonflikte Inkompatibilitäten Aggregationskonflikte
Konsolidierung der Heterogenitätsarten – Fortsetzung auf der nächsten Seite			

[256] vlg. Tabelle 5.30

Ebene	Art	Typ	Konflikt
	Struktur	bilaterale Korrespondenzen	Synonyme Homonyme Datentyp Genauigkeit Vagheit
			Defaultwert Schlüssel Integritätsconstraint
		Extensionale Konflikte	Extensionale Äquivalenz Einschluss/Teilmenge Überlappung Disjunktheit
		multilaterale Korrespondenzen	multilaterale Attributkorrespondenzen multilaterale Entitätskorrespondenzen fehlende Informationen
		Metaebenen Diskrepanzen	Daten-Attribute-Korrespondenzen Daten-Entitäten-Korrespondenzen fehlende Informationen
	Datenwerte		Attributwertkonflikte starke Attributäquivalenz schwache Attributäquivalenz disjunkte Attributäquivalenz
		Inkonsistenz- & Redundanzprobleme	Datenungenauigkeitskonflikte temporale Inkonsistenzkonflikte Redundanzprobleme fehlende Daten
Softwaresystem	Verhalten	constraint violations	pre conditions post conditions
		protocol violations	name conflicts Kompatibilität Austauschbarkeit
		property violations	deadlock
	Schnittstellen		mismatched interface types
	Funktionen	Trivialer Fall Einfacher Fall	1:1 Abbildung 1:1 Parameterabbildung 1:n Parameterabbildung n:1 Parameterabbildung
		Unabhängiger Fall	
		Abhängiger Fall	lineare Abhängigkeit 1:n Abhängigkeit n:1 Abhängigkeit
		Allgemeiner Fall	Iterative Abhängigkeit Kombination aller (Funtions-) Konflikte möglich
Softwaresystem	System		Datenmodelle concurrency control recovery
		property conflicts	communication mechanism technological characteristics
			Unspecified Data Destination Unspecified Data Location Multiple unsequenced Data Transfers Mismatched Data Transfer Assumptions Restricted Points of Data Transfer Uninitialized Data Transfer Restricted Points of Control Transfer Unspecified Control Destination Inhibited Rendezvous

Konsolidierung der Heterogenitätsarten – Fortsetzung auf der nächsten Seite

Ebene	Art	Typ	Konflikt
			Multiple Unsequenced Control Transfers Uninitialized Control Transfer
Plattform	Instanz		mehrere Instanzen Varianten einer Anwendung
	Frameworks		Unterschiedliche Technologische Plattformen
	Betriebssystem		File System Naming File Type Operation transaction support IPC
	Hardware		mehrere Rechner instruction set data representation/coding

Tabelle 5.32: Konsolidierung der Heterogenitätsarten

Die erste Ebene, die des Prozessmodells, bleibt in der Systematik der Konflikte mit Absicht leer. Auf der einen Seite wird diese Ebene für so wichtig erachtet, dass Sie analog zu der Arbeit von Vogler erwähnt werden muss, da auch dort Konflikte existieren. Zum anderen sind in der analysierten Literatur keine expliziten Konflikte enthalten, sodass an dieser Stelle auf den Ausblick in Kapitel 8 verwiesen werden muss, da eine systematische Erhebung dieser Konflikte den Umfang der Arbeit übersteigen würde.

Klassifikation der Integrationsmuster nach der Art der überbrückten Heterogenität: Aus der Literatur wurden verschiedene Heterogenitätskonflikte extrahiert und in einer gemeinsamen Systematik zusammengeführt (Tabelle 5.32). Um die Integrationsmuster entsprechend der Heterogenitätskonflikte zu klassieren, ist es notwendig zu untersuchen, inwieweit diese die Lösung eines Heterogenitätskonfliktes ermöglichen. Die wissenschaftliche Diskussion der Muster liefert nur wenige Hinweise darauf, welche Muster explizit bestimmten Konflikten zugeordnet werden. Eine der wenigen Arbeiten, die explizit auf diese Zuordnung eingeht, ist die von Hepner u.a.[257] Es findet eine Zuordnung auf zwei Ebenen statt. Zu den bei Hepner beschriebenen Konflikten wird zum einen gesagt, welche Mustertypen und Typkombinationen – im Sinne der Taxonomie aus Controller, Extender und Transformer – die beschriebenen Konflikte lösen. Zum anderen werden auch explizit Integrationsmuster inclusive Quelle benannt, die den beschriebenen Konflikt adressieren. In Tabelle 5.33 werden die den Konflikten zugeordneten Muster und Mustertypen aufgelistet. Die dargestellten Zuordnungen ermöglichen es, über die Mustertypen nach weiteren

Konflikt	Mustertyp	Muster und Quelle
restricted control transfer	Controller + Extender	n.a.
unspecified control destination	Extender	Proxy (Buschmann; Meunier u. a., 1996)
uninitialized control transfer	Controller (+ Extender)	Facade (Gamma u. a., 1995), Integration Facade (Lutz, 2000)
inhibited rendezvous	Extender	Wrapper (Mularz, 1995)
invalid data	Translator	Adapter (Gamma u. a., 1995)
inconsistent data	Extender + Controller (+Translator)	n.a.

Tabelle 5.33: Übersicht der Zuordnung Konflikte und lösende Muster nach Hepner
Quelle: (Hepner u. a., 2006)

möglichen Mustern zu suchen, welche dem gleichen Typ zugeordnet werden. Über diesen Weg

[257] Vgl. (Hepner u. a., 2006).

können potenzielle weitere Lösungen über die benannten hinaus gefunden werden. Für den letzten benannten Konflikt *inconsistent data* finden sich bei Davis und Gamble weitere Muster, die als Lösung angesehen werden. Dabei wird auch dort eine Kombination aus Extender, Controller und gegebenen Falls einem Translator als Lösung angesehen. Der Broker[258], der Workflow Manager[259] und Sahred Repository[260] werden als Lösungen des Konfliktes beschrieben, wobei Shared Repository als beste der drei Lösungen bezeichnet wird[261]. Die vorgenannten Ausführungen stellen die Gesamtheit der direkten Zuordnungen von Konflikten zu Integrationsmustern dar. Es finden sich nur für einen Bruchteil der in Tabelle 5.32 enthaltenen Konflikte direkt zugeordnete Integrationsmuster. Die nachfolgende Tabelle 5.34 zeigt die Klassierung der Integrationsmuster auf dieser Basis. Wie sich zeigt, ergibt sich eine sehr lückenhafte Zuordnung.

Pattern	**Quelle**	**Focus**
(Integration) Adapter	(Lutz, 2000),(Andersson und Johnson, 2001)	invalid data
Integration Messenger	(Lutz, 2000)	inconsistent data
Integration Mediator	(Lutz, 2000)	n.a.
Mediator	(Andersson und Johnson, 2001)	n.a.
Broker	(Trowbridge, 2003)	inconsistent data
Message Router	(Andersson und Johnson, 2001)	n.a.
Gateway	(Andersson und Johnson, 2001)	n.a.
Service Gateway	(Trowbridge, 2003)	n.a.
Integration Facade	(Lutz, 2000)	uninitialized control transfer
Service Interface	(Trowbridge, 2003)	uninitialized control transfer
Process Automator	(Lutz, 2000)	inconsistent data
Database Federation	(Andersson und Johnson, 2001)	n.a.
Desktop Integration	(Andersson und Johnson, 2001)	n.a.
Observer	(Trowbridge, 2003)	n.a.
Data Transfer Object	(Trowbridge, 2003)	n.a.
Singleton	(Trowbridge, 2003)	n.a.

Tabelle 5.34: Testklassifikation entsprechend der Facette der Heterogenität

Darüber hinaus wurden bereits Annahmen getroffen, dass die Zuordnung von Konflikten aus Tabelle 5.33 auf ähnliche Muster übertragbar ist. Dem Broker wird nach Davis und Gamble die Lösung eines *inconsistent data* Konflikt zugeordnet. Da der Integration Messenger nach Lutz eine höhere Abstraktion aufweist und für jedes Interaktionsmodell eine eigene Spezialisierung besitzt – welche im Fall des Brokers eine Spezialisierung auf Remote-Method-Invocation (RMI) bedeutet[262] – wurde diesem ebenfalls die Lösung des *inconsistent data* Konflikt zugeschrieben. Die anderen Spezialisierungen des Musters können weitere Konflikte lösen, weshalb die Zuordnung unvollständig ist.Dies führt dazu, dass disjunkte Fokusse einer Facette nicht gewährleistet werden können.

Durch die Granularität der Musterbeschreibungen gestaltet es sich schwierig – allein auf Grundlage der Beschreibung – eine direkte Zuordnung zu weiteren Konflikten zu treffen, was vor allem dann notwendig ist, wenn keine in der Literatur nachweisbaren Zuordnungen existieren. Nachfolgend wird dies exemplarisch an einem Muster illustriert. Es wird dafür der Message Translator[263] gewählt, da man aufgrund der intendierten Funktion – der Übersetzung von einem Nachrichtenformat in ein anderes – grundsätzlich damit rechnen kann, dass dieses Muster der Lösung von Heterogenitätskonflikten dient. Weiteren Aufschluss hierzu gibt die Arbeit von Scheibler[264], welcher sich mit ausführbaren Integrationsmustern auseinandersetzt und zu diesem

[258] Vgl. (Buschmann; Meunier u. a., 1996; Mularz, 1995).
[259] Vgl. (Mularz, 1995).
[260] Vgl. (ebenda).
[261] Vgl. (Davis und Gamble, 2000).
[262] Vgl. (Lutz, 2000, S. 68).
[263] Vgl. (Buschmann; Henney und Schmidt, 2007a, S. 229ff.).
[264] Vgl. (Scheibler, 2010).

Zweck Parameter für die Konfiguration bestehender Muster identifiziert. Zu diesem Zweck wurde auch die Parametrisierung der Muster genauer untersucht und spezifiziert. Auf den ersten Blick kommen für dieses Muster alle Konflikte der Ebene *Information* infrage. Als Parameter des Message Translator sind bei Scheibler die Struktur der Eingangsnachricht, die Struktur der Ausgangsnachricht und die Transformationslogik angegeben.[265] Damit umfasst der Parameter der Transformationslogik die konfliktlösenden Informationen. Eine weitere Detaillierung des Parameters ist nicht zu finden, sodass sich die anfängliche Vermutung bestätigt. Ein Message Translator ist nicht auf der gleichen Granularitätsstufe angesiedelt wie die einzelnen Konflikte der Systematik. Dies zeigt, dass eine detaillierte Zuordnung zu den einzelnen Konflikten auf dem – in der Literatur vorhandenen – Detaillierungsgrad von Mustern nicht möglich ist. Abhilfe könnten detailliertere Muster schaffen, welche anhand der Konflikte zu einer konkreten Ausprägung eines Message Translator konfiguriert werden.

Auf der Grundlage des Message Translator Beispiels lässt sich zusammenfassen, dass eine umfassende Klassierung anhand der vorliegenden Konfliktsystematik fehlgeschlagen ist. Diese Wirkung liegt begründet in der Granularität der Muster. Auf der einen Seite scheint es –wenn man die Existenz der ausgewerteten Literatur als Nachweis dieser Behauptung annimmt – von Interesse zu sein, Konflikte zwischen Integrationsobjekten detailliert zu untersuchen.[266] Auf der anderen Seite sind die Lösungsmuster –hinsichtlich dieser Facette – auf einem zu hohen Abstraktionsniveau beschrieben. Es ist nur möglich einem einzelnen Muster eine Menge von Konflikten, einen Typ zuzuordnen, nicht jedoch einen einzelnen Konflikt. Damit ist es aktuell nicht möglich, Muster anhand der detaillierten Konflikte auszuwählen und eine Lösung in diesem Sinne planvoll zu gestalten.

Da weder die Detaillierung der Muster, noch eine umfassende Untersuchung hinsichtlich der Lösung einzelner Konflikte Gegenstand dieser Arbeit sind, wird dazu übergegangen, die Muster auf der Typebene der Konflikte zu klassifizieren.[267] Für die Auswahl der Muster im Rahmen eines Lösungsprozesses kann so über den vorliegenden Konflikttyp die Menge der infrage kommenden Muster eingeschränkt werden. Zu diesem Zweck werden drei Konflikttypen eingeführt und diesen die zugehörigen Einzelkonflikte zugeordnet. Dabei werden bewusst die Ebenen, Arten und Typen, welche in Tabelle 5.32 enthalten sind, vernachlässigt. Damit gehen zwar einige Informationen verloren, es wird jedoch die Möglichkeit gewonnen, gleichartige Konflikte die auf unterschiedlichen Ebenen auftreten zu identifizieren. Dieses Vorgehen unterstützt die grundlegende These, dass die Konflikte bspw. im Bereich der Informationen auf unterschiedlichen Ebenen Analogien aufweisen und bisher die dadurch mögliche Übertragbarkeit von Lösungskonzepten nur in unzureichendem Maße genutzt wurde. Den Eigenschaften eines Integrationsmusters folgend, werden in Tabelle 5.35 vier Typen verwendet. Konflikte vom Typ Informationen beziehen sich auf alle Konflikte, die mit der strukturellen oder semantischen Repräsentation der Informationen zusammenhängen. Diesem Ansatz folgend, fallen auch strukturelle Konflikte von Schnittstellen in diesen Typ. Als Beispiel können unterschiedliche Bezeichnungen von Parametern einer Methode angeführt werden, die bei der Übergabe entsprechend zu transformieren sind. Die zweite Kategorie sind Konflikte vom Typ Verhalten. Unterschiedliche Vor- und Nachbedingungen fallen – exemplarisch angeführt – in diesen Typ. Der dritte Typ ist Transport, welcher alle Konflikte des Informationstransportes umfasst. Dazu zählen beispielhaft fehlende Zugriffspunkte sowie Unkenntnis der Kommunikationspartner.

[265] Vgl. (ebenda, S. 93).
[266] Vgl. Tabelle 5.32.
[267] Dieser Punkt wird im Ausblick ein wenig vertieft. Vgl. Kapitel 8.

Typ	**Konflikt**
Informationen	Skalierungs- und Einheitenkonflikte Repräsentationskonflikte surjektive Abbildungskonflikte Subsumptionskonflikte Überlappungskonflikte Inkompatibilitäten Aggregationskonflikte Synonyme Homonyme Datentyp Genauigkeit Vagheit Defaultwert Schlüssel Integritätsconstraint Extensionale Äquivalenz Einschluss/Teilmenge Überlappung Disjunktheit multilaterale Attributkorrespondenzen multilaterale Entitätskorrespondenzen fehlende Informationen Daten-Attribute-Korrespondenzen Daten-Entitäten-Korrespondenzen fehlende Informationen Attributwertkonflikte starke Attributäquivalenz schwache Attributäquivalenz disjunkte Attributäquivalenz Datenungenauigkeitskonflikte temporale Inkonsistenzkonflikte Redundanzprobleme fehlende Daten Naming File Type name conflicts mismatched interface types 1:1 Abbildung 1:1 Parameterabbildung 1:n Parameterabbildung n:1 Parameterabbildung instruction set data representation/coding Varianten einer Anwendung Datenmodelle File System Kompatibilität
Verhalten	pre conditions post conditions deadlock recovery Operation
Transport	Unabhängiger Fall lineare Abhängigkeit 1:n Abhängigkeit n:1 Abhängigkeit Iterative Abhängigkeit concurrency control Unspecified Data Destination Unspecified Data Location Multiple unsequenced Data Transfers Mismatched Data Transfer Assumptions Restricted Points of Data Transfer Uninitialized Data Transfer Restricted Points of Control Transfer

Typisierung der Interoperabilitätskonflikte – Fortsetzung auf der nächsten Seite

Typ	Konflikt
kombinierte Konflikte	Unspecified Control Destination Inhibited Rendezvous Multiple Unsequenced Control Transfers Uninitialized Control Transfer mehrere Rechner mehrere Instanzen transaction support communication mechanism IPC Austauschbarkeit Kombination aller (Funtions-) Konflikte möglich technological characteristics Unterschiedliche Technologische Plattformen

Tabelle 5.35: Typisierung der Interoperabilitätskonflikte

Der letzte Typ kombinierte Konflikte nimmt eine Sonderstellung ein. Diese Kategorie entsteht dadurch, dass die Quellen die Konflikte nicht immer auf dem gleichen Granularitätsniveau beschreiben. Die Austauschbarkeit von Komponenten hängt nach Leicher zum einen davon ab, ob die Schnittstellen der Struktur nach gleich sind und das Verhalten der Komponenten identisch. Streng genommen handelt es sich dabei um die Kombination zweier Konflikttypen, nämlich Information und Verhalten. Um alle Konflikte vollständig aufzulisten, wurde diese Kategorie eingeführt.

Im Anschluss an die Bildung von Konflikttypen können nun die bekannten Integrationsmuster klassifiziert werden, wozu die Konflikttypen als Fokusse übernommen werden. Daraus ergibt sich die folgende Darstellung in Tabelle 5.36.

Pattern	Quelle	Fokus
Adapter Style	(Andersson und Johnson, 2001, S. 229)	Information
Adapter static	(ebenda, S. 229)	Information
Adapter dynamic	(ebenda, S. 229)	Information
Adapter thin	(ebenda, S. 229)	Information
Adapter thick	(ebenda, S. 229)	kombinierte Konflikte
Adapter centralized	(ebenda, S. 229)	Information
Adapter distributed	(ebenda, S. 229)	Information
Adapter	(Khomh und Guéhéneuc, 2008a)	
(Integration) Adapter	(Lutz, 2000)	Information
Component Wrapper	(Goedicke und Zdun, 2002, S. 7ff.)	Information
Wrapper Facade	(Schmidt; Stal u. a., 2000, S. 47ff.)	
(Legacy) Wrapper	(Mularz, 1995)	Information
Facade	(Khomh und Guéhéneuc, 2008a)	
Integration Facade	(Lutz, 2000)	kombinierte Konflikte (Transformation)
Service Interface	(Trowbridge, 2003)	kombinierte Konflikte
Integration Messenger (a, broker)	(Lutz, 2000)	Transport
Integration Messenger (b, message queuing)	(ebenda)	Transport
Integration Messenger (c, publish subscribe)	(ebenda)	Transport
Proxy	(Buschmann; Meunier u. a., 1996, S. 263ff.)	Transport
Proxy	(Khomh und Guéhéneuc, 2008a)	
Mediator	(Gamma u. a., 1995, S. 277)	Transport
Integration Mediator	(Lutz, 2000, S. 71)	kombinierte Konflikte (Transport und Transformation)
Mediator	(Buschmann; Henney und Schmidt, 2007a, S. 411)	Transport
Mediator	(Khomh und Guéhéneuc, 2008a)	
Mediator Style	(Andersson und Johnson, 2001, S. 230)	kombinierte Konflikte
Message Router	(ebenda, S. 231)	Transport

Klassifikation entsprechend der Facette der Heterogenität – Fortsetzung auf der nächsten Seite

Pattern	Quelle	Fokus
Message Router	(Hohpe und Woolf, 2003, S. 81)	Transport
Message Router	(Buschmann; Henney und Schmidt, 2007a, S. 232)	Transport
Gateway	(Andersson und Johnson, 2001)	Information (kombinierte Konflikte thick)
Service Gateway	(Trowbridge, 2003, S. 293ff.)	kombinierte Konflikte (mit Transformation)
Broker	(ebenda, S. 207f.)	Transport
Direct Broker	(Trowbridge u. a., 2004, S. 155,215ff.)	kombinierte Konflikte (Transport)
Indirect Broker	(ebenda, S. 221)	kombinierte Konflikte (Transport)
Message Broker	(ebenda, S. 240f.)	kombinierte Konflikte (Transport)
Message Broker	(Hohpe und Woolf, 2003, S. 322ff.)	Transport
Broker	(Buschmann; Meunier u. a., 1996, S. 119ff.)	Transport
Broker Revisited	(Kircher; Voelter u. a., 2004, S. 581)	Transport
Broker	(Harrison und Avgeriou, 2007, S. 267f.), bezieht sich auf Buschmann	
Pipes and Filter	(Hohpe und Woolf, 2003, S. 322ff.) , (ebenda, S. 70ff.)	Transformation
Pipes and Filter	(Buschmann; Meunier u. a., 1996, S. 67ff.)	Transformation
Pipes and Filter	(Trowbridge u. a., 2004, S. 292f.)	Transformation
Pipes and Filters	(Garlan und Shaw, 1993, S. 7f.)	Transformation
Pipes and Filters	(Meunier, 1995, S. 437)	Transformation
Pipes and Filters	(Harrison und Avgeriou, 2007, S. 267f.) Bezieht sich auf Buschmann	
Observer	(Khomh und Guéhéneuc, 2008a)	
Observer	(Trowbridge, 2003, S. 129f.)	Transport
Publish and Subscribe	(Trowbridge u. a., 2004, S. 280f.)	Transport
Process Automator	(Lutz, 2000)	Verhalten
Database Federation	(Andersson und Johnson, 2001)	Information
Desktop Integration	(ebenda)	kombinierte Konflikte
Data Transfer Object	(Trowbridge, 2003, S. 229)	Transport
Singleton	(ebenda, S. 257)	Transport
Singleton	(Khomh und Guéhéneuc, 2008a)	
File Gateway	(Erl, 2009, S. 457)	kombinierte Konflikte
Process Centralizazion	(ebenda, S. 193)	Verhalten
Protocol Bridging	(ebenda, S. 687)	Information
Legacy Wrapper	(ebenda, S. 441)	kombinierte Konflikte
Service Broker (compund Pattern)	(ebenda, S. 707)	Information
Enterprise Service Bus (compund Pattern)	(ebenda, S. 704)	kombinierte Konflikte
Proxy Capability	(ebenda, S. 497)	Transport
Service Facade	(ebenda, S. 333)	kombinierte Konflikte
UI Mediator	(ebenda, S. 366)	Information
Data Model Transformation	(ebenda, S. 671)	Information
Data Format Transformation	(ebenda, S. 681)	Information
Asynchronous Queuing	(ebenda, S. 582)	Transport
Intermediate Routing	(ebenda, S. 549)	Transport

Tabelle 5.36: Klassifikation entsprechend der Facette der Heterogenität

Der Integration Messenger lässt sich eindeutig der Kategorie *Transport* zuordnen. Das Ziel dieses Musters ist die Minimierung der Kommunikationsabhängigkeiten zwischen den Integrationsobjekten. Dabei werden verschiedene Kommunikationsmodelle (1:1 synchron, 1:1 asynchron 1:N asynchron) unterstützt. Die Integrationskomponente ist für die Zustellung der Nachrichten verantwortlich, die Interaktionslogik verbleibt bei den Integrationsobjekten.[268]

[268] Vgl. (Lutz, 2000, S. 67f.).

Das zweite Muster aus der gleichen Quelle, der Integration Mediator verfolgt ähnliche Ziele. Ebenso wie der Integration Messenger dient der Integration Mediator dazu, die Kommunikationsabhängigkeiten zu minimieren. Der Unterschied besteht in der nun zentralisierten Interaktionslogik[269], welche dazu führt, dass der Integration Mediator im Gegensatz zu dem Integration Messenger integrationsobjektspezifisch ist. Damit ergibt sich ebenfalls die eindeutige Zuordnung zu dem Typ *Transport*.

Obwohl man eigentlich annehmen könnte, dass der Mediator identisch mit dem Integration Mediator ist, kann dieser nicht eindeutig in die Kategorie Transport eingeordnet werden. Das zentrale Konzept ist eine Integrationskomponente, welche bestimmte Aufgaben an einer Stelle zusammenfasst (Funktionszentralisierung). Neben verschiedenen Submustern (middle-tier, peer-to-peer, hub) – welche die Möglichkeiten von weiteren Kommunikationsverbindungen parallel zu dem Mediator einschränken und die Charakteristik der Integrationsobjekte im Sinne von Client und Server betreffen – werden auch verschiedene Funktionen benannt, die der zentralen Integrationskomponente zugeordnet werden können. Da hier neben Lastverteilung und Routing auch Transformation genannt wird, kann das Muster auch Konflikte des Typs Information lösen. Deshalb handelt es sich um ein gemischtes Muster der Typen Information und Transport, weshalb die Einordnung in *kombinierte Konflikte* erfolgt.

Der Broker wird dem Typ *Transport* zugeordnet, da er eine Lösung zur Kapselung und Isolierung der Kommunikationsmechanismen in verteilten Anwendungen beschreibt. Dabei bleiben die Mechanismen zur Überbrückung der physischen und logischen Verteilung vor den Integrationsobjekten verborgen, die Interaktionslogik – wann und welche Interaktion stattfindet – verbleibt jedoch bei den Integrationsobjekten.[270]

Obwohl mit dem Integration Mediator das grundlegende Konzept bereits in Form eines Musters beschrieben wurde, existiert mit dem Message Router ein weiteres Muster, welches auf dem gleichen Konzept basiert. Auch bei diesem Muster stehen die Isolation der Kommunikationsmechanismen und der Einsatz einer zentralen Komponente im Vordergrund.

Ein Gateway dient der direkten Abbildung zweier Schnittstellen [271] aufeinander. Es werden verschiedene Untervarianten dargelegt, die sich im Zeitpunkt der Bindung der Schnittstellen (static, dynamic) unterscheiden. Weiterhin wird zwischen einem *thin* Gateway, welcher nur die Bindung der Schnittstellen organisiert, und einem *thick* Gateway unterschieden. Die letztere Variante ist neben der reinen Bindung der Schnittstellen und deren Abbildung aufeinander in der Lage Transformationen durchzuführen. Damit handelt es sich bei einem Gateway in erster Linie um ein Muster vom Typ *Information*.

Der Service Gateway beschreibt auf den ersten Blick lediglich ein altes Konzept erweitert um die Paradigmen der Serviceorientierung. Die Kontraktbasis der Serviceorientierung führt dazu, dass der Service Gateway wesentlich mehr Funktionalität wie Kommunikationskanal, Datenformate, Service-Discovery, Process-Adapter und Calling-Semantics umfasst. Deshalb lässt er sich nur der Kategorie *kombinierte Konflikte* zuordnen, weil neben Transportkonflikten auch jene vom Typ Information gelöst werden können. Der einzige Unterschied, der nun noch zu Konzepten wie einem Mediator zu bestehen scheint, ist die Tatsache, dass von „The Service Gateway component encapsulates the low-level details of communicating with a service“[272] gesprochen wird und damit genau ein Service adressiert wird.

[269] Vgl. (ebenda, S. 71).
[270] Vgl. (Trowbridge, 2003, S. 199ff.).
[271] Vgl. (Andersson und Johnson, 2001, S. 228).
[272] (Trowbridge, 2003, S. 294)

Integration Facade ist ein Muster, welches Konflikte vom Typ *Information* löst. Es findet eine Abbildung zwischen der komplexen Schnittstelle einer Serverapplikation und einer vereinfachten, durch die Integration Facade zu Verfügung gestellten, Schnittstelle statt. Als Interaktionsmodell wird dabei nur Client-Server unterstützt, wobei es mehr als eine Client-Applikation geben kann. Dieser Charakterisierung zur Folge kann das Muster der Gruppe *Information* zugeordnet werden, da die Hauptleistung in der geeigneten, einfacheren Schnittstelle besteht.

Das Service Interface weist in seiner Beschreibung große Parallelität zu dem Service Gateway auf. Das Ziel des Musters ist es, die Kommunikationsbestandteile von der eigentlichen Geschäftslogik des Dienstes zu trennen. Zu diesem Zweck wird eine Serviceschnittstelle eingesetzt, welche die folgenden Details kapselt: Netzwerkprotokoll, Datenformate inclusive Transformation, Sicherheit, Dienstgütevereinbarungen. Da neben den reinen Kommunikationseigenschaften z. B. auch Transformationen enthalten sein können, handelt es sich nicht um ein reines Muster des Typs *Transport*, sondern um ein Gemischtes, welches auch Konflikte des Typs *Information* lösen kann. Daraus folgt der Fokus *kombinierte Konflikte*. Die Gemeinsamkeiten zu dem Muster Service Gateway entstehen dadurch, dass beide Muster einen Zugriffspunkt erschaffen. Mit dem Unterschied, dass ein Service Interface direkt auf die Geschäftslogik zugreift und ein Service Gateway möglicherweise nur auf eine Schnittstelle und nicht auf die Geschäftslogik direkt. Die beiden Muster stellen ein Beispiel für die im Ausblick (Kapitel 8) diskutierten Variationspunkte dar.

Der Process Automator ist ein Muster, welches in den Fokus *Verhalten* fällt. Die Hauptfunktion ist die Sequenzbildung und die Kontrolle von Aktivitäten[273], wodurch die Zuordnung des Focus begründet wird, da die Bildung von Sequenzen als die Sicherstellung des gewünschten Verhaltens gesehen wird.

Das Ziel der Database Federation ist es, heterogene Datenbanken zu integrieren und eine einheitliche Sicht auf die Datenbestände zu liefern. Aus diesem Grund handelt es sich um ein Muster des Fokus *Information*. Diese Einordnung vernachlässigt bewusst, die zusätzlich notwendige Lösung der Transportproblematik. Lutz spricht von einer Mediator-Lösung, welche die Kommunikation zu den einzelnen Datenbanken übernimmt und nach außen die einheitliche Sicht zur Verfügung stellt.

Im Wesentlichen löst das Muster Desktop Integration ein Transportproblem, in dem es einen Zugriffspunkt für eine Menge an Informationen zur Verfügung stellt. Dies trifft vor allem auf die schlanke Ausführung zu, welche keine neue Funktionalität zur Verfügung stellt.[274] Die etwas umfangreichere Ausführung fügt eine zusätzliche aktive gatewayartige Teilkomponente hinzu, welche Funktions- und Datensynchronisation zwischen den Komponenten verfügbar macht, auch vollständig neue Funktionalität ist möglich, von deren Art die weitere Klassierung abhängig wäre.[275] Daraus folgt die Einordnung in *Transport*.

Der Observer löst einen Konflikt vom Typ *Transport*. Die Besonderheit des Transportproblems besteht darin, dass die Bindung zwischen den kommunizierenden Integrationsobjekten lose sein soll. Das in Tabelle 5.36 referenzierte Muster stammt aus der Kategorie der Entwurfsmuster und zeigt eine Umsetzung in diesem Bereich. Wesentliche Elemente sind zwei Superklassen, durch welche mithilfe der Vererbung sichergestellt wird, dass ein standardisierter Aufbau umgesetzt wird. Dies ermöglicht Austauschbarkeit. Die dynamische Registrierung wird durch eine Interessentenliste[276] gelöst, über die bei Statusänderung iteriert wird. Diese Eigenschaften begründen die Einordnung in den Fokus *Transport*.

[273] Vgl. (Lutz, 2000, S. 73).
[274] Engl. thin desktop integration. Vgl. (Andersson und Johnson, 2001, S. 232).
[275] engl. thick desktop integration Vgl. (ebenda, S. 232).
[276] Liste der Beobachter

Das Muster Data Transfer Object löst eine Problematik, welche durch Latenzzeiten und Durchsatz bei entfernten Prozeduraufrufen entsteht(RPC[277]). Die Praxis möglichst feingranulare und atomare Funktionalität zur Verfügung zu stellen, steht im Kontrast dazu, dass für einen Schritt – wie die Übermittlung von Kundendaten – mehrere Funktionsaufrufe notwendig sind. Der Transport über ein Netzwerk führt dazu, dass Latenzzeiten und Durchsatzprobleme für jeden Aufruf entstehen und sich summieren. Ein Data Transfer Object übermittelt alle notwendigen Daten in einem Objekt und ermöglicht eine lokale Verarbeitung der Daten. Dieses Muster lässt sich eindeutig der Kategorie *Transport* zuordnen.

Singleton ist ein Muster, das infolge der Konzepte der Objektorientierung entsteht. Um sicherzustellen, dass von einer Klasse nur ein Objekt erzeugt werden kann und man immer genau dieses Objekt anspricht, stellt das Muster eine Lösung zur Verfügung. Es handelt sich um ein Problem der Adressierung, welche Bestandteil des Transportes ist, was dieses Muster zu einem Muster mit dem Fokus *Transport* macht.

In dieser Facette werden kombinierte Konflikte verwendet, was an dieser Stelle zweckmäßig erscheint, jedoch im Ausblick 8 jedoch eingehender diskutiert wird. Im Anschluss wird eine weitere funktionale Facette vorgestellt, die sich mit den Kommunikationseigenschaften eines Musters auseinandersetzt.

5.4.7 Facetten der Kommunikation und Entkopplung

In der vorliegenden Arbeit wird die Auffassung vertreten, dass die Lösung eines Integrationsproblems sowohl in der Überbrückung von Heterogenität als auch in der Lösung von Kommunikationsproblemen oder einer Kombination von beidem besteht. Eine umfangreiche Systematisierung von Kommunikationseigenschaften ist bei Wagner[278] zu finden. Die nachfolgende Aufzählung listet diese Typisierung auf und stellt sie bereits in der benötigten Form von Facetten und zugehörigen Fokussen auf.

1. Zweck der Kommunikation
   - Informationen (I)
   - Dienste (D)
   - Inanspruchnahme von Ressourcen (R)
   - Reservierung von Ressourcen (RR)
2. Synchronität
   - synchron (sync)
   - asynchron (async)
3. Verbindungsorientierung
   - verbindungsorientiert (vo)
   - verbindungslos (vl)
4. Kardinalität der Kommunikation
   - 1:1
   - m:1

[277] Engl. remote procedure call
[278] Vgl. (Wagner, 2006).

- 1:m
- m:n

5. Interaktionsmechanismus
   - Dateien (D)
   - Nachrichten (N)
   - Prozeduraufrufe (PA)
   - verteilte Objekte (vO)

Mit den Ausnahmen der Synchronität, Verbindungsorientierung und Kardinalität der Kommunikation finden sich diese Facetten identisch bei Wagner.[279] Die Ausnahmen sind aus dem dort beschriebenen Kommunikationsmodell entstanden, welches je nach Ausprägung eine Kombination der Fokusse der Facetten enthält. Hier war jedoch eine Aufspaltung in einzelne Facetten notwendig.

Weitere Facetten für diesen Bereich lassen sich bei Tichy[280] entnehmen. Die Arbeit wurde bereits in Kapitel 5.3.2 vorgestellt und auf Ihre Eignung hin geprüft wurde. Von besonderer Bedeutung sind die Facetten *Decoupling*, *Control* und *Distribution*. Alle drei adressieren relevante Eigenschaften für die Kommunikation.

Decoupling beschreibt die Aufteilung eines Softwaresystems in voneinander weitestgehend unabhängige Teile, welche getrennt voneinander erstellt, verändert und ausgetauscht werden können. Diese Eigenschaften treffen auf die meisten Integrationsobjekte zu, da diese zwar zur Abbildung bestimmter Arbeitsabläufe miteinander interagieren müssen (Integrationsnotwendigkeit), aber auch von diesen Schritten losgelöste Aufgaben haben, welche eine vom Integrationszustand unabhängige Existenz notwendig machen. Tichy ordnet dieser Facette lediglich Muster zu ohne detaillierte Fokusse zu beschreiben.[281] Aus diesem Grund wären lediglich die Fokusse supported und unsupported verwendbar. Der Grad der erreichbaren Entkopplung bliebe so unbestimmt und muss deshalb später weiter detailliert werden. Muster der Kategorie Control befassen sich nach Tichy mit der Kontrolle der Ausführung und Selektion der richtigen Methoden zur richtigen Zeit. Die hier beschriebenen Muster finden sich alle im Bereich der Muster wieder, die zum Typ Verhalten gehören.[282] Die Facette Interaktionslogik beschreibt einen wesentlichen Bestandteil einer Integrationslösung und bezieht sich auf die Fragen wann und in welcher Reihenfolge interagieren die Integrationsobjekte. Von Interesse für die Facette ist es, ob diese Logik zentralisiert *(z)* oder verteilt *(v)* ist. Nimmt man als Beispiel den Integration Messenger, so handelt es sich bei diesem Muster um eine verteilte Interaktionslogik, da diese weiterhin in den Integrationsobjekten implementiert wird. Der Integration Mediator hingegen ist ein Muster, welches eine zentralisierte Interaktionslogik umsetzt. In diesem Fall wird die Interaktionslogik zentral im Mediator implementiert. Weitere Einstufungen von Integrationsmustern hinsichtlich dieser Facette finden sich in Tabelle 5.37 mit Ausnahme der Synchronität diese fließt in Tabelle 5.38 im Rahmen der Facetten der Entkopplung ein.

[279] Vgl. (Wagner, 2006).
[280] Vgl. (Tichy, 1997).
[281] Vgl. (ebenda, S. 3ff.).
[282] Vgl. die Facette Interoperabilitätskonflikt, Heterogenität in Kapitel 5.4.6.

Pattern	Quelle	Zweck	Verbindungsorientierung	Kardinalität	Interaktionsmechanismus	Interaktionslogik
Adapter Style	(Andersson und Johnson, 2001, S. 229)	-	vo	1:1	PA	d
Adapter static	(ebenda, S. 229)	-	vo	1:1	PA	d
Adapter dynamic	(ebenda, S. 229)	-	vo	1:1	PA	d
Adapter thin	(ebenda, S. 229)	-	vo	1:1	PA	d
Adapter thick	(ebenda, S. 229)	-	vo	1:1	PA	d
Adapter centralized	(ebenda, S. 229)	-	vo	1:1	PA	d
Adapter distributed	(ebenda, S. 229)	-	vo	1:1	PA	d
Adapter	(Khomh und Guéhéneuc, 2008a)					
(Integration) Adapter	(Lutz, 2000)	-	vo	m:1		d
Component Wrapper	(Goedicke und Zdun, 2002, S. 7ff.)	-	vo	1:1	PA	d
Wrapper Facade	(Schmidt; Stal u. a., 2000, S. 47ff.)	-	vo	1:1	PA	d
(Legacy) Wrapper	(Mularz, 1995)	-		1:n		d
Facade	(Khomh und Guéhéneuc, 2008a)					
Integration Facade	(Lutz, 2000)			m:n		z
Service Interface	(Trowbridge, 2003)	-	vl	1:n	N	d
Integration Messenger (a, broker)	(Lutz, 2000)	-	vo	1:1	N	d
Integration Messenger (b, message queuing)	(ebenda)	-	vl	1:1	N	d
Integration Messenger (c, publish subscribe)	(ebenda)	-	vl	1:n	N	d
Proxy	(Buschmann; Meunier u. a., 1996, S. 263ff.)	-	vo	1:n	PA	d
Proxy	(Khomh und Guéhéneuc, 2008a)					
Mediator	(Gamma u. a., 1995, S. 277)	-	vo	1:n	PA	z
Integration Mediator	(Lutz, 2000, S. 71)	-	vl	m:n		z
Mediator	(Buschmann; Henney und Schmidt, 2007a, S. 411)			1:n		z
Mediator	(Khomh und Guéhéneuc, 2008a)					
Mediator Style	(Andersson und Johnson, 2001, S. 230)	-	vl	m:n		z
Message Router	(ebenda, S. 231)		vl	m:n	N	z
Message Router	(Hohpe und Woolf, 2003, S. 81)		vl	1:n	N	z
Message Router	(Buschmann; Henney und Schmidt, 2007a, S. 232)		vl	m:n	N	d
Gateway	(Andersson und Johnson, 2001)	-	vo	1:1	PA	d
Service Gateway	(Trowbridge, 2003, S. 293ff.)	Dienste	vl	m:n	N	d
Broker	(ebenda, S. 207f.)	-	vl	m:n	PA	d
Direct Broker	(Trowbridge u. a., 2004, S. 155,215ff.)	-		m:n		d
Indirect Broker	(ebenda, S. 221)	-		m:n		d
Message Broker	(ebenda, S. 240f.)	-	vl	m:n	N	d
Message Broker	(Hohpe und Woolf, 2003, S. 322ff.)	-	vl	m:n	N	d
Broker	(Buschmann; Meunier u. a., 1996, S. 119ff.)	-		m:n	PA	d
Broker Revisited	(Kircher; Voelter u. a., 2004, S. 581)	-		m:n		d
Broker	(Harrison und Avgeriou, 2007, S. 267f.), bezieht sich auf Buschmann					
Pipes and Filter	(Hohpe und Woolf, 2003, S. 322ff.) , (ebenda, S. 70ff.)	-	vl	1:1	N	d
Pipes and Filter	(Buschmann; Meunier u. a., 1996, S. 67ff.)	-	vl	1:1	N	d
Pipes and Filter	(Trowbridge u. a., 2004, S. 292f.)	-	vl	1:1	N	d
Pipes and Filters	(Garlan und Shaw, 1993, S. 7f.)	-	vl	1:1	N	d
Pipes and Filters	(Meunier, 1995, S. 437)		vl	1:1	N	d
Pipes and Filters	(Harrison und Avgeriou, 2007, S. 267f.) Bezieht sich auf Buschmann					
Observer	(Khomh und Guéhéneuc, 2008a)					
Observer	(Trowbridge, 2003, S. 129f.)	Informationen	vo	1:1	PA	d
Publish and Subscribe	(Trowbridge u. a., 2004, S. 280f.)	-	vl	m:n	N	d

Klassifikation entsprechend der Facetten der Kommunikation – Fortsetzung auf der nächsten Seite

Pattern	Quelle	Zweck	Verbindungsorientierung	Kardinalität	Interaktionsmechanismus	Interaktionslogik
Process Automator	(Lutz, 2000)			m:n		z
Database Federation	(Andersson und Johnson, 2001)	Informationen	vo	1:n	PA	d
Desktop Integration	(ebenda)		vo	1:n	PA	z
Data Transfer Object	(Trowbridge, 2003, S. 229)	Informationen	vo	1:1	PA	d
Singleton	(ebenda, S. 257)	Informationen	vo	1:n	PA	d
Singleton	(Khomh und Guéhéneuc, 2008a)					
File Gateway	(Erl, 2009, S. 457)	Informationen	vl	1:1	N	d
Process Centralizazion	(ebenda, S. 193)	Informationen		m:n	N	z
Protocol Bridging	(ebenda, S. 687)		vl	m:n	N	d
Legacy Wrapper	(ebenda, S. 441)			1:n	N	d
Service Broker (compund Pattern)	(ebenda, S. 707)		vl		N	
Enterprise Service Bus (compund Pattern)	(ebenda, S. 704)		vl	m:n	N	d
Proxy Capability	(ebenda, S. 497)		vl	1:1	N	d
Service Facade	(ebenda, S. 333)		vl	1:n	N	d
UI Mediator	(ebenda, S. 366)		vl	m:n	N	d
Data Model Transformation	(ebenda, S. 671)		vl	1:1	N	d
Data Format Transformation	(ebenda, S. 681)		vl	1:1	N	d
Asynchronous Queuing	(ebenda, S. 582)		vl	1:1	N	d
Intermediate Routing	(ebenda, S. 549)		vl	m:n	N	z

vo: verbindungsorientiert; vl: verbindungslos; D: Dateien; N: Nachrichten; PA: Prozeduraufrufe; vO: verteilte Objekte;z: zentralisiert; d: verteilt (distributed)

Tabelle 5.37: Klassifikation entsprechend der Facetten der Kommunikation

Die Bewertung des Broker nach Trowbridge[283] stellt für die Facetten der Synchronität, Verbindungsorientierung und Kardinalität vornehmlich auf die skizzierten Beispiele der Anwendung ab. Explizite Äußerungen zu diesen Eigenschaften waren der Arbeit nicht zu entnehmen.

Wurde eine Facette mit - bewertet, so kann dies bedeuten, dass diese Facette nicht auf das entsprechende Element zutrifft oder aber, dass diese Eigenschaft in der Quelle nicht nachweisbar ist und sich nicht anderweitig bestimmen lies.

Eine Sonderstellung nehmen die Facetten der Entkopplung im Rahmen der Kommunikation ein, weshalb diese nun gesondert betrachtet werden. Diese Facetten beziehen sich zwar auf Kommunikationseigenschaften sind jedoch eher ursächlich für diese Eigenschaften, denn ein Resultat derselben. Die Bestrebung bestimmte Teile eines Systems weitestgehend unabhängig voneinander zu entwickeln und zu pflegen, führen zu der Notwendigkeit bestimmte Kommunikationseigenschaften umzusetzen. Mit der Synchronität und Verbindungsorientierung wurden bisher verwandte Eigenschaften angesprochen, es existieren jedoch detailliertere Unterscheidungen zu den Eigenschaften der Entkopplung. In der Arbeit von Tichy wird Decoupling – also Entkopplung – als Kategorie für Muster verwendet. Im Abschnitt 5.3.2 wurden die einzelnen Kategorien bereits auf ihre generelle Bedeutung für die Integration untersucht. Die Kategorie Entkopplung wird nach Tichy als „divide a software system into several independent parts in such a way that they

[283] Vgl. (Trowbridge, 2003).

can be built, changed, and replaced independently as well as reused and recombined in unforeseen combinations“[284] beschrieben. Da Integrationsobjekte Träger betrieblicher Aufgaben sind, existieren diese unabhängig von Integrationsbeziehungen. Integrationsbeziehungen sind vielmehr eine Notwendigkeit, welche aus der Arbeitsteilung der betrieblichen Gesamtaufgabe über mehrere Aufgabenträger hinweg entstehen. Darüber hinaus sollte es möglich sein ein Integrationsobjekt über mehrere Integrationsbeziehungen mit verschiedenen anderen Integrationsobjekten integrieren zu können. Die Entkopplung im Sinne von Tichy ist demnach eine Facette, welche essenziell für Integrationsmuster ist, da erst diese Eigenschaften die Konstruktion komplexer Systeme ermöglichen. Verschiedene Kommunikationseigenschaften unterstützen unterschiedliche Grade der Entkopplung. Leider liefert die Arbeit von Tichy keine Hinweise auf mögliche Fokusse einer Facette Entkopplung, da lediglich zwei Mengen an Mustern identifizierbar sind. Jene, welche in der Kategorie Entkopplung sind und jede außerhalb der Kategorie. Als Fokusse wären – auf dieser Basis – nur plus und minus möglich, was hinsichtlich der Unterscheidungskraft bezüglich der Muster unbefriedigend erscheint. Abhilfe schaffen die Betrachtungen von Aldred u.a.[285], welche basierend auf Eugster[286] weitere Detaillierungen vornehmen und die Basis für die Facetten und Fokusse beisteuern. Es werden grundsätzlich unidirektionale Integrationsbeziehungen betrachtet. Bidirektionale Integrationsbeziehungen lassen sich durch zwei unidirektionale Beziehungen mit entgegengesetzten Richtungen ersetzen.

Entkopplung hat drei Dimensionen *Thread*, *Time* und *Space*. Die Dimension *Thread* teilt sich in zwei Facetten: Send und Receive. Die Facette Send kennt zwei Fokusse blocking-send *(bs)* und non-blocking-send *(nbs)*. Die Facette Receive kennt analog die Fokusse blocking-receive *(br)* und non-blocking-receive *(nbr)*.[287] Bei *Time* handelt es sich um eine Facette, welche den zeitlichen Aspekt der Entkopplung betont. Diese kennt zwei Fokusse, welche time-coupled *(tc)* und time-decoupled *(td)* sind. Für time-coupled gilt:„interactions,[...], cannot take place unless both endpoints are concurrently connected“[288]. Dabei wird davon ausgegangen, dass im Fall einer Zeitkopplung der Empfänger in dem Moment mit dem Empfangen beginnt, in dem der Sender beginnt zu senden. Damit ist die hier beschriebe Zeitkopplung in Tabelle 5.37 als Synchronität enthalten. *Space*, als Dimension der Kopplung, bezeichnet die Kenntnis der Kommunikationspartner voneinander. Es gibt keine direkten Referenzen der Sender auf die Empfänger und umgekehrt und auch keine Kenntnis der Anzahl der Kommunikationsteilnehmer.[289] Aldred u.a. bezeichnen dies als die Fähigkeit, zu kontrollieren welche Instanz in die Kommunikation involviert wird.[290] Die Fokusse werden nach Aldred u.a. als space-coupled *(sc)* und space-decoupled *(sd)* bezeichnet. Die Unterscheidung von space-decoupled, ob die schlussendliche Kommunikation einen von vielen oder mehrere von vielen möglichen Kommunikationspartnern betrifft, wird nicht mit übernommen.[291]

[284] (Tichy, 1997, S. 2)
[285] Vgl. (Aldred u. a., 2009).
[286] Vgl. (Eugster u. a., 2003).
[287] Vgl. (Aldred u. a., 2009, S. 2235ff.).
[288] (ebenda, S. 2239)
[289] Vgl. (Eugster u. a., 2003, S. 116).
[290] Vgl. (Aldred u. a., 2009, S. 2239f.).
[291] Vgl. (ebenda, S. 2240).

Pattern	Quelle	Send	Receive	Time/Synchronität	Space
Adapter Style	(Andersson und Johnson, 2001, S. 229)			tc	sc
Adapter static	(ebenda, S. 229)			tc	sc
Adapter dynamic	(ebenda, S. 229)			tc	sd
Adapter thin	(ebenda, S. 229)			tc	sc
Adapter thick	(ebenda, S. 229)			tc	sc
Adapter centralized	(ebenda, S. 229)			tc	sc
Adapter distributed	(ebenda, S. 229)			tc	sc
Adapter	(Khomh und Guéhéneuc, 2008a)				
(Integration) Adapter	(Lutz, 2000)			tc	sd
Component Wrapper	(Goedicke und Zdun, 2002, S. 7ff.)			tc	sc
Wrapper Facade	(Schmidt; Stal u. a., 2000, S. 47ff.)			tc	sc
(Legacy) Wrapper	(Mularz, 1995)			tc	sc
Facade	(Khomh und Guéhéneuc, 2008a)				
Integration Facade	(Lutz, 2000)			tc	sd
Service Interface	(Trowbridge, 2003)	nbs	nbr	td	sd
Integration Messenger (a, broker)	(Lutz, 2000)	nbs	nbr	tc	sd
Integration Messenger (b, message queuing)	(ebenda)	nbs	nbr	td	sd
Integration Messenger (c, publish subscribe)	(ebenda)	nbs	nbr	td	sd
Proxy	(Buschmann; Meunier u. a., 1996, S. 263ff.)	bs	br	tc	sd
Proxy	(Khomh und Guéhéneuc, 2008a)				
Mediator	(Gamma u. a., 1995, S. 277)	bs	br	tc	sd
Integration Mediator	(Lutz, 2000, S. 71)	nbs	nbr		sd
Mediator	(Buschmann; Henney und Schmidt, 2007a, S. 411)				sd
Mediator	(Khomh und Guéhéneuc, 2008a)				
Mediator Style	(Andersson und Johnson, 2001, S. 230)	nbs	nbr	td	
Message Router	(ebenda, S. 231)	nbs	nbr	td	sd
Message Router	(Hohpe und Woolf, 2003, S. 81)	nbs	nbr	td	sd
Message Router	(Buschmann; Henney und Schmidt, 2007a, S. 232)	nbs	nbr	td	sd
Gateway	(Andersson und Johnson, 2001)			tc	sc (sd dynamic)
Service Gateway	(Trowbridge, 2003, S. 293ff.)			td	sd
Broker	(ebenda, S. 207f.)			tc	sd
				tc	sc
Direct Broker	(Trowbridge u. a., 2004, S. 155,215ff.)			tc	sc
Indirect Broker	(ebenda, S. 221)			td	sd
Message Broker	(ebenda, S. 240f.)	nbs	nbr	td	sd
Message Broker	(Hohpe und Woolf, 2003, S. 322ff.)	nbs	nbr	td	sd
Broker	(Buschmann; Meunier u. a., 1996, S. 119ff.)	nbs	nbr	td (indirekt) tc (direct)	sd
Broker Revisited	(Kircher; Voelter u. a., 2004, S. 581)	nbs	nbr	td	sd

Klassifikation entsprechend der Facetten der Entkopplung – Fortsetzung auf der nächsten Seite

Pattern	Quelle	Send	Receive	Time/Synchronität	Space
Broker	(Harrison und Avgeriou, 2007, S. 267f.), bezieht sich auf Buschmann				
Pipes and Filter	(Hohpe und Woolf, 2003, S. 322ff.) , (ebenda, S. 70ff.)	nbs	nbr	td	sd
Pipes and Filter	(Buschmann; Meunier u. a., 1996, S. 67ff.)	nbs	nbr	td	sd
Pipes and Filter	(Trowbridge u. a., 2004, S. 292f.)	nbs	nbr	td	sd
Pipes and Filters	(Garlan und Shaw, 1993, S. 7f.)	nbs	nbr	td	sd
Pipes and Filters	(Meunier, 1995, S. 437)	nbs	nbr	tc	sd
Pipes and Filters	(Harrison und Avgeriou, 2007, S. 267f.) Bezieht sich auf Buschmann				
Observer	(Khomh und Guéhéneuc, 2008a)				
Observer	(Trowbridge, 2003, S. 129f.)	bs	br	tc	sc
Publish and Subscribe	(Trowbridge u. a., 2004, S. 280f.)	nbs		tc	sd
Process Automator	(Lutz, 2000)	nbs	nbr	td	sd
Database Federation	(Andersson und Johnson, 2001)			tc	sd
Desktop Integration	(ebenda)			tc	sc
Data Transfer Object	(Trowbridge, 2003, S. 229)	bs	br	tc	sc
Singleton	(ebenda, S. 257)	bs	br	tc	sc
Singleton	(Khomh und Guéhéneuc, 2008a)				
File Gateway	(Erl, 2009, S. 457)	nbs	nbr	td	sd
Process Centralizazion	(ebenda, S. 193)	nbs	nbr	td	sd
Protocol Bridging	(ebenda, S. 687)	nbs	nbr	td	sd
Legacy Wrapper	(ebenda, S. 441)	nbs	nbr	td	sd
Service Broker (compund Pattern)	(ebenda, S. 707)	nbs	nbr	td	sd
Enterprise Service Bus (compund Pattern)	(ebenda, S. 704)	nbs	nbr	td	sd
Proxy Capability	(ebenda, S. 497)	nbs	nbr	td	sd
Service Facade	(ebenda, S. 333)			tc	sc
UI Mediator	(ebenda, S. 366)	nbs	nbr	tc	sd
Data Model Transformation	(ebenda, S. 671)	nbs	nbr	tc	sd
Data Format Transformation	(ebenda, S. 681)	nbs	nbr	tc	sd
Asynchronous Queuing	(ebenda, S. 582)	nbs	nbr	td	sd
Intermediate Routing	(ebenda, S. 549)	nbs	nbr	td	sd

I: Informationen; D: Dienste; R: Inanspruchnahme von Ressourcen; RR: Reservierung von Ressourcen; sync: synchron; async: asynchron; vo: verbindungsorientiert; vl: verbindungslos; D: Dateien; N: Nachrichten; PA: Prozeduraufrufe; vO: verteilte Objekte; nbs: non blocking send; bs: blocking send; nbr: non blocking receive; br: blocking receive; tc: time coupled; td: time decoupled; sd: space decoupled; sc: space coupled

Tabelle 5.38: Klassifikation entsprechend der Facetten der Entkopplung

Mit der vorliegenden Facette wurde die Letzte der funktionalen Facetten beschrieben, die Eingang in die hier aufgestellte Klassifikation finden soll. Wie die Abbildungen 3.17 und 3.18 in Kapitel 3.5.5 jedoch zeigen, wählen Kunden Integrationslösungen – und damit die Architektur bestehend aus Integrationsmustern – nicht allein anhand funktionaler Kriterien aus. . Deshalb fasst das folgende Kapitel die nicht-funktionalen Facetten zusammen.

5.4.8 Facetten Qualitätseigenschaften der Muster

Neben der Erfüllung der funktionalen Anforderungen sind bei der Entwicklung einer Integrationslösung auch nicht funktionale Anforderungen zu berücksichtigen. Die Ergebnisse der empirischen Untersuchung haben gezeigt, dass diese sogar einen wesentlichen Teil der wichtigsten Auswahlkriterien bilden, anhand derer die Kunden von Integrationsdienstleistern sich für alternative Lösungen entscheiden.[292] Es wird zunächst dargelegt, welches der Kriterien sich auf der Ebene von Mustern und Architekturen überhaupt entscheiden lässt, bevor auf die Ausgestaltung der einzelnen Facetten eingegangen wird. Die Ausgestaltung der Facetten orientiert sich dabei weitestgehend an bestehenden Arbeiten.[293]

Im Rahmen der empirischen Untersuchung wurden die folgenden Kriterien mit absteigender Häufigkeit genannt:

1. Projektkosten,
2. TCO,
3. Flexibilität, Änderbarkeit,Skalierbarkeit,
4. Zeit bis zur Fertigstellung der Lösung,
5. Betreibbarkeit, Bedienbarkeit,
6. Zuverlässigkeit, Stabilität,
7. Performance,
8. Zukunfsfähigkeit,
9. Standardkonformität,
10. Monitoring und Nachvollziehbarkeit,
11. Erfüllung funktionaler, technischer Anforderungen,
12. politische Entscheidungen,
13. Wiederverwendbarkeit,
14. Kosten-Nutzen-Analyse,
15. Geschäftsprozessoptimierung,
16. Kompetenzen des Entwicklers,
17. Testbarkeit,
18. Minimal-Invasivität,
19. Kundenzufriedenheit.

Die am höchsten priorisierten Kriterien – die Projektkosten und Total-Cost-of-Ownership (TCO) – lassen sich nicht auf dem Niveau einer Architektur entscheiden. Sie hängen von konkreten Integrationshilfsmitteln, deren Implementierungstechnologien sowie verfügbaren Experten (Humanressourcen) ab. Dies ist auch der Grund, warum eine Kosten-Nutzen-Analyse nur auf einer detaillierteren Ebene möglich ist. Gleiches gilt für die Zeit bis zur Fertigstellung der Lösung. Sind sehr erfahrene Experten verfügbar, so kann ein Integrationsprojekt dadurch signifikant beschleunigt werden und die Projektdauer sinkt.

[292] Vgl. Kapitel 3.5.5.
[293] Vgl. (Schmidt, 2010) und (Gebauer und Schmidt, 2012)

Betreibbarkeit und Bedienbarkeit sind abhängig von den Integrationshilfsmitteln. Ein mögliches Indiz zur Bewertung wäre die Zahl der Architekturelemente je mehr Elemente es gibt, desto umfangreicher sind die notwendigen Kenntnisse der Experten. Die Abbildung auf die Integrationshilfsmittel spielt aber eine nicht zu unterschätzende Rolle. Vereint ein Hilfsmittel verschiedene Architekturelemente, so kann dennoch eine relativ gute Bettreibbarkeit gegeben sein.

Über die Flexibilität, Änderbarkeit und Skalierbarkeit kann teilweise auf der Ebene einer Architektur entschieden werden. Es lassen sich qualitative Aussagen ableiten, welche eine positive Beeinflussung dieser Kriterien bei Einsatz eines Musters belegen. Gleiches gilt für die Zuverlässigkeit und die Stabilität. Bereits bei der Architekturgestaltung können bestimmte Elemente bewusst redundant ausgelegt werden, was bei Ausfällen von Vorteil sein kann. Ebenso wie bei der Betreibbarkeit spielt auch hier die Abbildung auf die eingesetzten Hilfsmittel eine Rolle. Diese müssen ebenso wie die Architekturelemente redundant ausgelegt werden, damit die geplanten Effekte voll zum Tragen kommen.

Performance, Messungen lassen sich nur auf der Ebene der Hilfsmittel mit absoluten Zahlen hinterlegen. Man kann jedoch davon ausgehen, dass die Zahl der Indirektionen der Kommunikation – welche sich aus einem Architekturentwurf ableiten lassen – qualitative Aussagen über die Performancebeeinflussung erlauben. Je mehr Indirektionen und Stellvertreterkommunikationen vorliegen, desto geringer wird die Performance ausfallen, da diese Form der Entkopplung der Kommunikationspartner mit zusätzlichen Verarbeitungsschritten verbunden ist, welche Rechenleistung und damit Zeit verbrauchen. Konkrete Aussagen sind jedoch nur im Einzelfall möglich, da immer berücksichtigt werden muss, ob ein technischer Performance Rückgang auf die Performance des Geschäftsprozesses einen Einfluss hat, oder ob der Rückgang unerheblich ist. Erwartet ein Kunde eine Rückmeldung über eine Aktion nur innerhalb einer Tagesfrist, so ist die technische Ausgestaltung nicht von Millisekunden abhängig. Für derartige Entscheidungen sind detaillierte Zeitanforderungen der zu unterstützenden Geschäftsprozess unabdingbar.

Die Zukunftsfähigkeit einer Lösung hängt davon ab, ob Entwicklungspfade strategisch geplant werden. Es ist möglich im Rahmen der Architekturgestaltung diese Eigenschaft positiv zu beeinflussen in dem auf entsprechende Anforderungen eingegangen wird. Zukunftsfähigkeit ist aber keine Eigenschaft eines Musters an sich.

Standardkonformität hängt auch von den Hilfsmitteln ab, man kann jedoch die Umsetzbarkeit von Standards verbessern, in dem eine Architektur direkt darauf ausgelegt wird. Schreibt ein Standard z. B. eine nachrichtenbasierte Kommunikation vor, so lässt sich dies auf der Ebene der Architektur planen. Die Umsetzung sollte dann über die Formulierung konkreter funktionaler Anforderungen erfolgen.

Die Eigenschaft Monitoring und Nachvollziehbarkeit kann bspw. durch das Vorsehen entsprechender Komponenten gesteigert werden, entspricht dann aber einer funktionalen Anforderung an die Architektur und ist damit nicht Gegenstand dieses Kapitels.

Eine Erfüllung funktionaler, technischer Anforderungen, wurde bereits in den vorangegangenen Facetten behandelt. Politische Entscheidungen, wie die Festlegung bestimmter Technologien sind kein Gegenstand der Betrachtungen dieser Arbeit.

Die Wiederverwendbarkeit einer Architektur wird grundsätzlich durch die Verwendung von Mustern unterstützt, da diese von der Implementierung abstrahieren. Es handelt sich demnach um eine Eigenschaft, welche durch die in der Arbeit vorgeschlagene Vorgehensweise positiv beeinflusst wird.

Entscheidungen ob eine Geschäftsprozessoptimierung bzw. effiziente Abbildung der Prozesse auf die Architekturelemente vorliegt, sind prinzipiell auf der Ebene einer logischen Facharchitektur

möglich. Diese Entscheidungen betreffen die Abbildung auf die logischen Applikationskomponenten, welche die Geschäftslogik automatisieren. Eine Integrationsarchitektur zielt auf die Teile der Architektur ab, welche keine Geschäftslogik automatisieren.[294] Deshalb ist hier keine Entscheidung im Rahmen der Betrachtungen möglich, da dies den Betrachtungs- und teilweise den Gestaltungsgegenstand der Arbeit verändern würde. Eine Geschäftsprozessoptimierung ist auch möglich, ohne die IT zu verändern und läge damit außerhalb des Fokus.

Die vorhandenen Kompetenzen des Entwicklers sind ein Gegenstand der Gestaltung des Dienstleistungsbündels und keines der Architekturgestaltung. Sie finden auf einer der Ausgestaltung einer Architektur mit Mustern vorgelagerten Ebene statt.

Die Testbarkeit einer Lösung hängt von vielen Detailfaktoren auf der Ebene der Integrationshilfsmittel ab und ist im Rahmen der vorliegenden Informationen eines Architekturentwurfes nicht entscheidbar.

Obwohl für das Kriterium der Minimal-Invasivität bisher keine Messbarkeitskriterien vorliegen, lässt sich eine qualitative Aussage auf der Ebene einer Architektur treffen.

Die Messung der Kundenzufriedenheit ist ein Prozess, welcher erst nach dem Einsatz einer Lösung stattfinden kann. Ein mittelbarer Einfluss ist über die Formulierung entsprechender Anforderungen gegeben.

Wie die vorangegangenen Ausführungen gezeigt haben, lassen sich einige der Auswahlkriterien mit einer detaillierten Untersuchung der Muster zumindest auf qualitativer Ebene – im Sinne einer Aussage über eine positive bzw. negative Beeinflussung – treffen, einige Kriterien sind aber Gegenstand der in Kapitel 4.3 geschilderten Kommunikationsprobleme und einer möglichst umfassenden Formulierung von Anforderungen an die Integrationslösung. Im weiteren Verlauf entwickeln werden aus diesen Problemen die qualitativen Facetten der Klassifikation entwickelt. Dabei stützen sich die Ausführungen auf die Arbeiten von Schmidt[295] und Gebauer und Schmidt[296].

Der Zusammenhang zwischen Mustern und Qualität wurde bereits zum Teil untersucht, da Mustern grundsätzlich ein positiver Einfluss auf die Qualität zugeschrieben wird. Dies gilt jedoch nicht uneingeschränkt, da eine – die Qualität positiv beeinflussende – Entscheidung im Rahmen des Architekturentwurfes durch eine schlechte Umsetzung z. B. Implementierung wieder zu Nichte gemacht werden kann.[297] Qualität kann dabei als die „Gesamtheit von Merkmalen einer Einheit bezüglich ihrer Eignung, festgelegte und vorausgesetzte Erfordernisse zu erfüllen“[298] gesehen werden. Auf der Basis dieser Definition lässt sich die Frage formulieren, welche Eigenschaften und Merkmale relevant sind.

Die Definitionen verschiedener Qualitätsmerkmale und Eigenschaften sind in der Literatur nicht einheitlich. So verwenden Harrison und Avgeriou[299], Khohm und Guéhéneuc[300], sowie Majidi, Alemi und Rashidi[301] bereits ein breites Spektrum an verschiedenen Qualitätskriterien. In der Tabelle 5.39 werden die Qualitätsmerkmale dieser Publikationen gegenüber gestellt.

Dabei bleiben Standards und Normen wie die ISO/IEC 9126 häufig außer Acht, obwohl sich in allen drei Beispielen sowohl Qualitätsmerkmale (**) als auch Qualitätseigenschaften (*) finden

[294] Vgl. Kapitel 5.1.3.
[295] Vgl. (Schmidt, 2010).
[296] Vgl. (Gebauer und Schmidt, 2012).
[297] Vgl. (Bass; Clements und Kazman, 2003, S. 30f.) und (Malich, 2008, S. 48).
[298] (Schatten u. a., 2010, S. 110)
[299] Vgl. (Harrison und Avgeriou, 2007).
[300] Vgl. (Khomh und Guéhéneuc, 2008a; Khomh und Guéhéneuc, 2008b).
[301] Vgl. (Majidi; Alemi und Rashidi, 2010).

Qualitätsmerkmal	Publikation			
	(Harrison und Avgeriou, 2007)	(Khomh und Guéhéneuc, 2008b)	(Majidi; Alemi und Rashidi, 2010)	(Andersson und Johnson, 2001)
Expandability		X		
Simplicity		X		
Reuseability		X	X	
Learnability*		X		
Understandability*		X		
Modularity		X		
Generality		X		
Modularity at runtime		X		
Scalability		X	X	X
Robustness		X		
Reliability**	X		X	X
Portability**	X		X	
Implementability	X			
Usability**	X			X
Security*	X		X	X
Maintainability**	X			
Efficiency**	X			
Testability*			X	
Modifiability			X	X
Availability			X	
Performance			X	X
Integrability				X
Upgradeability				X
Data consistency				X
Transaction atomicity				X
Isolation				X
Durability				X
Accessibility				X
Component data accessibility				X
Component functionality accessibility				X
User data accessibility				X
User functionality accessibility				X

Tabelle 5.39: Vergleich von beschriebenen Qualitätsmerkmalen für Muster
Quelle: in Anlehnung an (Schmidt, 2010, S. 20)

lassen, die in der eben angeführten Norm enthalten sind[302], stellt keines der Beispiele vollständig auf die Norm ab. Die Tabelle 5.40 stellt die Qualitätsmerkmale und -teilmerkmale der ISO-Norm dar. Ein Qualitätsmerkmal kann durch verschiedene Teilmerkmale bestimmt werden.

Mit dieser Systematik liefert die ISO/IEC 9126 einen einheitlichen Beschreibungsrahmen, mit dem existierende Aussagen zu qualitativen Eigenschaften von Integrationsmustern einheitlich beschrieben werden können. Es finden sich jedoch nicht zu allen Qualitätsmerkmalen des Standards Aussagen in der untersuchten Literatur.[303] Die folgenden Ausführungen stellen die Abbildung der Qualitätsmerkmale einzelner Autoren auf die ISO/IEC 9126 Norm dar. Durch diese Abbildung wird die Klassifikation nachvollziehbarer, weshalb die Qualitätsmerkmale mit (**) und Qualitätseigenschaften mit (*) gekennzeichnet sind, da es gelegentlich vorkommt, dass sich ein Musterautor gemischt, sowohl auf Eigenschaften als auch auf Merkmale bezieht. Dabei werden hier nur Publikationen angeführt, die sich nicht bereits explizit zu der Verwendung der ISO/IEC Norm bekennen – wie z. B. Harrison und Avgeriou[304] – oder welche die Eigenschaften nur textuell formulieren, ohne explizit eine Merkmalsystematik zu verwenden. Die letzteren werden auf die

[302] Vgl. (Schmidt, 2010, S. 20).
[303] Vgl. (Gebauer und Schmidt, 2012, S. 31), sowie (Schmidt, 2010, S. 18ff.).
[304] Vgl. (Harrison und Avgeriou, 2007, S. 266).

Qualitätsmerkmal	**Teilmerkmal**
Funktionalität (Functionality)	Genauigkeit (Accuracy) Angemessenheit (Suitability) Interoperabilität (Interoperability) Sicherheit (Security)
Zuverlässigkeit (Reliability)	Reife (Maturity) Fehlertoleranz (Fault tolerance) Wiederherstellbarkeit (Recoverability)
Benutzbarkeit (Usability)	Verständlichkeit (Understandability) Erlernbarkeit (Learnability) Bedienbarkeit (Operability) Attraktivität (Attractiveness)
Effizienz (Efficiency)	Zeitverhalten (Time behavior) Verbrauchsverhalten (Resource behavior)
Änderbarkeit (Maintainability)	Analysierbarkeit (Analyzability) Veränderlichkeit (Changeability) Stabilität (Stability) Prüfbarkeit (Testability)
Übertragbarkeit (Portability)	Anpassbarkeit (Adaptability) Installierbarkeit (Installability) Austauschbarkeit (Replaceability) Koexistenz (Co-existence)

Tabelle 5.40: Qualitätsmerkmale nach ISO/IEC 9126
Quelle: (Malich, 2008, S. 63f.)

Merkmalsystematik abgebildet, ohne jede einzelne Zuordnung gesondert aufzuführen. Das einzige Merkmal was von Harrison und Avgeriou genutzt wird und sich nicht in der ISO/IEC Norm wiederfindet ist Implementability[305] welches sich nicht auf die Produktqualität bezieht und deshalb nicht abgebildet wird. Die Tabelle 5.41 zeigt die Abbildung der Merkmale nach Khomh und Guéhéneuc auf die Norm.

Khomh und Guéhéneuc	**ISO/IEC 9126**
Expendability[306]	Changeability (*)
Simplicity	Understandability
Generality	Suitability
Modularity	keine Abbildung möglich
Mod. at runtime	keine Abbildung möglich, siehe Modularity
Learnability	Learnability
Understandability	Understandability
Reusability	Reusability
Scalability	Adaptability
Robustness	Fault tolerance

Tabelle 5.41: Abbildung der Qualitätsmerkmale nach Khomh und Guéhéneuc auf ISO/IEC 9126

Expandability ist definiert als der Grad der Erweiterbarkeit eines Systems und wird deshalb auf *Changeability* abgebildet. Obwohl *Simplicity* als Einfachheit übersetzt werden kann, verstehen Khohm und Guéhéneuc darunter die Verständlichkeit des Systemdesigns, weshalb *Understandability* das passende ISO/IEC Merkmal ist. Modularität an sich ist kein Qualitätsmerkmal. Die Angemessenheit der Modularität müsste geprüft werden. Diese kann Merkmale wie die Wartbarkeit und Verständlichkeit beeinflussen. Die Definition der Anpassbarkeit nach Malich führt Benutzerzahlen – die ein Grund für die Skalierung sind– explizit auf, weshalb *Scalability* auf *Adaptability* abgebildet wird.

[305] Vgl. (Harrison und Avgeriou, 2007, S. 266).
[306] sic! es ist eigentlicht Expandability gemeint.

Durch die Trennung der Qualitätsattribute in drei Gruppen – Attribute mit Bezug zum Design, zur Implementierung und zur Laufzeit – müssen sowohl Simplicity, als auch Unterstandability auf das Merkmal Understandability nach ISO/IEC abgebildet werden. Ergeben sich bei beiden Merkmalen unterschiedlich Bewertungen so resultiert daraus eine gemischte Bewertung nach ISO/IEC in Form von *+/-*

Andersson und Johnson	**ISO/IEC 9126**
Modifiability	Changeability
Integrability	Co-existence
Scalability	Adaptability
Upgradeability	Replaceability
Performance	Efficiency
Usability	Usability
Reliability	Reliability
Data consistency	Abbildung nicht möglich
Transaction atomicity	Abbildung nicht möglich
Isolation	Abbildung nicht möglich
Durability	Abbildung nicht möglich
Security	Security
Accessibility	Functionality
Component Data accessibility	Abbildung nicht möglich
Component functionality accessibility	Abbildung nicht möglich
User Data accessibility	Abbildung nicht möglich
User functionality accessibility	Abbildung nicht möglich

Tabelle 5.42: Abbildung von Andersson und Johnson auf ISO/IEC 9126

Die Integrierbarkeit (*Integrability*) wird am ehesten dadurch erschwert, dass es Auswirkungen und Einschränkungen in der Softwareumgebung gibt, weshalb Co-existence als Abbildung gewählt wird. Es lässt sich aus der Definition von *Performance* leider nicht entnehmen, ob mit „how well the computer system responds to its inputs“[307] das Zeit- oder das Ressourcenverbrauchsverhalten gemeint ist. Modifiability – „The ability of a system to be extended to accomplish additional functionality“[308] – kann mittelbar auf Maintainability abgebildet werden, da sich das Teilmerkmal Changeability unter anderem auf Änderungen bezüglich der Anforderungen bezieht. Bestehen diese in funktionalen Anforderungen, so kann es sich um zusätzliche Funktionalität handeln. Das Kriterium Replaceability adressiert explizit auch den Austausch von Teilen einer Anwendung.

Buschmann; Meunier u. a.	**ISO/IEC 9126**
Changeability	Maintainability
Maintainability	Changeability
Extensibility	Adaptability, Replaceability
Restructuring	Adaptability
Portability	Portability
Interoperability	Interoperability
Efficiency	Efficiency
Reliability	Reliability
Fault tolerance	Fault tolerance
Robustness	Abbildung nicht möglich
Testability	Testability
Reusability	Reusability

Tabelle 5.43: Abbildung der Qualitätsmerkmale nach Buschmann; Meunier u. a. auf ISO/IEC 9126

[307] (Andersson und Johnson, 2001, S. 227)
[308] (ebenda, S. 227)

Buschmann ordnet Maintainability, Extensibility, Restructuring und Portability als Teilmerkmale von Changeability an. Dies stimmt so nicht mit der ISO/IEC Norm überein, welche Changeability als Teilmerkmal von Maintainability sieht. Betrachtet man jedoch die jeweiligen Begriffsdefinitionen, so entsteht die in Tabelle 5.43 enthaltene über Kreuz Zuordnung. Extensibility zielt nicht nur auf die Erweiterung, sondern auch auf den Austausch von Teilkomponenten und das Entfernen unnötiger Funktionalität ab.[309] Damit werden Adaptability und Replaceability nach ISO/IEC angesprochen, weshalb auf beide Merkmale abgebildet wird. Findet sich eine positive Bewertung von Extensibility nach Buschmann, so wird beiden Merkmalen nach ISO/IEC eine positive Bewertung zugeordnet.

Das Merkmal Robustness lässt sich nicht auf die ISO/IEC Norm abbilden. Im Gegensatz zu Fault-Tolerance muss eine Software im Fehlerfall die Arbeit nicht fortsetzen, sondern ein Terminieren mit Erreichen eines definierten Zustandes ist bei dieser Eigenschaft ausreichend. In der ISO/IEC Norm findet sich keine Entsprechung.

Nach der nun erfolgten Abbildung der Merkmale auf die einheitliche ISO-Systematik müssen aus diesen die Facetten ausgewählt werden. Als Facetten bieten sich die Qualitätsmerkmale an, da diese häufiger in der Literatur zu finden sind als Aussagen zu einzelnen Teilmerkmalen. Es kommt dennoch vor, dass Musterautoren einzelne Qualitätseigenschaften ansprechen. In der Analyse wird dann so vorgegangen, dass bei ausschließlich positiven Qualitätseigenschaften das Qualitätsmerkmal ebenfalls positiv bewertet wird. Bei ausschließlich negativen Eigenschaften wird das Merkmal negativ bewertet und sind sowohl negative als auch positive Eigenschaften vorhanden, so bekommt das zugehörige Merkmal eine gemischte positive/negative Bewertung. Eine Ausnahme bildet die Facette Reusability – die Wiederverwendbarkeit – obwohl nicht in der ISO/IEC Norm enthalten, wurde sie von so vielen Musterautoren angeführt, dass sie in die Klassifikation mit aufgenommen wurde. „Reuse is the use of existing assets in the development of other software with the goal of improving productivity, quality, and other factors (e.g., usability)“[310]. Bei der Wiederverwendbarkeit handelt es sich um ein abgeleitetes Merkmal, welches sich durch die Nutzbarkeit für Entwickler, Adaptierbarkeit und Portierbarkeit bestimmt.[311] Stützle betont, dass es sich bei diesen Merkmalen nur um diejenigen handelt, welche wiederverwendbare Software in besonderem Maße haben muss, in jedem Fall in besserer Ausprägung als nicht wiederverwendbare Software. Es müssen jedoch weitere Kriterien erfüllt sein, die aber auch bei nicht wiederverwendbarer Software erfüllt sein müssen. Dazu gehören nach Stützle Spezifikationstreue der Funktionalität, Zuverlässigkeit, Benutzbarkeit und Effizienz.[312]

Viele Musterautoren beschreiben die Auswirkungen auf die Qualitätsmerkmale, in dem sie von einem positiven oder negativen Einfluss sprechen. „Given that the gateway is well implemented, it will have moderately negative effects on performance, and no effects on reliability or security per se“[313]. Aus derartigen Aussagen lassen sich weder absolute Aussagen ableiten, noch sind die Aussagen so miteinander vergleichbar, dass Reihenfolgen gebildet werden können. Obwohl für zwei Muster die Aussage verfügbar sein kann, dass beide ein Merkmal positiv beeinflussen, kann nicht abgeleitet werden, welches der beiden Muster das Merkmal stärker verbessert.

Aus diesen Gründen finden als Fokusse für alle qualitativen Facetten *+* als Indikator für einen positiven Einfluss, - für einen negativen Einfluss Verwendung. *+/-* wird genutzt, wenn eine Eigenschaft nicht eindeutig als positiv oder negativ zu bewerten ist und *n*, wenn ein Muster explizit als neutral bezüglich des Merkmals ausgewiesen wird. Der Einflussgrad *n* sollte nicht mit

[309] Vgl. (Buschmann; Meunier u. a., 1996, S. 406).
[310] (Lim, 1998, S. 7), ähnliche Definitionen finden sich bei (Krueger, 1992), (Stützle, 2002) und (Jonge, 2003).
[311] Vgl. (Stützle, 2002, S. 13).
[312] Vgl. (ebenda, S. 13).
[313] (Andersson und Johnson, 2001, S. 229)

+/- verwechselt werden. Bei Ersterem kann man keinen Einfluss auf ein Merkmal feststellen und bei Letzterem, finden sich – je nach Betrachtungsweise – sowohl Argumente für eine positive als auch eine negative Beeinflussung desselben Merkmals. In der nachfolgenden Tabelle werden nur die Qualitätsmerkmale dargestellt. Die Herleitung der Bewertung aus den Teilmerkmalen ist im Anhang enthalten.

Pattern	**Quelle**	**Functionality**	**Reliability**	**Usability**	**Efficiency**	**Maintainability**	**Portability**	**Reusability**
Adapter Style	(Andersson und Johnson, 2001, S. 229)	+				-		
Adapter static	(ebenda, S. 229)	+				-		
Adapter dynamic	(ebenda, S. 229)	+				-		
Adapter thin	(ebenda, S. 229)	+				-		
Adapter thick	(ebenda, S. 229)	+				-		
Adapter centralized	(ebenda, S. 229)	+				-		
Adapter distributed	(ebenda, S. 229)	+	+			-		
Adapter	(Khomh und Guéhéneuc, 2008a)	+	-	+/-		+	-	+
(Integration) Adapter	(Lutz, 2000)						-	+
Component Wrapper	(Goedicke und Zdun, 2002, S. 7ff.)	+		-	-	+	+	+
Wrapper Facade	(Schmidt; Stal u. a., 2000, S. 47ff.)	-		+	-	+	+	+
(Legacy) Wrapper	(Mularz, 1995)							
Facade	(Khomh und Guéhéneuc, 2008a)	+	-	+		+	-	-
Integration Facade	(Lutz, 2000)							
Service Interface	(Trowbridge, 2003)	-			-	+/-	+	
Integration Messenger (a, broker)	(Lutz, 2000)							
Integration Messenger (b, message queuing)	(ebenda)							
Integration Messenger (c, publish subscribe)	(ebenda)							
Proxy	(Buschmann; Meunier u. a., 1996, S. 263ff.)				+/-	+	-	+
Proxy	(Khomh und Guéhéneuc, 2008a)	+	-	+/-		-	-	+
Mediator	(Gamma u. a., 1995, S. 277)			+		+	+	+/-
Integration Mediator	(Lutz, 2000, S. 71)					+		+
Mediator	(Buschmann; Henney und Schmidt, 2007a, S. 411)		-		-	-		
Mediator	(Khomh und Guéhéneuc, 2008a)		-	+/-		+	-	-
Mediator Style	(Andersson und Johnson, 2001, S. 230)	+	-		+		+	
Message Router	(ebenda, S. 231)	+/-	-				+	
Message Router	(Hohpe und Woolf, 2003, S. 81)			-	-	+/-	+	
Message Router	(Buschmann; Henney und Schmidt, 2007a, S. 232)			-	-	+/-		
Gateway	(Andersson und Johnson, 2001)	n	n		-	+/-		
Service Gateway	(Trowbridge, 2003, S. 293ff.)						+/-	+
Broker	(ebenda, S. 207f.)	+		+	-		+	
Direct Broker	(Trowbridge u. a., 2004, S. 155,215ff.)							
Indirect Broker	(ebenda, S. 221)							
Message Broker	(ebenda, S. 240f.)	+	-		-	+/-	+	
Message Broker	(Hohpe und Woolf, 2003, S. 322ff.)							
Broker	(Buschmann; Meunier u. a., 1996, S. 119ff.)	+	-		-	+/-	+	+
Broker Revisited	(Kircher; Voelter u. a., 2004, S. 581)		-		-		+	
Broker	(Harrison und Avgeriou, 2007, S. 267f.)	+	n	+	n	+	+	
Pipes and Filter	(Hohpe und Woolf, 2003, S. 70ff.,S. 322ff.)				+/-	+		+
Pipes and Filter	(Buschmann; Meunier u. a., 1996, S. 67ff.)				+/-	+	+	+
Pipes and Filter	(Trowbridge u. a., 2004, S. 292f.)			-	+/-	-	+/-	+
Pipes and Filters	(Garlan und Shaw, 1993, S. 7f.)			+	+/-	+		+
Pipes and Filters	(Meunier, 1995, S. 437)			+	+/-	+	+	+
Pipes and Filters	(Harrison und Avgeriou, 2007, S. 267f.)	-	-	-	+/-	+	+	

Nicht-funktionale Klassifikation der Integrationsmuster – Fortsetzung auf der nächsten Seite

Pattern	Quelle	Functionality	Reliability	Usability	Efficiency	Maintainability	Portability	Reusability
Observer	(Khomh und Guéhéneuc, 2008a)	+	-	+/-		+	-	+
Observer	(Trowbridge, 2003, S. 129f.)			-	-	-	+	+
Publish and Subscribe	(Trowbridge u. a., 2004, S. 280f.)	+			-	+/-	+	
Process Automator	(Lutz, 2000)		+			+	+	
Database Federation	(Andersson und Johnson, 2001)	+	-			+		
Desktop Integration	(ebenda)	+		-		-		
Data Transfer Object	(Trowbridge, 2003, S. 229)			+	+/-	+/-		
Singleton	(ebenda, S. 257)	+		-	-	+		
Singleton	(Khomh und Guéhéneuc, 2008a)	-	-	+		-	-	-
File Gateway	(Erl, 2009, S. 457)				-			-
Process Centralizazion	(ebenda, S. 193)							
Protocol Bridging	(ebenda, S. 687)		-		-			
Legacy Wrapper	(ebenda, S. 441)				-			
Service Broker (compund Pattern)	(ebenda, S. 707)							
Enterprise Service Bus (compund Pattern)	(ebenda, S. 704)							
Proxy Capability	(ebenda, S. 497)		-					
Service Facade	(ebenda, S. 333)				-			
UI Mediator	(ebenda, S. 366)				-			+
Data Model Transformation	(ebenda, S. 671)				-			-
Data Format Transformation	(ebenda, S. 681)				-			
Asynchronous Queuing	(ebenda, S. 582)		+/-		-	+/-		
Intermediate Routing	(ebenda, S. 549)	-	-		-	+/-		

+: positiv; -: negativ; +/-: posititv/ negativ; n: neutral

Tabelle 5.44: Nicht-funktionale Klassifikation der Integrationsmuster

Wie die Klassifikation darstellt, finden sich für gleiche Muster verschiedene Bewertungen je nach dem, welche Quelle dafür ausgewertet wurde. Auch wurden Quellen separat aufgeführt, wenn Sie sich z. B. nur mit den qualitativen Facetten befasst haben und auf eine Beschreibung der funktionalen Eigenschaften verzichtet wurde.

5.4.9 Zusammenfassung der Klassifikation

An dieser Stelle werden noch einmal alle Facetten und Fokusse zusammengefasst, um die Klassifikation als Ganzes darzustellen. Dies ermöglicht einen leichten Einstieg in die Nutzung und vor allem Erweiterung des Klassifikationsschemas.

Facette	Fokusse
Subsystem BIS	Idiom Entwurfsmuster Architekturmuster Organisationsmuster Hardwaremuster
Integrationsoperation	Erstellung Vernichtung Restrukturierung Schnittstellenergänzung Schnittstellenentfernung Adaption

Facetten und Fokusse der Klassifikation – Fortsetzung auf der nächsten Seite

Facette	**Fokusse**
	Desintegrierung
	direkte Kopplung
	indirekte Kopplung
	Verschmelzung
	Entkopplung
Integrationsarchetyp	Alignment
	Ableitung
	Bindung
	Vereinigung
Komponente der Referenzarchitketur	Monitoring
	Protokollierung
	Reporting
	Ausfallsicherheit
	Lastverteilung
	Fehlerbehandlung
	fachliche Transformation
	technische Transformation
	Cleansing
	Lieferung
	dyn.&stat. Adressierung
	Massendatenzugriff
	Einzeldatenzugriff
	Stubs f. domänspez. Services
	Identizierung& Authentifizierung
	Zugriffsschutz& Benutzerkanalsicherung
	Single Sign-on
	Steuerung von Portalarbeitsschrittfolgen
	Komposition
	Personalisierung
	Collaboration& Community
	Content Management
	Tracking
	Multikanalunterstützung
	Darstellung
	Dialogsteuerung
	Process Engine
	Transaktionen
	Ereignisverwaltung
	Suchen & Ändern
	manuelle Dubletten Bereinigung
	Datenmodell
	Hierarchien, Taxonomien, Relationen
	Datenbasis
	historische Daten
	Postkorb
	Dialogeinsprung
	Organisationsrepository
	Prozessrepository
	Servicerepository
	Collection/Distribution
	Governance
Heterogenitätskonflikt	Information
	Verhalten
	Transport
	kombinierte
	Konflikte
Zweck der Kommunikation	Informationen
	Dienste
	Ressourcen
	Reservierung von Ressourcen
Synchronität	sync (synchron)
	async (asynchron)
Verbindungsorientierung	vo (verbindungsorientiert)
	vl (verbindungslos)

Facetten und Fokusse der Klassifikation – Fortsetzung auf der nächsten Seite

Facette	**Fokusse**
Kardinalität d. Komm	1:1 m:1 1:m m:n
Interaktionsmechanismus	D (Dateien) N (Nachrichten) PA (Prozeduraufrufe) vO (verteilte Objekte)
Send	bs (blocking send) nbs (non blocking send)
Receive	br (blocking read) nbr (non blocking read)
Time/Synchronität	tc (time coupled) td (time decoupled)
Space	sc (space coupled) sd (space decoupled)
Testability	- (negativ) + (positiv) +/- (positiv/ negativ) n (neutral)
Reliability	- (negativ) + (positiv) +/- (positiv/ negativ) n (neutral)
Analyzability	- (negativ) + (positiv) +/- (positiv/ negativ) n (neutral)
Portability	- (negativ) + (positiv) +/- (positiv/ negativ) n (neutral)
Reusability	- (negativ) + (positiv) +/- (positiv/ negativ) n (neutral)
Efficiency	- (negativ) + (positiv) +/- (positiv/ negativ) n (neutral)
Interoperability	- (negativ) + (positiv) +/- (positiv/ negativ) n (neutral)
Security	- (negativ) + (positiv) +/- (positiv/ negativ) n (neutral)
Maintainability	- (negativ) + (positiv) +/- (positiv/ negativ) n (neutral)

Tabelle 5.45: Facetten und Fokusse der Klassifikation

5.5 Höhere Integrationsmuster durch Generalisierung

Auf den ersten Blick scheint die Generalisierung nicht geeignet im Nachgang einer Klassifikation angewendet werden zu können, ist diese Abstraktionsart doch eine Voraussetzung für die Klassifikation. Dennoch wird hier der Versuch unternommen, die klassifizierten Muster einer weiteren Generalisierung zu unterziehen. Durch die bisherige Arbeit an der Klassifikation ist die Hypothese entstanden, dass sich alle Muster auf einige wenige Grundprinzipien reduzieren lassen, welche sich auf allen Ebenen von BIS wiederfinden. Es wird versucht diese Hypothese an zwei Gruppen von Mustern nach zu weisen, welche sich in der bisher verwendeten Musterauswahl finden. Zum einen ist dies die Gruppe der Muster Adapter, Wrapper, Facade und Gateway (Kapitel 5.5.1) und zum anderen die Gruppe der Muster Broker, Mediator, Router (Kapitel 5.5.2). Diese beiden exemplarischen Anwendungen der Generalisierung werden im Ausblick in Kapitel 8.2 zusammengefasst und es wird aufgezeigt, inwieweit sich dieser Mechanismus fortsetzen lässt.

5.5.1 Adapter, Wrapper, Facade und Gateway, der Typ zusätzlicher Zugriffspunkt

Dieser Abschnitt untersucht die Muster Adapter und Wrapper, Facade sowie Gateway hinsichtlich eines gemeinsamen, grundlegenden Konzeptes. Als Ausgangsbasis dienen die bereits klassifizierten Muster, ergänzt um die Ausführungen von Gamma zu Entwurfsmustern. Ausgehend von der Benennung der Muster fällt auf, dass sich in dem Paper von Gamma u.a. auf der ECOOP[314] 1993[315] die Bezeichnung Wrapper für ein Muster in Bezug auf Compounds findet. In dem Buch von 1995[316] – welches weltweite Verbreitung gefunden hat – taucht der Begriff Wrapper zur Bezeichnung eines Musters nicht mehr auf. Im Index ist unter Wrapper aber „see Adapter, Decorator“[317] verzeichnet. Augenscheinlich ist der Decorator an die Stelle des 1993 als Wrapper beschriebenen Musters getreten, die Beschreibungen sind nahezu identisch. Die Entwurfsmuster nach Gamma u. a.[318] befassen sich mit Problemen der objektorientierten Softwareentwicklung, weshalb Klassen und Objekte zu den Integrationsobjekten zählen. Das dort beschriebene Wrapper Muster verfolgt die Absicht zusätzliche Services, Eigenschaften oder Verhalten zu einem Objekt hinzuzufügen. Die Motivation dafür ist explizit, derartige Erweiterungen bei individuellen Objekten vorzunehmen anstelle von Klassen. Als Alternative auf Klassenbasis wird Vererbung genannt, wobei dabei der Nachteil bestünde, dass alle Instanzen die erweiterten Eigenschaften erhielten. Diesen Umstand soll der Wrapper explizit vermeiden. Er schließt das betroffene Objekt ein und leitet Anfragen an das Interface des umschlossenen Objektes weiter. Es wird darauf verwiesen, dass der Wrapper zusätzliche Funktionalität – eben jene hinzuzufügenden Dinge – umsetzen kann, bevor oder nach dem er eine solche Anfrage weiter geleitet hat. Gamma u. a. stellen insbesondere die Anwendbarkeit bei Basisklassen von Bibliotheken heraus, die nicht so einfach verändert werden können.

Unter der Beschreibung des Adapters wird unter also-known-as als Synonym Wrapper aufgeführt. Es werden zwei Arten von Adaptern unterschieden, Klassen und Objektadapter. Der Klassenadapter basiert auf dem Konzept der multiplen Vererbung und der Objektadapter basiert auf Objektkomposition. Anzumerken ist an dieser Stelle, dass die Entwurfsmuster auf C++ basieren und in anderen Sprachen wie beispielsweise Java eine Mehrfachvererbung – Interfaces

[314] European Conference on Object-Oriented Programming
[315] Vgl. (Gamma u. a., 1993).
[316] Vgl. (Gamma u. a., 1995).
[317] (ebenda, S. 395)
[318] Vgl. (Gamma u. a., 1993).

ausgenommen – nicht möglich ist. Die Umsetzung des Musters ähnelt in diesen Sprachen dem Objektadapter.[319]

Im Vergleich zu einem Adapter ändert der Wrapper nur die Eigenschaften eines Objektes und nicht das Interface. Ein Adapter hingegen gibt einem Objekt ein völlig neues Interface. Weiterhin wird ausgeführt, dass ein Wrapper als ein degeneriertes Composite mit nur einer Komponente gesehen werden kann. Beiden gemeinsam ist jedoch, dass sie einen neuen Zugriffspunkt erschaffen.

Mularz [320] stellt bei der Definition des (Legacy-) Wrappers explizit auf Integrationsmuster ab. So verfolgt das Wrapper Muster die Absicht fortdauernden Zugriff auf eine Applikation zu gewährleisten, während die Funktionalität der Applikation verändert wird. Perspektivisch ist auch eine Ablösung der Applikation möglich. Das Muster wird eingesetzt, wenn die alte Funktionalität weiterhin von Nöten zusätzlich aber eine Erweiterung notwendig ist. Insbesondere dann, wenn der Zugriff von unterschiedlichen, heterogenen Arbeitsumgebungen aus notwendig ist. Dies ist das kurzfristige Ziel. Das langfristige Ziel besteht in einer potenziellen, inkrementellen Ablösung der Anwendung bei weiterhin transparentem Zugriff durch die Nutzer. Anwendbar ist das Muster bei jeder Applikation mit einem definierten Application-Programming-Interface (API), womit der Einsatzbereich nun über Legacy-Anwendungen hinaus geht, bei der ein erneutes Erstellen der Schnittstelle nicht realisierbar, gewollt ist. Mularz gibt dabei auch noch Hinweise, inwieweit sich das Muster mit dem Broker ideal kombinieren lässt.

Eine analoge Absicht verfolgt der Component Wrapper von Goedicke und Zdun.[321] Dieser stellt einen alternativen Zugriffspunkt für Black-Box-Komponenten zur Verfügung um diese – über eine bei derartigen Komponenten übliche Parametrisierung hinaus – anpassen zu können.

Schmidt; Stal u. a. betrachten in ihrer Publikation in der Reihe Pattern-oriented-Software-Architecture (POSA) die Wrapper Facade.[322] Wesentliches Merkmal der dort vorgestellten Wrapper Facade ist, dass dieses Muster das Ziel verfolgt den Zugriff von objektorientierten Applikationen auf nicht objektorientierte APIs (funktionale Programmiersprachen) zu gestalten. Statt die API direkt anzusprechen, wird diese in Klassen gekapselt, deren Methoden die Aufrufe an die funktionale API weiterleiten. Dadurch wird eine Komplexitätsreduktion erreicht. Die Portabilität verbessert sich, wobei diese ggf. durch die Portabilität der gekapselten funktionalen API beschränkt wird. Für das Untersuchungsziel – der Identifikation gemeinsamer wesentlicher Konzepte verschiedener Muster – ist die von Schmidt; Stal u. a. vorgenommene Abgrenzung[323] hilfreich. Gegenüber der Facade nach Gamma u. a.[324] besteht der wesentliche Unterschied darin, dass die Wrapper Facade nicht objektorientierte APIs kapselt. Warum eine Abgrenzung zum Decorator[325] erfolgt, bleibt unklar. Es wird ausgeführt, dass der Decorator[326] Funktionalität erweitern will, was zu Performanceeinbußen führt. Das Ziel eine andersartige API objektorientiert zugreifbar zu machen – wie es die Wrapper Facade verfolgt – besteht nicht. Bridge und Adapter[327] führen im Vergleich ebenfalls eine Indirektion ein, welche einen Overhead darstellen kann. Diese Gründe stellen – so Schmidt; Stal u. a. – die unzureichende Eignung, und damit auch die Abgrenzung, zu der Wrapper Facade dar. Gemeinsam mit allen anderen Mustern von denen dieses abgegrenzt werden soll ist aber, dass ein neuer Zugriffspunkt geschaffen wird.

[319] Vgl. (Metsker und Wake, 2006).
[320] Vgl. (Mularz, 1995).
[321] Vgl. (Goedicke und Zdun, 2002, S. 7f.).
[322] Vgl. (Schmidt; Stal u. a., 2000, S. 47).
[323] Vgl. (ebenda, S. 49).
[324] Vgl. (Gamma u. a., 1995).
[325] Vgl. (ebenda).
[326] Das Decorator-Muster wird in (Gamma u. a., 1993) auch als Wrapper bezeichnet.
[327] Vgl. (Gamma u. a., 1995).

Andersson und Johnson[328] beschreiben umfangreich den Adapter-Style sowie den Gateway-Style. Die Hauptmotivation des Adapter-Musters ist das Binden und Transformieren von Schnittstellen, hauptsächlich um Integrationsobjekte mit einem Mediator zu verbinden. Für ein Gateway wird als Funktionalität die Abbildung von Schnittstellen zwischen einem Quell- und einem Zielsystem benannt. In der Gateway beschreibung heißt es: „Application gateways, or adapters, encapsulate the application logic and the data by providing the source component with an application interface compatible with the target component“[329]. Die separate Aufführung des Adapters trotz großer Nähe zum Gateway Muster wird mit: „it is however presented as a style in its own right because of its common usage“[330] begründet. Auch bei diesen Beschreibungen wird die funktionale Gemeinsamkeit eines alternativen, zusätzlichen Zugriffspunktes mehr als deutlich.

Bei Lutz [331] findet sich ebenfalls eine Trennung von zwei Mustern, diesmal in Integration Adapter und Integration Facade. Der durch die Integration Facade zur Verfügung gestellte Zugriffspunkt weist im Vergleich zu dem ursprünglichen Mechanismus der Nutzung des Integrationsobjektes eine vereinfachte Schnittstelle auf.[332] Im Gegensatz dazu stellt der – durch einen Integration Adapter verfügbar gemachte – Zugriffspunkt meist den Einzigen für sogenannte Alt-Anwendungen dar. Diese verfügen häufig nicht über Schnittstellen auf der Basis moderner Technologien.[333] Als Ziel wird dabei eine wiederverwendbare Schnittstelle verfolgt.

Dem Service Gateway[334] liegt die Intention zugrunde den Vertrag für die Nutzung eines Service von dessen Implementierung zu entkoppeln und so – durch den Einsatz mehrerer Service Gateway – multiple Verträge für ein und denselben Service zu erlauben.

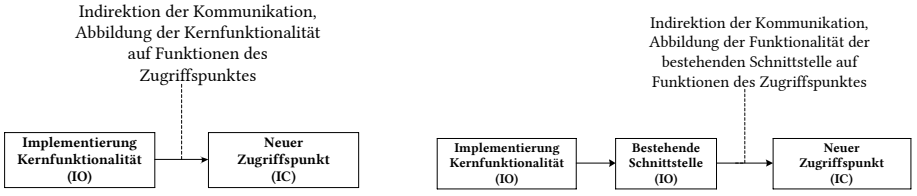
Einen schönen Vergleich erlaubt Erl[335], da dort sowohl Service Facade als auch Service Wrapper sowie ein File Gateway beschrieben werden. Wie bei dem durch Trowbridge u. a.[336] beschriebenen Service Gateway ist das Ziel der Service Facade[337] eine negative Kopplung von Kontrakt und Servicelogik zu vermeiden, um zu verhindern, dass sich bspw. Kontraktänderungen direkt auf die Servicelogik auswirken. Finden solche Änderungen bei Nutzung einer Service Facade statt, so können diese Änderungen durch Änderungen an der Facade kompensiert werden und die Servicelogik bleibt unangetastet. Der Legacy Wrapper[338] wird analog zu dem Integration Adapter nach Lutz beschrieben. Der hier zur Verfügung gestellte Zugriffspunkt erschließt explizit die Funktionalität einer Alt-Anwendung und vermeidet so Abhängigkeiten zwischen den Servicenutzern und der Alt-Anwendung. Das File Gateway stellt ebenfalls einen Zugriffspunkt zur Verfügung, in dem es eine Komponente mit Serviceschnittstelle umsetzt, die auf der anderen Seite als Input- und Output mit Dateien umgehen kann. Wiederum kann als gemeinsame funktionale Eigenschaft der Muster die Verfügbarmachung eines Zugriffspunktes abgeleitet werden.

In die vorangegangene Diskussion sind bei Weitem nicht alle bekannten Muster eingeflossen. Nobel und Biddle bezeichnen bspw. Decorator als Synonym für Wrapper[339], da sich diese Publikation nicht in der Grundgesamtheit befindet, wurde sie hier nicht berücksichtigt. Eine im Ausblick diskutierte Erweiterung der Mustermenge könnte dieses Muster jedoch berücksichtigen.

[328] Vgl. (Andersson und Johnson, 2001).
[329] (ebenda, S. 228)
[330] (ebenda, S. 229)
[331] Vgl. (Lutz, 2000).
[332] Vgl. (ebenda, S. 70).
[333] Vgl. (ebenda, S. 66f.).
[334] Vgl. (Trowbridge, 2003, S. 293ff.).
[335] Vgl. (Erl, 2009).
[336] Vgl. (Trowbridge u. a., 2004).
[337] Vgl. (Erl, 2009, S. 333ff.).
[338] Vgl. (ebenda, S. 441ff.).
[339] Vgl. (Noble und Biddle, 2006).

Im Folgenden wird nun das **grundlegende Konzept** der vorgenannten Muster beschrieben und die vernachlässigten Eigenschaften diskutiert.

Alle in dieser Arbeit vorgestellten Ausprägungen der Muster Adapter, Wrapper, Facade und Gateway sind Vertreter des Konstruktionsprinzips der Trennung von Zugriffspunkt und Implementierung. Dabei kann auch eine erneute Indirektion – das Vorschalten eines veränderten Zugriffspunktes vor einen anderen – erfolgen. Als Integrationskomponente wird von jedem Muster ein neuer Zugriffspunkt (Access Point) geschaffen. Dabei besteht die Möglichkeit, den neuen Zugriffspunkt nach den vorherrschenden Anforderungen zu gestalten. Es findet aber in jedem Fall eine Abbildung von Funktionalität der Implementierung auf die Funktionalität statt, welche durch den neuen Zugriffspunkt zur Verfügung gestellt wird. Liegen in der Implementierung und dem Zugriffspunkt z. B. identische Methoden Signaturen vor, so handelt es sich streng genommen dennoch um eine Abbildung, da es eigenständige Funktionen sind. Die nachfolgenden Abbildungen



(a) Direktzugriff auf Kernfunktionalität

(b) Indirektion eines Zugriffspunktes

Abbildung 5.8: Schema des Musters zusätzlicher Zugriffspunkt

Alle weiteren Veränderungen sind dann als Spezialfälle einer solchen Abbildung zu sehen. Die nachfolgende Abbildung 5.9 stellt die Spezialisierungen bzw. Verfeinerungen dieses Grundprinzipes dar.

Im Wesentlichen werden drei Eigenschaften weiter eingeschränkt, von denen das Muster Zugriffspunkt abstrahiert hat. Dabei handelt es sich um die *beteiligten Objekte*, die *Art der Abbildung* der Funktionalität und die *Art des Zugriffspunktes*. Die *beteiligten Objekte* betreffen das Ziel der Kommunikation und den Typ des Integrationsobjektes. Im Fall des Adapter Style Musters nach Andersson und Johnson[340] wird das Ziel der Kommunikation festgelegt, da das dort beschriebene Muster der Kopplung von Komponenten mit einem Mediator dient. Für eine Spezialisierung des *Typs des Integrationsobjektes* existieren drei Beispiele. Das File Gateway nach Erl[341] bestimmt Dateien zu den Integrationsobjekten. Im Gegensatz dazu ermöglichen die Wrapper Facade[342] und der Integration Adapter[343] die Integration von nicht Objektorientierten Integrationsobjekten. Die Art der Abbildung der Funktionalität einer bestehenden Schnittstelle oder der Implementierung der Funktionalität lässt sich anhand der untersuchten Beispiele in zwei Arten einteilen. Zum einen besteht eine Spezialisierung der Art der Abbildung darin, festzulegen, dass die Abbildung eine Vereinfachung erzielen muss. Diese Bestimmung trifft auf die Integration Facade[344] und den Legacy Wrapper[345] zu. Im Gegensatz dazu legen der Component Wrapper und der Legacy Wrapper fest, dass die Abbildung unter Beibehaltung der vollständigen Funktionalität erfolgen muss, wobei diese um zusätzliche Funktionen

[340] Vgl. (Andersson und Johnson, 2001).
[341] Vgl. (Erl, 2009).
[342] Vgl. (Schmidt; Stal u. a., 2000).
[343] Vgl. (Lutz, 2000).
[344] Vgl. (ebenda).
[345] Vgl. (Erl, 2009).

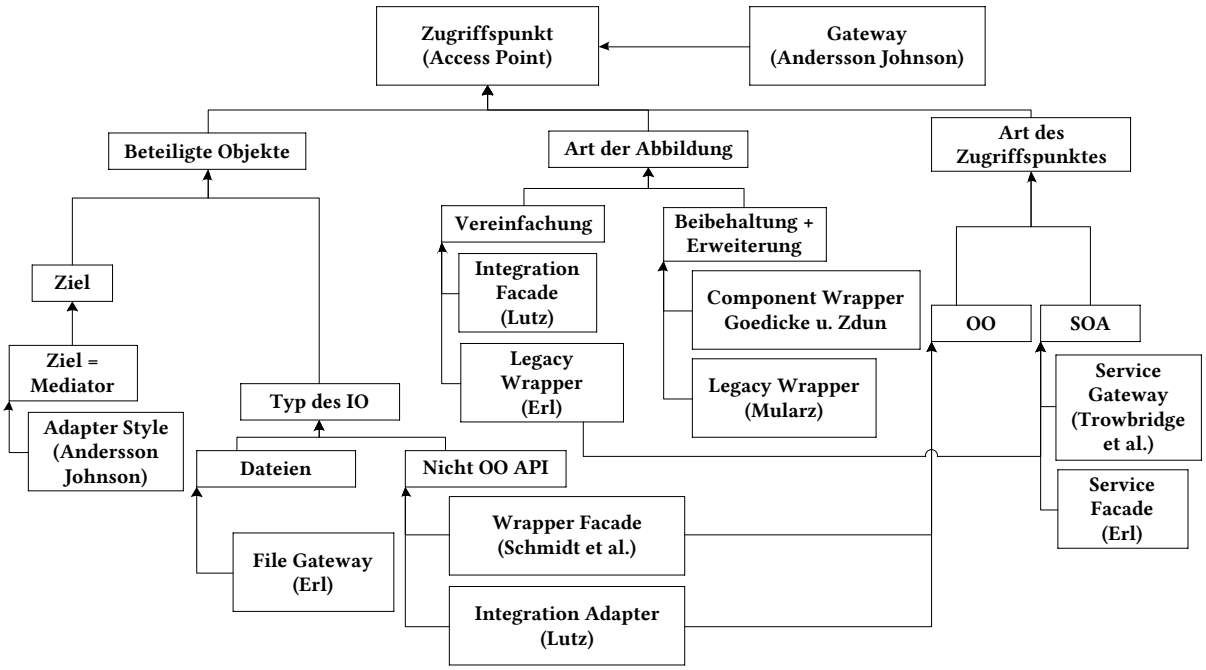


Abbildung 5.9: Spezialisierungen des Musters Zugriffspunkt

erweitert werden kann. Die Art des Zugriffspunktes legen das Service Gateway[346] und die Service Facade[347] fest. Beide Muster spezifizieren, dass der Zugriffspunkt den Paradigmen einer serviceorientierten Architektur gehorchen muss. Gleiches gilt für den bereits erwähnten Legacy Wrapper von Erl.[348] Damit handelt es sich bei diesem um das erste untersuchte Muster, welches zwei Eigenschaften spezialisiert. In die gleiche Kategorie gehören die Wrapper Facade[349] und der Integration Adapter[350], diese spezialisieren ebenfalls eine weitere Eigenschaft, nämlich die Art des Zugriffspunktes. Sie legen fest, dass es sich um eine Objekt-orientierte Schnittstelle handeln muss.

Alle anderen Eigenschaften bleiben von den Betrachtungen hier unberührt, sie werden vernachlässigt und so wird von diesen abstrahiert. Untersuchungen bezüglich der Abhängigkeiten von funktionalen Merkmalen können diese Aussage aufweichen. Erste Anzeichen dafür liefert die Spezialisierung der Art des Zugriffspunktes. Durch das Prinzip serviceorientierter Architekturen werden teilweise Eigenschaften der Kopplung und der Synchronität festgelegt bzw. eingeschränkt, was aber im Rahmen dieser Arbeit nicht betrachtet wird. In dem sich anschließenden Kapitel wird exemplarisch eine zweite Mustergruppe untersucht und aus dieser ebenfalls ein abstrakteres, höher stehendes Muster abgeleitet.

5.5.2 Broker, Mediator, Router, der Typ Vermittler

Wie bei der vorangegangenen Mustergruppe werden hier die Muster Broker, Mediator, Router und Messenger auf ein gemeinsames Grundkonzept hin untersucht, was einer Generalisierung dieser Muster und damit der Entwicklung eines abstrakteren Musters entspricht. Wiederum

[346] Vgl. (Trowbridge, 2003).
[347] Vgl. (Erl, 2009).
[348] Vgl. (ebenda).
[349] Vgl. (Schmidt; Stal u. a., 2000).
[350] Vgl. (Lutz, 2000).

existieren verschiedene – in dieser Arbeit bereits untersuchte – Beschreibungen dieser Muster, welche im Folgenden auf das gemeinsame Konzept hin untersucht werden.

Der Integration Messenger nach Lutz[351] kennt drei verschiedene Ausprägungen, womit bereits ein erster Ansatz zu einer Generalisierung gegeben ist. Alle drei Ausprägungen variieren den Kommunikationsmechanismus. Es handelt sich bei der ersten Variante um die Umsetzung eines 1:1, synchronen Kommunikationsverhaltens im Request-Reply-Verfahren, welche Remote-Method-Invocation verwendet. Die zweite Variante setzt auf Message-Queuing und kann deshalb einen asynchronen 1:1-Kommunikationsmechanismus umsetzen. Die dritte Variante bricht die bisher realisierte 1:1-Kardinalität auf und erzielt eine asynchrone 1:n-Kommunikation in dem ein Publish-Subscribe-Mechanismus verwendet wird. Gemeinsam ist allen drei Varianten jedoch, dass sie eine zentrale Integrationskomponente zur Verfügung stellen, welche dabei hilft die Kommunikationspartner räumlich voneinander zu entkoppeln und damit die Kommunikationsabhängigkeiten unter den Integrationsobjekten zu minimieren.[352] Dabei bleibt jedes Integrationsobjekt für die Interaktionslogik selbst verantwortlich und muss also die Interaktionsauslösung selbst initiieren.

Der Integration-Mediator[353] des gleichen Autors setzt ebenfalls auf die Entwicklung einer zentralen Integrationskomponente, welche die Kommunikationsabhängigkeiten minimiert. Im Gegensatz zu dem Integration-Messenger ist dessen Umsetzung jedoch nicht mehr unabhängig von den Integrationsobjekten, da der wesentliche Unterschied in der Entkopplung und Zentralisierung der Interaktionslogik besteht.[354] Das Grundprinzip des Integration-Messenger wurde also um die Zentralisierung der Interaktionslogik erweitert.

Der Mediator nach Gamma[355] stellt in einem gewissen Sinne den Prototypen aller anderen Muster dieser Gruppe dar. Da dieses Muster auf der Abstraktionsebene von Entwurfsmustern angesiedelt ist, sind die Objekte – im Sinne der objektorientierten Entwicklung – als die Integrationsobjekte anzusehen. Im Zentrum steht die räumliche Entkopplung dieser Objekte. „[...] in the worst case, every object ends up knowing about every other“[356]. Damit ist die Gemeinsamkeit mit den bereits angeführten Mustern identifiziert auch hier wird eine Zentrale Komponenten geschaffen, welche die Kommunikationsabhängigkeiten minimiert. Da der Mediator von Buschmann u. a.[357] eine weitgehende Wiedergabe des Musters nach Gamma darstellt, fügt diese erneute Beschreibung keine weiteren Erkenntnisse zu der Analyse hinzu.

Der Message Router, bei Buschmann et al. beschrieben[358], ist auf der Ebene eines Architekturmusters angesiedelt. Auch hier wird eine zentrale Integrationskomponente verwendet, welche auf der Basis von Bedingungen die Nachrichten eines Nachrichtenkanals auf verschiedene Kanäle verteilt. Das Grundprinzip ist damit wiederum gegeben. Die Musterbeschreibung nach Buschmann et al. ist weitgehend identisch mit der des Message Router nach Hohpe und Woolf[359], welche das gleiche Prinzip propagiert.

Die Beschreibung des Mediator Style nach Andersson und Johnson[360] erweitert die funktionalen Eigenschaften über das Routing – also die räumliche Entkopplung der Kommunikationspartner hinaus – um Lastverteilungs- und Transformationsfunktionalität. Der Message Router[361] wird

[351] Vgl. (Lutz, 2000).
[352] Vgl. (ebenda, S. 68).
[353] Vgl. (ebenda).
[354] Vgl. (ebenda, S. 71).
[355] Vgl. (Gamma u. a., 1995).
[356] (ebenda, S. 273)
[357] Vgl. (Buschmann; Henney und Schmidt, 2007a, S. 411).
[358] Vgl. (ebenda, S. 232).
[359] Vgl. (Hohpe und Woolf, 2003, S. 81).
[360] Vgl. (Andersson und Johnson, 2001, S. 230).
[361] Vgl. (ebenda, S. 231).

explizit als eine Anwendung des Mediator Style gekennzeichnet: „The message router is a hub and peer-to-peer mediator style with a central component that directs information between the base components based on the source of the information. All internal component communication is routed through the central component“[362].

Ein Muster mit dem Namen Broker wird durch Trowbridge[363] beschrieben. Es handelt sich dabei um eine Wiedergabe desselben Prinzips, welches auch von Buschmann u. a. mit dem Broker-Muster beschrieben wurde.[364] Beide Muster zielen auf die Kapselung von Kommunikationsdetails ab und versuchen entfernte Methodenaufrufe so zu kapseln, dass der Client den Server nicht kennen muss.

Dieses Prinzip wird in Form des Direct Broker[365] und des Indirect Broker[366] ebenfalls verwendet. Wobei hier jedoch unterschieden wird, ob nur eine Vermittlung des Partners (Direct Broker) stattfindet oder aber die dauerhafte Vermittlung der Kommunikation (Indirect Broker). Bei Ersterem findet die eigentliche Kommunikation dann direkt zwischen den Integrationsobjekten statt.

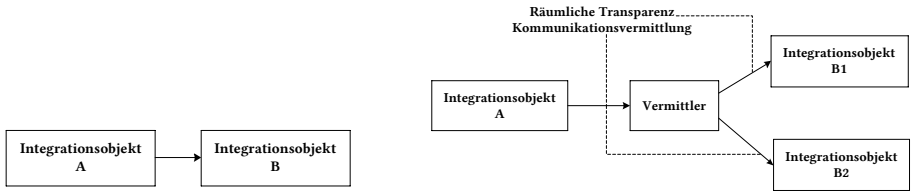
Auch die als Broker Revisited bezeichnete erneute Beschreibung [367] fügt keine weitere Sichtweise zu den bisherigen Beschreibungen hinzu. Die Veränderungen sind in der Umsetzung und der Verteilung der Zuständigkeiten einzelner Teilkomponenten des Musters zu finden. Diese Veränderungen beeinträchtigen das grundsätzliche Prinzip des Musters jedoch nicht.

Erl beschreibt einen UI Mediator[368], welcher das bisher identifizierte Prinzip der räumlichen Entkopplung der Kommunikationsteilnehmer auf Benutzerschnittstellen anwendet. Dabei wird aber zusätzlich das Ziel verfolgt, eine Integrationskomponente zu schaffen, die dem Nutzer eine Statusmitteilung anzeigen kann, obwohl die angestoßene Verarbeitung noch nicht beendet ist.

Mit dem Service Broker (COMP)[369] stellt Erl ein sogenanntes zusammengesetztes Muster (Compound) vor.[370] Es besteht aus den zusammengesetzten Mustern Data Model Transformation, Data Format Transformation und Protocol Bridging, bei denen es sich ausnahmslos um Transformationsmuster handelt. In einer freien Interpretation kann auch dieses Muster als die Umsetzung des Prinzips der räumlichen Entkopplung der Kommunikationspartner bezeichnet werden, sofern man die Transformationsmuster als eine Festlegung eines Teils der Integrationsobjekte auffasst und der Service Broker sicherstellt, dass diese auch erreicht werden können. Diese Ansicht findet sich jedoch so nicht direkt in den Ausführungen von Erl, weshalb dieses Muster im Vergleich zu den bisher Untersuchten, aus dem Rahmen fällt.

Es bleibt festzuhalten, dass das **grundlegende Konzept** der in diesem Kapitel analysierten Muster die Schaffung einer Integrationskomponente ist, welche als Architekturelement die Rolle des Kommunikationsvermittlers übernimmt. Das verfolgte Ziel besteht darin, die Kommunikation an ihr Ziel zu bringen und eine räumliche Entkopplung der Integrationsobjekte zu ermöglichen, sodass diese wechselseitig keine Kenntnis voneinander haben müssen. Die Abbildung 5.10(a) zeigt die Notwendigkeit der wechselseitigen Kenntnis der Integrationsobjekte, wenn kein Vermittler verwendet wird. In der Abbildung 5.10(b) wird deutlich, dass das Integrationsobjekt A nun nur

[362](ebenda, S. 231)
[363]Vgl. (Trowbridge, 2003, S. 207f.).
[364]Vgl. (Buschmann; Meunier u. a., 1996, S. 119ff.).
[365]Vgl. (Trowbridge u. a., 2004, S. 155, 215ff.).
[366]Vgl. (ebenda, S. 221).
[367]Vgl. (Kircher; Voelter u. a., 2004, S. 581).
[368]Vgl. (Erl, 2009, S. 366).
[369]Vgl. (ebenda, S. 707).
[370]Im Unterschied zu einem Composite handelt es sich bei einem Compund um die gemeinsame Verwendung der enthaltenen Muster und nicht um deren Verbindung zu einem komplexeren Muster. Vgl. (ebenda, S. 698).



(a) Punkt-zu-Punkt Verbindung (b) Vermittlung der Kommunikation

Abbildung 5.10: Grundsätzliches Prinzip eines Vermittlers

noch Kenntnis über den Vermittler benötigt, nicht aber über die anderen Integrationsobjekte, was zu der räumlichen Transparenz führt. Die Unkenntnis des Kommunikationspartners ermöglicht nun verschiedene andere Lösungen, welche in den Spezialisierungen des Grundmusters ausgeführt werden.

Durch diese Generalisierung wurden einige, bereits in den Beschreibungen angeführten, Merkmale ausgeblendet. Im Folgenden werden diese wieder als Spezialisierungen des Grundkonzeptes dargestellt und so aufgezeigt, dass ein abstrakteres, hochstehendes Muster identifiziert wurde. Die Abbildung 5.11 veranschaulicht die einzelnen Eigenschaften, die spezialisiert wurden und die zugehörigen detaillierteren Muster.

Als Spezialisierungen des Musters Vermittler lassen sich Einschränkungen der Eigenschaften Kommunikationsrichtung, Kommunikationsform, weitere Funktionalität, Art der Integrationsobjekte, Art der Vermittlung, Ermittlung des Kommunikationspartners und Zentralisierung der Interaktionslogik identifizieren. Die Eigenschaft Kommunikationsrichtung begrenzt ob alle Integrationsobjekte, die mithilfe dieses Vermittlers verbunden werden, gleichermaßen Initiator und Empfänger einer Kommunikation sein können. Der Fall Hub Mediator[371] nach Andersson und Johnson stellt dabei den allgemeinen uneingeschränkten Fall dar, in dem jeder mit jedem kommunizieren kann. Eine in der Abbildung 5.11 nicht dargestellte Eigenschaft dieses Musters ist die, dass es sich gleichzeitig auch um eine dauerhafte Vermittlung handelt. Eine Einschränkung dieser Eigenschaft bildet der Peer-to-peer Mediator[372] hier werden eins zu eins Kommunikationspaare festgelegt. Der Middle-tier Mediator[373] schränkt dabei noch die Kommunikationsinitiierung ein, welche nur von Clients ausgeht und an Server gerichtet ist.

Eine Eigenschaft, die ebenfalls eingeschränkt werden kann, ist die Kommunikationsform. In der untersuchten Menge an Mustern sind die Verfahren Publish-Subscribe, Message-Queuing und Request-Reply enthalten. Der Integration Messenger[374] kennt Ausprägungen für alle drei Kommunikationsformen. Darüber hinaus ist der Message Router nach Andersson und Johnsson ebenfalls ein Vertreter des Message-Queuing, ebenso wie die Muster Message Router nach Buschmann[375] und Message Router nach Hohpe und Woolf[376]. Bei den beiden Letztgenannten handelt es sich um die einzigen Muster in Abbildung 5.11, bei denen die Spezialisierung von zwei Eigenschaften in der Abbildung gekennzeichnet wurde. Aus Gründen der Übersichtlichkeit musste dies für alle anderen Muster unterlassen werden. Diese beiden Muster stellen Beispiele für die Spezialisierung Art der Ermittlung des Kommunikationspartners dar. Dies geschieht in diesen Fällen auf der Basis von Bedingungen und ermöglicht so ein situativ angepasstes

[371] Vgl. (Andersson und Johnson, 2001).
[372] Vgl. (ebenda).
[373] Vgl. (ebenda).
[374] Vgl. (Lutz, 2000).
[375] Vgl. (Buschmann; Henney und Schmidt, 2007a, S. 232).
[376] Vgl. (Hohpe und Woolf, 2003, S. 81).

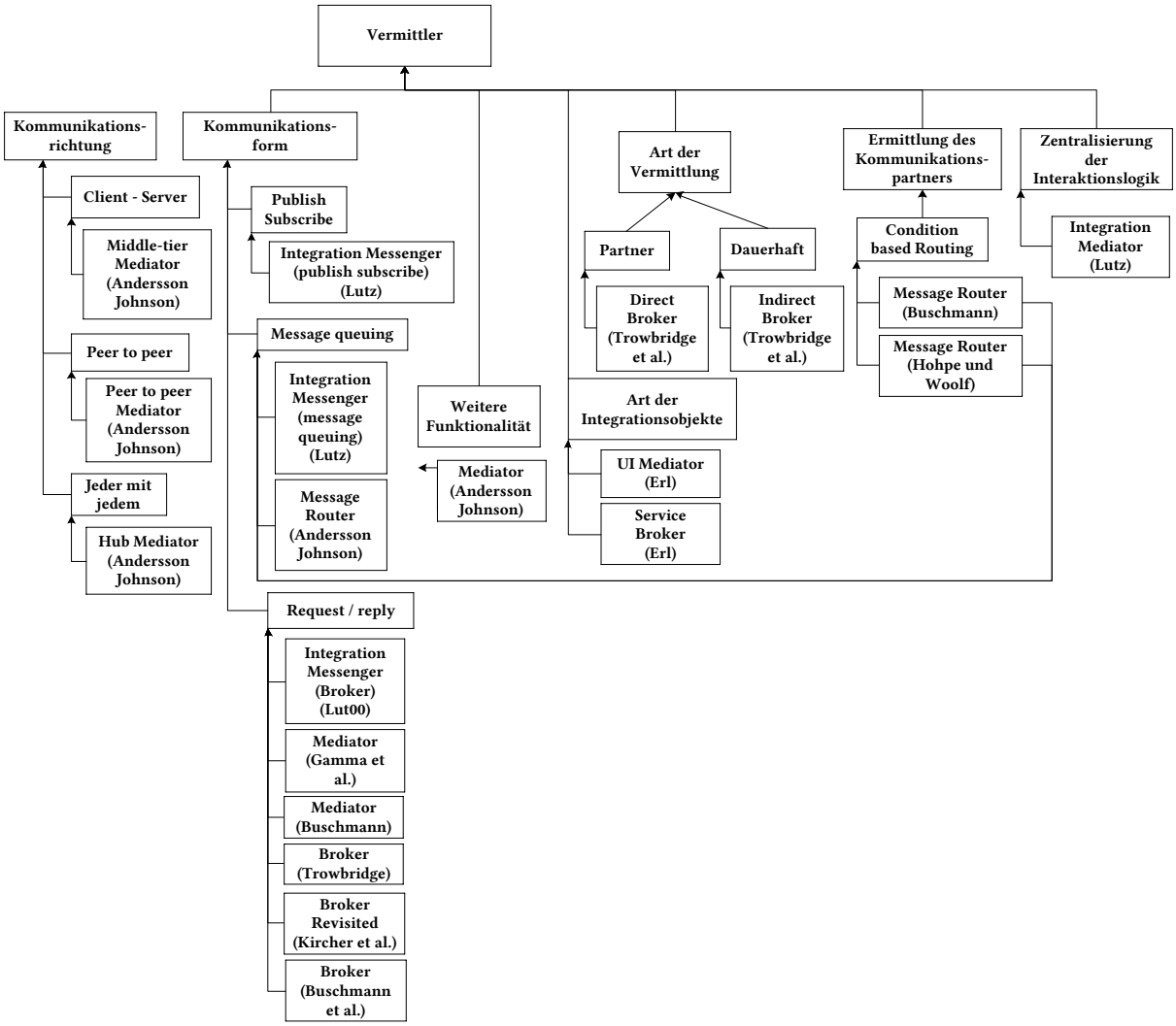


Abbildung 5.11: Spezialisierungen des Musters Vermittler

Routing-Verhalten. Für die letzte Kommunikationsart existieren die meisten Vertreter. Für eine Spezialisierung auf Request-Reply sind der Integration Messenger (Broker)[377], der Mediator[378], der Mediator[379], Broker[380], Broker Revisited[381] und der Broker[382] bekannt.

Weitere Funktionalität zu integrieren ist die Spezialisierung des Musters Mediator nach Andersson und Johnson.[383] Dabei kann es sich sowohl um Transformations- als auch Lastverteilungsfunktionalität handeln.

Alle beiden Muster von Erl, welche in diese Untersuchung eingeflossen sind, legen die Art der Integrationsobjekte fest. Zum einen Benutzerschnittstellen in Form des UI Mediator[384] und zum anderen in Form von Transformationsservices mit dem Service Broker[385]. Selbstverständlich sind beide Muster auch in der Eigenschaft Kommunikationsform auf nachrichtenbasierte Kommunikation spezialisiert.

Bei der Art der Vermittlung lassen sich die Partnervermittlung und die dauerhafte Vermittlung unterscheiden. Die Partnervermittlung vermittelt nur den Kommunikationspartner, kontrolliert aber die Kommunikation danach nicht mehr z. B. in Form des Direct Broker.[386] Die Alternative ist eine dauerhafte Vermittlung, bei der jeder Teil der Kommunikation durch den Vermittler läuft, z. B. im Fall des Indirect Broker.[387]

Als letzte und abschließende Eigenschaft verbleibt die der Zentralisierung der Interaktionslogik. Alle bisher genannten Muster sind Vertreter der Ausprägung verteilt, bei der jedes Integrationsobjekt selbst für die passgenaue Auslösung der Interaktion zuständig ist. Im Gegensatz zu dieser Ausprägungsart entkoppelt und zentralisiert der Integration Mediator die Interaktionslogik in der Integrationskomponente.[388]

Mit diesen Ausführungen wurde nachgewiesen, dass sich alle untersuchten Muster als Spezialisierungen des Musters Vermittler auffassen lassen, weshalb es sich bei diesem um ein neues hochstehendes Integrationsmuster handelt.

5.5.3 Zusammenfassung und Weiterführung der Generalisierung

Streng genommen wurde mit der vorliegenden Generalisierung einen Teil der Funktion und die Art der durch die Muster eingeführten Integrationskomponenten erfasst und so die Grundlage für eine weitere Facette gelegt. Sind für ein Merkmal verschiedene Ausprägungen (Fokusse) vorhanden, lässt sich daraus eine Art Featuremodell erstellen, welches eine sukzessive Auswahl und schrittweise Verfeinerung ermöglicht.

Immer dann, wenn eine Entkopplung der Kommunikationsteilnehmer erreicht wurde, sei es durch die Indirektion bei der Trennung von Implementierung und Zugriffspunkt oder durch die Entkopplung der Kommunikationspartner, können diese Integrationskomponenten für eine weitere Zentralisierung von Funktionalität, sowie eine Durchführung von Erweiterungen etc.

[377] Vgl. (Lutz, 2000).
[378] Vgl. (Gamma u. a., 1995, S. 277).
[379] Vgl. (Buschmann; Henney und Schmidt, 2007a, S. 411).
[380] Vgl. (Trowbridge, 2003, S. 207f.).
[381] Vgl. (Kircher; Voelter u. a., 2004, S. 581).
[382] Vgl. (Buschmann; Meunier u. a., 1996, S. 119ff.).
[383] Vgl. (Andersson und Johnson, 2001, S. 230).
[384] Vgl. (Erl, 2009, S. 366).
[385] Vgl. (ebenda, S. 707).
[386] Vgl. (Trowbridge u. a., 2004, S. 155, 215ff.).
[387] Vgl. (ebenda, S. 221).
[388] Vgl. (Lutz, 2000, S. 71).

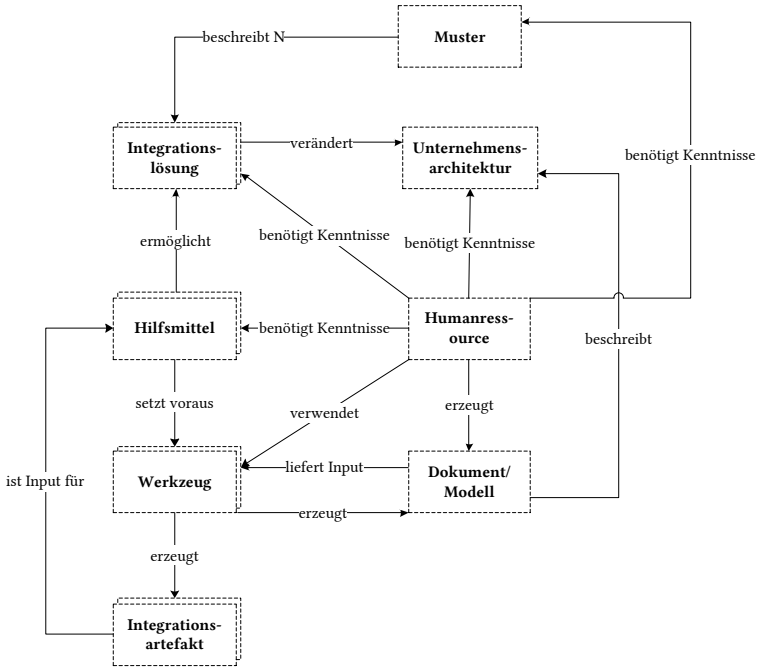


Abbildung 5.12: Integrationsdienstleistungen und Muster

eingesetzt werden. Als Voraussetzung ist dabei die neu geschaffene Integrationskomponente zu nennen, da erst durch diese ein Architekturelement geschaffen wird, in dem diese Funktionen angesiedelt werden können, ohne die Integrationsobjekte verändern zu müssen.

Durch die exemplarische Generalisierung eines Teils der untersuchten Muster in den beiden vorangegangenen Kapiteln ist die Überzeugung entstanden, dass sich Integrationsmuster in Basisbausteine zerlegen lassen, von denen eine begrenzte Anzahl existiert, mit deren Hilfe alle Integrationsmuster als Kombination dieser Bausteine ausdrückbar sind. Die weiteren vermuteten Bausteine werden in Kapitel 8 diskutiert.

Darüber hinaus sind in Kapitel8 auch weitere Mechanismen enthalten die geeignet sind, um höhere Integrationsmuster zu bilden, die hier bisher nicht genutzt wurden. Ein Beispiel ist der Mechanismus der Komposition.

Im Anschluss wird untersucht, ob die Musternutzung in bestehenden Vorgehen der Integration berücksichtigt wird, bevor die Ergebnisse in einem Anwendungsfall in Kapitel 6 evaluiert werden.

5.6 Berücksichtigung von Integrationsmustern in Vorgehen und Vorgehensmodellen

Die Auswirkungen der Verwendung von Mustern können anhand der folgenden Abbildung dargelegt werden. Diese positioniert die Muster im Zusammenhang mit den Ressourcentypen, welche für die Integration notwendig sind.

Wie bereits dargelegt sind Humanressourcen der zentrale Ressourcentyp bei Integrationsdienstleistungen, da es sich um eine Dienstleistung mit personeller Dienstleistungsproduktion handelt. Muster stellen abstraktes Lösungswissen zur Verfügung, welches sich für mehrere Integrationslösungen eignet, da es von Werkzeugtypen und konkreten Hilfsmitteln abstrahiert. Können nun die

mit der Integrationslösung betrauten Mitarbeiter auf abstrahiertes Lösungswissen in Form von Integrationsmustern zurückgreifen, so vereinfacht sich die Lösungserstellung. Die Humanressource als zentrales Element der Dienstleistung erlernt so Lösungskonzepte, welche in sehr vielen Kontexten verwendbar ist. Damit kann eine Effizienzsteigerung erzielt werden, wobei sich gleichzeitig die Qualität der Lösung sowie deren identische Reproduzierbarkeit erhöht.

In der Literatur sind einige Publikationen zu finden, die sich mit der Durchführung von Integrierungsaufgaben beschäftigen. Dabei sind Vorgehensmodelle und Schrittfolgen voneinander zu unterscheiden. Vorgehensmodelle legen detailliert fest, was, wann, wie, von wem auszuführen ist. Eine Schrittfolge enthält zumeist nur das was und vernachlässigt die anderen Dimensionen. Die folgende Tabelle gibt einen Überblick über die betrachteten Werke. Im Anschluss daran werden diese miteinander verglichen, mit dem Ziel aufzuzeigen inwieweit der Einsatz von Mustern Berücksichtigung findet.

Literaturquelle	Art
(Lam und Shankararaman, 2004)	Schrittfolge
(Erasala; Yen und Rajkumar, 2003)	Schrittfolge
(Schwinn, 2005)	Vorgehen
(Tiedemann, 2006)	Vorgehen
(Vogler, 2006)	Methoden
(Neubauer, 2007)	Vorgehensmodell[389]
(Stadlbauer, 2007)	Vorgehensmodell
(Engels; Hess u. a., 2008)	Schrittfolge
(Thränert, 2008)	Vorgehensmodell

Tabelle 5.46: Schrittfolgen und Vorgehensmodelle der Integrierung

Lam und Shankararaman[390] haben sich mit dem Prozess der Unternehmensintegration, den Ergebnissen und den Risiken auseinandergesetzt, wobei die Autoren nicht auf eine einzige Art der Unternehmensintegration abstellen. Enterprise-Integration oder auch Unternehmensintegration kann neben der Legacy-System-Integration auch Enterprise-Application-Integration, sowie unternehmensübergreifende Integration (B2B) umfassen. Ein Integrationsprozess unterteilt sich nach den Autoren in die in Tabelle 5.47 dargestellten Phasen.

Prozessschritt	Ergebnisse	Risisken
Understand the end-to-end business process	Usage scenarios	Integration does not reflect business process Unclear business need for integration
plan the integration	Integration actions and plans	unestimated work items Chaotic project execution
produce the architecture	Integration Architecture Feasibility studies	Performance issues Integration showstoppers
Derive the requirements	Integration requirements	Inappropriate or inadequate integration architecture
Map the process onto components	Integration components	Gaps in integration Inefficiencies and redundancies in integration flows

Tabelle 5.47: Schritte nach Lam und Shankararaman

An den Phasen ist kritisch zu beurteilen, dass sie keinen Schritt beinhalten, in dem die Verknüpfung der bestehenden Geschäftsprozesse mit den eingesetzten Systemen analysiert wird. Geht man davon aus, dass Integration keine einmalige Aufgabe ist und nie eine Veränderung aller

[389] Vorgehen und Vorgehensmodell werden hier synonym gebraucht.
[390] Vgl. (Lam und Shankararaman, 2004).

Teilkomponenten möglich ist, so wird in nahezu allen Fällen nur ein Teilprozess umstrukturiert und zum Teil auf IT-Systeme abgebildet. Eine Untersuchung der bestehenden IT-Landschaft und die Bestimmung möglicher Auswirkungen sind unumgänglich, bevor eine Entscheidung zugunsten einer konkreten Lösung gefällt wird. Dabei sind die Kosten der IT[391], der damit verbundenen Organisationsentwicklung und die erreichten Verbesserungen im Prozessablauf abzuwägen.

Ebenso wie Lam und Shankararaman führen Erasala u. a. einzelne Schritte, die sie als Basisschritte bezeichnen, zur Erstellung einer EAI-Lösung an. Defition und Katalogisierung von:

- Geschäftsereignissen,
- Geschäftsprozessregeln und Nachrichten,
- IT-Komponenten,
- Nachrichteninhalten.[392]

Die Definition und Katalogisierung der Geschäftsprozesse dient dabei der Identifikation des Geltungsbereiches und hilft bei der Aufnahme von Anforderungen an die Integrationslösung. Geschäftsereignisse sind aus Sicht der Autoren wichtig, da sie Geschäftsprozesse anstoßen. Aus diesem Grund sollten der Inhalt und das Format der Ereignisse definiert werden. In gängigen Unternehmensarchitekturframeworks findet man eine Ebenenunterscheidung zwischen der Geschäftsebene, einer logischen Sicht auf die Applikationskomponenten und einer physischen Sicht auf die tatsächliche Verteilung. Erasala u. a. sprechen lediglich von IT-Komponenten und unterscheiden nicht zwischen Applikationen und Infrastruktur. Die systematische Aufnahme der IT-Komponenten führt aus ihrer Ansicht zu einer Auflistung der Komponenten von denen die Geschäftsprozesse abhängen. Auf Basis dieser Analyse lassen sich Aussagen über fehlende oder zu entwickelnde Komponenten treffen. Geschäftsregeln definieren die Sequenz des Aufrufs von IT-Komponenten. Auf dieser Basis kann die Ablauffolge zwischen den IT-Komponenten bestimmt werden und damit auch die Ablauffolge zwischen IT-Komponente und Integrationslösung.[393] Die Nachrichteninhalte legen die erwarteten Inhalte und Formate für jede Komponente fest, für die diese Nachricht von Bedeutung ist. Der Nachrichtenanalyse kommt darüber hinaus Bedeutung zu um Gemeinsamkeiten in den Formaten zu identifizieren, wann immer es möglich ist. Die Ausführungen der Autoren lassen sich dem Problemraum zuordnen, da sie notwendige Informationen umreißen, auf deren Basis eine Integrationslösung ausgewählt werden kann. Die bei Lam und Shankararaman angebrachte Kritik lässt sich nicht wiederholen, da im Schritt IT-Components explizit darauf hingewiesen wird die Komponenten zu erheben, von denen ein Geschäftsprozess abhängt. Einen Entwurf auf der Basis wiederverwendbaren Lösungswissens sucht man vergebens.

Neubauer[394] siedelt seine Arbeit im Themenfeld informationstechnischer Systeme zur Dokumentation im Operationssaal an. Bestandteil der Arbeit ist unter anderem die Herleitung eines Vorgehensmodells zur Systemintegration in diesem Bereich. Um sich der Thematik zu nähern, vergleicht Neubauer zunächst bestehende Modelle. In seinen Vergleich fließen das Wasserfallmodell, das V-Modell, das Spiralmodell, der Systems-Engineering-Ansatz und der DMAIC-Zyklus ein. Neubauer kommt zu dem Schluss, dass diese Modelle sich nicht direkt auf das Problem der Systemintegration übertragen lassen.[395] Das daraufhin definierte Vorgehensmodelle beinhaltet 5 Projektphasen welche sich an den Phasen des DMAIC-Zyklus orientieren. Muster finden keinen direkten Eingang in das Modell.

[391] Kosten der Einführung und die Kosten des Betriebs der Lösung
[392] Vgl. (Erasala; Yen und Rajkumar, 2003).
[393] Die Autoren zielen Explizit auf EAI Lösungen ab, die in die Klasse der Kopplungssysteme fallen.
[394] Vgl. (Neubauer, 2007).
[395] Vgl. (ebenda, S. 62).

Mit dem Werk „Quasar Enterprise Anwendungslandschaften serviceorientiert gestalten“[396] – im Folgenden kurz Quasar genannt – liegt ein weiteres Werk aus der Unternehmenspraxis – diesmal von Capgemini – vor, in dem eine Referenzarchitektur beschrieben wird. Quasar ist wesentlich komplexer als der hier vorgestellte Teil, da es die serviceorientierte Gestaltung von Anwendungslandschaften zum Ziel hat. Neben Integrationslösungen spielt dabei auch die Auswahl und Gestaltung von Anwendungskomponenten eine Rolle. Diese Betrachtungen werden jedoch ausgeblendet und nur die integrationsrelevanten Teile betrachtet.

Neben der Referenzarchitektur für die serviceorientierte Gestaltung von Anwendungslandschaften werden in dem Buch zwei Methoden beschrieben, die relevant sind. Zum einen ist dies eine Methode zur Gestaltung von Integrationsarchitekturen und zum anderen eine Methode zur Definition und zum Aufbau einer Integrationsplattform. Eine existierende Integrationsplattform bildet dabei die Voraussetzung für die Anwendung der Methode zur Gestaltung der Integrationsarchitektur. Die Gestaltung der Integrationsarchitektur wird von den Autoren als Teilschritt der Gestaltung der Anwendungslandschaft gesehen. Input ist dabei die Ist-Anwendungslandschaft sowie die Soll-Anwendungslandschaft mit Komponenten, Schnittstellen und der Kopplungsarchitektur.

Die Autoren geben an, dass die Methode iterativ zu durchlaufen ist. Dabei werden keine Aussagen über Rückschritte gemacht. So könnte eine Festlegung der verbleibenden Schnittstellen ebenfalls durch Musteranwendung weiter verfeinert werden. Da im Rahmen eines Integrationsprojektes auch – um im Sprachgebrauch der Quasar Autoren zu verbleiben – über die Erweiterung, Ergänzung, Neueinführung einer Integrationsplattform oder Teilen dieser entschieden wird, muss im Folgenden auch der Teil der Quasar Methode betrachtet werden, der sich mit der Gestaltung von Integrationsplattformen beschäftigt. Das von den Autoren angeführte Beispiel kann nicht als typisch für eine Integration im Sinne dieser Arbeit eingestuft werden. Wie in der Abbildung auf Seite 217 deutlich zu sehen ist, handelt es sich bei 5 von 6 Komponenten der Anwendungslandschaft um neu zu gestaltende. Nach gemachter Erfahrung ist eine Integrationssituation typischerweise durch den umgekehrten Umstand geprägt, dass die existierenden Komponenten überwiegen. Unter Umständen ist diese Kritik an der Stelle auch unberechtigt, da es sich um logische AL-Komponenten handelt über deren physische Realisierung entschieden wird.

Im weiteren werden die einzelnen Schritte der Methodik anhand des Beispiels vorgestellt. Dabei führen die Autoren im Schritt *Integrationsmuster anwenden* aus, dass lediglich das Muster zur Realisierung einer Prozesskomponente in Anwendung kommt. Im Schritt *Gestaltung der physischen Kopplung* hingegen wird angeführt, dass für die Gestaltung der Kopplung zwischen 2 Systemen ein UTM-Adapter zum Einsatz kommen muss, um eine ESB Schnittstelle zur Verfügung zu stellen. Weiterhin werden asynchrones Messaging und synchrone RPC Aufrufe angeführt. Die im Rahmen dieser Arbeit analysierten Integrationsmuster weisen jedoch ebenso Adapter, Messaging und RPC als Muster aus, sodass es auch im Schritt Gestaltung der physischen Kopplung zur Anwendung von Integrationsmustern kommt. Darüber hinaus weisen die Autoren in Kapitel 6.3 – in dem Integrationsmuster vorgestellt werden – selbst auf Werke als Quellen hin, die zum einen dieser Arbeit hier zugrunde liegen und zum anderen Adapter u.ä. Muster als Integrationsmuster ausweisen.[397] Daraus können 2 Schlussfolgerungen abgeleitet werden. Erstens, das Verständnis der Autoren über Integrationsmuster unterscheidet sich grundlegend von dem welches dieser Arbeit zugrunde liegt. Zweitens, es gibt Integrationsmuster auf unterschiedlichen Abstraktionsniveaus. Einmal auf dem Niveau *Realisierung einer Prozesskomponente durch Orchestrierung* und einmal auf dem Abstraktionsniveau *UTM-Adapter*. Die Quasar Autoren werten nur das höhere Abstraktionsniveau als Integrationsmuster.

[396] (Engels; Hess u. a., 2008)
[397] Vgl. (ebenda, S. 221).

Bis zum Schritt der Finalisierung haben die Autoren lediglich die geplante Ausgestaltung der Integrationsbeziehungen festgelegt. Bis zu diesem Zeitpunk wurde nicht geprüft, welche technischen Services dafür notwendig sind und ob diese in der existierenden Integrationsplattform existieren bzw. es geplant, wünschenswert, ökonomisch sinnvoll ist diese in die existierende Integrationsplattform zu integrieren. Der Schritt Finalisierung entspricht genau diesen Prüfungen. Zum einen wird gegen die Referenzarchitektur abgeglichen und zum anderen festgelegt, welche technischen Services genutzt werden. Darüber hinaus werden aber auch Fragen der Sicherheit festgelegt. So wird angeführt, dass die technischen Services der Integrationsplattform nur für eine Komponente zum Einsatz kommen und für die übrigen festgelegt wurde, dass diese sich auf die Authentifizierung des Nutzers gegenüber der Komponente verlassen bzw. erweiterte Mechanismen innerhalb ihrer Fachlogik umsetzen. Aus Sicht der hier vertretenen Auffassung sollten diese Entscheidungen nicht an dieser Stelle getroffen werden, sondern bereits in einem vorgelagerten Schritt als Anforderungen an logische AL-Komponenten festgelegt werden, sodass hier nur eine technische Detaillierung notwendig ist. Dies sollte aber ebenso nicht im Schritt Finalisierung stattfinden, sondern kann z. B. durch die Anwendung von Sicherheitsmustern bereits bei der Anwendung von Integrationsmustern stattfinden. Damit bleibt festzuhalten, dass die Anwendung von Integrationsmustern in der vorgeschlagenen Methodik zwar prinzipiell Beachtung findet, aber nur unzureichend und unvollständig umgesetzt wurde.

Thränert erarbeitet ein Vorgehensmodell für die Integrierung, dabei werden 4 Phasen Analyse, Planung, Umsetzung und Übergabe unterschieden. Das Vorgehensmodell ist umfangreich und unterscheidet neben einer Haupt- und Mittelebene auch eine Detailebene. Die Haupt- und Mittelebene liefern für die Identifikation von Schritten – die ein Framework unterstützen sollte – keine geeigneten Ansatzpunkte, da sie zwar spezifisch für die Entwicklung eines Softwaresystems aber nicht integrationsspezifisch sind. Die folgende Aufzählung listet die Schritte der Haupt- und Mittelebene kurz auf:

- Initialisierung,
- Iterieren,
  - Umsetzungsplanung,
  - Umsetzung,
  - Evaluierung,
  - Freigabe,
- Abbruch,
- Abschluss.

Die Aktivitätstypen und Methodentypen der Haupt- und Mittelebene würde man auch in einem Vorgehensmodell zur iterativen Softwareentwicklung erwarten. Dies ist nicht weiter verwunderlich, da ein Großteil eines Integrationsprojektes sich aus Tätigkeiten der Softwareentwicklung zusammensetzt. Die Betrachtungen beschränken sich deshalb auf die Detailebene. Diese gliedert sich in vier Aufgabenbereiche: Problemverständnis schaffen und Erwartungen abgleichen, technische Integrationslösung erarbeiten, Testfälle entwickeln, Wiederverwendung ermöglichen.

Stadlbauer hat in seiner Arbeit ein Vorgehen für die Integrierung entwickelt, welches hauptsächlich auf Unternehmensnetzwerke abzielt. Die betrachteten Integrationsobjekte sind Anwendungssysteme, jedoch wird die Einbindung in unternehmensübergreifende Abläufe untersucht. Dabei stehen die Identifikation von Integrationsbereichen, die Konzeption und Bewertung der Integrationslösung sowie eine inkrementelle Vorgehensweise unter der Beachtung von Abbruchstellen im

Vordergrund.[398] Eigenschaften, die dieses Vorgehen vergleichbar mit dem von Thränert machen, da dort ähnliche Rahmenbedingungen gelten.

Zusammenfassend kann das Fazit gezogen werden, dass Integrationsmuster in bekannten Vorgehen und Vorgehensmodellen nur ungenügende Beachtung finden und sich aus diesem Blickwinkel leicht erklären lässt, warum ein Einsatz in der Praxis ausbleibt. Geht ein Dienstleister nach einem dieser Vorgehen vor, so ist der Einsatz von Integrationsmustern schlichtweg nicht vorgesehen. Weitere Auswirkungen von Integrationsmustern werden am konkreten Beispiel im Rahmen einer Fallstudie in Kapitel 6.6 diskutiert.

[398] Vgl. (Stadlbauer, 2007, S. 126).

6 E-Procurement Integration eines ERP-Herstellers

Die in diesem Abschnitt beschriebene Fallstudie verwendet die in den vorangegangenen Kapiteln erarbeiteten Muster, die Klassifikation und die generellen Prinzipien der Integration zur Lösungsgestaltung. Zunächst werden der Integrationsdienstleister und das Kundenunternehmen anonymisiert beschrieben. Danach wird der relevante Ausschnitt der Unternehmensarchitektur des Kunden vorgestellt und im Anschluss daran die gewünschte Veränderung – der Sollzustand – skizziert. Daran schließen sich die Analyse der Integrationsobjekte und eine kurze Beschreibung der üblichen Vorgehensweise des Integrationsdienstleisters an, bevor die Lösungsgestaltung auf der Basis der entwickelten hochstehenden Integrationsmuster und der Klassifikation erfolgt. Die Umsetzung dieser Lösung – inclusive Integrationshilfsmittelauswahl – beispielhaft im Integrationslabor rundet die Darstellung ab. Auf der Basis dieser Erfahrungen wird dann das nun veränderte Dienstleistungsbündel des Integrationsdienstleisters beschrieben und aufgezeigt, inwieweit dieser zukünftig von der Anwendung der Integrationsmuster profitieren kann.

Bei der hier vorgestellten Fallstudie handelt es sich um das Beispiel eines Anwendungspartners aus einem BMBF geförderten Forschungsprojekt. Neben diesem Anwendungspartner – der in der Rolle eines Integrationsdienstleisters agiert – war als assoziierter Partner auch ein Kunde des Dienstleisters in das Projekt einbezogen. Das eröffnet die Möglichkeit zum einen die angebotene Integrationsdienstleistung als solche zu untersuchen und zum anderen eine konkrete Umsetzung.

Der Integrationsdienstleister bietet seine Dienstleistungen produktbegleitend, rund um das von ihm vertriebene ERP-System an. Die Zielgruppe des ERP-Systems sind produzierende mittelständische Unternehmen, da es besondere Stärken in der Produktionsplanung und -steuerung (PPS) aufweist. Einige Kunden setzen das System auch ausschließlich zu diesen Zwecken ein und ergänzen bestehende ERP-Systeme großer Hersteller mit den PPS-Funktionen.

Der Integrationsdienstleister ist seit 1990 am Markt tätig und verfügt in dem Bereich, welcher mit dem ERP-System und arrondierten Dienstleistungen beschäftigt ist, über ca. 20 Beschäftigte. Die Gesamtzahl der Beschäftigten überschreitet 50 Mitarbeiter nicht. Auch dem Umsatz nach handelt es sich um ein mittelständisches Unternehmen. Die Integrationsdienstleistungen werden produktbegleitend, rund um das vertriebene ERP-System angeboten. Die Zielgruppe des ERP-Systems sind produzierende mittelständische Unternehmen, da das System besondere Stärken in der Produktionsplanung und -steuerung (PPS) aufweist. Einige Kunden setzen das System auch ausschließlich zu diesen Zwecken ein und ergänzen bestehende ERP-Systeme großer Hersteller mit den PPS-Funktionen.

Durch die zentrale Stellung des ERP-Systems in der Unternehmensarchitektur mittelständischer Kunden wird der ERP-System-Hersteller bei vielen IT-Entscheidungen des Kunden zurate gezogen. Insbesondere dann, wenn das ERP-System mit von diesen Entscheidungen betroffen ist. Das ist häufig zwangsläufig der Fall, da Daten aus dem ERP-System benötigt oder an dieses übergeben werden müssen. Deshalb bietet der ERP-System-Hersteller neben seinem Produkt auch umfassende Integrationsdienstleistungen an. In den letzten Jahren konnte der Trend beobachtet werden, dass immer mehr produzierende Unternehmen im Bereich der Beschaffung nach Automatisierungspotenzial gesucht haben. Der hier beschriebene ERP-System-Hersteller hat sich auf diese Anfragen eingestellt und bietet spezielle E-Procurement-Integrationsdienstleistungen an. Dabei erfolgt die Erbringung dieser Dienstleistung bisher immer individuell und von Grund

auf neu, obwohl sich die Entwickler und Projektleiter des Wiederholungscharakters bewusst sind. Zur Zeit der Durchführung des Projektes gab es noch zwei weitere Kundenprojekte, welche sich mit derselben Problemstellung befassten. Dies verdeutlicht den Wiederholungsgrad der angebotenen Dienstleistung, welcher eine wesentliche Basis für die Rentabilität der Entwicklung und Nutzung von Mustern bildet. Dies haben einzelne Personen im Unternehmen bereits erkannt, sodass erste wiederverwendbare Artefakte entwickelt wurden.

Zunächst wird die konkrete Situation des Kunden beschrieben und dann auf die Integrationsobjekte und durch die Integration zu unterstützenden Prozesse eingegangen. Danach wird erläutert, wie die Erbringung der Dienstleistung bisher erfolgte. Dieses Vorgehen lässt sich aus untersuchten Kundenprojekten des Dienstleisters ableiten. Im Anschluss daran wird die Lösung auf der Basis der hochstehenden Integrationsmuster dargelegt und eine Beispielimplementierung beschrieben, die auf den Kunden zugeschnitten ist.

Diese Fallstudie integriert die konkrete Problemstellung **eines Kunden** des ERP-System-Herstellers. Wie nicht anderes zu erwarten setzt dieser das ERP-System ein, um sein Geschäft mit Metallerzeugnissen IT-seitig zu unterstützen. Die Produkte und Halbzeuge finden in den Bereichen der Küchen- und Medizintechnisch, sowie der Drahttechnik Einsatz. Das Unternehmen ist bereits seit 1870 tätig und verfügt damit über eine mehr als 130-jährige Präsenz am Markt. Mit mehr als 300 Mitarbeitern ist das Unternehmen nach den Kriterien der Europäischen Union als Großunternehmen einzustufen.[1] Dennoch wird es umgangssprachlich als Mittelstand gesehen und stellt einen typischen Vertreter der Kunden des Integrationsdienstleisters dar. Die Produkte werden weltweit in 60 Länder exportiert, wobei es 22 eigene Niederlassungen bzw. Vertriebsvertretungen gibt.

6.1 Das Integrationsszenario

Das Integrationsszenario wird beschrieben, in dem zunächst der bestehende Unternehmensarchitekturausschnitt des Kunden mit den von der Integration betroffenen Systemen und deren bestehenden Systembeziehungen erfasst wird. Danach erfolgt die Darlegung der IST-Situation des abzubildenden Prozesses und im Anschluss daran die der gewünschten Soll-Situation mit den Anforderungen des Kunden.[2]

6.1.1 Unternehmensarchitekturausschnitt mit bestehenden Systembeziehungen

Zunächst ist es notwendig, die zentrale Stellung des ERP-Systems in dem vorgestellten Unternehmen zu erkennen. Die Abbildung 6.1 zeigt funktionale Schritte im Unternehmen gekennzeichnet als Prozess. In der Mitte sind die Funktionen des ERP-Systems in einer Prozesskomponente zusammengefasst. Die Darstellung orientiert sich an einem Durchlauf eines Kundenauftrages durch das Unternehmen. An erster Stelle steht der Kundenkontakt, der z. B. in Form einer Anfrage stattfinden kann. In weiteren – hier zusammengefassten - Prozessschritten folgen die Aufgaben des *Vertriebs*, bestehend aus Angebot und Kundenauftrag, zusammengefasst in einer Komponente. Liegt ein Kundenauftrag vor, so folgt die *Konstruktion* mit Stammdaten, Stücklisten und Zeichnungen. Auf der Basis der Konstruktionsinformationen kann die *Arbeitsvorbereitung* erfolgen, welche ausgehend von einer Bedarfsermittlung Arbeitspläne erstellt und eine Terminierung des Auftrages vornehmen kann. Bevor die eigentliche *Fertigung* starten kann, sind natürlich die notwendigen Eingangsstoffe, Werkzeuge und Betriebsmittel durch den *Einkauf* zu beschaffen.

[1] Vgl. (Europäische Gemeinschaften, 2006, S. 14).
[2] Vgl. auch (Gebauer; Thränert u. a., 2008) und (Schenderlein u. a., 2009).

Für die Fertigung werden z. B. Daten für die Grobplanung und konkrete Fertigungsaufträge benötigt. Die hergestellten Produkte werden einem *Lager* zugeführt, dessen Daten neben der eigenen Bestandsführung auch im ERP-System hinterlegt werden müssen. Für den *Versand* mit der Komissionierung, Lieferzuteilung und Fakturierung ist eine Integration mit dem externen Anwendungssystem des Speditionsdienstleisters notwendig, da diese Leistungen durch einen Dienstleister erbracht werden. Alle diese Schritte involvieren neben weiteren Systemen auch

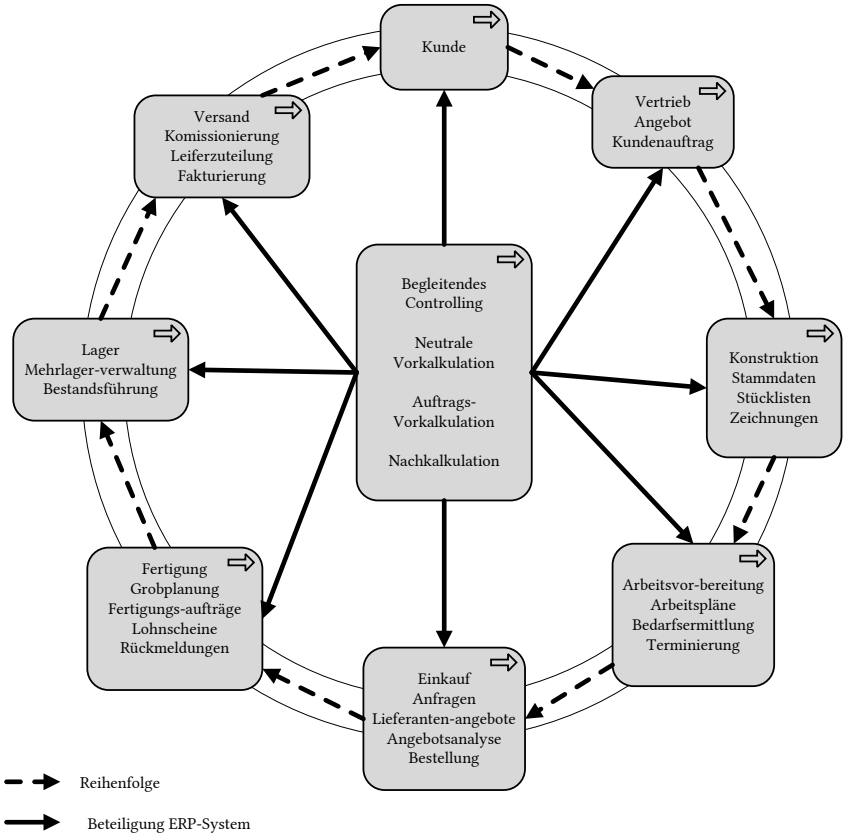


Abbildung 6.1: Funktionale Stellung des ERP-Systems im Unternehmen

jedes Mal das ERP-System, weshalb die Abbildung 6.2 die bestehenden Integrationsbeziehungen des ERP-Systems zeigt. Die zentrale Stellung des Systems wird sehr gut sichtbar, weshalb alle Integrationsbeziehungen auch durch den hier beschriebenen Dienstleister realisiert wurden. Die Integrationsbeziehungen wurden mit und durch das ERP-System als Punkt-zu-Punkt Verbindungen realisiert. Dies führt zum einen dazu, dass das ERP-System sämtliche Interaktionen mitbekommt und die im ERP-System vorgehaltenen Daten immer aktuell sind.

Ein einfacher Pfeil symbolisiert eine unidirektionale Beziehung der Systeme, wohingegen ein Doppelpfeil eine bidirektionale Beziehung der Systeme darstellt. Diese Beziehung ist nicht mit dem bisher verwendeten Begriff Integrationsbeziehung zu verwechseln. Eine Systembeziehung kann mehrere Integrationsbeziehungen umfassen, da die Granularität der Systeme Interaktionen in mehreren Kontexten erfordern kann.

Obwohl es sich um Punkt zu Punkt Verbindungen handelt, hat der Dienstleister bereits darauf geachtet fortschrittliche Technologien einzusetzen, weshalb – sofern technisch möglich – Web Services zum Einsatz kommen. Diese werden jedoch direkt aufgerufen, weshalb kein Vermittler zum Einsatz kommt und weiterhin direkte Abhängigkeiten der Integrationsobjekte bestehen bleiben, da die Kommunikationspartner mit allen notwendigen Daten der Kommunikation dem

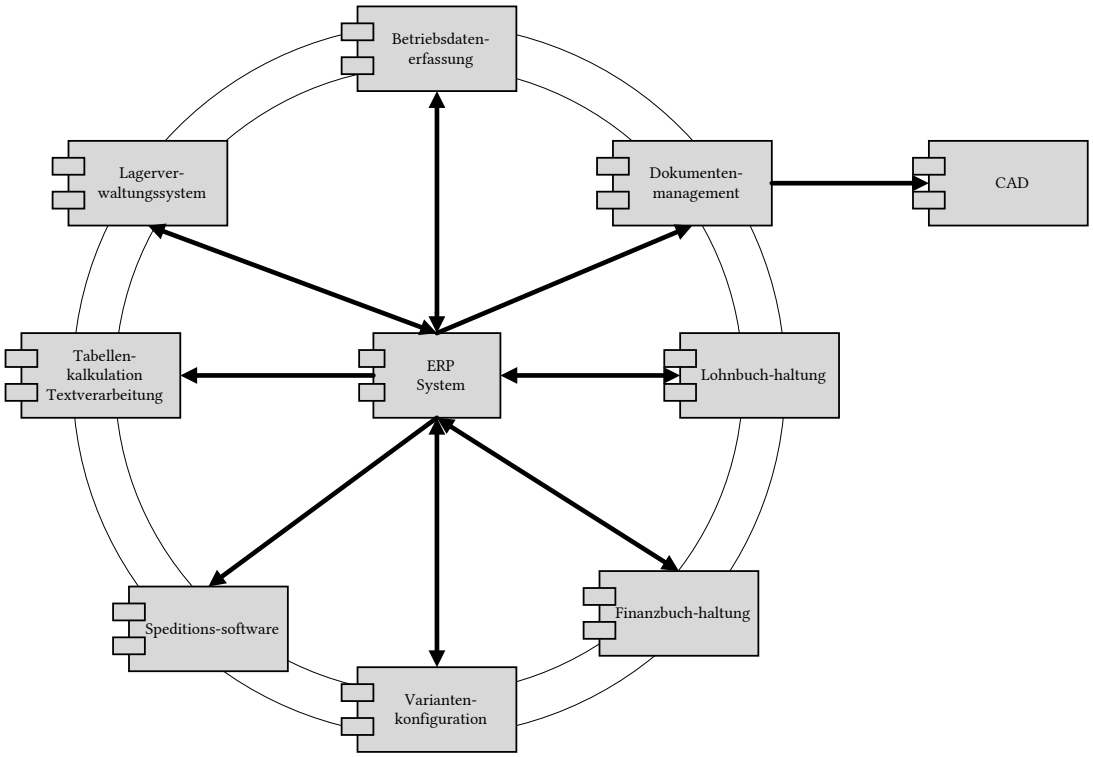


Abbildung 6.2: Architekturausschnitt des Anwendungsfalles

jeweiligen Integrationsobjekt bekannt sein müssen bzw. aufseiten des Aufrufers mit codiert werden. Dies ermöglicht Architekturveränderungen, welche im Rahmen der Beschreibung des Sollzustandes mit erläutert werden. Diese notwendigen Veränderungen ergeben sich aus einem massiven Defizit in der vorliegenden Architektur. Das Defizit resultiert aus der zentralen Stellung des ERP-Systems, welches sich nicht nur in funktionalen Abhängigkeiten niederschlägt, sondern durch die Lösung aller Integrationsbeziehungen mit und durch das ERP-System dazu führt, dass jedwede Kommunikation von dem Funktionieren des ERP-Systems abhängt. Damit wurde eine singuläre Fehlerquelle (Single-Point-of-Failure, SPOF) geschaffen, ein Ausfall des ERP-Systems bedeutet im schlimmsten Fall einen Stillstand der Produktion. Überspitzt formuliert führt ein Wegfall der Möglichkeit des Buchens und Kontieren zu einem Ausfall der Kerntätigkeit des Unternehmens, dem Produzieren, führen. Dies resultiert aus der Tatsache, dass das ERP-System zur Realisierung von Funktionalität verwendet wird, die der Rolle eines Anwendungssystems mit überwiegend betriebswirtschaftlichen Funktionen nicht entspricht.

6.1.2 Die Ist-Situation

Einige inländische Tochterfirmen des Kundenunternehmens verarbeiten – neben dem Vertrieb von Fertigprodukten – auch Halbzeuge der Muttergesellschaft und stellen damit spezialisierte Produkte für ihr jeweiliges Marktsegment her. Die Beschaffung dieser Halbzeuge erfolgt derzeit manuell mit Telefon, Fax und Papier. Die Bestellungen werden bei der Muttergesellschaft dann manuell im ERP-System erfasst. Daraus resultiert auf der einen Seite ein massiver Personalbedarf zur Abdeckung dieser Tätigkeiten durch die Muttergesellschaft und auf der anderen Seite ein Zeitversatz des Informationsflusses im Vergleich zur Produktion mit Auswirkung auf die Tochtergesellschaften. Die Terminierung der Lieferung der Halbzeuge in Abhängigkeit der Produktionsauslastung der Muttergesellschaft erfolgt erst nach der manuellen Erfassung. Im Fall von Planungsänderungen

der Muttergesellschaft entsteht dann ein Zeitversatz. Die Muttergesellschaft ändert den Produktionsplan und muss diese Veränderung manuell an das Tochterunternehmen kommunizieren. Kann ein Halbzeug nicht zum vereinbarten Termin geliefert werden, muss die Produktionsplanung des Tochterunternehmens entsprechend angepasst werden. Zusammengefasst handelt es sich um einen Fall des überbetrieblichen Prozessmanagements[3], da die Tochtergesellschaften als rechtlich selbstständige Unternehmen zu sehen sind.

6.1.3 Beschreibung des gewünschten Soll-Zustandes

Diese Ineffizienzen sollen durch eine Automatisierung der Bestellung von Halbzeugen – aus Sicht der Tochtergesellschaft also der Beschaffung – beseitigt werden, in dem ein Shop-System eingesetzt wird. Für die Tochtergesellschaften entsteht eine Verbesserung der Informationsqualität, da diese aktueller sind mit dem Effekt einer Verbesserung der eigenen Produktionsplanung, was zu einer höheren Termintreue gegenüber den eigenen Kunden führt.

Die Muttergesellschaft wird in die Lage versetzt das hoch qualifizierte Personal, welches derzeit mit der Bestellerfassung beschäftigt ist, anderweitig einzusetzen. Damit kann auf aufwendige externe Personalgewinnung verzichtet und den internen Mitarbeitern interessantere Tätigkeiten geboten werden.

In der ersten Ausbaustufe der Lösung sollen zunächst die Tochterfirmen ihre Beschaffung von Halbzeugen mithilfe eines durch den Integrationsdienstleister auszuwählenden Shop-Systems durchführen. Idealerweise soll jeder nur die Produkte angezeigt bekommen, die auch für ihn bestellbar sind. Die vertraglich zugesicherten Mindestlieferzeiten von zwei Tagen für vorrätige Artikel und vierzehn Tagen für nicht vorrätige Artikel müssen eingehalten werden.

In einer weiteren Ausbaustufe ist angedacht, dass die Tochterfirmen ihre eigenen Kataloge pflegen können und so auch Beschaffungsprozesse unter den Tochterfirmen ermöglicht werden.

Zukünftig ist weiterhin eine Anbindung von ca. 12 Auslandsniederlassungen geplant, weshalb die zu planende Lösung dahin gehend erweiterbar sein soll.

Neben den Kunden als Benutzergruppe ist für eine der späteren Ausbaustufen vorgesehen, den Vertriebsmitarbeitern, welche die Kunden betreuen, eine Abrufmöglichkeit für aktuelle Vertriebsinformationen (Umsätze etc.) zu ermöglichen. Aktuell werden diese als Hardcopy versandt, was bedeutet, dass sie zum Zeitpunkt des Eintreffens bereits veraltet sind. Dazu kann eine webgestützte Benutzerschnittstelle umgesetzt werden, die die Eingabe von Informationen mit Zielrichtung ERP-System bspw. der Kundenchronologie oder Spesenabrechnung der Außendienstler ermöglicht.

Weitere Anforderungen:
Grundsätzlich wünscht dieser Kunde aus der bisherigen Erfahrung heraus eine möglichst billige Lösung, die eine hohe Qualität und Verfügbarkeit aufweist. Es sollen möglichst wenig manuelle Eingaben notwendig sein.

Zusätzlich zu diesen Szenarien bestehen die nachfolgenden technischeren Anforderungen für die Umsetzung der Ausbaustufe eins, der Shop-basierten Halbzeugbeschaffung. Die Produktsicht des Shop-Systems stellt eine Teilmenge der Artikelsicht des ERP-Systems dar. Der Produktbestand der Kataloge des Shop Systems soll, aufgrund des Umfangs und der Aktualisierungshäufigkeit der Artikel im ERP-System, tagesaktuell sein.

Eine Aussage über die Verfügbarkeit eines Produktes – in Form von ist verfügbar oder ist nicht verfügbar – bedeutet für eine Tochtergesellschaft eigentlich, dass sie mit der Wiederbeschaffungs-

[3] Vgl. (Alt, 2008).

zeit des Produktes kalkulieren muss. Diese Information wird im ERP-System gepflegt und soll mit an das Shop-System übergeben werden, damit sie dort abgefragt werden kann. Zusätzlich soll die Verfügbarkeitsinformation zum Zeitpunkt der Abfrage aktuell sein.

Das System soll bei der Muttergesellschaft betrieben und in der demilitarisierten Zone (DMZ) platziert werden. Weitergehende Sicherheitsanforderungen existieren dahin gehend, dass nur explizit ausgewiesene Mitarbeiter der Tochterfirmen Zugriff auf das Shop-System erhalten sollen.[4]

6.1.4 Abzubildender Prozess

Im Zuge der Einführung des neuen Shop Systems ergeben sich aus der Architektursicht zahlreiche Veränderungen. In Abbildung 6.3 ist ein frühes Modell – Ergebnis der Grobanalyse – des aktuellen Zustands der Bestellverarbeitung durch die Muttergesellschaft dargestellt. Die Grafik zeigt ein Archimate 1.0 Modell.[5] Es handelt sich dabei um einen Modellierungsstandard der Object Management Group, welcher für Unternehmensarchitekturen verwendet wird. Die Entscheidung für den Standard erfolgte auf der Basis einer Analyse bestehender Konventionen in diesem Bereich. Das Modell wurde mit Hilfe eines – im Rahmen des Forschungsprojektes entwickelten – Prototypen erzeugt, welcher den Standard mithilfe des Eclipse-Modeling-Framework und des Graphical-Modeling-Framework umzusetzen versucht. Nähere Details dazu finden sich in Kapitel 8.

Es lassen sich drei Ebenen unterscheiden, wobei die oberste die geschäftliche Ebene darstellt. Die mittlere Ebene umfasst die Applikationssicht, wobei es sich um logische Applikationskomponenten handelt, die nach funktionaler Zusammengehörigkeit gebildet werden. Auch wenn ein ERP-System aus technischen Gesichtspunkten z. B. einer Dreischichtenarchitektur folgt und aus diesem Grund ein Datenbankmanagementsystem verwendet, resultiert daraus lediglich eine Anwendungskomponente. Die Abbildung beinhaltet zwei Rollen *MA Tochter* und *Mutter Disposition*, welche beide

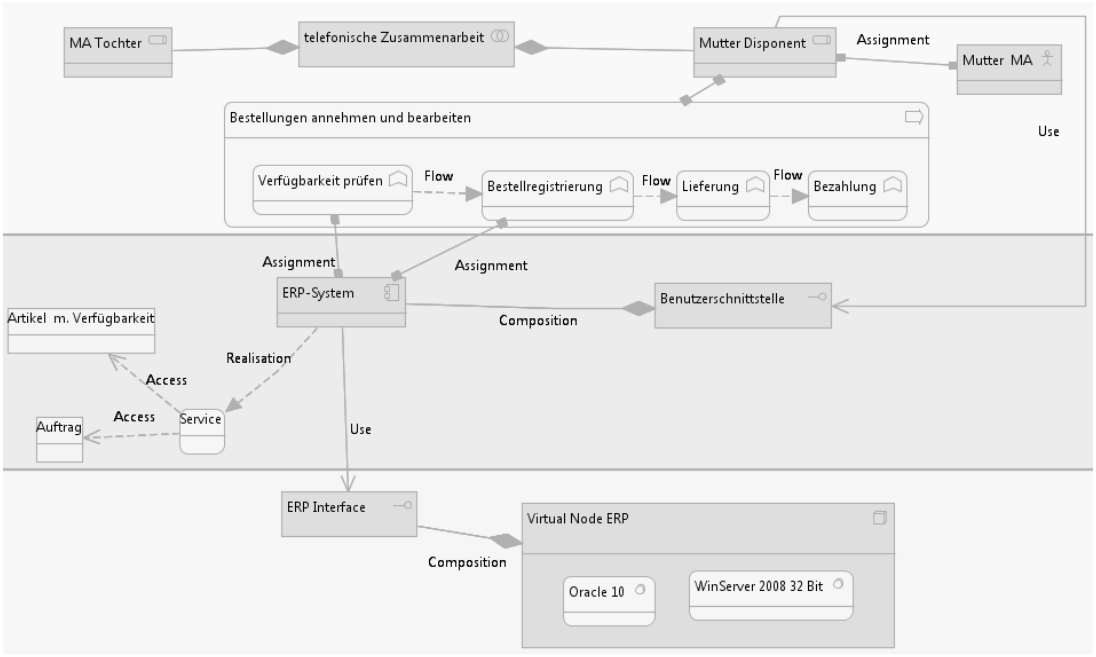


Abbildung 6.3: Ist-Zustand des Kunden als Archimate 1.0 Modell

[4]Sicherheitsanforderungen spielen bei dem Kundenunternehmen eine herausgehobene Rolle. Der E-Mailempfang wird z.B. über ein White-List-Verfahren organsisiert.

[5]Vgl. (The Open Group, 2009).

eine telefonische Zusammenarbeit durchführen. Der Modellierungsstandard sieht dafür explizit das Konstrukt einer Business-Collaboration[6] vor. Exemplarisch ist der Rolle *Mutter Disposition* ein Mitarbeiter zugeordnet. Durch die Assignment-Relation, welche die Rolle *Mutter Disposition* mit dem Prozess *Bestellungen annehmen und bearbeiten* verbindet wird die Verantwortung für die Durchführung des Prozesses verdeutlicht, die bei dem die Rolle ausfüllenden Mitarbeiter der Muttergesellschaft liegt.

Die Prozessschritte Verfügbarkeit prüfen und Bestellregistrierung werden mithilfe der Applikationskomponente ERP-System realisiert, was durch die Assignment-Relation ausgedrückt wird. Dafür wird die Benutzerschnittstelle des Systems verwendet. Als Repräsentation der Datenobjekte sind Artikel m. Verfügbarkeit und Auftrag im Modell enthalten. Obwohl fachlich unzureichend, ist in der dargestellten Version des Modells, keine einem Auftrag vorausgehende Bestellanforderung enthalten. Die Modellierung der Umsetzung der Datenobjekte über ein Service-Konstrukt, welches durch die Applikationskomponente realisiert wird und auf die Datenobjekte zugreift entspricht dem Modellierungsstandard kann aber einfacher ausgedrückt werden, wenn der Prototyp geerbte Relationen[7] umsetzen würde. Die Darstellung der technischen Basis wurde recht knappgehalten und ist – der Grobanalyse geschuldet – nicht ganz vollständig. Es wird deutlich, dass die technische Umsetzung auf einer Microsoft Betriebssystembasis aufsetzt und eine Oracle Datenbank verwendet.

Die weiteren Ausführungen erfordern eine genaue Darstellung des neu umzusetzenden Prozesses nebst den beteiligten Integrationsobjekten auf der Ebene logischer Applikationskomponenten und zusätzlich deren technischer Realisierung. Die Abbildung 6.4 stellt dies in Form eines Archimate 1.0 Modells dar und bietet die Basis für die weiteren Analysen. Das dargestellte Modell entstammt einer späten Phase des Projektes, was die Detailunterschiede erklärt, da eine iterative Verfeinerung vorgenommen wurde. Betrachtet man die obere – geschäftliche – Ebene, so kann man schnell die Detaillierung des Prozesses im Vergleich zu dem in Abbildung 6.3 erkennen. Im Gegensatz zu der dortigen Darstellung ist der Rolle *MA Tochter* nun ein Teil des Prozesses verantwortlich zugeordnet. Diesen Teil führt der Mitarbeiter selbstständig ohne die Mitarbeit der Rolle *MA Mutter* durch. Das frei werden der Arbeitskraft ist sehr gut daran erkennbar, dass der Rolle MA Mutter kein Prozessteil mehr zugeordnet ist. Es ist demnach fraglich, ob die Modellierung der Zusammenarbeit der beiden Rollen – in Form von Halbzeug Bestellung – fachlich noch korrekt ist. Dieses Modellierungselement wurde hier beibehalten, um die sich verändernden Beziehungen zu verdeutlichen.

Ein weiterer Unterschied im Hinblick auf den Istzustand ist die Modellierung der Geschäftsobjekte, die an dem Prozess beteiligt sind. Die explizite Darstellung und die Zuordnung zu den Datenobjekten der jeweiligen logischen Applikationskomponenten helfen bei der Zuordnung korrespondierender Datenobjekte. Dies ist zum Beispiel bei *Artikel* und *Produkt* der Fall. Beide realisieren dasselbe Geschäftsobjekt *Produkt*, welches in unterschiedlichen Teilschritten des Prozesses Halbzeugbestellung zugegriffen wird. Beide Teilschritte werden durch dieselbe Applikationskomponente unterstützt, sodass zunächst kein Integrationsbedarf zu vermuten ist. Da das Shop System jedoch erst auszuwählen ist und erst danach mit Daten befüllt werden kann

[6] Vgl. (ebenda).

[7] Diese Definition geht von einem Metamodell aus und legt fest, dass eine vererbte Relation eine Relation zwischen zwei Elementen ist, zwischen denen sich ein Weg über andere Elemente hinweg finden lässt. Die Relation verbindet die Elemente dann direkt und ist von dem Typ, der in einer Rangfolge der schwächsten Relation auf dem gefundenen Weg entspricht. Der Prototyp implementierte diese Funktionalität erst in einer späteren Version. Ein erneutes Erstellen aller bis dahin bereits entwickelten Modelle war aus Zeitgründen nicht möglich. Automatische Migrationen waren zu diesem Zeitpunkt erst Gegenstand der Forschung im Bereich der modellgetriebenen Entwicklung auf deren Entwicklungsprinzipien der Prototyp beruht.

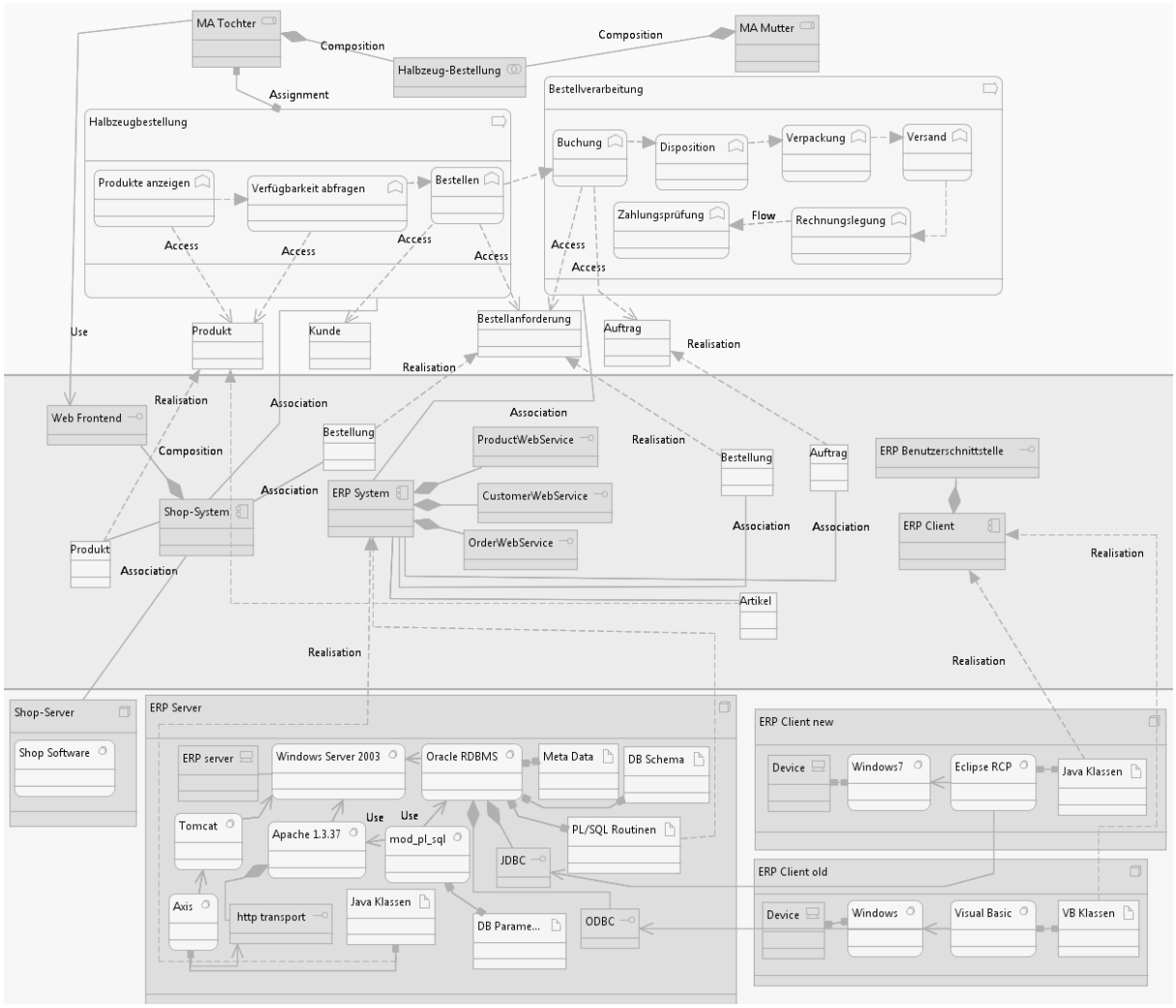


Abbildung 6.4: Soll-Zustand des Kunden als Archimate 1.0 Modell

wird schnell die erste Integrationsbeziehung zwischen dem ERP-System und dem Shop-System klar. Es muss ein Produktimport durchgeführt werden.

Die zweite Integrationsbeziehung wird ebenfalls schnell deutlich. Sie betrifft die Bestellanforderungen. Dieses Geschäftsobjekt wird von den Teilschritten Bestellen und Buchung zugegriffen, welche durch unterschiedliche Applikationskomponenten unterstützt werden. Beide Applikationskomponenten setzen ein Datenobjekt Bestellung um, welches eine Bestellanforderung realisiert. Durch den Verlauf des Prozesses wird schnell klar, dass die Bestellanforderungen im Shop-System auflaufen und zur weiteren Verarbeitung an das ERP-System übergeben werden müssen. Auf diese Weise wurde die zweite Integrationsbeziehung identifiziert.

Bevor die Beschreibung des Integrationsobjektes ERP-System und die Auswahl des Shop-Systems bearbeitet werden, wird zunächst noch die abgeleiteten Anforderungen an die Lösung in der Sprache des Integrationsdienstleisters beschrieben.

Aus der Anforderung, dass nur Mitarbeiter der Tochtergesellschaften den Shop nutzen sollen, leitet sich eine geschlossene Kundengruppe ab. Die Darstellung des Katalogs muss in Abhängigkeit des Log-ins variierbar sein, oder es muss die Möglichkeit geben sein, verschiedene Kataloge anzuzeigen und zu pflegen. Die letztere Variante erhöht den Datenpflegeaufwand in Abhängigkeit

der voneinander zu unterscheidenden Nutzergruppen. Durch die potenzielle Anbindung der Auslandsgesellschaften und die damit verbundenen Probleme der Lokalisierung sowie die potenziell sich rasch vergrößernde Benutzerzahl wird nach einer skalierbaren und leicht anpassbaren Lösung gesucht. Eine sofortige Verfügbarkeitsanzeige entspricht einem Vergleich der Bestellmenge mit dem zum Zeitpunkt aktuellen Lagerbestand des Artikels. An dieser Stelle wurde eine weitere Integrationsbeziehung aufgedeckt, da diese Information in einem Teilprozessschritt zugegriffen wird, der durch das Shop-System verwaltet wird, aber in diesem nicht gepflegt wird. Aufgrund der geforderten Aktualität der Information muss eine Abfrage des ERP-Systems erfolgen, da in diesem die aktuellen Bestände gepflegt werden. Der tägliche Abgleich der Artikel des ERP-Systems mit den Produkten bzw. dem Katalog des Shop-Systems entspricht einem zeitgesteuerten Batch-Lauf. Aufgrund der Aussagen des Kunden soll die Rechnungslegung weiterhin per Post oder E-Mail erfolgen, weshalb auch keine Applikationskomponente mit dem entsprechenden Prozessschritt verbunden ist. Da Produkte immer bestellbar sein sollen, auch wenn sie gerade nicht verfügbar sind, ist außer dem Vergleich mit dem Lagerbestand keine Rückkopplung zwischen ERP-System und Shop-System über die bisher benannten Integrationsbeziehungen hinaus notwendig. Die Integrationsbeziehungen werden in der Abbildung 6.5 noch einmal veranschaulicht.

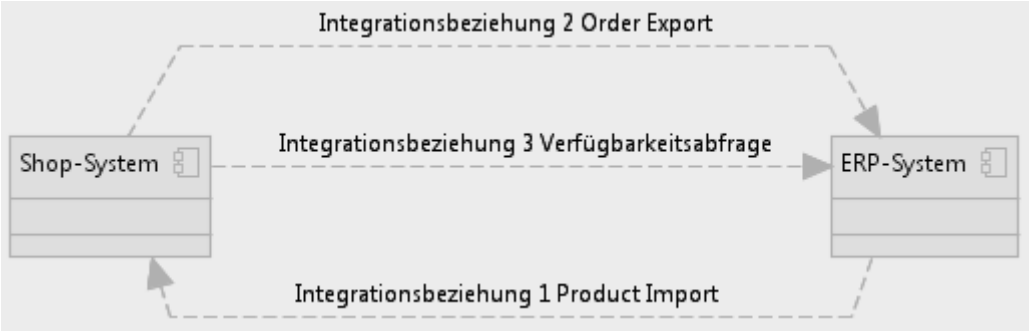


Abbildung 6.5: Integrationsbeziehungen des Szenarios

6.1.5 Integrationsobjekte

Da das Integrationsobjekt Shop-System noch auszuwählen ist, wird erst die in allen Integrationsobjekten feste Seite beschrieben. Aus der Sicht des Integrationsdienstleisters ist handelt es sich dabei um das ERP-System. Diese Beschreibung in Form eines Modells wie in der Abbildung 6.4 dargestellt, ist aus der Sicht eines Integrationsdienstleisters ein widerverwendbares Asset. Es lässt sich nicht nur für die Kommunikation mit dem Kunden nutzen, sondern kann auch bei der Einarbeitung neuer Mitarbeiter eine wertvolle Hilfe sein. Die hier vorgenommene Beschreibung geht explizit von der im Integrationslabor installierten Instanz aus, was z. B. bestimmte Softwareversionen erklärt.

Entgegen der Darstellung in Abbildung 6.3 handelt es sich bei dem umgangssprachlich als ERP-System bezeichneten Anwendungssystem nicht um eine, sondern um zwei logische Applikationskomponenten. Die Erste ist der Client, mit dem ein Mitarbeiter auf das eigentliche ERP-System im Sinne eines Servers zugreift. Dieser Client ist auf dem Gerät des Mitarbeiters zu installieren. Obwohl der Modellierungsstandard es ermöglicht Netzwerke und die Verbindung der technischen Komponenten durch diese mit zu modellieren, verzichtet die Abbildung 6.4 aus Platzgründen auf eine Darstellung. Für den Client ist zwischen einer alten ERP Client old und einer neueren Ausführung (ERP Client new) zu unterscheiden. Diese sind technisch mit unterschiedlichen Mitteln realisiert. Bei dem alten Client kommt Visual Basic (VB) zum Einsatz, wohingegen der neue Client mit Java auf Basis der Eclipse-Rich-Client-Platform (Eclipse-RCP)

realisiert wurde. Beide unterscheiden sich durch die Schnittstelle, über welche sie mit dem Server integriert sind. Der alte Client nutzt Open-Database-Connectivity (ODBC) und der neue Client Java-Database-Connectivity (JDBC). Auf den ersten Blick mag die Integration des Client mit dem Server so aussehen, als ob das ERP-Sytem eine Zweischichtenarchitektur aufweist, dies ist jedoch nicht der Fall. Das ERP-System basiert auf einer Oracle-Datenbank, welche neben dem Datenbankschema auch einen abgegrenzten Bereich enthält, in dem die Metadaten vorgehalten werden. Die Auswirkungen dieses Ansatzes werden an späterer Stelle erläutert und spielen bei der durch den Integrationsdienstleister vorgeschlagenen Lösung eine Rolle. Das es sich um eine Dreischichtenarchitektur handelt wird dadurch, deutlich, dass der Funktionskern des ERP-Systems auf der Basis von PL/SQL Routinen entwickelt wurde und durch eine strenge Überwachung der Entwickler keine Zugriffe direkt auf die Datenbank erfolgen, sondern nur auf PL/SQL Routinen.

Die dargestellte Variante des ERP-Systems beinhaltet bereits eine sogenannte Web-Option, welche in der Lage ist Web-Services zur Verfügung zu stellen. Das ERP-System ist bei einigen Kunden auch ohne diese Option im Einsatz, was die Ausgangslage für die Integration verändern würde. Für die Bereitstellung von Services wie unter anderem dem ProductWebService wird Apache Axis als Framework verwendet, welches mithilfe eines Tomcat Webservers ausgeführt wird. Der Zugriff der Services auf die PL/SQL Routinen erfolgt durch eine Schnittstelle http Transport, welche proprietäres XML (Extensible-Markup-Language) verarbeitet. In dieses XML sind die Aufrufe der Routinen verpackt, welche wiederum durch einen Apache Webserver – der über ein spezifisches Modul zur Verbindung mit der Datenbank (mod_pl_sql) verfügt an die Oracle Datenbank übermittelt werden. Die Rückrichtung mit den Ergebnissen der Routinen erfolgt in umgekehrter Reihenfolge. Damit wird deutlich, dass die logische Applikationskomponente ERP-System auf der technischen Ebene, welche mitbestimmend für die Auswahl einer Integrationslösung ist, interessante Details aufweist, die eine Integration herausfordernd machen.

6.2 Typischer Lösungsvorschlag des Integrationsdienstleisters

Der Integrationsdienstleister hat auf der Basis seiner bisherigen Erfahrungen einen Lösungsvorschlag erarbeitet, welcher hier vorgestellt werden soll, um im Anschluss daran die Veränderung in der Lösungsgestaltung aufzuzeigen, welche mit der Anwendung der Ergebnisse dieser Arbeit einhergeht.

Der Dienstleister fasst den Prozess wie folgt zusammen: 1) der Mitarbeiter einer Tochtergesellschaft meldet sich im Shop an, 2) er sieht sein spezifisches Produktsortiment, welches auf den Artikelstammdaten aus dem ERP-System basiert (Integrationsbeziehung 1), 3) die Verfügbarkeit der einzelnen Produkte wird in Form einer Farbcodierung angezeigt, genannt Ampelfunktionalität, welche auf die Bestandsfunktion des ERP-Systems zurückgreift (Integrationsbeziehung 3), 4) die Bestellanforderung wird automatisch an das ERP-System übertragen (Integrationsbeziehung 2), 5) die Muttergesellschaft generiert manuell aus der Bestellanforderung eine Bestellung im Rahmen eines Überprüfungsworkflows. Eine Integration von Bezahlsystemen ist nicht notwendig, da dies weiterhin wie bisher zwischen Mutter- und Tochtergesellschaften üblich ablaufen soll.

Daraus kann für die Auswahl des Shop-Systems die Funktionen Login, (Multi-)Katalog, Verfügbarkeit, Warenkorb, Bestellung und Bestellhistorie festgehalten werden. Auf der Basis bisheriger Erfahrungen schlug der Integrationsdienstleister die Verwendung eines EasyShop-Systems vor, welches auf dem in Phython programmierten Content-Management-System Plone aufsetzt.

Die Basis des Lösungsvorschlags bildet zunächst ein Erkennen von Bekanntem in der Problemstellung des Kunden. Dabei handelt es sich um zwei domänenspezifische Integrationsszenarien, welche der Dienstleister, nach eigenem Bekunden, wiederkehrend löst. Dabei handelt es sich um

den *Product Import*, also den wiederkehrenden Abgleich der Produkte des Shop Systems mit dem ERP-System und den *Order Export*, den Transport der Bestellanforderungen in das ERP-System.

Grundsätzlich soll die Kommunikation mit dem ERP-System über die – durch die Weboption verfügbaren – Web Services erfolgen. Standardmäßig wird diese Lösung offeriert und Alternativen nur in Betracht gezogen, wenn der Kunde dies explizit in den Anforderungen festlegt oder dem Lösungsvorschlag widerspricht. Für den *Product Import* schlägt der Integrationsdienstleister eine zeitgesteuerte, asynchrone Variante vor, die einmal am Tag des Nachts den Abgleich durchführt.

Die Verfügbarkeitsabfrage soll möglichst aktuell sein, weshalb hier eine synchrone Kommunikation vorgeschlagen wird. Synchron wird hier im Sinne eines blockierenden Senden verwendet, was ein Warten des Shop-Systems bis zum Erhalt einer Antwort durch das ERP-System zur Folge hat.

Am Beispiel der letzten Integrationsbeziehung, dem Bestellexport (Order Export), werden die bereits geschaffenen, wiederverwendbaren Artefakte des Integrationsdienstleisters erläutert. Neben den Beschreibungen des Prozesses und seiner Teilprozesse sind auch die Datenbankfelder sowie die korrespondierenden Elemente der XML-Struktur detailliert beschrieben. Diese werden hier nicht im Detail vorgestellt. Die Abbildung 6.6 zeigt den Hauptprozess des Bestellexports. Es handelt sich dabei um eine mögliche Instanz. Es wurden keine Alternativen Ausprägungen aufgenommen, die in Abhängigkeit einer Kundenentscheidung zu instanziieren sind. So geht die

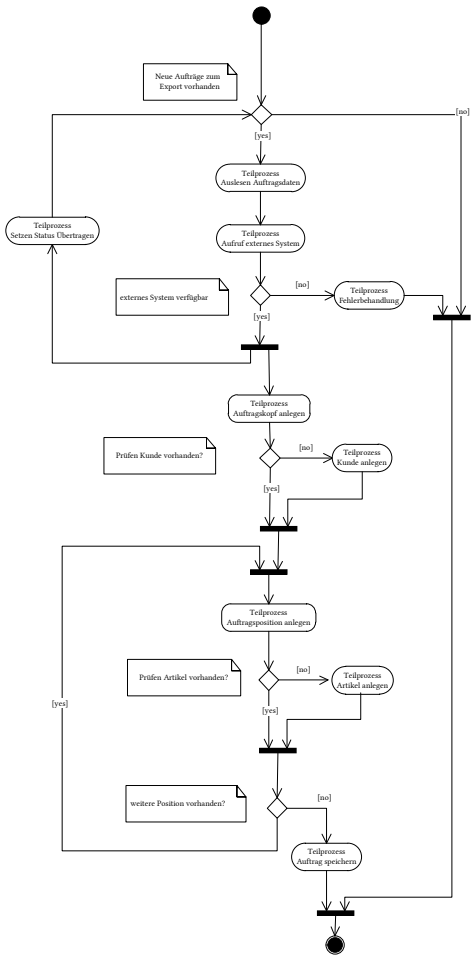
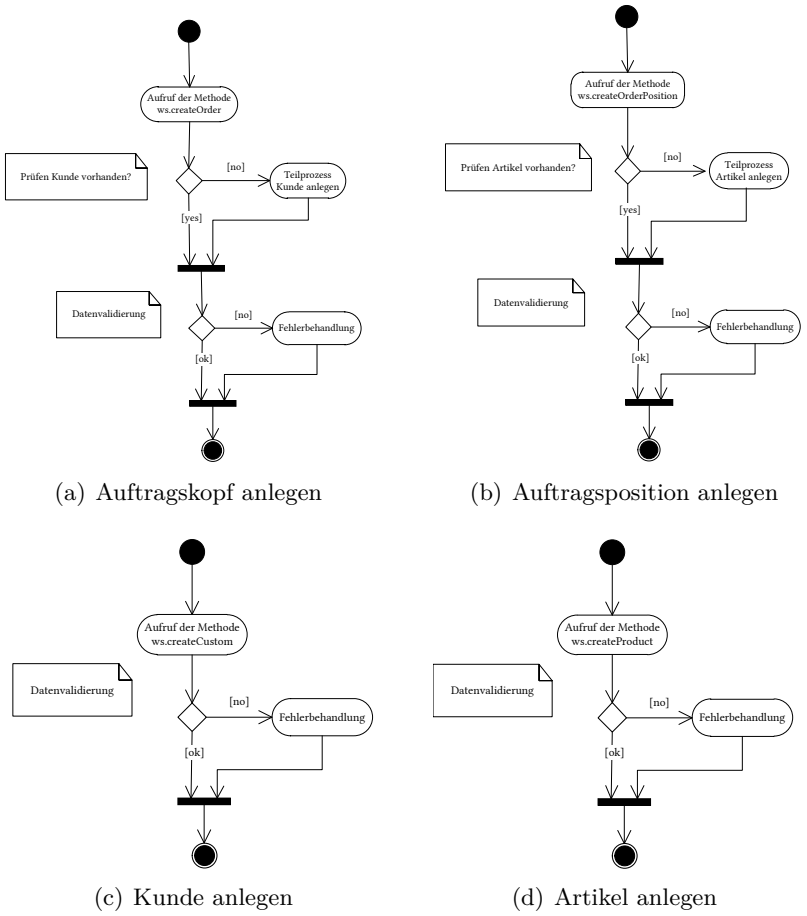


Abbildung 6.6: Hauptprozess des Bestellexports

hier dargestellte Variante dieses Prozesstyps davon aus, dass die Teilprozesse Kunde anlegen (Abbildung 6.7(c)) und Artikel anlegen (Abbildung 6.7(d)) durchlaufen werden, wenn der Artikel

oder der Kunde nicht im ERP-System existiert. In dem hier zu realisierenden Szenario können diese beiden Teilprozess ausgespart werden. Bei einer geschlossenen Benutzergruppe tritt der Fall nicht ein, dass sich ein Kunde am Shop-System registriert, der vorher noch nie etwas bestellt hat und deshalb als Kunde nicht im ERP-System existiert. Als das führende System für die Pflege von Artikelstammdaten und damit den bestellbaren Produkten ist ganz klar das ERP-System ausgewiesen. Damit werden keine Produkte in dem Shop-System hinzugefügt, weshalb auch dieser Teilprozess nicht betrachtet werden muss. Diese Fragen klärt der Integrationsdienstleister unter Verwendung der in den Abbildungen vorgestellten Prozessschablone in einem Gespräch mit dem Kunden ab. Das Integrationsszenario vernachlässigt auch bewusst die Möglichkeit des Einsatzes eines Produktdatenmanagementsystems. Derartige Systeme kommen zum Einsatz, weil in den Artikelstammdaten des ERP-Systems nicht alle notwendigen Daten für den Verkauf über ein Shop-System enthalten sind. Bilddaten oder Packmaße gehören nicht zu diesen Stammdaten und müssen extern gepflegt werden. In dem vorliegenden Szenario erfolgt diese Pflege direkt im Shop-System und nicht in einem Produktdatenmanagementsystem. Der Einsatz eines solchen Systems würde sich anbieten, wenn mehr als ein System mit solchen zusätzlichen Produktdaten versorgt werden müsste und eine zentrale Pflege angestrebt wird. Im Gegensatz dazu müssen die Teilprozesse Auftragskopf anlegen (Abbildung 6.7(a)) und Auftragsposition anlegen (Abbildung 6.7(b)) in jeder Variante des Die Teilprozesse Auftragskopf anlegen und Auftragsposition anlegen zeigen



(a) Auftragskopf anlegen

(b) Auftragsposition anlegen

(c) Kunde anlegen

(d) Artikel anlegen

Abbildung 6.7: Teilprozesse des Bestellexports

Diskrepanzen in der Terminologie des Integrationsdienstleisters auf. So ist auf der einen Seite von einem Bestellexport die Rede und auf der anderen Seite vom Anlegen eines Auftrages. An dieser

Stelle kann davon ausgegangen werden, dass es sich hier um das Anlegen der Bestellanforderung handelt. Im Allgemeinen wird erst nach der Bestätigung einer Bestellanforderung aus dieser ein Auftrag. Im Fall des untersuchten Integrationsszenarios handelt es sich dabei um einen manuell durchzuführenden Schritt, der bisher in keinem Modell berücksichtigt wurde und später noch Anlass zur Kritik geben wird.

Auch diese letzte Integrationsbeziehung soll – wie durch den Dienstleister vorgeschlagen – als Punkt-zu-Punkt Verbindung unter Verwendung der jeweiligen Web-Services implementiert werden. Dies entsprich der bei den Mitarbeitern des Dienstleisters verbreiteten Sichtweise, dass mit Web-Services im Sinne einer Service Facade oder eines Service Wrapper eine Kapselung von entfernten Methodenaufrufen ermöglichen. Insgesamt wurde für diesen Lösungsvorschlag inclusive Dokumentation und Schulung ein Zeitaufwand von ca. 45 Manntagen veranschlagt.

6.3 Dienstleistungsmodell des ERP-Herstellers

Um diesen Lösungsvorschlag und damit die bisherige Lösung gleichartiger Probleme durch den Integrationsdienstleister zu bewerten, reicht es nicht aus, die Problemstellung zu kennen. Auf der Basis der empirischen Untersuchung lässt sich sagen, dass das konkrete Dienstleistungsbündel des Integrationsdienstleisters eine ebenso wichtige Rolle spielt. Aus diesem Grund werden an dieser Stelle relevante Ausschnitte des Dienstleistungsmodells des Integrationsdienstleisters vorgestellt. Wie in Kapitel 4.2 dargelegt, besteht ein Dienstleistungsmodell aus verschiedenen Teilen. Zunächst wird mit dem Komponentenmodell begonnen, welches die Servicekomponenten einer Dienstleistung enthält. Für den Dienstleister wurden diese Informationen direkt aus konkreten Angeboten dieser Dienstleistung abgeleitet, was z. B. die Zuordnung von Aufwandsgrößen ermöglicht. Die Tabelle 6.1 stellt die angebotenen Servicekomponenten mit den Ergebnissen der empirischen Untersuchung gegenüber und zeigt auf, wie sich die Servicekomponenten des Dienstleisters – im Sinne von Instanzen – auf die Servicekomponenten der Integration[8] abbilden lassen. Die Tabelle 6.2 detailliert die relevanten Eigenschaften der einzelnen – in dem hier untersuchten Fall – angebotenen Servicekomponenten. In der Tabelle wurde nur die integrationsbezogenen

Komponenten des Referenzmodells	**angebotene Servicekomponenten**
Pre-Sales Analyse Planung	Spezifikation/Abstimmungen
Durchführung Durchführung Durchführung Durchführung	Einrichtung Onlinekennzeichen in ERP Einrichung Batch-Job für Artikelexport Einrichtung der Web Services Implementierung/Verwendung WebServices auf Shop Seite Order Export, Verfügbarkeit
Rollout nicht angebbar Schulung	Installation Dokumentation Schulung

Tabelle 6.1: Angebotene Servicekomponenten

Servicekomponenten aufgeführt. Das Gesamtangebot des Dienstleisters beinhaltet eigentlich noch eine Dienstleistung Shop-System-Auswahl und Einrichtung mit entsprechenden Teilkomponenten. Sofern kein direkter Integrationsbezug zu erkennen war, wurden diese Servicekomponenten hier nicht mit aufgeführt. Im Vergleich zu den Servicekomponenten der Integration weisen die durch den Dienstleister angebotenen Komponenten eine deutlich kleinere Zahl auf. Außerdem sind sie nicht so feingranular. Die Komponenten *Spezifikation/Abstimmung* vereint die Komponenten Pre-

[8] Vgl. Tabelle 4.2

Sales, Analyse und Planung. Wobei davon auszugehen ist, dass dies nur teilweise der Fall ist, da das Ergebnis dieser Komponenten ein Angebot ist. Deshalb kann nicht davon ausgegangen werden, dass auch Teilkomponenten wie Termin- und Ressourcenplanung in dieser Komponente enthalten sind. Insofern ist das hier skizzierte Dienstleistungsmodell des ERP-Herstellers unvollständig bzw. nur grob spezifiziert. Im Gegensatz dazu weist das Modell der Servicekomponenten der Integration, welches auf Basis der empirischen Untersuchung erstellt wurde, nur eine Teilkomponenten Durchführung auf. Diese wurde jedoch im vorliegenden Modell des ERP-System-Herstellers drei Mal in unterschiedlichen Ausprägungen instanziiert. Mit Installation und Schulung sind zwei Teilkomponenten der Transition enthalten. Die Komponente Dokumentation hingegen taucht in dem empirisch ermittelten Modell gar nicht auf und muss in diesem entsprechend ergänzt werden.

Nun werden die Eigenschaften der Servicekomponenten des ERP-System-Herstellers betrachtet, welche in Tabelle 6.2 dargestellt sind. Über einige Eigenschaften kann keine Aussage getroffen werden, da in den analysierten Unterlagen die Servicekomponenten der Einführung des Shop-Systems mit denen der Integration vermischt wurden. Der Dienstleister hat dem Kunden eine Offerte bestehend aus zwei Teildienstleistungen gemacht. Darüber hinaus werden einige Eigenschaften von Servicekomponenten in der Tabelle nicht aufgeführt, da sie eine für alle Servicekomponenten gleiche Ausprägung aufweisen oder die Ausprägung nicht bestimmt werden konnte. Dies betrifft z. B. das Bezahlverfahren, welches für alle dargestellten Servicekomponenten die Ausprägung Überweisung nach Rechnung aufweist. Weiterhin wurde auf die Darstellung von Evaluationskriterien verzichtet, da diese häufig nur in der Form angegeben wurden, ob die Servicekomponente erfolgreich durchgeführt wurde oder nicht. Eine qualitative Bewertung, mit deren Hilfe sich die Servicekomponentenqualität messen lassen würde, war in der Praxis nicht zu finden, womit diese Eigenschaft sich – im vorliegenden Fall – als unbestimmbar erwies. Die Ziele der einzelnen Schritte sind leicht nachvollziehbar und beginnen mit der Festlegung des Integrationsszenarios. Die Einrichtung Online-Kennzeichen und Einrichtung Web Services stellen die Integrationsfähigkeit des ERP-Systems her. Die letztendliche Realisierung der Integrationsbeziehungen erfolgt durch Einrichtung Batch-Job und und Implementierung Verwendung der Web-Services. Dabei geschieht Ersteres durch eine Veränderung des Integrationsobjektes ERP-System und Letzteres durch eine Veränderung des Integrationsobjektes Shop-System.

Lediglich für die Servicekomponenten Spezifikation/Abstimmungen, Schulung und Installation kann die Eigenschaft lokale und temporale Verfügbarkeit exakt angegeben werden, dass diese beim Kunden vor Ort erfolgen. Alle anderen Servicekomponenten können sowohl beim Dienstleister als auch beim Kunden erfolgen. Was durch die Angabe Dienstleister/Kunde verdeutlicht wird. Dies hängt davon ab, ob die Entwicklung mit Hilfe von Entwicklungssystemen (Replikationen der Integrationsobjekte) erfolgt und wo sich diese befinden, bzw. ob diese einen Remote-Zugriff erlauben. Mit der Unbestimmtheit dieser Eigenschaft wurde ein Konfigurationspunkt der Dienstleistung des ERP-System-Herstellers identifiziert.

Für alle Servicekomponenten ist auf der Basis eines konkreten Kundenangebotes für den vorliegenden Fall die Eigenschaft Dauer ermittelt worden. Eine Ausnahme bildet die Installation. Durch die Vermischung der Leistungen für die Auswahl und Installation des Shop-Systems mit denen der Integration kann hier nur die Dauer für beide Dienstleistungen zusammen angegeben werden. Deshalb findet hier die Zeit für beide Leistungen als obere Grenze Verwendung, was durch maximal 1,0 Manntage (MT) angezeigt wird. Teilweise gestaltet sich die Angabe von Ressourcen, welche in eine Servicekomponente eingehen oder durch diese entstehen als schwierig. Als Beispiel kann hier die Komponente Dokumentation dienen. Genau genommen handelt es sich hier um eine kontinuierliche Tätigkeit, welche parallel zu den einzelnen Entwicklungsschritten erfolgt. Durch die Angabe der Ressourcen Lösung auf der Input-Seite und Dokumentation auf der Output-Seite

Service-komponente	**Ziele**	**Ressourcen**	**lokale und temporale Verfügbarkeit**	**Dauer**	**Konsequenzen**
Spezifikation/ Abstimmungen	Integrationsszenario festlegen	In:IT-Leiter, Out:Angebot	Kunde	4 MT	undefinierter Umfang
Einrichtung Online Kennzeichen in ERP-System	Festlegung der Artikel zum Export in Shop-System	In:ERP-System (Datenbankschema, Datenbestand), Datenbankentwickler Out: ERP-System (Datenbankschema, Datenbestand erweitert)	Dienstleister/ Kunde	0,5 MT	Artikelmenge zu gering oder null
Einrichtung Batch-Job	Export der Artikel in ERP-System	In:ERP-System, Datenbankentwickler Out: verändertes ERP-System	Dienstleister/ Kunde	1,0 MT	kein regelmäßiger Artikelexport/ Produktimport
Einrichtung der Web Services	Zugreifbarkeit der Daten	In:ERP-System(), Datenbankentwickler, Java-Entwickler Out: ERP-System(Web-Option)	Dienstleister/ Kunde	9,0 MT	Daten nicht abrufbar
Implementierung Verwendung der Web Services	Zugriff auf die Daten	In: Shop-System(), Java-Entwickler Out: Shop-System(erweitert)	Dienstleister/ Kunde	12,0 MT	Datenabruf findet nicht statt
Installation	Lösung ist beim Kunden installiert	In: Java-Entwickler, Datenbankentwickler, ERP-System(K), Shop-System(K) Out: veränderte Unternehmensarchitektur	Kunde	maximal 1,0 MT	Lösung ist nicht installiert
Dokumentation	Lösung ist dokumentiert	In: Lösung Out: Dokumentation	Dienstleister	1,0 MT	Lösung ist nicht dokumentiert
Schulung	Mitarbeiter des Kunden können mit der Lösung umgehen	In: Director Business Solutions, Mitarbeiter Kunde (ungeschult) Out:Mitarbeiter Kunde (geschult)	Kunde	1,0 MT	Mitarbeiter sind nicht geschult

Tabelle 6.2: Eigenschaften der angebotenen Servicekomponenten

Ressource	Type	Resource identity	Resource quantity	Capacity	Resource Relations	Resource hierarchy	Source Type	Tangibility	Mobility	Activity
Java Entwickler	HR	JE	3 Pers	5 MT pro Woche	coup (ECLI)	-	p	t	m	uop
Datenbankentwickler	HR	DBE	5 Pers	5MT pro Woche	coup (SQLD)	-	p	t	m	uop
Account Manager	HR	ACM	3 Pers	5MT pro Woche	-	-	p	t	m	uop
Projektleiter	HR	PL	1 Pers	5MT pro Woche	-	-	p	t	m	uop
IT-Leiter	HR	ITL	1 Pers	n.a.	-	-	c	t	m	uop
Eclipse	H/W	ECLI	n.a.	1 N/ZE	coup (JE)	-	p	i	m	uop
Oracle SQL Developer	H/W	SQLD	n.a.	1 N/ZE	coup (DBE)	-	p	i	m	uop
Angebot	D/M	ANG	1 Stck	n.a.	-	-	p	i	m	op/ uop
ERP-System	IO	IO1	1 AWS	n.a.	-	-	c	t	im	op
Shop-System	IO	IO2	1 AWS	n.a.	-	-	c	t	im	op
Order Export Process	D/M	OEX	-	-	-	-	p	i	m	uop
Product Import Process	D/M	PI	-	-	-	-	p	i	m	uop

HR: Humanressource, D/M: Document/Model, H/W: Hilfsmittel/Werkzeug, p: provider, c: customer, t: tangible, i: intangible, uop: usage operator, op: operand, ZE: Zeiteinheit, N: Nutzer, n.a.: nicht angebbar,-: Merkmal nicht ausgeprägt, coup: coupled,IO: Integrationsobjekt, AWS: Anwendungssystem, Stck: Stück, Pers: Person, m: mobil, im: immobile, OEX: Order Export, PI: Product Import

Tabelle 6.3: Ressourcenmodell des Integrationsdienstleisters

könnte der Eindruck entstehen, dass es sich um einen separaten, einmaligen Schritt handelt. Undeutlich bleibt, welche Veränderungen an dem Integrationsobjekt Shop-System zu erbringen sind, um von diesem die Web-Services des ERP-Systems aufrufen zu können. Es ist von einer Erweiterung und damit einer Veränderung des Integrationsobjektes Shop-System auszugehen, gekennzeichnet durch die Veränderung der Ressource von Shop-System() zu Shop-System(erweitert). Alle anderen Veränderungen von Ressourcen sind analog gekennzeichnet. Es wird jedoch darauf verzichtet, eine Abnutzung z. B. den Zeitverbrauch einer Humanressource als Veränderung der Ressource darzustellen. Stattdessen wird davon ausgegangen, dass die Ressourcen über die gesamte Dauer der Servicekomponente eingesetzt werden und sich deren Zeitbeanspruchungen anhand der Dauer der Komponente ablesen lassen. Nachdem nun die Servicekomponenten mit ihren Eigenschaften betrachtet wurden und bereits die Ressourcen mit ihren Zustandsänderungen verzeichnet sind, wendet sich die Tabelle 6.3 den Eigenschaften der Ressourcen zu. Dabei werden wieder die Eigenschaften genutzt, welche in Kapitel 4.2 identifiziert wurden.

Für die in dem untersuchten Fall vorliegende Ausprägung einer Integrationsdienstleistung können die nachfolgend beschriebenen Ressourcen mit ihren Eigenschaften festgehalten werden. Es finden sich fünf Ausprägungen von Humanressourcen. In vier Fällen handelt es sich um Ressourcen des Integrationsdienstleisters und in einem Fall um eine Ressource des Kunden. Bei der Kundenressource handelt es sich um den IT-Leiter, welcher in das Integrationsprojekt maßgeblich in der Komponente Spezifikation/Abstimmungen eingebunden ist. Die Quantität der Ressource (Quantity) wird bei den Humanressourcen in Form von Manntagen angegeben, wobei dabei von einer normalen wöchentlichen Arbeitszeit von 40 Stunden ausgegangen wird und somit ein Manntag gleich acht Stunden ist. An dieser Stelle lässt sich eine Abhängigkeit der Art der Kapazität vom Typ der Ressource feststellen. Diese ändert sich, wenn man Ressourcen vom Typ Hilfsmittel oder Werkzeug betrachtet, denn hier wird die Kapazität – wie im Fall von Eclipse – in Form von Nutzern pro Zeiteinheit angegeben. Die Quantität diese Ressourcen Typs ließ sich für den untersuchten Fall nicht bestimmen. Für die Humanressourcen hingegen war die Zahl der Personen mit einer entsprechenden Qualifikation ermittelbar. Es lassen sich Humanressourcen wie Java-Entwickler und Datenbankentwickler finden, welche dem gemeinsamen Obertyp Entwickler zugeordnet werden können und eine wechselseitige Kopplung mit Ressourcen vom Typ Hilfsmittel aufweisen. Ein Entwickler mit der Qualifikation Java ist an die Ressource Eclipse gekoppelt, da er diese Entwicklungsumgebung benötigt um seine Leistung erbringen zu können. Gleiches gilt in Bezug auf den SQL-Developer für die Ressource des Datenbankentwicklers. Umgekehrt gilt diese Kopplung natürlich auch. Dies bedeutet, dass die Ressourcen in den meisten Fällen nur gemeinsam eingesetzt werden können. Die Ausnahme bilden die Humanressourcen Projektleiter als auch der Account Manager, für welche keine derartigen Kopplungen in dem vorliegenden Fall bestehen. Es lassen sich für alle Ressourcen keine maßgeblichen Hierarchien feststellen, was daran liegt, dass hier eine konkrete Dienstleistung für einen dedizierten Kunden betrachtet wird. Ausgestaltungen ähnlicher Dienstleistungen, die sich aus einem Modell mit entsprechenden Konfigurationsoptionen ableiten lassen, werden hier nicht betrachtet. Dazu müsste eine Vielzahl an Dienstleistungen des Dienstleisters untersucht und aus den Einzelausprägungen ein Modell erstellt werden, aus dem sich diese ableiten lassen. Dies würde dann auch abstrakte Ressourcen wie die – bereits angedeutete – Ressource eines Entwicklers implizieren, aus der sich dann verschiedene Spezialisierungen hinsichtlich der Beherrschung von Programmiersprachen ableiten lassen. Die Anfassbarkeit der Ressourcen lässt sich z. B. bei Humanressourcen eindeutig bestimmen. Im Fall des Angebots oder der Modelle des Order Export und Product Import kann diese Eigenschaft kontrovers diskutiert werden. Zwar kann das Papier, auf dem diese dokumentiert sind angefasst werden, sind diese jedoch nur digital vorhanden, so ist diese Argumentation hinfällig, weshalb diese Ressourcen als nicht-anfassbar (intangible) einzustufen sind. Bis auf die Anwendungssysteme, die Integrationsobjekte, werden alle Ressourcen als mobil eingestuft. Die Anwendungssysteme weisen eine weitere Besonderheit auf, die aus der Tabelle nicht deutlich wird. Sie können in verschiedenen Ausprägungen existieren. Je nach dem ob ein Entwicklungssystem, ein Testsystem und ein Produktivsystem unterschieden werden existieren verschiedene Instanzen dieser Ressourcen. Wird sich auf das Produktivsystem bezogen – welches typischerweise beim Kunden vor Ort betrieben wird – so wird dies mit (K) gekennzeichnet, also ERP-System(K). Damit erfolgt eine Bezugnahme auf eine domänenspezifische Eigenschaft. Die letzte zu diskutierende Eigenschaft ist Activity. Bis auf Angebot und die Integrationsobjekte weisen alle anderen Ressourcen Usage-Operator aus, was bedeutet, dass diese nicht verändert werden. Im Gegensatz dazu stehen die Integrationsobjekte, welche bei der durch den Dienstleister angestrebten Integration verändert werden müssen und deshalb mit Operand gekennzeichnet sind. Die Ressource Angebot ist ambivalent, da sie in der Teilkomponente in der sie erstellt wird, natürlich mit Operand zu bewerten ist, danach aber nicht mehr verändert, sondern genutzt wird

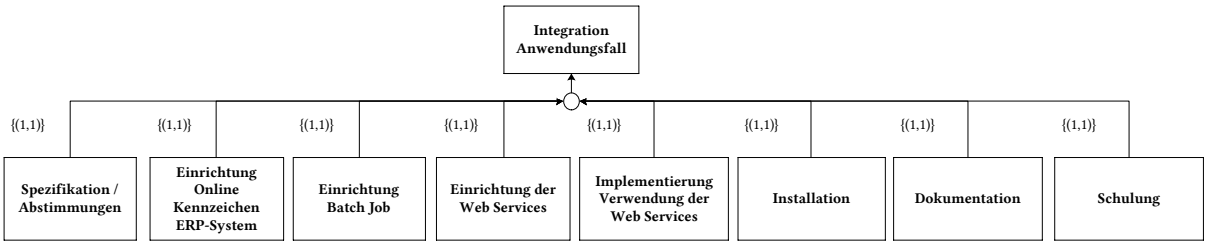


Abbildung 6.8: Produktmodell der Integrationsdienstleistung

und damit die Bewertung Usage-Operator zutreffender ist. Die Eigenschaft Activity ist demnach von der Servicekomponente abhängig.

Das Produktmodell der betrachteten Dienstleistung enthält keine Überraschungen. Der Dienstleister verwendet jede Servicekomponente genau einmal und formt damit die Dienstleistung. Die Abbildung 6.8 verdeutlicht dies durch die dargestellte Kardinalität 1:1. Wie bereits beschrieben, wird momentan eine konkrete Dienstleistung betrachtet. Erstellt man ein Dienstleistungsmodell für mehrere und für Varianten einer Dienstleistung bekommen die Kardinalitäten eine stärkere Bedeutung. Das Prozessmodell der Integrationsdienstleistung wird nicht explizit dargestellt, da es einem einfachen linearen Durchlaufen der Servicekomponenten – in der Reihenfolge ihrer Darstellung in Tabelle 6.2 – entspricht.

Bisher wurde nicht auf die Art des zweiten Integrationsobjektes eingegangen. Dies geschah, um zunächst das Ressourcenmodell analysieren zu können, da sich mit diesem die Wahl des Shop-Systems durch den Dienstleister leichter begründen lässt.

Dem Kunden wurde der Einsatz des auf Plone basierten EasyShop-Systems vorgeschlagen. Bei Plone handelt es sich um ein Content-Management-System, dass durch EasyShop um Shop-Funktionalität erweitert wird. Plone setzt wiederum auf dem Web-Application-Framework Zope auf, welches in Phyton geschrieben ist. Bei Python handelt es sich um eine interpretierte Sprache welche für die Java- und .Net-Virtual-Machine portiert wurde. Die Auswahl dieses Shop-Systems basiert auf der Qualifikation einzelner Personen des Dienstleisters, weshalb sich diese Qualifikation nicht in dem Ressourcenmodell widerspiegelt. Andere Shop Systeme, welche mit PHP realisiert wurden haftet aus der Sicht des Dienstleisters der Makel an, dass die genutzte Programmiersprache nicht professionell ist. Im Gegensatz zu dieser Einsicht steht die Tatsache, dass nahezu alle erfolgreichen Shop-Systeme mit PHP realisiert wurden. Plone, und damit EasyShop bringt – mit einer Web-Service-API – eine einfache Möglichkeit mit, die es erlaubt die gewählte Integrationsstrategie – des direkten Web-Service-Aufrufs durch die Integrationsobjekte – einfach zu realisieren.

Nachdem nun die Integrationslösung und die zugehörige Dienstleistung dargestellt wurde, wird die Lösung kurz bewertet. Sie steht in der Tradition aller anderen bereits bei dem Kunden in Betrieb befindlichen Integrationsbeziehungen, bei denen es sich ausnahmslos um Punkt-zu-Punkt Verbindungen handelt. Deshalb gilt die hier vorgetragene Kritik an den neuerlich zu realisierenden Beziehungen zwischen dem ERP-System und dem Shop-System stellvertretend für alle weiteren Beziehungen, die bereits aktiv sind. In dem Dienstleistungsmodell und in der Beschreibung der technischen Details der Integrationsbeziehungen findet keine explizite Berücksichtigung von notwendigen Transformationen statt. Diese könne sowohl bei dem Import von Artikeldaten als Produktdaten in das Shop-System als auch bei dem Export der Bestellungen notwendig sein. Die Transformationslogik verteilt sich so über die Integrationsobjekte, was es schwierig macht Wiederverwendungseffekte einzelner Transformationen zu nutzen. Darüber hinaus belastet die notwendige Rechenleistung für die Realisierung der Transformationen die

Integrationsobjekte. Vermutlich ist ERP-System – aufgrund seiner zentralen Stellung – davon besonders betroffen. Dies kann aber nicht mit abschließender Gewissheit gesagt werden, da sich aus den Beschreibungen nicht herauslesen lässt, ob eine Transformation vor oder nach dem passieren der Anwendungssystemgrenze erfolgt. Die bisherige Vorgehensweise des Dienstleisters sieht jedoch vor, für etwaige Datenerfordernisse externer Systeme auf der Ebene der Oracle Datenbank eine Sicht zu erzeugen, welche die Daten in der benötigten Form enthält, was eine überproportionale Belastung des ERP-Systems darstellt. Dies geschieht zumeist aus dem Grund, dass andere Systeme sich aus der Sicht des Dienstleisters als unkontrollierbar und mit den vorhandenen personellen Ressourcen als schwer veränderbar darstellen. Das selbst entwickelte ERP-System unterliegt in dieser Hinsicht keinen Beschränkungen. Diese Vorgehensweise bringt jedoch einen weiteren Effekt mit sich. Jedes ERP-System eines Kunden wird dadurch zu einem Unikat und verliert seine Releasefähigkeit. Eine Aktualisierung auf ein neues Release kann nicht automatisch vollzogen werden, da zuvor alle kundenindividuellen Anpassungen zu prüfen sind. Die hier dargestellten Prozesse, welche durch die Integrationsbeziehungen realisiert werden, zeigen auf, dass ihre Bedeutung für das Kundenunternehmen eine andere Vorgehensweise nicht zulässt. Sie müssen in jedem Fall funktionsfähig sein. Diese Kritik betrifft auch die anderen Integrationsobjekte und hängt mit jeder Veränderung der Integrationsobjekte zusammen. Begründet werden kann das Vorgehen des Dienstleisters damit, dass versucht wird die Wichtigkeit des Dienstleisters für den Kunden dadurch zu steigern, in dem ein großer Teil der IT-Landschaft des Kunden von dem ERP-System abhängig gemacht wird. Die Punkt-zu-Punkt Verbindungen führen auch zu einer schlechten Wartbarkeit, da z. B. Änderungen am ERP-System zu Änderungen an der – in den anderen Integrationsobjekten implementierten – Integrationslogik führen können. Damit müssen bei einer Veränderung eines Integrationsobjektes alle anderen Integrationsobjekte potenziell auch verändert werden. Mit jeder neuen Verbindung, die hinzukommt, wird diese Problematik weiter verstärkt. Auf der anderen Seite sind diese Beziehungen natürlich außerordentlich stabil. Fällt eine dieser Komponenten aus, so funktionieren alle anderen Beziehungen nach wie vor. Dies gilt allerdings nur mit der Ausnahme des ERP-Systems, welches eine zentrale Stellung einnimmt und bei einem Ausfall alle Integrationsbeziehungen beeinflusst und zu deren Ausfall führt.

In der weiteren Darstellung wird auf die Lösungsauswahl mithilfe der erarbeiteten Klassifikation und den weiteren Generalisierungen der Integrationsmuster eingegangen. Es wird aufgezeigt, wie das vorliegende Problem des Kunden auf einer abstrakteren Ebene gelöst werden kann. Dies erfolgt durch einen Architekturentwurf und erst im Anschluss durch die konkrete Auswahl von Hilfsmitteln. Nach dieser Darstellung der Veränderungen auf der konkreten Ebene eines Integrationsproblems wird dargelegt, wie sich dadurch das Dienstleistungsmodell des ERP-System-Herstellers verändert und welche Auswirkungen dies auf seine weitere Tätigkeit hat. Obwohl bei der Analyse des konkreten Kundenfalls die Konzentration auf den neu zu realisierenden Integrationsbeziehungen zwischen dem ERP-System und dem Shop-System lag, wird einen Exkurs angeschlossen, wie sich die Unternehmensarchitektur des Kunden durch die eingeführte Veränderung der Integrationsarchitektur weiter verbessert lässt, in dem weitere der bestehenden Punkt-zu-Punkt Verbindungen einer Revision unterzogen werden.

6.4 Lösungsauswahl auf der Basis hochstehender Integrationsmuster

Die Lösungsauswahl auf der Ebene der Integrationsmuster setzt voraus, dass die Integrationsprobleme auf der Abstraktionsebene der Integrationsmuster erfasst werden. Zu diesem Zweck sind noch einmal die zu realisierenden Integrationsbeziehungen der Abbildung 6.5 zu betrachten. Zwischen einem noch näher zu bestimmenden Integrationsobjekt Shop-System und dem

ERP-System des Dienstleisters sind drei Integrationsbeziehungen zu realisieren: Product Import, Verfügbarkeitsabfrage und Order Export. Alle drei geben sowohl den Kontrollfluss, weiter und haben zusätzlich eine Datenübergabeproblematik.

Zunächst werden die **Kommunikationseigenschaften** analysiert. Sowohl der Produktimport als auch der Bestellexport eignen sich nach den Vorgaben des Kunden für eine asynchrone Kommunikation. Genauer gesagt bezieht sich diese Information eigentlich auf die Interaktionsauslösung, welche lediglich einmal am Tag erfolgen muss. Über die eigentlichen Eigenschaften der Synchronität also die zeitliche Kopplung wird nichts gesagt. Ein täglicher Abgleich lässt sich sowohl synchron als auch asynchron abwickeln, da dies eine Aussage darüber darstellt, dass die Integrationsobjekte zur gleichen Zeit in Betrieb sein müssen. Es lassen sich also wie vom Dienstleister vorgeschlagen asynchrone Verfahren welche für Time/Synchronity den Fokus td aufweisen anwenden. Bis zu diesem Punkt deckt sich die Analyse mit der des Dienstleisters.

Für die Integrationsbeziehung Verfügbarkeitsabfrage wurde durch den Dienstleister eine synchrone Ausführung vorgeschlagen. Das Shop-System soll warten, bis es eine Antwort vom ERP-System erhalten hat, und es so in die Lage versetzt wird, die Verfügbarkeitsinformationen anzuzeigen. Zunächst ist anzumerken, dass es sich hier nicht um eine zeitliche Kopplung der Integrationsobjekte handelt, sondern dass eigentlich ein blockierendes Senden verlangt wird. Die Fortführung des Bestellvorganges im Shop-System soll bis zum Eintreffen der Antwort des ERP-Systems angehalten werden. Dabei soll nach Vorschlag des Integrationsdienstleisters diese Information direkt aus dem ERP-System geholt werden, was wiederum Last auf diesem erzeugt. In der Beispielimplementierung wird zwar dieser Zugriff auf das ERP-System realisiert werden, es soll jedoch darauf hingewiesen werden, dass diese Ausgestaltung aus Sicht des Geschäftsprozesses angezweifelt werden muss. Sie ergibt sich so nicht zwingend aus dem zu unterstützenden Geschäftsprozess. Der Terminologie des Kundenunternehmens folgend, entsteht eine definitive Auftragsbestätigung – und damit die Zusicherung eines Liefertermins – erst bei einem manuellen Versand einer Auftragsbestätigung. Dieser Schritt ist dem Order Export und damit der Verfügbarkeitsabfrage nachgelagert und wird durch das ERP-System unterstützt. Die bevorzugte Integrationsmethode des zeitversetzten Exports der Bestellungen aus dem Shop-System an das ERP-System führt aber zu dem Effekt, dass die Reihenfolge des Exportes und die Reihenfolge der Abarbeitung im ERP-System maßgebend dafür werden, in welcher Reihenfolge die Bestellungen den Lagerbestand der bestellten Artikel dezimieren. Daraus bestimmen sich etwaige Lieferverzögerungen, sofern ein Lagerbestand unterschritten wird. Allein durch diese Konstellation hat die Aussage der Verfügbarkeitsabfrage eine geringe Aussagekraft, da nach einer Bestellung bei der eine positive Verfügbarkeit signalisiert wurde, eine weitere Bestellung erfolgen kann, bei der ebenfalls eine positive Verfügbarkeitsinformation an den Kunden gegeben wird. Die Bestellungen wurden bis zu diesem Zeitpunkt nicht exportiert und so wurde die Bestellmenge zwei Mal mit demselben Lagerbestand verglichen. Sind nun beide Bestellungen so groß, dass sie den Bestand eines Artikels auf null reduzieren, so kann die, zweite Bestellung nicht mehr ausgeführt werden. Der Kunde – das Tochterunternehmen – erhält eine Auftragsbestätigung, welche die Wiederbeschaffungszeit des Artikels ausweist. Das widerspricht aber der zuvor im Shop System angezeigten Verfügbarkeit.

Daraus lassen sich die folgenden Schlüsse ziehen. Eine synchrone Verfügbarkeitsabfrage ist nutzlos, wenn keine Reservierung von Lagerbeständen im ERP-System erfolgt. Zumal in dem vorliegenden Szenario völlig offen gelassen wird, ob der Lagerbestand auch durch andere Vertriebskanäle verringert werden kann. Die Reihenfolge des Bestelleinganges muss bei der Übermittlung und weiter Verarbeitung im ERP-System berücksichtigt werden. Die Lagerbestände können auch einmal am Tag in das Shop-System importiert werden (asynchron), der Informationsgehalt würde sich dadurch nicht verändern.

Bis jetzt werden also drei Muster benötigt, welche die Lieferung realisieren. Die Richtung des Datenflusses ist bei zwei Beziehungen gleich, wobei diese sich dahin gehend unterscheiden, dass eine – die Verfügbarkeitsabfrage – für die Facette Send die Eigenschaft bs verlangt. Die dritte Integrationsbeziehung ist entgegengesetzt. Die zwei Integrationsbeziehungen – die einander entgegengesetzt sind – können durch Muster der Facette Time/Synchronity mit der Eigenschaft td gelöst werden.

Für die Eigenschaft bs ist anzumerken, dass sich die Auswahl bei diesen Facetten schwierig gestaltet, da sie nur eine prinzipielle Unterstützung anzeigen. Ein Muster mit dem Fokus nbs unterstützt diese Ausgestaltung, kann häufig aber auch in der Form bs eingesetzt werden, da es sich dabei um ein Merkmal der Ausgestaltung auf der Aufruferseite handelt. Wird dort gewartet (bs) kann die Übermittlung der Informationen auch mit einer asynchronen Technologie wie Nachrichten erfolgen.

6.4.1 Auswahl der Muster mit Hilfe der Klassifikation

Würde auf der Basis dieser Charakteristika nach Mustern gesucht, wären die folgen Ergebnisse das Resultat. Aus Platzgründen ist die vollständige Tabelle der Klassifikation – welche die Basis für die hier vorgenommene Auswahl bildet – in Anhang C enthalten.

Das Kriterium der Lösung eines Lieferproblems wird durch 22 Muster erfüllt. Diese Menge schränkt sich für den Fokus td der Facette Time/Synchronity auf 14 Muster ein, wobei dazu noch zwei Muster hinzukommen. Für die zwei zusätzlichen Muster konnte die Eigenschaft Time/Synchronity im Rahmen der Klassifikation nicht spezifiziert werden. Diese sollten dennoch zusätzlich auf die Erfüllung der Eigenschaft geprüft werden, da – auf Basis einer unvollständigen Beschreibung – nicht ausgeschlossen werden kann, dass sie diese erfüllen. In der Summe wären also bis jetzt 16 Muster zu untersuchen und kämen als potenzielle Lösungen infrage.

Ein direktes bs erfüllen nur zwei Muster. Geht man jedoch davon aus, dass ein Muster mit der Eigenschaft nbs auch ein blockierendes Senden realisieren kann, so kommen zu den zwei Mustern 17 weiter hinzu. Berücksichtigt man dann noch, dass es wieder Muster gibt, bei denen sich die Eigenschaft aus der Quellenanalyse nicht ermitteln lies, aber ebenso wenig ausgeschlossen werden konnte, so sind zusätzlich drei Muster zu analysieren. In der Summe wären also 22 Muster für diesen Fall zu untersuchen.

Diese Menge an Mustern ist aber verteilt über unzählige Publikationen beschrieben, womit das Beispiel eindrucksvoll das Auswahlproblem illustriert, welches zu der – in der empirischen Untersuchung[9] festgestellten – geringen Nutzung von Mustern bei der Integration führt. Die Menge der in der Praxis zu berücksichtigenden Muster ist weitaus größer, da die Menge der hier untersuchten Muster nachgewiesener Maßen sehr unvollständig ist und nur 66 Einzelbeschreibungen von Mustern enthält. Die bestehende Klassifikation vereinfacht bereits den Zugang, da nicht alle Muster geprüft werden müssen. Es ist ausreichend Muster zu untersuchen, welche der Lieferung zuzuordnen sind. Damit ist es gelungen einen ersten Fortschritt zu erzielen, denn in der Praxis steht eine solche Klassifikation derzeit nicht zur Verfügung. Der Einsatz der Klassifikation führt zu einer zielgerichteten, begründbaren Auswahl und damit zu einem fundierten Architekturentwurf.

Wird die Klassifikation aber im Detail betrachtet, so ist festzustellen, dass zahlreiche Varianten von Mustern beschrieben werden, die einem gemeinsamen Konzept folgen. Sie weisen lediglich Detailunterschiede auf, die zu unterschiedlichen Musterbeschreibungen führen. An dieser Stelle

[9] Vgl. Kapitel 3.

lassen sich die Ergebnisse der weiteren Generalisierung aus Kapitel 5.5 verwenden, was im nachfolgenden Abschnitt detailliert dargelegt wird.

6.4.2 Auswahl der Muster auf Basis der generalisierten Konzepte

Dieser Abschnitt legt die Auswahl der Muster mithilfe der hochstehenden Integrationsmuster bzw. der generalisierten Konzepte dar. Dabei wird auch das noch ausstehende Integrationsobjekt Shop-System festgelegt und der daraus resultierende logische Architekturentwurf vorgestellt.

Um die generalisierten Konzepte in Anwendung zu bringen, ist es sinnvoll sich noch einmal die Schwächen des üblichen Lösungsansatzes des Dienstleisters zu vergegenwärtigen. An dieser Stelle gehen wird nur auf die Erfordernisse der Lieferung eingegangen, über Probleme der Transformation oder des Zugriffes kann noch keine Aussage getroffen werden, da das Integrationsobjekt Shop-System noch nicht genau bestimmt wurde. Die Interaktionslogik verteilt sich über alle Integrationsobjekte und unterliegt keiner zentralen Kontrolle, was zu einer schlechten Wartbarkeit führt. Weiterhin ist die Stellung des ERP-Systems zu zentral. Diese Aussage bezieht sich nicht auf die fachlich zentrale Stellung dieses Systems, sondern auf die Zentralität in der Abwicklung der Kommunikation. Zusätzlich soll mit dem Architekturvorschlag eine Perspektive für eine Überarbeitung der bereits bestehenden Integrationsbeziehungen gegeben werden.[10]

Die genannten Nachteile werden durch die Muster des Typs Vermittler vermieden.[11] Alle Muster dieses Typs verwenden das Konzept einer zentralen Komponente, welche für die Weiterleitung der Kommunikation verantwortlich ist. Die Abbildung 5.11 zeigt die Spezialisierung und damit die Muster, welche diesem Prinzip folgen. Im Gegensatz zur Auswahl auf der Basis der Klassifikation wurde nun die Auswahl weiter vereinfacht, da auf der Basis des generalisierten Konzeptes ausgewählt und das hochstehende Integrationsmuster im Anschluss anhand der Anforderungen weiter verfeinert werden kann. Es wird also von dem zu lösenden Integrationsproblem ausgegangen, ein lösendes Konzept ausgewählt und dieses sukzessive durch die Anwendung der Anforderungen verfeinert, um eine konkrete Ausprägung des Konzeptes zu selektieren.

Eine wesentliche Anforderung in dem vorliegenden Integrationsszenario ist die Art der Kommunikation auf Web-Services und die damit verbundenen Prinzipien zu beschränken. Diese Art der Kommunikation legt die Eigenschaft der zeitlichen Entkopplung fest. Prinzipiell kommen dafür alle Varianten infrage, bei denen es sich um Message-Queuing handelt. Request-Reply kann aufgrund der mangelnden zeitlichen Entkopplung ausgeschlossen werden, wäre aber geeignet die Integrationsbeziehung, bei der ein blockierendes Senden gefordert ist zu lösen. Es kommen also für die zeitlich entkoppelten Integrationsbeziehungen Varianten des Integration Messenger und der Message Router infrage. Zur Realisierung des blockierenden Sendens die Realisierungen des Brokers und Mediators. Damit hat sich die Menge der zu prüfenden Muster auf 10 reduziert, was im Vergleich zu der Auswahl mithilfe der Klassifikation – von 22 Mustern – eine Reduktion um über die Hälfte darstellt. Perspektivisch kann auch noch der Integration Mediator in Betracht gezogen werden, da dieser explizit eine Zentralisierung der Interaktionslogik umsetzt. Auch mit dieser Erweiterung wäre immer noch eine Reduktion der infrage kommenden Muster um 50% gegeben.

Für die Bestimmung weiterer Integrationsmuster ist zunächst das Integrationsobjekt des Shop-Systems näher festzulegen. Der Vorschlag des Integrationsdienstleisters wurde bereits als nicht optimal eingestuft, da ein Shop-System zum Einsatz kommen soll, welches die Qualifikation Python-Programmierung erfordert. Diese Qualifikation ist aber nur bei einem Mitarbeiter vorhanden und die Unternehmensleitung sieht keine Veranlassung, diese in breiter Menge auszubauen.

[10] Vgl. Abbildung 6.2.
[11] Vgl. Kapitel 5.5.2.

Dem Ressourcenmodell zur Folge gibt es aber Mitarbeiter, welche die Qualifikation Java Programmierung aufweisen. Diese ist – im Gegensatz zur Python breiter im Unternehmen einsetzbar, da bereits die Web-Option des ERP-Systems (siehe Abbildung 6.4) und der neue Client des ERP-Systems diese Qualifikation benötigen. Das ERP-System stellt das Kernprodukt des Dienstleisters dar, weshalb davon ausgegangen werden kann, dass es eine langfristige Bestrebung geben wird, diese Qualifikation im Unternehmen vorzuhalten. Kann auf diese Qualifikation im Rahmen der Einrichtung und Anpassung einer das ERP-System begleitenden Dienstleistung zurückgegriffen werden, was den Einsatz bereits vorhandener Personalressourcen – impliziert, so ist dies als der bessere Ansatz zu sehen.

Es wird also versucht das vom Dienstleister vorgeschlagene Shop-System durch eines zu ersetzen, welches in Java geschrieben ist. Zum Zeitpunkt der Durchführung der Fallstudie war die Auswahl an Systemen, die diese Anforderung erfüllten gering. Die Masse der Systeme setzt auf PHP als Sprache. Neben einem kommerziellen System der Enfinity Suite von Intershop konnten zwei Open-Source-Varianten gefunden werden, welche keine Lizenzkosten verursachen würden. Der ursprüngliche Vorschlag des Dienstleisters setzte ebenfalls auf dieses Lizenzmodell. Es handelt sich bei den Systemen um Apache OFBiz[12] und den 123JavaShop.[13] Apache OFBiz wird von der Apache Software Foundation als Community Projekt betrieben und hat im Vergleich zu dem 123JavaShop einen umfassenderen Ansatz. Es bietet neben einer Shop-Komponente[14] weitere Komponenten, die für ein ERP- und Customer-Relationship-System benötigt werden. Aufgrund der fortschrittlichen Architektur fiel die Entscheidung für OFBiz und der geplante Python-Shop wurde dadurch ersetzt. OfBiz ist in verschiedenen Versionen verfügbar, so gibt es die Version 9.04 und 10.04. Die Version 9.04 weist jedoch Probleme im Umgang mit komplexen Datentypen auf, die bei der vorliegenden Integration die Arbeit erheblich erleichtern, da ein Verzicht auf diese zu Performance Einbußen sowie einer schlechteren Übersichtlichkeit und Handhabbarkeit führen würde. Die Version 10.04. weist im Vergleich zu den aktuellen Entwicklerversionen Nachteile im Serviceframework auf, welches umfassend überarbeitet wurde. Aus diesem Grund wurde die zum Zeitpunkt der Fallstudie aktuelle Entwicklerversion von OFBiz verwendet. In der zwischen Zeit hat sich der Versionsstand auf die Version 10.04.02 verändert. In Abbildung 6.9 ist eine knappe Zusammenfassung der Technologien von OFBiz dargestellt. Aus dieser lässt sich ersehen, dass OFBiz eine Menge an modernen Technologien vereinigt. OFBiz ist modular aufgebaut, wobei die Abhängigkeiten zwischen den einzelnen Modulen groß sind. Diese sind stark miteinander verwoben. So werden für einen Betrieb des Point-of-Sale-Moduls die Module Auftragsverwaltung, Lagerverwaltung und Einkauf weiterhin benötigt. Eine umfassende Deaktivierung verschiedener Hauptmodule, die nicht benötigt werden, war nicht realisierbar. Soweit als möglich wurden jedoch Spezialmodule deaktiviert. Dennoch ist ein Betrieb von OFBiz als reine Shop Lösung nur unter der Nutzung des Point-of-Sale Moduls möglich.

Für die integrierte Katalogverwaltung sind Schnittstellen für den Import und Export von XML-Dateien und CSV-Dateien verfügbar. Es lassen sich beliebige CSV-Formate durch ein Import-Werkzeug importieren, wobei das jeweilige Dateiformat mittels einer XML-Datei beschrieben wird. Ebenfalls ist der Zugriff per RMI und SOAP ist ebenfalls möglich. Dabei bleibt anzumerken, dass Schnittstellen in Form eines Service in OFBiz erst verfügbar gemacht werden müssen. Die technologische Basis wird mitgeliefert, der Service ist jedoch über eine Beschreibung erst zu spezifizieren. Für den Einsatz der generalisierten Muster bedeutet dies, dass ein Zugriffspunkt benötigt wird. Da die Verbesserungen möglichst nah an dem Vorschlag des Dienstleisters liegen sollen, kommt nur eine Web-Service basierte Schnittstelle infrage. Aus den Spezialisierungen des Musters Zugriffspunkt lässt sich schnell entnehmen, dass es sich dabei um eine Einschränkung

[12] Vgl. (Apache Software Foundation, 2012).
[13] Vgl. (123JavaShop, 2012).
[14] In der OFBiz Domäne Point-of-Sale (POS) genannt.

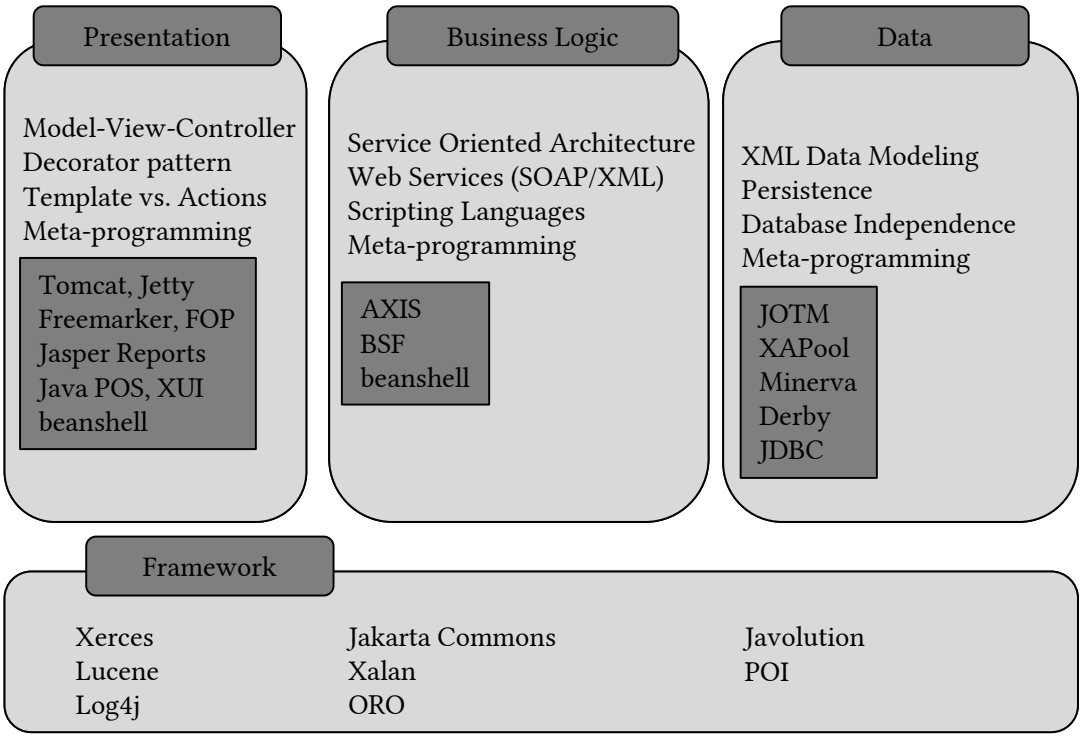


Abbildung 6.9: Architektur und Technologien von Apache OFBiz
Quelle: (Chen, 2005)

der Art des Zugriffspunktes handelt und die Service Facade oder das Service Gateway als einzige Muster möglich sind.[15]

Aus den ermittelten generalisierten Konzepten resultiert das in der Abbildung 6.10 dargestellte abstrakte Bild der zu realisierenden Architektur. Die Abbildung ist auf der Ebene logischer Applikationskomponenten, also dem Application-Layer in der Sprache von Archimate, angesiedelt. Die grau unterlegten Komponenten stellen den durch die Muster spezifizierten Teil der Architektur dar. Der Architekturentwurf weist eine zentrale Vermittlerkomponente auf, welche die Interaktionslogik zentralisiert. Die gestrichelten Pfeile sind Flow Relationen, deren Nummerierung die Reihenfolge und die Richtung der Pfeile die Richtung der Interaktionen kennzeichnet. Der Vermittler stellt *(1)* aktiv eine Nachfrage bei OFBiz nach neuen Bestellungen, welche durch OFBiz *(2)* beantwortet werden. Dazu wird der neu zu gestaltende Zugriffspunkt verwendet. Durch das inhärente Transformationsproblem ist der Vermittler gezwungen *(3)* einem Transformator einen Transformationsauftrag für die erhaltenen Bestellungen zu geben, um diese in das Auftragsformat des ERP-Systems umzuwandeln, welches er in Schritt *(4)* erhält. Nun ist der Vermittler in der Lage den Order-Web-Service des ERP-Systems aufzurufen und *5* das Anlegen der neuen Aufträge zu veranlassen.

Die Nutzung von Transformationsmustern wurde in der Fallstudie ausgeblendet, sie wird aus wissenschaftlicher Sicht im Ausblick diskutiert.[16] Damit sind bis auf die Transformation alle notwendigen Muster auf der Ebene der generalisierten Konzepte und der infrage kommenden konkreten Muster definiert. Die weitere Auswahl richtet sich in wesentlichem Maße nach den Integrationshilfsmitteln, da diese die Realisierbarkeit der Muster beschränken können. Deshalb

[15] Vgl. Abbildung 5.9.
[16] Vgl. Kapitel 8.

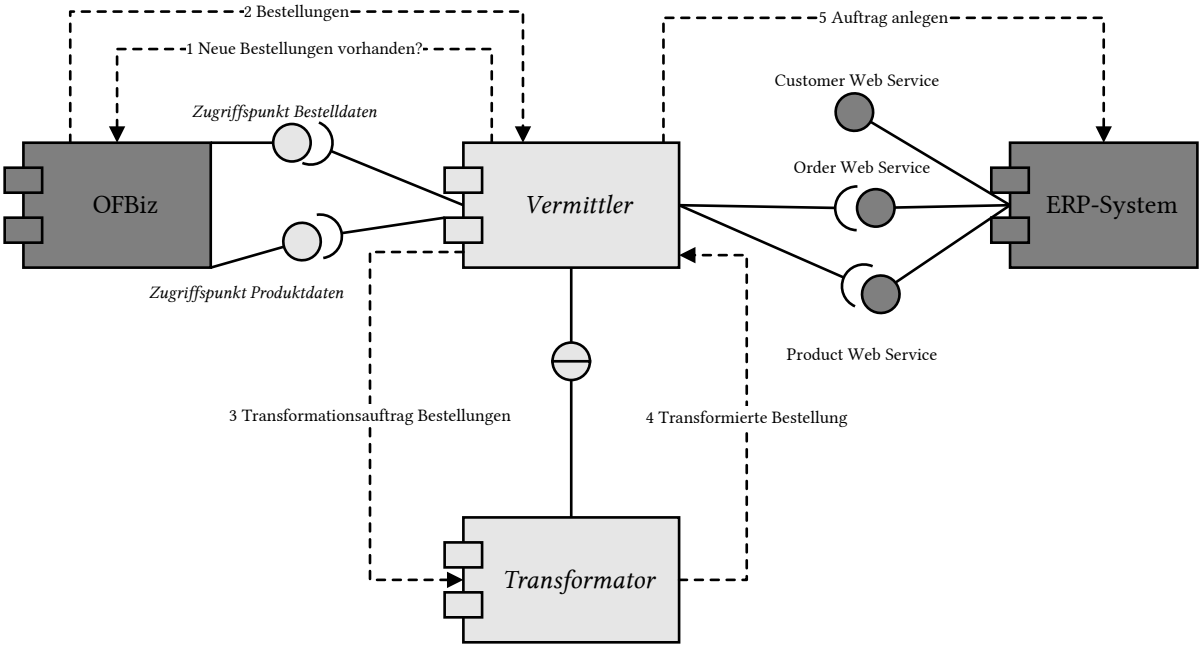


Abbildung 6.10: Logische Architektur bei Verwendung von Zugriffspunkt und Vermittler

stellt das nachfolgende Kapitel die Beispielimplementierung mit der Auswahl eines Integrationshilfsmittels vor.

6.5 Beispielimplementierung im Integrationslabor

Für die ausgewählte Lösung wurde eine Beispielimplementierung im Integrationslabor vorgenommen. Das Integrationslabor ist eine Virtualisierungsumgebung auf VMware ESXi Basis, mit der es möglich ist, verschiedene Integrationsobjekte, sowie Integrationshilfsmittel zu betreiben. Die Virtualisierung gestattet es, für Integrationsobjekte verschiedene Integrationshilfsmittel zu verwenden und so alternative Integrationsszenarien zu realisieren. Durch die Konfigurationsmöglichkeiten des internen Netzwerkes zwischen den virtuellen Maschinen lassen sich auch diesbezüglich realistische Szenarien konstruieren. Der Aufbau und die Überwachung dieser Szenarien werden derzeit manuell realisiert, wobei ein weiterer Ausbau und eine verbesserte Automatisierung und besseres Monitoring anzustreben sind. Dieser Umstand wird in Kapitel 8 weiter diskutiert.

Der nächste Schritt bei der Umsetzung einer praktischen Realisierung ist die Auswahl eines geeigneten Integrationshilfsmittels, welches es ermöglicht eines oder mehrere Muster der Lösungsauswahl zu realisieren. Die Wahl des Shop-Systems wurde bereits kritisch bewertet und eine Veränderung zugunsten eines Java basierten Systems vorgenommen. Die Humanressourcen des Integrationsdienstleisters machen eine Fokussierung auf Java basierte Integrationshilfsmittel aus den gleichen Gründen sinnvoll wie die Auswahl des Shop-Systems. Durch Java als Programmiersprachen werden die vorhandenen Humanressourcen vielfältiger einsetzbar, ein Mitarbeiter kann dann neben der Entwicklung am ERP-System auch bei Integrationsprojekten und E-Commerce- bzw. E-Procurement-Projekten eingesetzt werden. Dieses Vorgehen ermöglicht auch weiterführende Konzepte der Personalwirtschaft wie Job-Rotation, da die Basisqualifikation bei vielen Mitarbeitern vorhanden ist.

Für die Auswahl an Integrationshilfsmitteln auf Java-Basis wäre eine umfangreiche Evaluation notwendig gewesen, da es – im Gegensatz zu den Shop-Systemen – in diesem Bereich eine

weitaus größere Auswahl gibt. Dieser Prozess wurde drastisch verkürzt, in dem die Nennungen der empirischen Untersuchung bezüglich der verwendeten Hilfsmittel ausgewertet wurden. Dort wurde von den Open-Source-Hilfsmitteln der JBoss-ESB mit 6 Nennungen – nach IBM (9 Nennungen), Microsoft (7 Nennungen) und SAP PI/XI (7 Nennungen) – am häufigsten erwähnt.[17] Es handelt sich dabei um ein Java basiertes Hilfsmittel, was dem Ziel des verbesserten Einsatzes der Humanressourcen entspricht. Aufgrund dieser Ergebnisse wurde eine umfangreiche Evaluation vernachlässigt und der JBoss-ESB als Hilfsmittel ausgewählt. Der JBoss-ESB kann sowohl alleinstehend betrieben werden als auch als Teil eines JBoss-Application-Servers. Der ESB spielt besonders bei der Umsetzung des Vermittlers eine Rolle, bevor dies jedoch erläutert wird, findet eine Betrachtung der Umsetzung des Musters Zugriffspunkt statt.

Umsetzung des Musters Zugriffspunkt am Beispiel des Produktimports:
Die Umsetzung der Muster vom Typ Zugriffspunkt erfolgt im Integrationsobjekt OFBiz. Es sind zwei Zugriffspunkte zur Verfügung zu stellen. Einmal ein Zugriffspunkt, welcher in der Lage ist Bestelldaten nach außen zu geben, und zum anderen ein Zugriffspunkt, welcher Produktdaten entgegen nehmen kann. Da eine Beschränkung auf Web Services im Rahmen der Anforderungen formuliert wurde und OFBiz auf einer eigenen Komponentensystematik basiert, war zunächst eine Komponente zu definieren. Die Komponente ERPshowcase beinhaltet hauptsächlich die zwei Web-Services, welche die Zugriffspunkte realisieren, sowie zusätzliche Funktionalität für eine Ampelanzeige der Verfügbarkeit von Produkten. Die Definition einer solchen Komponente erfolgt in Form von XML und bindet im Wesentlichen die Definitionen der weiteren Ressourcen – wie die der Services – in einer Datei zusammen ein. Der Quellcode der Komponente ist im Anhang E in Listing E.1 enthalten. Die dort eingebundene Datei services.xml enthält dann die Definitionen der Zugriffspunkte. Diese beschreiben nur den Zugriffpunkt und stellen eine Indirektion auf die Realisierung dar. Diese erfolgt in Form von Java Klassen, wodurch sich die verbesserte Auslastung der Humanressource Java Entwickler ergibt. Die Realisierung dieser Java-Klassen ist im Anhang in Listing E.3 enthalten.

Der Service importERPProduct realisiert die Zugreifbarkeit der für das Anlegen und Aktualisieren von Produkten benötigten Funktionalität. Die Schnittstelle ist im Anhang E in Listing E.2 enthalten. Für das Anlegen eines Produktes werden ein Produktschlüssel (productId), ein Name (productName), eine Beschreibung (description) und als optionale Angabe auch eine längere Beschreibung (longDescription), sowie eine Kategorie (categoryId) benötigt. Mithilfe des Produktschlüssels wird intern geprüft, ob das bestehende Produkt bereits existiert. Ist dies der Fall, findet nur eine Aktualisierung mit den übermittelten Daten statt. Sofern keine Kategorie angegeben wird, kommt der Standardwert ShowcaseCategory1 zum Tragen. Das Löschen eines Produktes oder die Entfernung aus einer Kategorie ist über den Zugriffspunkt nicht möglich. Bei einem erfolgreichen Import wird der Produktschlüssel zurückgegeben, was einen Abgleich der Schlüssellisten mit dem ERP-System ermöglicht. Die Abbildung 6.11 zeigt die Anzeige der Produkte im Shop-System. Es fällt auf, dass der Preis jedes Produktes auf einen Euro festgesetzt wird. Dies entspricht einem Standardpreis, da zwischen Mutter- und Tochtergesellschaft die Leistungsverrechnung auf andere Weise erfolgt, und Preise keine Rolle spielen. Eine Anlage eines Produktes in OFBiz war ohne sehr tief greifende Änderungen am System nicht ohne Preisangabe möglich, weshalb auf diesen symbolischen Preis ausgewichen wurde. Aus Gründen der Anonymität des Kunden zeigt die Abbildung 6.11 keine typischen Produkte des Kunden, da dies Rückschlüsse ermöglichen würde.

Die Umsetzung der Zugriffspunkte kann als Service Facade oder Service Gateway interpretiert werden, da nicht die gesamte Funktionalität nach außen gegeben wird. Eine finale Unterscheidung der beiden Muster ist nur anhand der Eigenschaften der zeitlichen und räumlichen

[17] Vgl. (Gebauer und Stefan, 2011b, S. 146).

Abbildung 6.11: Anzeige importierter Produkte im Shop-System

Kopplung sowie der Kardinalität möglich. Die Eigenschaft der 1:n-Kardinalität spricht für das Muster Service Facade. Es findet jedoch keine zusätzliche Verarbeitung statt, sodass die Umsetzung auch als Service Interface aufgefasst werden kann. Was ein Hinweis auf eine Erweiterung der in das Konzept Zugriffspunkt eingeflossenen Muster ist.

Der Zugriffspunkt für den Bestellexport wurde auf die gleiche Weise umgesetzt. Das Integrationsobjekt OFBiz wurde nicht weiter verändert, um z.B. eine Auslösung von Interaktionen und damit einen Teil der Interaktionslogik zu realisieren. Mit dieser Entscheidung wurde festgelegt, dass die nachfolgend beschriebene Umsetzung des Vermittlermusters eine Zentralisierung der Interaktionslogik vornehmen muss.

Umsetzung des Musters Vermittler am Beispiel des Produktimports:
Aus der Auswahl an konkreten Mustern des Typs Vermittler erfüllt der Message Router – sowohl in der Variante nach Andersson und Johnson als auch in der nach Hohpe und Woolf – die Eigenschaft einer zentralisierten Interaktionslogik. Diese Eigenschaft geht nur aus der Klassifikation hervor und wurde aus Gründen der Übersichtlichkeit in Abbildung 5.11 nicht mit dargestellt. Die Umsetzung mithilfe des JBoss-ESB folgt einem Modulkonzept. Ein Modul wird in einem Deployment-Container – in einer .esb Datei – zusammengefasst und ähnelt den bekannten Java-Web-Archiven (.war). Sie können zur Laufzeit in das Verzeichnis server/default/deploy kopiert werden. Im vorliegenden Fall sind die folgenden detaillierten Realisierungen enthalten. Eine grundlegende Konfiguration mit der Definition von Services und Providern ist in der jboss-esb.xml enthalten. Die Datei für den Produktimport ist im Anhang in Listing E.4 enthalten. In diesem Artefakt findet sich bereits die Lösung der zeitgesteuerten Interaktionsauslösung. Der ESB bietet die Möglichkeit, zeitgesteuerte Provider zu definieren. Im dargestellten Beispiel wurde zu Testzwecken ein Intervall von 3600 Sekunden verwendet. Diese Datei ist auch für die Umsetzung der zentralisierten Interaktionslogik verantwortlich, hier in Form von sequenziellen Actions.

Für den Produktimport steht aufseiten des ERP-Systems ein WebService zur Verfügung (ERP-ProductWebService). Welcher eine Methode (getItemsByIdRane) enthält, welche eine Menge an Artikeln zurückliefert. Zunächst werden durch ERPProductExportParameterMapping die notwendigen Zugangsdaten in eine Nachricht verpackt, welche dann in der nächsten Action einem

SOAP-Aufruf an das ERP-System übermittelt wird. Die zurückgelieferte Nachricht wird in Form einer Map abgelegt. Die nachfolgende Aktion ist für die Transformation verantwortlich. In der Musterauswahl wurde dieser Aspekt nicht berücksichtigt. Es liegen jedoch vereinzelte Muster wie Data Format Transformation vor, die diese Probleme adressieren. Die Menge der bekannten Muster ist jedoch in diesem Bereich zu gering, was in Kapitel 8 ausgeführt wird. Die Transformation erfolgt mithilfe des im ESB enthaltenen Smooks-Framework, welches auch in den ESBs Synapse und Mule-ESB zum Einsatz kommt. Im vorliegenden Fall wird im Wesentlichen das Java-Binding verwendet, mit dem es möglich ist, den Inhalt der Nachricht in ein Java Objekt umzuwandeln und dieses dann weiter zu verarbeiten. Nach der erfolgreichen Transformation wird der OFBiz-Web-Service aufgerufen, für den ein eigener HTTP-Aufruf umgesetzt werden musste, da der SOAP-Request anzupassen war. Hier kommen die transformierten Daten zum Einsatz, da nun auf Java-Funktionalität zurückgegriffen werden kann, um den Request zu konstruieren. Die Aktion verweist auf eine Java-Klasse, welche die eigentliche Funktionalität umsetzt. Diese ist in Listing E.5 enthalten.

Die beschriebene Realisierung steht exemplarisch auch für den Order Export. Dieser folgt dem gleichen Realisierungsprinzip, ebenso wie die Verfügbarkeitsanfrage, weshalb beide hier nicht noch einmal im Detail erläutert werden. Das folgende Unterkapitel beschäftigt sich mit den Auswirkungen, die sich aus der realisierten Dienstleistung sowohl für den Kunden als auch für den Dienstleister ableiten lassen.

6.6 Auswirkungen des neuen Lösungsvorschlages

Aus der erarbeiteten Lösung ergeben sich mehrere Aspekte, die sich in Auswirkungen für den Kunden und in Auswirkungen für den Dienstleister trennen lassen. Insbesondere die Beurteilung der Auswirkungen für den Dienstleister greift auf das Dienstleistungsmodell der Integration zurück und betrachtet, wie diese singuläre Dienstleistung in ein verstetigtes Angebot überführt werden kann.

Für den Kunden des Dienstleisters kann bewertet werden, inwieweit die Lösung das aktuelle Problem umsetzt und mit den Zielen des Kunden übereinstimmt. Die Kosten der mithilfe der hochstehenden Muster vorgeschlagenen Lösung haben sich – zumindest was die Lizenzkosten angeht – nicht verändert, es kommen ausnahmslos Open-Source-Produkte zum Einsatz, welche keine Lizenzkosten verursachen. Für die Web-Option des ERP-Systems wird vom Hersteller ein zusätzlicher Server empfohlen. Dieser kann – sofern die Ausstattung dies zulässt – auch für den Betrieb des Shop-Systems, sowie den Betrieb des JBoss ESB verwendet werden. Diese Form hat den Nachteil, dass sich die Anwendungssysteme auf demselben physischen System befinden, was im Falle eines Hardwareausfalls als negativ zu bewerten ist. Es werden jeweils separate Hardwareumgebungen vorgeschlagen. Möglicherweise wäre durch die steigende Rechenlast auf der Hardware der Integrationsobjekte – in Folge der durch den Dienstleister vorgeschlagenen Punkt-zu-Punkt Verbindungen – eine Aufrüstung notwendig geworden, die nun alternativ in separate Hardware investiert werden kann. Manuelle Eingaben sind bei der Realisierung der Integrationsbeziehungen nicht notwendig. Die Integrationsobjekte werden im Gegensatz zu dem Vorschlag des Integrationsdienstleisters nicht verändert und behalten deshalb ihre Releasefähigkeit.

Zusätzlich geht von einer Integrationsplattform wie dem JBoss-ESB auch eine Wirkung bezüglich der Architekturentwicklung aus. Werden noch einmal die bestehenden Punkt-zu-Punkt Verbindungen betrachtet so ermöglicht die nun eingeführte Plattform eine Migration der Integrationsbeziehungen, welche im Rahmen des untersuchten Projektes nicht betrachtet wurden. Dies bietet die Möglichkeit die Releasefähigkeit anderer Integrationsobjekte wieder herzustellen, da kundenspezifische Anpassungen zurück gebaut werden können und nicht mehr beachtet

werden müssen. Der Grad der Entkopplung der Integrationsobjekte steigt und sie können sich unabhängig voneinander weiter entwickeln, vorausgesetzt die – für die Integration genutzten – Schnittstellen bleiben weiterhin stabil. Weiterhin kann die Interaktionslogik so an einer zentralen Stelle zusammengeführt werden, was die Kontrolle der bestehenden Integrationsbeziehungen im Unternehmen vereinfacht. Von dieser Strategie geht aber auch ein Nachteil aus, der ESB wird damit zu einem Single-Point-Of-Failure (SPOF). Diesem Problem kann mit gezielter Redundanz begegnet werden, in dem z. B. eine zweite Instanz des ESB eingesetzt wird, welche im Fehlerfall übernimmt. Auch eine wechselseitige Verwendung beider Instanzen mithilfe einer Lastverteilung ist denkbar, erfordert aber zusätzlichen Aufwand. Alles in allem lässt sich sagen, dass die Lösung aus der Sicht die bestehenden Probleme löst und immense Vorteile für die Zukunft bietet.

Für den Integrationsdienstleister eröffnen sich neue Möglichkeiten der Wiederverwendung. Die domänenspezifischen Artefakte der technischen Prozesse – wie der dargestellte Order-Export-Prozess – beschreiben die Interaktionen zwischen Anwendungssystemen unter der Annahme einer spezifischen Funktionsverteilung im Sinne der unterstützten Funktionen eines Geschäftsprozesses. Durch die Zentralisierung der Interaktionslogik entsteht ein Pendant zu dem abstrakten Artefakt, was die Entwicklung wiederverwendbarer Implementierungsartefakte möglich macht.[18]

Darüber hinaus baut der Dienstleister Wissen auf, welches sich in einem veränderten Dienstleistungsbündel niederschlägt. Aus der Umsetzung ergeben sich Veränderungen an den Servicekomponenten und Ressourcen. Die Tabelle 6.4 zeigt das veränderte Ressourcenmodell.

Der Vergleich des veränderten Ressourcenmodells erfolgt mit der Variante aus Tabelle 6.3 und geht von der Erkenntnis aus, dass die Transformation nicht mehr durch das Erzeugen einer Sicht in der Datenbank erfolgen muss. Deshalb kann davon ausgegangen werden, dass der Dienstleister die Web-Option und damit die Web-Services des ERP-Systems standardisieren wird. Diese Situation bildet den Ausgangspunkt für die Betrachtung des veränderten Ressourcenmodells und führt zu einer Veränderung von Activity von op zu uop, das ERP-Sytem wird nicht mehr verändert. Für das Shop-System stellt sich dies anders dar, denn in diesem werden Services verfügbar gemacht. Es kann aber kontrovers diskutiert werden, ob es sich im Fall von OFBiz dabei nicht um eine Konfiguration handelt und nicht um eine Veränderung im engeren Sinne. Weist ein derartiges System eine standardisierte Schnittstelle auf, so verändert sich Activity auch zu uop. Die Humanressource des Datenbankentwicklers kommt durch die veränderte Position der Transformation in der Integrationsarchitektur im Ressourcenmodell gar nicht mehr vor. Man kann jedoch nicht von einer Personaleinsparung sprechen, da die Quantität der Ressourcen vom konkreten Fall abhängt. Hinzugekommen ist das Hilfsmittel JBoss ESB, welches von nun an auch für weitere Integrationsprojekte verwendet werden kann. Selbstverständlich werden die genutzten Muster nun summarisch als Integrationsmuster mit aufgeführt.

Die veränderte Ausgestaltung der Integrationsarchitektur wird auch durch ein verändertes Servicekomponentenmodell deutlich, welches in Tabelle 6.5 im Detail betrachtet wird. Diese Variante wird mit der ursprünglichen aus Tabelle 6.2 verglichen.

Wiederum ist von einer Veränderung des Angebotes des ERP-Systems auszugehen und es kann unterstellt werden, dass dieses in der Zukunft über standardisierte Web-Service-Schnittstellen verfügen wird. Wobei damit nicht nur die prinzipielle technologische Standardisierung –es werden Web-Services mit WSDL-Schnittstelle angeboten – sondern eine inhaltliche Standardisierung meinen. Innerhalb einer solchen Standardisierung wird festgelegt, dass es einen Product-Web-Service mit festgelegten Methoden nebst Input- und Rückgabewerten gibt. Durch die Verwendung des ESB entfällt die Komponente Einrichtung Batch-Job ersatzlos. In diesem Entfallen schlägt sich auch die nicht mehr notwendige Veränderung des ERP-Systems wieder. Darüber hinaus ist davon

[18] Vgl. Kapitel E.

Ressource	Type	Resource identity	Resource quantity	Capacity	Resource Relations	Resource hierarchy	Source Type	Tangibility	Mobility	Activity
Java Entwickler	HR	JE	3 Pers	5 MT pro Woche	coup (ECLI)	-	p	t	m	uop
Account Manager	HR	ACM	3 Pers	5MT pro Woche	-	-	p	t	m	uop
Projektleiter	HR	PL	1 Pers	5MT pro Woche	-	-	p	t	m	uop
IT-Leiter	HR	ITL	1 Pers	n.a.	-	-	c	t	m	uop
Eclipse	H/W	ECLI	n.a.	1 N/ZE	coup (JE)	-	p	i	m	uop
Angebot	D/M	ANG	1 Stck	n.a.	-	-	p	i	m	op/ uop
ERP-System	IO	IO1	1 AWS	n.a.	-	-	c	t	im	uop
Shop-System	IO	IO2	1 AWS	n.a.	-	-	c	t	im	op
Order Export Process	D/M	OEX	-	-	-	-	p	i	m	uop
Product Import Process	D/M	PI	-	-	-	-	p	i	m	uop
JBoss ESB	H/W	ESB	-	-	-	-	p	i	im	uop
Integrationsmuster	D/M	PI	-	-	-	-	p	i	m	uop

HR: Humanressource, D/M: Document/Model, H/W: Hilfsmittel/Werkzeug, p: provider, c: customer, t: tangible, i: intangible, uop: usage operator, op: operand, ZE: Zeiteinheit, N: Nutzer, n.a.: nicht angebbar,-: Merkmal nicht ausgeprägt, coup: coupled,IO: Integrationsobjekt, AWS: Anwendungssystem, Stck: Stück, Pers: Person, m: mobil, im: immobile, OEX: Order Export, PI: Product Import

Tabelle 6.4: Verändertes Ressourcenmodell des Integrationsdienstleisters

auszugehen, dass ein ERP-System, welches mit Web-Option ausgeliefert wird keine Komponente Einrichtung Online-Kennzeichen im Sinne des Hinzufügens einer entsprechenden Eigenschaft in den Datenbanktabellen der Artikel benötigt, sondern dass maximal diese Eigenschaft für, die zu verkaufenden, Produkte gesetzt werden muss, was auch von einem ERP-Nutzer über den Client erfolgen kann. Aus diesem Grund entfällt auch diese Komponente. Eine weitere Folge ist der Entfall der Komponente Einrichtung der Web Services. Auf der Seite der Komponenten, die das Shop-System betreffen, verändert sich die Implementierung Verwendung der Web Services hin zu einer bloßen Einrichtung OFBiz Schnittstelle welche nur im konkreten Fall der Verwendung von OFBiz existiert und bei Shop-Systemen mit bestehender Schnittstelle entfällt. Der Einsatz des ESB führt zu einer neuen Servicekomponente Implementierung der Integrationsbeziehungen, welche die Umsetzung der Integrationsbeziehungen im JBoss ESB erfasst. Da für neue Servicekomponenten und die veränderten Komponenten keine verlässlichen Aufwände abgeschätzt werden konnten, ist diese Eigenschaft mit offen gekennzeichnet. Es kann aber davon ausgegangen werden, dass der Hauptteil des Aufwandes in der Implementierung der Integrationsbeziehungen geleistet werden muss.

Service-komponente	**Ziele**	**Ressourcen**	**lokale und temporale Verfügbarkeit**	**Dauer**	**Konsequenzen**
Spezifikation/ Abstimmungen	Integrations-szenario festlegen	In:IT-Leiter, Out:Angebot	Kunde	4 MT	undefinierter Umfang
Einrichtung OFBiz Schnittstelle	Kommunikations-möglichkeit mit dem Shop-System	In: Shop-System(), Java-Entwickler Out: Shop-System(erweitert)	Dienstleister/ Kunde	offen	Kommunikation nicht möglich
Implementierung der Integrationsbeziehungen	Umsetzung der Integrationsbeziehungen	In: JBoss ESB(), Java-Entwickler, Integrationsmuster Out: JBoss ESB(), Implementierungsartefakte	Dienstleister/ Kunde	offen	Systeme sind nicht integriert
Installation	Lösung ist beim Kunden installiert	In: Java-Entwickler, Datenbankentwickler, ERP-System(K), Shop-System(K) Out: veränderte Unternehmensarchitektur	Kunde	maximal 1,0 MT	Lösung ist nicht installiert
Dokumentation	Lösung ist dokumentiert	In: Lösung Out: Dokumentation	Dienstleister	1,0 MT	Lösung ist nicht dokumentiert
Schulung	Mitarbeiter des Kunden können mit der Lösung umgehen	In: Director Business Solutions, Mitarbeiter Kunde (ungeschult) Out:Mitarbeiter Kunde (geschult)	Kunde	1,0 MT	Mitarbeiter sind nicht geschult

Tabelle 6.5: Verändertes Modell der Servicekomponenten

Der Dienstleister hat nun zusätzliches Wissen im Bereich des ESB-Einsatzes aufgebaut und kann, ausgehend von den hier vorgestellten Modellen der Dienstleistungen, versuchen ein konfigurierbares Modell – und damit auch Dienstleistungsangebot – für Integrationsdienstleistungen zu erstellen. Eine Ausweitung auf andere Bereiche als die E-Procurement Problemstellung ist denkbar. Als erster Anknüpfungspunkt bietet es sich an, die Punkt-zu-Punkt Verbindungen bei dem vorgestellten Kunden zu überarbeiten und auf die neue Integrationsplattform zu migrieren.

Damit kann zusammengefasst werden, dass sich die in dieser Arbeit entwickelte Klassifikation der Muster, die generalisierten, hochstehenden Konzepte und das Dienstleistungsmodell im vorliegenden Anwendungsfall nutzbringend einsetzen ließen. Die Integrationslösung konnte nachvollziehbarer ausgewählt werden und die Wechselwirkungen zwischen Ressourcen untereinander und zwischen Ressourcen und Servicekomponenten konnten aufgezeigt werden. Die Dienstleistungserbringung wurde so qualitativ verbessert und konnte beschleunigt werden, darüber hinaus ist sie für den Dienstleister exakter planbar geworden. In den sich anschließenden Kapiteln wird die Arbeit in Kapitel 7 zusammengefasst, bevor in Kapitel 8 eine kritische Würdigung der Ergebnisse, sowie ein Ausblick auf zukünftige Arbeiten erfolgt.

7 Zusammenfassung

Die vorliegende Arbeit beschäftigt sich mit einer Thematik – der Integration – die den vermeintlichen Zenit in der wissenschaftlichen Beachtung mit dem Abklingen der Enterprise-Application-Integration Welle überschritten zu haben scheint. Bei genauerer Betrachtung offenbart sich jedoch eine fortwährende – und weit über diese Arbeit anhaltende – Bedeutung. Systemintegration ist eine Aufgabe, die in allen künstlichen Systemen auftritt, wenn diese nicht monolithisch sind, sondern auf verschiedene Aufgabenträger zur Erfüllung des Systemzwecks setzen. Für die Gestaltung von Unternehmensarchitekturen und Informationssystemen trifft dies zweifelsohne zu. Die Gestaltung dieser Systeme wird im Rahmen von betrieblichen Informationssystemen sowohl durch die Informatik als auch der Wirtschaftsinformatik aus unterschiedlichen Gesichtspunkten betrachtet. Diese Themen haben natürlich für Unternehmen eine außerordentlich hohe praktische Relevanz, sind sie doch von einer gut funktionierenden Informationsverarbeitung abhängig.

Zwischen dem wissenschaftlichen Diskurs in diesem Themenbereich und der Praxis in den Unternehmen ist eine Kluft auszumachen bezüglich der diskutierten Techniken, Methoden und Technologien.

Wissenschaftlich diskutierte Aspekte sind in der Praxis teilweise irrelevant oder sind nicht wie intendiert anwendbar und müssen erheblich anders eingesetzt werden. Praktische, pragmatische Problemlösungen finden hingegen selten Eingang in die Wissenschaft.

Die Problemstellung der Praxis wird maßgeblich dadurch verstärkt, dass die Organisation der Informationsverarbeitung und die damit zusammenhängende Systemintegration keine Kernkompetenz der meisten Unternehmen ist. Sie sind deshalb gezwungen diese Leistungen am Markt einzukaufen und einen mehr oder weniger großen Teil der Realisierung der Integration in fremde Hände geben. Eine fortschreitende Dynamisierung der Geschäftstätigkeit führt dazu, dass die Systemintegration ebenfalls eine Beschleunigung erfährt und vor allem einem wachsenden Rationalisierungsdruck ausgesetzt ist.

In diesem Spannungsfeld hat sich die Arbeit mit den Fragen beschäftigt, wie einmal erworbenes Lösungswissen konserviert und nutzbringend wieder eingesetzt werden kann und wie die Erbringung von Integrationsdienstleistungen mit IT-Werkzeugen geeignet zu unterstützen ist. Aus diesen Fragen leiteten sich die Forschungsprobleme ab, die untersucht worden. In Integrationsprojekten weisen die Phasen Analyse und Planung ein höheres Gewicht im Vergleich zu der Entwicklungsphase auf, als in Softwareentwicklungsprojekten. Dennoch scheinen Praxisprojekte diese Phasen nur unzureichend zu beachten und sich frühzeitig auf technologische Details wie die Auswahl eines Hilfsmittels zu konzentrieren.

Die zentrale These dieser Arbeit ist daher, dass abstraktes wissenschaftlich diskutiertes Lösungswissen existiert – insbesondere in Form von Mustern – findet aber aufgrund mangelhafter Abstraktion keinen Eingang in die Praxis. Darüber hinaus ist die Eigenschaft der Integration als Dienstleistung bisher wissenschaftlich zu wenig beachtet worden, weshalb zentrale Wirkungszusammenhänge nicht untersucht sind.

Im Rahmen der Arbeit erfolgte eine qualitative empirische Untersuchung, welche erstmals im deutschsprachigen Raum Wirkungszusammenhänge und Entscheidungsmechanismen in Integrationsprojekten erhebt. Es wurden die Interviews von führenden Mitarbeitern von 30 Integrationsdienstleistern aus dem deutschsprachigen Raum ausgewertet. In die Arbeit ist nur der relevante Teil der Untersuchung eingeflossen, die andere Hälfte der Untersuchung liefert die Basis für

eine verwandte Arbeit. Durch diese Untersuchung kann zum einen die These, welche der Arbeit zugrunde liegt, bestätigt und deren praktische Relevanz nachgewiesen werden. Zum Anderen war es möglich, aus der Untersuchung die Richtung der Lösungsfindung abzuleiten. Das Konzept von Mustern – welches der Softwareentwicklung entstammt und im Bereich der Integration auch wissenschaftlich untersucht wird – ist in der Praxis bekannt. Es wird jedoch nicht wissentlich verwendet. Als Ursache kann ein nicht adäquater Abstraktionsgrad der bestehenden Muster identifiziert werden. Die Beschreibungen enthalten eine große Zahl an Details, was dazu führt das etliche Varietäten der gleichen Konzepte beschrieben werden. Darüber hinaus stammen diese von den unterschiedlichsten Autoren, was eine publikationsübergreifende Systematik vermissen lässt.

Bevor die Lösung dieser Missstände angestrebt wird, erfasst die Arbeit die Zusammenhänge, die in einem Integrationsprojekt wirken, in dem auf der Basis der empirischen Untersuchung das Dienstleistungsmodell für Integrationsdienstleistungen entwickelt wird. Zum Einsatz kommt dabei eine Beschreibungsmethode aus der Services-Science, welche neben einem Ressourcen- ein Komponenten-, Produkt- und Prozessmodell umfasst. Dieses Modell ist geeignet die Integrationsdienstleistungen der Unternehmen zu erfassen, welche an der empirischen Untersuchung beteiligt waren. Zusätzlich zu diesem Modell wurde ein GAP-Modell für Integrationsdienstleistungen entwickelt, welches auf einer umfassenden Literaturanalyse bezüglich der Herausforderungen und Probleme der Integration basiert.

Die Lösung des Abstraktionsproblems von Integrationsmustern wird dadurch eingeleitet, dass der Terminus Integrationsmuster definiert wird. Zu diesem Zweck wurden auf der Basis expliziter Definitionen von Integrationsmustern, den Definitionen von Architekturmustern – in deren Gruppe Integrationsmuster eingeordnet werden – und den Definitionen von Integrationsarchitekturen konstituierende Eigenschaften extrahiert und in eine Definition überführt. Diese Definition wurde anhand explizit ausgewiesener Integrationsmuster evaluiert und geprüft, ob sich diese auch durch ihre Eigenschaften als Integrationsmuster ausweisen lassen, bevor eine Grundgesamtheit an Mustern aus einer Literaturauswahl extrahiert wurde, welche als Basis der weiteren Untersuchungen diente.

Da der Abstraktionsgrad auf dem die Muster beschrieben sind – den Ergebnissen der Untersuchung und der eingangs dargestellten Forschungsprobleme zur Folge – zu deren mangelhaftem Einsatz in der Praxis führt, wurden im nächsten Schritt Abstraktionsarten nebst deren impliziter Nutzung durch das Musterkonzept ermittelt. Als Beschreibungsmodell wurden Muster dazu mit Hilfe des semiotischen Dreiecks charakterisiert.

Die Anpassung des Abstraktionsgrades der Muster der Grundgesamtheit erfolgte mithilfe von zwei Abstraktionsarten, welche eng miteinander zusammenhängen. Es wurde sowohl die Klassifikation als auch die Generalisierung ausgewählt. Im Rahmen der Klassifikation wurde ein erweiterbares und flexibles Klassifikationsverfahren – die Facettenklassifikation – ausgewählt. Diese ermöglicht jederzeit das Hinzufügen weiterer Facetten und weist zusätzlich die Eigenschaft auf, dass nur die Einordnung eines Musters in einer Facette disjunkt sein muss. Für jedes Muster ist immer nur ein Fokus einer Facette zutreffend, es kann aber in anderen Facetten eingeordnet werden, muss dies aber nicht. Es können also auch Muster in einer Klassifikation enthalten sein, die in einigen Facetten keine Bewertung aufweisen. Für die Durchführung der Klassifikation der Grundgesamtheit wurden verschiedene Facetten entwickelt, die sowohl dem Problembereich als auch dem Lösungsbereich entstammen. Es wurden die folgenden Facetten definiert und deren Fokusse festgelegt:

- Subsystem des betrieblichen Informationssystems,
- Integrationsoperation,
- Integrationsarchetyp,

- Komponenten der Referenzarchitektur,
- Interoperabilitätskonflikt, Heterogenität,
- Zweck der Kommunikation,
- Verbindungsorientierung,
- Kardinalität der Kommunikation,
- Interaktionsmechanismus,
- Interaktionslogik,
- Send,
- Receive,
- Time/Synchronität,
- Space Decoupling,
- Functionality,
- Reliability,
- Usability,
- Efficiency,
- Maintainability,
- Portability,
- Reusability.

Jeder Facette liegt dabei eine umfassende Analyse für die Definition der Fokusse zugrunde. Im Anschluss daran wurde die Grundgesamtheit der Muster in die entsprechenden Facetten eingeordnet, klassiert und so eine Gesamtklassifikation gebildet.

Diese Klassifikation bildet den Ausgangspunkt der Generalisierung, die Muster mit ähnlichen bzw. identischen Ausprägungen erfasst und diese auf ein gemeinsames Konzept untersucht. Diese Generalisierung wurde für zwei Mustergruppen durchgeführt und der Typ zusätzlicher Zugriffspunkt sowie der Typ Vermittler entwickelt.

Die so entwickelten Konzepte flossen in eine umfangreiche Evaluation ein welche am Beispiel einer konkreten Dienstleistung im Bereich der E-Procurement-Integration eines Integrationsdienstleisters durchgeführt wurde. Weder der Dienstleister noch der Kunde waren an der empirischen Untersuchung beteiligt, was den Evaluationscharakter unterstreicht. Die Besonderheiten der Evaluation bestehen darin, dass neben dem Integrationsdienstleister auch der Kunde sowie die – konkret zur Umsetzung anstehende – Integrationsdienstleistung untersucht werden konnten. So erfolgte nach der umfangreichen Darstellung und Analyse des Integrationsszenarios zunächst auch erst eine Erfassung der üblichen Vorgehensweise des Integrationsdienstleisters. Diese umfasst nicht nur den Architekturentwurf des Dienstleisters, sondern auch eine Instanziierung des Dienstleistungsmodells für den konkreten Fall des Dienstleistungsangebotes an den Kunden.

Danach wurde die Lösungsauswahl auf der Basis der erarbeiteten Klassifikation und der hochstehenden, abstrakten Integrationsmuster durchgeführt. Die beiden Lösungsansätze wurden einander gegenübergestellt und die Vorteile des – in dieser Arbeit entwickelten – Konzeptes diskutiert. Der erarbeitete Lösungsvorschlag wurde in einem Integrationslabor implementiert, wobei das vollständige Integrationsszenario auf der Basis einer Virtualisierungsumgebung möglichst realistisch nachgebildet wurde. Es wurden Instanzen der Kundensysteme mit identischem Versions- und Patch-Stand sowie Datenbestände aus dem Produktivsystem verwendet. Aus

Anonymisierungsgründen konnten leider nicht alle Systeme benannt werden, die dargestellten Architekturen und Listings entsprechen jedoch bis auf Umbenennungen den Originalen. Für die Integrationshilfsmittel wurde auf die gleiche Weise eine Ablaufumgebung hergestellt.

Abschließend wurden die Auswirkungen des Lösungsvorschlages mithilfe der Konzepte der Arbeit diskutiert. Neben den Auswirkungen für die Integrations- und Unternehmensarchitektur des Kunden wurde auch das durch die Lösung entwickelte Potenzial der weiteren Entwicklung der Unternehmensarchitektur dargelegt. Die Auswirkungen für den Dienstleister manifestieren sich in den Veränderungen seines Dienstleistungsmodells und den Möglichkeiten, das einmalige Angebot in ein konfigurierbares Standarddienstleistungsangebot zu überführen. Eine verbesserte Ressourcennutzung wird ebenfalls anhand des Dienstleistungsmodells nachgewiesen.

Damit konnte im Rahmen der Arbeit ein Ansatz entwickelt werden, der die empirisch belegten Abstraktionsprobleme behebt und die Einsetzbarkeit bestehenden Lösungswissens verbessert. Gleichzeitig werden die Wirkungsmechanismen und Entscheidungszusammenhänge durch das Dienstleistungsmodell besser erfass-, erklär- und vor allem planbar. Die Evaluation liefert den Nachweis der praktischen Einsetzbarkeit. Die Verbesserung der Werkzeugunterstützung wurde nur in Randaspekten gestreift und wird in der kritischen Würdigung der Ergebnisse und dem Ausblick auf zukünftige Forschung im nächsten und abschließenden Kapitel dargelegt.

8 Ausblick und weiterführende Überlegungen

In diesem Kapitel werden verschiedene Einzelaspekte der Arbeit kritisch gewürdigt und ein Ausblick auf weitere Forschung, mit der die vorliegende Arbeit fortgeführt werden kann, gegeben. Es lassen sich drei große Bereiche unterscheiden. Der Erste betrifft das in dieser Arbeit entwickelte Dienstleistungsmodell, der Zweite die Klassifikation und weitere Abstraktion der Integrationsmuster, wohingegen der Dritte Teil sich mit der in dieser Arbeit zu kurz gekommenen Werkzeugunterstützung der Integration befasst. Für alle drei Bereiche werden immer erst die kritischen Punkte und dann die Möglichkeiten der Fortführung vorgestellt.

8.1 Dienstleistungsmodell

Grundsätzlich lässt sich an dem entwickelten Dienstleistungsmodell der Gültigkeitsbereich kritisieren. Es ist für die befragten Unternehmen und die Beteiligten der Evaluation verwendbar. Eine darüber hinaus gehende Einsetzbarkeit muss erst nachgewiesen werden. Grundsätzlich soll das Dienstleistungsmodell in weiteren Fällen verwendet und iterativ verbessert werden.

Im Servicekomponentenmodell scheint – belegt durch die ermittelten Ausprägungen der Komponente Durchführung in der Fallstudie – eine Verfeinerung dieser Komponenten angezeigt.

Das Produktmodell ist in den Phasen Test und Betrieb deutlich unterspezifiziert. Neue Trends in der IT wie das Cloud-Computing machen es aber notwendig diese Phasen weiter zu detaillieren, da Integrationsplattformen in der Zukunft auch als Dienst betrieben werden können, wodurch sich auch das Dienstleistungsmodell verändern wird.

Das Prozessmodell stellt nur auf eine sequenzielle Verknüpfung der Aktivitäten ab. An dieser Stelle zeigt sich weiterer Bedarf bezüglich der Analyse bestehender Prozessmodelle und einer Abbildung der dort beschriebenen Schritte auf die Servicekomponenten des Dienstleistungsmodells. Ebenso kann deren Ablauf analysiert und Bezüge zu bestehenden Vorgehensmodellen der Softwareentwickelung hergestellt werden.

Das GAP-Modell der Integration weist den Nachteil auf, dass die Einarbeitung aller Details des Integrationsdienstleistungsmodells noch aussteht. Eine genaue Analyse der Interdependenzen der Lücken und der Problemgruppen kann an helfen. Darüber hinaus betont das GAP-Modell stark die Auswirkungen hinsichtlich der Architektur und blendet die Entstehung der Integrationsanforderungen aus.

8.2 Abstraktion der Integrationsmuster

Grundsätzlich kann die Menge der untersuchten Muster kritisiert werden – welche dem Umfang der Arbeit geschuldet – begrenzt ist. Diese Menge kann – unterstützt durch das gewählte Klassifikationsverfahren – vergrößert werden in dem unberücksichtigte Quellen untersucht werden. Es sollten auch weitere Typen von Mustern hinzugefügt werden. So fehlen zum Beispiel Architekturmuster für Anwendungslandschaften, Muster für Unternehmensarchitekturen sowie Hardwaremuster[1] und Organisationsmuster. Ziel der Aufnahme dieser Musterarten ist es, für

[1] Vgl. (Day, 2008).

die Organisation von Systemen aus heterogenen Komponenten Grundprinzipien zu entwickeln, die sich auf der Hardware-, Software- und sozio-ökonomischen Systemebene finden lassen, im Sinne fundamentaler Integrationsarchitekturprinzipien von Informationssystemen. Erst durch die Hinzunahme dieser Mustertypen entfaltet auch die Archteyp-Facette ihren Nutzen wenn Systemübergreifende – im Sinne der Subsysteme eines BIS – Koordinationsmechanismen untersucht werden.

Die Komposition als Generalisierungsart wurde nicht verwendet, obwohl sich in der Literatur mit einer Vielzahl an *uses* Beziehungen Anhaltspunkte bestehen diese Abstraktionsart zu analysieren. Weitere Abstraktionsarten, die bereits eingesetzt werden lassen sich bestimmen in dem die Gruppierungskriterien von Musterkatalogen identifiziert und systematisiert werden.

Bei der Analyse der Klassifikation lassen sich diejenigen Kategorien identifizieren, in denen sich Muster voneinander unterscheiden. Zusammen mit den weiter generalisierten Mustern lässt sich so ein Variabilitätsmodell erstellen, welches die Konfigurationspunkte für die abstrakten Muster enthält. Ein Konzept lässt sich in mehr als einem Bereich anwenden. Damit ist der Bereich die Konfigurationsoption, durch die sich zwei Muster unterscheiden. Auf diesem Weg entstehen erste Anhaltspunkte, welche bei der Entwicklung eines Systems (von Mustern) bzw. einer Sprache behilflich sind.[2] Dazu müssen die bestehenden Muster in Subkonzepte zerlegt und deren Kombinationsmöglichkeiten analysiert werden.

Der Klassifikation lassen sich weitere Facetten hinzufügen. Eine Facette ist der betroffene Elementtyp der Unternehmensarchitektur. In Archimate werden Strukturelemente, Verhaltenselemente und Informationselemente unterschieden. Muster lassen sich nach den gleichen Kriterien strukturieren. Aus der bei Khesav und Gamble[3] diskutierten Controller-Extender-Translator-Systematik lässt sich ebenfalls eine Facette entwickeln, welche in die Richtung der Bildung von kombinierbaren, feingranulareren Mustern geht.

Weiterhin kann die Facette des Interoperabilitäts- bzw. Heterogenitätskonfliktes verbessert werden. Die Entwicklung einer Konfliktsystematik scheint ratsam. Welche Konflikte treten gemeinsam auf? Bedingen verschiedene Konflikte einander? Kann von einem auf den anderen geschlossen werden? Im Hinblick auf die Subsysteme eines BIS stellt sich auch die Frage, ob sich Konflikte auf verschiedenen Ebenen replizieren und welche Ebene die auslösende ist.

Auch hat die verwendete Menge der Konflikte noch nicht den maximalen Grad der Konsolidierung erreicht. Indem z. B. Konflikte der Art *Frameworks* eingeführt wurde, findet die Nutzung unterschiedlicher Granularitätsstufen von Konflikten statt. Frameworks stellen einen Mechanismus der Wiederverwendung dar, indem sie basale Funktionalität zur Verfügung stellen auf der aufgebaut werden kann. In dieser Funktionalität enthaltene Konflikte lassen sich durch die Konflikte der Ebene Softwaresystem beschreiben. Diese Konfliktkategorie bietet jedoch die Möglichkeit der Abstraktion indem für den Vergleich von Frameworks – welche Alternativen darstellen – nicht die Menge der Einzelkonflikte angegeben werden muss, sondern lediglich die Heterogenität der Frameworks. Diese ließe sich aber auch durch einen mehrstufigen Prozess der Konfliktanalyse abdecken und sollte noch einmal überdacht werden.

Aktuell werden nur Konflikttypen verwendet. Es wäre zu prüfen, ob sich festlegen lässt, welche Eingenschaft eines Musters – bei kombinierten Konflikten – dafür verantwortlich ist, dass ein einzelner Konflikt gelöst wird. Ist dies möglich, so müssen Muster weiter heruntergebrochen und in Teilkomponenten zerlegt werden, die komponierbar sind. Dann können die Konflikte Einzug in die Systematik der Klassifikation halten. Alternativ lassen sich die Typen der Muster verwenden, indem aus jedem Typ eine separate Facette gebildet wird.

[2] Vgl. Abschnitt 2.3.
[3] Vgl. (Keshav und Gamble, 1998).

Viele Muster weisen unterschwellig weitere Funktionalität aus, welche in der Generalisierung nur angeklungen ist, da der Fokus auf der Primärfunktionalität lag. Bei der Klassierung kann nach primärer und sekundärer Funktionalität des Musters unterschieden werden. Für das Muster Database Federation besteht die primäre Problematik in der Lösung eines Transportkonfliktes und die sekundäre Problematik in der Lösung nachfolgender Informationsprobleme.

Die Qualitätseigenschaften der Muster wurden auf die Qualitätsmerkmale reduziert. Eine Erweiterung der Klassifikation in dem alle Teilmerkmale der Tabelle 5.40 als Facetten aufgenommen werden sollte angestrebt werden. Dabei muss dann allerdings überlegt werden, inwieweit die Teilmerkmale das Qualitätsmerkmal vollständig bestimmen und ob dieses dann lediglich eine Zusammensetzung aus den Ausprägungen der Teilmerkmale darstellt und damit dann ein abhängiges Merkmal darstellt, also aus den Ausprägungen der Teilmerkmale heraus bestimmt werden kann. Die Zusammenhänge zwischen den Qualitätsmerkmalen wurden demnach bisher nicht betrachtet.

Die Einschränkung auf die Qualitätsmerkmale und das Mapping waren notwendige Schritte. Es sollte sich aber eine Neu-Untersuchung der Muster anschließen, welche eine Neubewertung der Muster mit einer einheitlichen Qualitätsmerkmalsystematik vornimmt. Dazu sollten zu jedem Muster Messbarkeitskriterien für das jeweilige Merkmal definiert werden, um ein Benchmarking zu ermöglichen. Hilfreich könnte hier eine Befragung von Experten sein, denen Muster gezielt zur Eigenschaftsbewertung vorgelegt werden. Bei einigen Bewertungen der Muster ist unklar, was bewertet wurde. Nimmt man das Beispiel Replaceability, so kann zum einen bewertet werden, wie leicht sich das Muster wieder austauschen lässt. Es kann aber auch bewertet werden, inwieweit sich nach Anwendung des Musters die Integrationsobjekte austauschen lassen, ohne die Beziehung verändern zu müssen.

Darüber hinaus beziehen sich die qualitativen Facetten der Klassifikation auf die ISO/IEC 9126. In der zwischen Zeit wurde die DIN ISO/IEC 25000 herausgegeben, sodass die Facetten an diese neue Norm angepasst werden müssen, sofern sich die Qualitätsmerkmale verändert haben. Zusätzlich muss angeführt werden, dass sich die Norm nur auf die Produktqualität bezieht und den zugehörigen Prozess der Erstellung außer Acht lässt. Dies ist für den Gegenstand der Arbeit ausreichend, in späteren Untersuchungen sollte der Einfluss eines Musters auf die Erstellung der Integrationslösungen und damit auf die Qualität der Integrationsdienstleistung untersucht werden.

Zusätzlich kann die Wahl der ISO/Norm kann kritisch beurteilt werden, da es fraglich ist, ob eine Architektur (Muster gestalten Architekturen) durch die gleichen Qualitätsmerkmale beschreibbar ist wie ein Softwareprodukt. So lassen sich einige Attribute bezüglich der Muster nicht mit der gewählten Systematik an Qualitätsattributen abbilden, da diese die Produktqualität erfasst. So wird z. B. bei Trowbridge für das Muster Data Transfer Object [4] angegeben, dass die Umsetzung des Musters einen erhöhten Programmieraufwand mit sich bringt, wodurch die Umsetzung ermüdend bzw. langweilig wird. Für die optimale Gestaltung eines Dienstleistungsbündels sind diese Eigenschaften zu berücksichtigen. Ist es möglich die gleiche Produktqualität durch ein alternatives Muster bei vergleichbar geringerem Programmieraufwand, der in der Umsetzung nicht zu einer Eintönigkeit führt, zu erreichen, so wäre dies vorzuziehen.

Das Qualitätsmerkmal Reusability müsste eigentlich in zwei einzelne Merkmale aufgespalten werden, und zwar Development-For-Reuse und Development-With-Reuse. Ein Muster kann dazu führen, dass der entwickelte Baustein einer Architektur besser wieder verwendbar ist (Development-For-Reuse) oder dazu, dass bestehende Teile leichter wieder verwendet werden

[4] Vgl. (Trowbridge, 2003, S. 236).

können (Development-With-Reuse). Ein Beispiel einer detaillierten Beschreibung beider Eigenschaften findet sich im Fall der Pipes und Filter Architektur bei Meunier.[5]

Die Generalisierung der Muster zeigt einen guten Ansatz auf, muss aber im Umfang gesteigert werden. Das Ziel ist die Identifikation von fundamentalen Grundprinzipien. Bereits vermutet werden können:

Entkopplung/Indirektion: Die Muster, welche den Zugriff und die Verteilung organisieren, verfolgen alle das gemeinsame Grundprinzip den Zugriffspunkt von dem Integrationsobjekt zu trennen. Muster, welche die Kommunikation organisieren, versuchen die Kommunikationspartner (Integrationsobjekte) voneinander zu entkoppeln. Damit findet eine 2-fache Entkopplung statt. Zum einen die der Integrationsobjekte voneinander (Unkenntnis der Kommunikationspartner) und dann die der Implementierung und der Zugriffspunkte statt.

Funktionszentralisierung: Insbesondere die Transformationslogik und die Interaktionslogik in Integrationsarchitekturen sind wesentliche, von vielen Integrationsobjekten genutzte Bestandteile der Architektur. Viele Muster ersetzen die verteilte Implementierung – in den Integrationsobjekten oder in der Nähe – durch zentrale Funktionsbausteine. Damit wird die Pflege an zentraler Stelle ermöglicht jedoch auch ein Single-Point-of-Failure (SPOF) eingeführt. Die Stabilität der Architekturen sinkt. Mit gezielt erzeugten Redundanzen kann diesem Effekt entgegen gewirkt werden.

Hybridmuster mit Heterogenitätsüberbrückung: Das letzte identifizierte Konstruktionsprinzip der systematisierten Integrationsmuster ist das der Hybridisierung zum Zwecke der Überbrückung von Heterogenität. Dabei wird nicht auf den Heterogenitätskonflikt im Detail eingegangen, sondern eine prinzipielle Erweiterung bestehender Muster angedeutet. Dies stellt einen Nachteil dar, denn bei großen Integrationsarchitekturen mit einer hohen Last oder gar Plattformen, welche Integrationsleistungen nach einem Software-as-a-Service-Modell anbieten, ist damit zu rechnen, dass ein wesentlicher Teil der Rechenlast auf diesen Teil der Architektur entfällt. Eine Separierung dieser Funktionalität würde die Möglichkeit mit sich bringen, Mechanismen der Lastverteilung und der gezielten Redundanz zu nutzen.

Die zur Evaluation ausgewählte Fallstudie weist Unvollständigkeiten des betrachteten Integrationsszenarios auf. Produktdaten werden nur zu einem Teil in einem ERP-System gepflegt. Informationen wie Packmaße, Gewichte und auch Bilder der Produkte werden nicht in einem ERP-System gepflegt. Zu diesem Zweck sind Produktdatenmanagementsysteme üblich. Für eine vollständige Realisierung wäre auch ein Import der dort verwalteten Produktdaten in das Shop-System notwendig. Verschiedene Szenarien lassen sich in der Realität nicht auf veränderte Lastbedingungen etc. untersuchen. Verlässliche Daten für die Konstruktion von Simulationen scheinen nicht gegeben zu sein, sodass der Aufbau eines Integrationslabors, in dem reale Anwendungsszenarien nachgestellt werden, um verschiedene Szenarien erproben zu können, Abhilfe schaffen kann. Besondere Bedeutung kommt dabei der Generierung von Bewegungsdaten in den Integrationsobjekten zu. Aus dieser Tätigkeit lassen sich nicht nur Informationen und Erkenntnisse für die Konstruktion von Integrationslösungen ableiten, sondern auch für das Testen und die Überwachung, denn diese Punkte sind für ein Integrationslabor zu lösen. Weiterhin werden so Ansätze und Daten für die Simulation von Unternehmensarchitekturen gewonnen.

8.3 Werkzeugunterstützung der Integration

Obwohl dieses Problem mit untersucht werden sollte, konnte es nur in Teilen bearbeitet werden, da die bisher dargestellten Teilprobleme fokussiert wurden. Dieses Kapitel zeigt, welche Ansätze

[5] Vgl. (Meunier, 1995, S. 437).

durch die Arbeit erkannt wurden und vielversprechend für eine weitere Vertiefung sind. Im Rahmen der empirischen Untersuchung wurde bemängelt, dass zwar die Phase der Entwicklung einer Integrationslösung gut mit Werkzeugen unterstützt ist, aber in den Phasen der Analyse und Planung erhebliche Defizite bestehen. Dies drückt sich in der nicht standardisierten Dokumentation mit Hilfe von Tabellenkalkulations- und Textverarbeitungsprogrammen aus. Dabei entstehen viele unterschiedliche Dokumente, welche keinen einheitlichen Aufbau haben. Darüber hinaus gestaltet es sich auf dieser Basis schwierig, alle Beteiligten in Ihrer Domäne abzuholen und dennoch alle relevanten Daten in einem Dokument zusammenzuführen. Betriebswirtschaftler sind typischerweise auf der geschäftlichen Ebene verankert und mit den Domänenkonstrukten dort vertraut. Zu realisierende Prozesskennzahlen wie Durchlauf- und Antwortzeiten sind für die Integration relevant, sofern der zugehörige Prozess in seiner technischen Realisierung durch heterogene Systeme unterstützt wird und damit Integrationsbeziehungen aufweist. Bereits die Abbildung der Prozesse auf die IT-Systeme – auch Business-IT-Alignment genannt[6] – erfordert die Zusammenarbeit von Menschen mit unterschiedlichem Domänenwissen und stellt in der Praxis ein größeres Problem dar. Die Zuordnung erfolgt jedoch nur zu logischen Anwendungskomponenten, welche auf eine technische Detaillierung meist vollständig verzichten. Welches Datenbanksystem im Rahmen eines ERP-Systems zum Einsatz kommt und ob von diesem ein signifikanter Einfluss auf die Verarbeitungsgeschwindigkeit und damit letztendlich eine Beschränkung der realisierbaren Durchlaufzeiten der teilautomatisierten Geschäftsprozesse ausgeht, bleibt meist unberücksichtigt und wird erst bei der Verletzung z. B. garantierter Durchlaufzeiten bemerkt. Eine verbesserte Werkzeugunterstützung muss deshalb diese Kommunikationsprobleme berücksichtigen.

Im Rahmen der Arbeit wurden bereits Modelle in der Archimate Notation gezeigt, welche Mithilfe eines Prototypen der diesen Standard in der Version 1.0 umzusetzen versucht entwickelt wurden.[7] Es handelt sich dabei um einen Modellierungsstandard für Unternehmensarchitekturen, welche versuchen die vorgenannten Perspektiven in einem Modell mit Teilmodellen zu vereinen. Damit erscheinen diese prinzipiell geeignet bei der Analyse und Planung von Integrationsprojekten hilfreich zu sein, zumal die Integrationsarchitektur einen Teil der Unternehmensarchitektur darstellt. Erste Evaluationen von Modellierungswerkzeugen in diesem Bereich ergaben, dass eine Erweiterbarkeit der Modelle um Aspekte der Integration, z. B. die Hinterlegung von Prozesskennzahlen oder Schnittstellenversionen und -beschreibungen, selten möglich ist. Auf der Basis dieser Erkenntnis wurde eine Analyse durchgeführt welche Werkzeuge in diesem Bereich auf dem Markt befindlich sich und welche Standards diese umsetzen. Analysiert wurden 36 Werkzeuge, von denen es sich bei 30 um kommerzielle – und damit für wissenschaftliche Versuche nicht veränderbare – handelt. Die restlichen Werkzeuge verteilen sich auf zwei universitäre Prototypen und vier Open-Source-Imlementierungen. Neben der Unterstützung verschiedener konzeptioneller Unternehmensarchitektur-Frameworks wie TOGAF konnte bei der Analyse der eingesetzten Modellierungsstandards festgestellt werden, dass Archimate mit sechs Implementierungen der mit Abstand am häufigsten verwendete Standard war. Dies veranlasste zu der Entwicklung des Prototypen ArchiLE[8], welcher Mithilfe der modellgetriebenen Entwicklung auf der Basis des Eclipse-Modelling-Framework (EMF) und des Graphical-Modelling-Framwork (GMF) entwickelt wurde. Dieser Prototyp unterstützt die Hinterlegung von generischen Attributen an allen Modellierungselementen und kann durch den Einsatz modellgetriebener Technologien leicht um feste Attribute, sowie zusätzliche Modellierungselemente erweitert werden. Der Archimate-Standard sieht dafür entsprechende Konzepte vor, sodass eine Standardkonformität gewahrt werden kann. Mit dieser Ausgangsbasis kann in Zukunft ein iterativer Verbesserungsprozess gestartet werden, der es ermöglicht sukzessiv alle – für die Integration relevanten – Informationen zu sammeln und

[6] Vgl. Kapitel 4.3 für die Problembeschreibung, sowie die Details eines idealen Werkzeugs in Kapitel 3.5.4.
[7] Vgl. Abbildung 6.4 und 6.4.
[8] Vgl. (Bretschneider und Gebauer, 2011).

der Modellierungskonvention hinzuzufügen. Anschließend ist es möglich, eine Erweiterung des Archimate Standards für die Integration vorzuschlagen.

Weiterhin ermöglich die Modellierungskonvention eine automatisierte Ableitung von Integrationsbeziehungen. Dies geschieht mit dem Ziel, in späteren Ausbaustufen eine Entscheidungsunterstützung bieten zu können. Eine Integration kann aus verschiedenen Gründen notwendig werden. Dabei wird (a) die Weitergabe des Kontrollflusses, (b) der Zugriff auf Daten einer anderen Anwendung, (c) die Nutzung von Funktionen einer anderen Anwendung unterschieden. Nachdem die Integrationskandidaten bestimmt sind, müssen Aussagen über die Kommunikation und die Heterogenität der Applikationen abgeleitet werden. Bis zu diesem Punkt werden ausschließlich bilaterale Integrationsbeziehungen betrachtet. In einem sich anschließenden Analyseschritt soll Optimierungspotenzial identifiziert werden. Dabei ist zu untersuchen, inwieweit die gleichen Informationen an verschiedenen Stellen benötigt werden oder Applikationen in multiplen Interaktionen beteiligt sind. Bevor darstellt wird, wie diese Analyseziele mithilfe eines Archimate-Modells unterstützt werden können, wird eine mengentheoretische Formalisierung eines Architmate-Modells dargelegt. Dies ermöglicht es, die Analysemuster als Mengenoperationen auszudrücken. Die Formalisierung wurde nach dem Erscheinen einer Publikation von von Šaša und Krisper – welche eine Analyse der Unterstützung von Geschäftsprozessen auf der Basis von Archimate-Modellen durchführen[9] – überarbeitet. Da die Gestaltung von Integrationsarchitekturen verfolgt wird, kann die Arbeit von Šaša und Krisper als Grundlage dienen. Die Formalisierung umfasst darüber hinaus alle Elemente des Application-Layers und betrachtet auch den bei Šaša und Krisper nicht enthaltenen Technology-Layer. Beide Arbeiten können so zusammen verwendet werden und ergeben eine Einheit. Details der Formalisierung sind in Anhang F enthalten.

Über die Formalisierung hinaus liefert die Arbeit von Šaša und Krisper auch Analysemuster, welche auf Basis der Formalisierung entwickelt wurden. Um den ersten Schritt in Richtung einer Entscheidungsunterstützung zu gehen, wurden diese Muster analysiert und auf Ihre Einsetzbarkeit für eine Analyse der Archimate-Modelle zum Zwecke der Integrationsanalyse geprüft. Die Tabelle 8.1 fasst alle bekannten Muster und Ihre Eignung für die Integrationsanalyse zusammen. Dabei steht (++) für eine uneingeschränkte Verwendbarkeit bei der Integrationsanalyse, (+) für eine bedingte Verwendbarkeit (das Muster muss angepasst werden) und (-) dafür, dass die Verwendung ausgeschlossen ist. Die Details sind im Anhang F.2 enthalten.

[9] Vgl. (Šaša und Krisper, 2011).

Muster	Gleichung	Eignung
$FABP_{AnBP}$	F.27	++
$PABP_{AnBP}$	F.28	++
$UABP_{AnBP}$	F.29	-
$FMPB_{AnBP}$	F.30	-
$PMBP_{AnBP}$	F.31	-
$MUSBP_{AnBP}$	F.32	-
$USBP_{AnBP}$	F.33	-
$PSBP_{AnBP}$	F.34	-
$FSBP_{AnBP}$	F.35	-
$HeASBP_{AnBP}$	F.36	-
$HoASBP_{AnBP}$	F.37	-
$RFABP_{AnBP}$	F.38	++
$NRFABP_{AnBP}$	F.39	-
$MFMBP_{AnBP}$	F.40	-
$SFMBP_{AnBP}$	F.41	-
$RFSBP_{AnBP}$	F.42	-
$NRFSBP_{AnBP}$	F.43	-
$FABO_{AnBO}$	F.44	++
$PABO_{AnBO}$	F.45	++
$UABO_{AnBO}$	F.46	-
$FCBO_{AnBO}$	F.47	-
$PCBO_{AnBO}$	F.48	-
$UCBO_{AnBO}$	F.49	-
$FRBO_{AnBO}$	F.50	-
$URBO_{AnBO}$	F.51	-
$PRBO_{AnBO}$	F.52	-
$MFRBO_{AnBO}$	F.53	-
$SFRBO_{AnBO}$	F.54	+
$MFABO_{AnBO}$	F.55	++
$SFABO_{AnBO}$	F.56	++
$MFCBO_{AnBO}$	F.57	-
$SFCBO_{AnBO}$	F.58	-

-: nicht geeignet; ++: sehr gut geeignet; +: geeignet

Tabelle 8.1: Eignung der vorgestellten Muster für die Integrationsanalyse

Aufbauend auf diesen Mustern lassen sich Muster für die Integrationsanalyse formulieren. Es wurden ja bereits verschiedene eine Integration notwendig machende Phänomene wie ein Kontrollflussübergang oder eine Datenübergabe angedeutet.

Control-flow-hand-over-Muster: Dieses Muster befasst sich mit der Analyse, ob zwischen zwei Applikationen eine Integration notwendig ist, weil der Kontrollfluss von der einen Anwendung zur anderen weitergegeben werden muss. Da der Business Layer die eigentliche geschäftliche Tätigkeit definiert ist zunächst zu untersuchen, welche Abhängigkeiten zwischen Elementen des Business Layers bestehen können. Relevant für die Abfolge der Verhaltenselemente[10] sind die dynamischen Relationen. Diese sind Triggering und Flow. Dabei beschreibt die Triggering-Relation temporale oder kausale Zusammenhänge zwischen Prozessen, Funktionen, Interaktionen und Events.[11] Im Gegensatz dazu stellt die Flow Relation auf den Transfer von z. B. Informationen zwischen den Elementen ab.[12] Für die Kontrollfluss Weitergabe spielt demnach die Triggering-Relation die wesentliche Rolle.

Als nächster Aspekt ist die Abbildung der Verhaltenselemente auf die Elemente des Application-Layer entscheidend. Von Interesse im Application-Layer können dabei Application-Component, Application-Interface, Application-Service und Application-Function sein. Das Element Application-Component ist das am nahesten liegende Strukturelement des Application-Layer, da es einen

[10] Business-Behavior-Element (BBE)
[11] Vgl. (The Open Group, 2009, S. 60).
[12] Vgl. (ebenda, S. 61).

modularen, verteilbaren[13], austauschbaren Teil eines Systems, der seinen Inhalt kapselt und seine Funktionalität durch sein Set von Schnittstellen nach außen gibt. Einem Verhaltenselement des Business Layer kann eine Applikationskomponente durch eine Assignment-Relation zugeordnet werden. Abbildung 8.1 verdeutlicht diesen Fall. Der Archimate Standard weist in einer in den Anhängen dargestellten Tabelle für die Zuordnung der Elemente andere Relationen aus. Dabei ist die Assignment-Relation nicht enthalten. Die in den Anhängen aufgelisteten Relationen werden als kumulativ zu denen betrachtet, welche in den Diagrammen des Standards dargestellt werden.

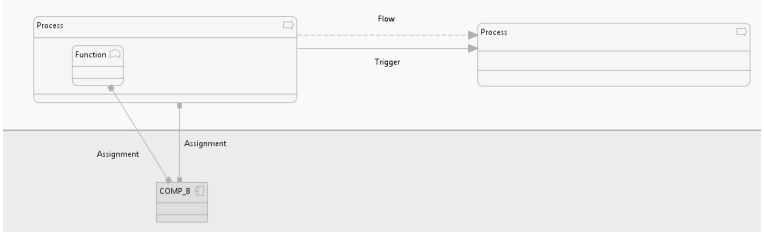


Abbildung 8.1: Zuordnung von Verhaltenselementen des Business-Layer zu Application-Component

Aus dieser Darstellung wird allerdings nicht deutlich, über welche Schnittstelle bzw. Service der Zugriff erfolgt. Aus diesem Grund kann mithilfe dieses Kriteriums lediglich ein Hinweis abgeleitet werden. Ein solches Modell wird als unterspezifiziert bezeichnet. Diese Unterspezifizierung lässt sich beseitigen, in dem Application-Interface, Application-Service etc. hinzugefügt werden. Abbildung 8.2 fasst alle Möglichkeiten in einem Diagramm zusammen. Es enthält – der Übersichtlichkeit halber – neben dem Business-Process auch eine Business-Function, welche Bestandteil (Composition-Relation) des Prozesses ist.

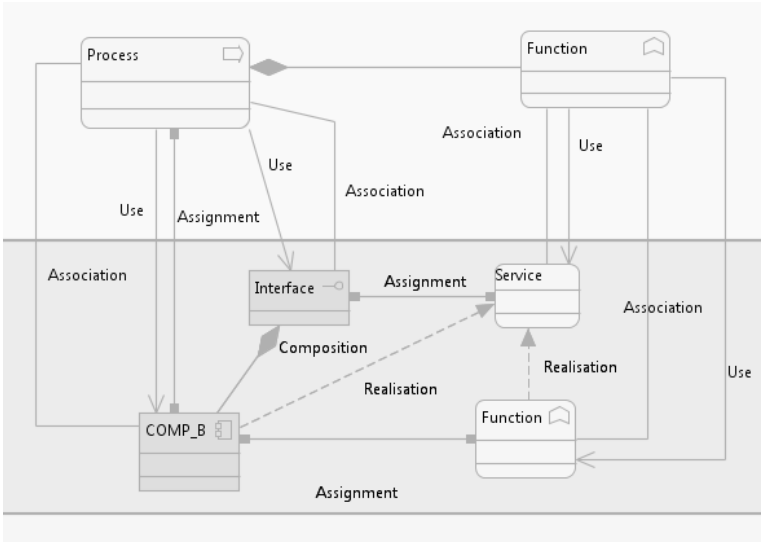


Abbildung 8.2: Zuordnung von Verhaltenselementen des Business-Layer zum Application-Layer

Auffällig ist zunächst, dass es mehrere Relationen gibt, mit denen die Verwendung eines Application-Interface oder eines Application-Service ausgedrückt werden kann. Neben einer Use-Relation ist auch jeweils eine Association-Relation möglich. Die Use-Relation wird innerhalb von Archimate im Vergleich zur Association-Relation als stärker bewertet. Daraus lässt sich ableiten, dass für

[13]Engl. deployable.

den Ausdruck des Mappings eines Geschäftsprozesses auf die Elemente des Application-Layer diese bevorzugt verwendet werden sollte. Wird lediglich ein Application-Service hinzugefügt, der eine unmittelbare Relation (Realization) zu einer Application-Component hat oder eine mittelbare über eine Application-Function, so ist ein solches Modell in einem geringeren Maße unterspezifiziert als eine direkte Zuordnung eines Verhaltenselementes zu einer Application-Component. In diesem Fall können aber immer noch keine Aussagen über die verwendete Schnittstelle gemacht werden. Deshalb wird als Modellierungskonvention die Verwendung einer Schnittstelle (Application-Interface) für die Zuordnung von Verhaltenselementen des Business Layers vorgeschlagen.

Die wesentliche Relation ist die Trigger Relation.

$$
\begin{aligned}
CFHOP = \{(a,b) | a \in AnBP \wedge b \in AnBP \wedge (ac1, ac2) | ac1, ac2 \in AC : \\
(\exists (a,b) \in Trigger \wedge (a, ac1) \in Assignment \wedge (b, ac2) \in Assignment) \\
\vee (\exists spa \in SP(a) \wedge (spa, b) \in Trigger \\
\qquad \wedge (spa, ac1) \in Assignment \wedge (b, ac2) \in Assignment) \\
\vee (\exists spb \in SP(b) \wedge (a, spb) \in Trigger \\
\qquad \wedge (a, ac1) \in Assignment \wedge (spb, ac2) \in Assignment) \\
\vee (\exists spa \in SP(a) \wedge \exists spb \in SP(b) \wedge (spa, spb) \in Trigger \\
\wedge (spa, ac1) \in Assignment \wedge (spb, ac2) \in Assignment)\}
\end{aligned}
\tag{8.1}
$$

Data-exchange-Muster: In diesem Muster spielt die Flow Relation die wesentliche Rolle. Ziel ist es, notwendige Datenübergaben zu erkennen.

$$
\begin{aligned}
DEP = \{(a,b) | a \in AnBP \wedge b \in AnBP \wedge (ac1, ac2) : ac1, ac2 \in AC \\
\wedge bo \in BO \wedge do1, do2 \in DO \\
\exists (a,b) \in Flow \wedge (a, ac1) \in Assignment \wedge (b, ac2) \in Assignment) \\
\vee (\exists spa \in SP(a) \wedge (spa, b) \in Flow \\
\qquad \wedge (spa, ac1) \in Assignment \wedge (b, ac2) \in Assignment) \\
\vee (\exists spb \in SP(b) \wedge (a, spb) \in Flow \\
\qquad \wedge (a, ac1) \in Assignment \wedge (spb, ac2) \in Assignment) \\
\vee (\exists spa \in SP(a) \wedge \exists spb \in SP(b) \wedge (spa, spb) \in Flow \\
\wedge (spa, ac1) \in Assignment \wedge (spb, ac2) \in Assignment)\}
\end{aligned}
\tag{8.2}
$$

Dieser Ansatz muss in Zukunft umfassend ausgebaut werden so tritt häufig eine Kombination von Kontrollflussübergabe und Datenaustauschnotwendigkeit auf, auch sind Phänomene wie eine physisch-digitale Synchronisation vorstellbar. Wird ein Geschäftsobjekt sowohl durch eine Repräsentation als auch durch ein Datenobjekt realisiert, so besteht die Notwendigkeit des Abgleichs zwischen der physischen und der digitalen Repräsentation. Dies kann z. B. bei Papierbelegen wie Rechnungen der Fall sein, zu denen ein digitales Pendant gespeichert wird. Über diese Ansätze hinaus wurde in der Arbeit eine Verbindung zwischen dem Modellierungsprototyp und einem Regelsystem (Drools) geschaffen, um die Muster als Regeln implementieren und so automatisiert prüfen zu können. Dieser Schritt ist der Erste auf dem Weg einer technischen Relaisierung der Ent-

scheidungsunterstützung für die Integration auf der Basis von Unternehmensarchitekturmodellen und kann in der Zukunft weiter verfolgt werden.

A Phasen von Integrationsprojekten

Teilphase	Beschreibung	beteiligte Rollen	Deliverables
n.a.	Verständnis schaffen (2N.)	n.a.	Solution Outline (erster Entwurf) (1N.)

Tabelle A.1: Presales-Phase
Quelle: (Gebauer und Stefan, 2011b, S. 37)

Teilphase	Beschreibung	beteiligte Rollen	Deliverables
Ist Analyse (6N.)	Analyse Ist Systeme, Daten, Kompetenzen, Umfeld, Business Value Workshop, geplante Unternehmensziele ermitteln		Unternehmensziele, Ist Stand
Soll Analyse (3N.)	aufnehmen von Vorgaben, Integrationsbeziehungen	Consultants	Soll Konzept
Fachliche Analyse (12N.)	Systemlandschaft aufnehmen (7N.)		Bebauungsplan
	Prozessaufnahme und Kontextanalyse (8N.), verstehen der abzubildenden Geschäftsthematik		Prozesslandkarte, Use Cases, Anforderungen
	Prozess-System-Mapping (3N.)		Prozesslaufschema, grobes Interaktionsmodell
	Bewertung (6N.); nach fachlichem Nutzen		bewertete Ist-Architektur, Potentialanalyse
Technische Analyse (9N.)	Schnittstellenidentifikation und Analyse (6N.), Klassifikation der Schnittstellen		identifizierte und klassifizierte Schnittstellen je System
	Kommunikationstypen, -muster und -protokolle (2N.)		den Systemen zugeordnete Kommunikationstypen und -muster
	technische Bewertung; Integrationsmöglichkeiten, Implementierungstechnologie,		technisch bewertete Systeme, detailliertes Interaktionsmodell
	Mappings; Mapping Notwendigkeit (1N.) identifizieren und Mappings ausarbeiten	Anwendungsberater	Mappingbeschreibungen

Tabelle A.2: Analysephase
Quelle: (Gebauer und Stefan, 2011b, S. 39)

Teilphasen	Beschreibung	beteiligte Rollen	Deliverables
Konzeption (6N.)	Entwicklung, Design des Lösungskonzeptes u.U. auf Basis von Referenzarchitekturen	Produktmanager, Anwendungsspezialist, Entwickler, externe Berater	Blueprint, Architekturvorschlag, Realsierungskonzept, Roadmap
Prototyping (1N.), Scope selection (1N.)	vorsichtiges Annähern an die Lösung; erster Test; Pilotierung		Prototyp, 1 - 2 Prozesse für Pilotprojekte ausgewählt
Evaluierung (5N.)	Auswahl und Bewertung von Technologien und Tools, auf Basis des Prototypen		Technologie- und Toolmatrix
Termin und Ressourcenplanung (5N.)	Planung der Ressourcen gemäß des erstellten Konzeptes		Lastenheft, Projektplan, Personaleinsatzplan, Angebot
Kundeninteraktion (3N.)	Abstimmungen mit Kunden; Konzept oder Managementpräsentation		Präsentation, Konzeptabnahme

Tabelle A.3: Planungsphase
Quelle: (Gebauer und Stefan, 2011b, S. 44)

Teilphase	Beschreibung	beteiligte Rollen	Arbeitsergebnisse
Umgebung aufsetzen	Installieren der Entwicklungsumgebung bzw. der Integrationsobjekte		Entwicklungsszenario
Durchführung	allgemein angesprochen (3N.); entspricht Implementierung (4N.); entspricht teilweise Customizing, Anpassung (2N.)		Implementierungskonzept, Installationspakete
Abnahme	interne Abnahme der Entwicklung		

Tabelle A.4: Fullfillment-Phase
Quelle: (Gebauer und Stefan, 2011b, S. 46)

Teilphase	Beschreibung	beteiligte Rollen	Deliverables
Test, Qualitätssicherung (4N.)	Sicherstellen der Qualität der Arbeitsergebnisse		gestestete Integrationsartefakte

Tabelle A.5: Testphase
Quelle: (Gebauer und Stefan, 2011b, S. 48)

Teilphase	Beschreibung	beteiligte Rollen	Deliverables
Umsetzungskonzept (1N.)	Planung der Einrichtung der Lösung beim Kunden		Umsetzungsplanung
Rollout (1N.)	Installation beim Kunden mit eigenen Tests		installierte Lösung
Schulung (2N.)	schulen der Endanwender		geschulte Anwender
Transition, Parallelbetrieb (4N.)	paralleles Betreiben der neuen Lösung		Erfahrungen im Realbetrieb
Endabnahme (3N.)	abschließende Abnahme der Lösung durch den Kunden		abgenommene Lösung

Tabelle A.6: Transitionsphase
Quelle: (Gebauer und Stefan, 2011b, S. 48)

Teilphase	Beschreibung	beteiligte Rollen	Deliverables
produktiver Betrieb der Lösung, Wartungstätigkeiten und Betriebsleistungen je nach Vereinbarung			betriebene Lösung, erbrachte Wertschöpfung in Richtung Geschäftsprozess

Tabelle A.7: Betriebsphase
Quelle: (Gebauer und Stefan, 2011b, S. 49)

B Bewertung der nicht funktionalen Eigenschaften

In diesem Anhang werden die nicht funktionalen Eigenschaften der Muster hergeleitet. Zu diesem Zweck werden die in der Literatur geäußerten Bewertungen der Qualitätsmerkmale und Teilmerkmale zu einer Bewertung des Qualitätsmerkmals zusammengefasst. Dabei resultiert eine posititive Bewertung aus ausschließlich posititven Bewertungen, ebenso wie eine negative Bewertung aus ausschließlich negeativen Bewertungen resultiert. Sind die Einzelwerte gemischt positiv und negativ so erfolgt die Bewertung ebenfalls gemischt um auf diese gegenläufigen Tendenzen hin zu weisen. Die Ausnahme bildet dabei die Bewertung neutral (n). Wurde in der Literatur auch das Qualitätsmerkmal anstelle oder zusammen mit den eigentlichen Teilmerkmalen bewertet, so wird diese Bewertung wie ein Teilmerkmal behandelt und fließt so in die Gesamtbewertung mit ein.

Den Anfang machen die Ausführungen zum Merkmal der Funktionalität (*Functionality*).

Pattern	**Quelle**	**Bewertung**	Functionality	Accuracy	Suitability	Interoperability	Security
Adapter Style	(Andersson und Johnson, 2001, S. 229)	+	+				
Adapter static	(ebenda, S. 229)	+	+				
Adapter dynamic	(ebenda, S. 229)	+	+				
Adapter thin	(ebenda, S. 229)	+	+				
Adapter thick	(ebenda, S. 229)	+	+				
Adapter centralized	(ebenda, S. 229)	+	+				
Adapter distributed	(ebenda, S. 229)	+	+				
Adapter	(Khomh und Guéhéneuc, 2008a)	+			+		
(Integration) Adapter	(Lutz, 2000)						
Component Wrapper	(Goedicke und Zdun, 2002, S. 7ff.)	+			+		
Wrapper Facade	(Schmidt; Stal u. a., 2000, S. 47ff.)	-	-				
(Legacy) Wrapper	(Mularz, 1995)						
Facade	(Khomh und Guéhéneuc, 2008a)	+			+		
Integration Facade	(Lutz, 2000)						
Service Interface	(Trowbridge, 2003)	-	-				
Integration Messenger (broker)	(Lutz, 2000)						
Integration Messenger (message queuing)	(ebenda)						
Integration Messenger (publish subscribe)	(ebenda)						
Proxy	(Buschmann; Meunier u. a., 1996, S. 263ff.)						
Proxy	(Khomh und Guéhéneuc, 2008a)	+			+		
Mediator	(Gamma u. a., 1995, S. 277)						
Integration Mediator	(Lutz, 2000, S. 71)						
Mediator	(Buschmann; Henney und Schmidt, 2007a, S. 411)						
Mediator	(Khomh und Guéhéneuc, 2008a)				-		
Mediator Style	(Andersson und Johnson, 2001, S. 230)	+	+				+
Message Router	(ebenda, S. 231)	+/-	-				+
Message Router	(Hohpe und Woolf, 2003, S. 81)						
Message Router	(Buschmann; Henney und Schmidt, 2007a, S. 232)						
Gateway	(Andersson und Johnson, 2001)	**n**					n
Service Gateway	(Trowbridge, 2003, S. 293ff.)						
Broker	(ebenda, S. 207f.)	+					+
Direct Broker	(Trowbridge u. a., 2004, S. 155,215ff.)						

Bewertung des Qualitätsmerkmals Functionality – Fortsetzung auf der nächsten Seite

Pattern	Quelle	**Bewertung**	Functionality	Accuracy	Suitability	Interoperability	Security	
Indirect Broker	(Trowbridge u. a., 2004, S. 221)							
Message Broker	(ebenda, S. 240f.)	+				+	+	
Message Broker	(Hohpe und Woolf, 2003, S. 322ff.)							
Broker	(Buschmann; Meunier u. a., 1996, S. 119ff.)	+				+		
Broker Revisited	(Kircher; Voelter u. a., 2004, S. 581)							
Broker	(Harrison und Avgeriou, 2007, S. 267f.)	+					+	
Pipes and Filter	(Hohpe und Woolf, 2003, S. 70ff., 322ff.)							
Pipes and Filter	(Buschmann; Meunier u. a., 1996, S. 67ff.)							
Pipes and Filter	(Trowbridge u. a., 2004, S. 292f.)							
Pipes and Filters	(Garlan und Shaw, 1993, S. 7f.)							
Pipes and Filters	(Meunier, 1995, S. 437)							
Pipes and Filters	(Harrison und Avgeriou, 2007, S. 267f.)	-					-	
Observer	(Khomh und Guéhéneuc, 2008a)	+			+			
Observer	(Trowbridge, 2003, S. 129f.)							
Publish and Subscribe	(Trowbridge u. a., 2004, S. 280f.)	+					+	
Process Automator	(Lutz, 2000)							
Database Federation	(Andersson und Johnson, 2001)	+					+	
Desktop Integration	(ebenda)	+					+(thick)	
Data Transfer Object	(Trowbridge, 2003, S. 229)							
Singleton	(ebenda, S. 257)	+	+					
Singleton	(Khomh und Guéhéneuc, 2008a)	-			-			
File Gateway	(Erl, 2009, S. 457)							
Process Centralizazion	(ebenda, S. 193)							
Protocol Bridging	(ebenda, S. 687)							
Legacy Wrapper	(ebenda, S. 441)							
Service Broker (COMP)	(ebenda, S. 707)							
Enterprise Service Bus (COMP)	(ebenda, S. 704)							
Proxy Capability	(ebenda, S. 497)							
Service Facade	(ebenda, S. 333)							
UI Mediator	(ebenda, S. 366)							
Data Model Transformation	(ebenda, S. 671)							
Data Format Transformation	(ebenda, S. 681)							
Asynchronous Queuing	(ebenda, S. 582)							
Intermediate Routing	(ebenda, S. 549)	-					-	

+: positiv; -: negativ; +/-: posititv/ negativ; n: neutral

Tabelle B.1: Bewertung des Qualitätsmerkmals Functionality

Pattern	Quelle	**Bewertung**	Reliability	Maturity	Fault tolerance	Recoverability
Adapter Style	(Andersson und Johnson, 2001, S. 229)					
Adapter static	(ebenda, S. 229)					
Adapter dynamic	(ebenda, S. 229)					
Adapter thin	(ebenda, S. 229)					
Adapter thick	(ebenda, S. 229)					
Adapter centralized	(ebenda, S. 229)					
Adapter distributed	(ebenda, S. 229)	+				+
Adapter	(Khomh und Guéhéneuc, 2008a)	-			-	
(Integration) Adapter	(Lutz, 2000)					
Component Wrapper	(Goedicke und Zdun, 2002, S. 7ff.)					

Bewertung des Qualitätsmerkmals Reliability – Fortsetzung auf der nächsten Seite

Pattern	Quelle	**Bewertung**	Reliability	Maturity	Fault tolerance	Recoverability
Wrapper Facade	(Schmidt; Stal u. a., 2000, S. 47ff.)					
(Legacy) Wrapper	(Mularz, 1995)					
Facade	(Khomh und Guéhéneuc, 2008a)	-			-	
Integration Facade	(Lutz, 2000)					
Service Interface	(Trowbridge, 2003)					
Integration Messenger (broker)	(Lutz, 2000)					
Integration Messenger (message queuing)	(ebenda)					
Integration Messenger (publish subscribe)	(ebenda)					
Proxy	(Buschmann; Meunier u. a., 1996, S. 263ff.)					
Proxy	(Khomh und Guéhéneuc, 2008a)	-			-	
Mediator	(Gamma u. a., 1995, S. 277)					
Integration Mediator	(Lutz, 2000, S. 71)					
Mediator	(Buschmann; Henney und Schmidt, 2007a, S. 411)	-	-			
Mediator	(Khomh und Guéhéneuc, 2008a)	-			-	
Mediator Style	(Andersson und Johnson, 2001, S. 230)	-	-			
Message Router	(ebenda, S. 231)	-	-			
Message Router	(Hohpe und Woolf, 2003, S. 81)					
Message Router	(Buschmann; Henney und Schmidt, 2007a, S. 232)					
Gateway	(Andersson und Johnson, 2001)	n	n			
Service Gateway	(Trowbridge, 2003, S. 293ff.)					
Broker	(ebenda, S. 207f.)					
Direct Broker	(Trowbridge u. a., 2004, S. 155,215ff.)					
Indirect Broker	(ebenda, S. 221)					
Message Broker	(ebenda, S. 240f.)	-				-
Message Broker	(Hohpe und Woolf, 2003, S. 322ff.)					
Broker	(Buschmann; Meunier u. a., 1996, S. 119ff.)	-			-	
Broker Revisited	(Kircher; Voelter u. a., 2004, S. 581)	-	-			
Broker	(Harrison und Avgeriou, 2007, S. 267f.)	n	n			
Pipes and Filter	(Hohpe und Woolf, 2003, S. 70ff., 322ff.)					
Pipes and Filter	(Buschmann; Meunier u. a., 1996, S. 67ff.)					
Pipes and Filter	(Trowbridge u. a., 2004, S. 292f.)					
Pipes and Filters	(Garlan und Shaw, 1993, S. 7f.)					
Pipes and Filters	(Meunier, 1995, S. 437)					
Pipes and Filters	(Harrison und Avgeriou, 2007, S. 267f.)	-	-			
Observer	(Khomh und Guéhéneuc, 2008a)	-			-	
Observer	(Trowbridge, 2003, S. 129f.)					
Publish and Subscribe	(Trowbridge u. a., 2004, S. 280f.)					
Process Automator	(Lutz, 2000)	+	+			
Database Federation	(Andersson und Johnson, 2001)	-	-			
Desktop Integration	(ebenda)					
Data Transfer Object	(Trowbridge, 2003, S. 229)					
Singleton	(ebenda, S. 257)					
Singleton	(Khomh und Guéhéneuc, 2008a)	-			-	
File Gateway	(Erl, 2009, S. 457)					
Process Centralizazion	(ebenda, S. 193)					
Protocol Bridging	(ebenda, S. 687)	-	-			
Legacy Wrapper	(ebenda, S. 441)					
Service Broker (COMP)	(ebenda, S. 707)					
Enterprise Service Bus (COMP)	(ebenda, S. 704)					
Proxy Capability	(ebenda, S. 497)	-	-			
Service Facade	(ebenda, S. 333)					

Bewertung des Qualitätsmerkmals Reliability – Fortsetzung auf der nächsten Seite

Pattern	Quelle	**Bewertung**	Reliability	Maturity	Fault tolerance	Recoverability
UI Mediator	(Erl, 2009, S. 366)					
Data Model Transformation	(ebenda, S. 671)					
Data Format Transformation	(ebenda, S. 681)					
Asynchronous Queuing	(ebenda, S. 582)	+/-	-		+	
Intermediate Routing	(ebenda, S. 549)	-			-	

+: positiv; -: negativ; +/-: posititv/ negativ; n: neutral

Tabelle B.2: Bewertung des Qualitätsmerkmals Reliability

Pattern	Quelle	**Bewertung**	Usability	Unterstandability	Leranability	Operability	Attractiveness
Adapter Style	(Andersson und Johnson, 2001, S. 229)						
Adapter static	(ebenda, S. 229)						
Adapter dynamic	(ebenda, S. 229)						
Adapter thin	(ebenda, S. 229)						
Adapter thick	(ebenda, S. 229)						
Adapter centralized	(ebenda, S. 229)						
Adapter distributed	(ebenda, S. 229)						
Adapter	(Khomh und Guéhéneuc, 2008a)	+/-		-	+		
(Integration) Adapter	(Lutz, 2000)						
Component Wrapper	(Goedicke und Zdun, 2002, S. 7ff.)	-		-			
Wrapper Facade	(Schmidt; Stal u. a., 2000, S. 47ff.)	+		+			
(Legacy) Wrapper	(Mularz, 1995)						
Facade	(Khomh und Guéhéneuc, 2008a)	+		+	+		
Integration Facade	(Lutz, 2000)						
Service Interface	(Trowbridge, 2003)						
Integration Messenger (broker)	(Lutz, 2000)						
Integration Messenger (message queuing)	(ebenda)						
Integration Messenger (publish subscribe)	(ebenda)						
Proxy	(Buschmann; Meunier u. a., 1996, S. 263ff.)						
Proxy	(Khomh und Guéhéneuc, 2008a)	+/-		+/-	-		
Mediator	(Gamma u. a., 1995, S. 277)	+		+			
Integration Mediator	(Lutz, 2000, S. 71)						
Mediator	(Buschmann; Henney und Schmidt, 2007a, S. 411)						
Mediator	(Khomh und Guéhéneuc, 2008a)	+/-		+/-	+		
Mediator Style	(Andersson und Johnson, 2001, S. 230)						
Message Router	(ebenda, S. 231)						
Message Router	(Hohpe und Woolf, 2003, S. 81)	-		-			
Message Router	(Buschmann; Henney und Schmidt, 2007a, S. 232)	-		-			
Gateway	(Andersson und Johnson, 2001)						
Service Gateway	(Trowbridge, 2003, S. 293ff.)						
Broker	(ebenda, S. 207f.)	+			+		
Direct Broker	(Trowbridge u. a., 2004, S. 155,215ff.)						
Indirect Broker	(ebenda, S. 221)						
Message Broker	(ebenda, S. 240f.)						
Message Broker	(Hohpe und Woolf, 2003, S. 322ff.)						
Broker	(Buschmann; Meunier u. a., 1996, S. 119ff.)						

Bewertung des Qualitätsmerkmals Usability – Fortsetzung auf der nächsten Seite

Pattern	Quelle	**Bewertung**	Usability	Unterstandability	Leranability	Operability	Attractiveness
Broker Revisited	(Kircher; Voelter u. a., 2004, S. 581)						
Broker	(Harrison und Avgeriou, 2007, S. 267f.)	+	+				
Pipes and Filter	(Hohpe und Woolf, 2003, S. 70ff., 322ff.)						
Pipes and Filter	(Buschmann; Meunier u. a., 1996, S. 67ff.)						
Pipes and Filter	(Trowbridge u. a., 2004, S. 292f.)	-			-		
Pipes and Filters	(Garlan und Shaw, 1993, S. 7f.)	+		+			
Pipes and Filters	(Meunier, 1995, S. 437)	+		+			
Pipes and Filters	(Harrison und Avgeriou, 2007, S. 267f.)	-	-				
Observer	(Khomh und Guéhéneuc, 2008a)	+/-		+/-	+		
Observer	(Trowbridge, 2003, S. 129f.)	-		-			
Publish and Subscribe	(Trowbridge u. a., 2004, S. 280f.)						
Process Automator	(Lutz, 2000)						
Database Federation	(Andersson und Johnson, 2001)						
Desktop Integration	(ebenda)	-	-				
Data Transfer Object	(Trowbridge, 2003, S. 229)	+		+			
Singleton	(ebenda, S. 257)	-			-		
Singleton	(Khomh und Guéhéneuc, 2008a)	+		+	+		
File Gateway	(Erl, 2009, S. 457)						
Process Centralizazion	(ebenda, S. 193)						
Protocol Bridging	(ebenda, S. 687)						
Legacy Wrapper	(ebenda, S. 441)						
Service Broker (COMP)	(ebenda, S. 707)						
Enterprise Service Bus (COMP)	(ebenda, S. 704)						
Proxy Capability	(ebenda, S. 497)						
Service Facade	(ebenda, S. 333)						
UI Mediator	(ebenda, S. 366)						
Data Model Transformation	(ebenda, S. 671)						
Data Format Transformation	(ebenda, S. 681)						
Asynchronous Queuing	(ebenda, S. 582)						
Intermediate Routing	(ebenda, S. 549)						

+: positiv; -: negativ; +/-: posititv/ negativ; n: neutral

Tabelle B.3: Bewertung des Qualitätsmerkmals Usability

Pattern	Quelle	**Bewertung**	Efficiency	Time behavior	Resource behavior
Adapter Style	(Andersson und Johnson, 2001, S. 229)				
Adapter static	(ebenda, S. 229)				
Adapter dynamic	(ebenda, S. 229)				
Adapter thin	(ebenda, S. 229)				
Adapter thick	(ebenda, S. 229)				
Adapter centralized	(ebenda, S. 229)				
Adapter distributed	(ebenda, S. 229)				
Adapter	(Khomh und Guéhéneuc, 2008a)				
(Integration) Adapter	(Lutz, 2000)				
Component Wrapper	(Goedicke und Zdun, 2002, S. 7ff.)	-	-		
Wrapper Facade	(Schmidt; Stal u. a., 2000, S. 47ff.)	-		-	
(Legacy) Wrapper	(Mularz, 1995)				

Bewertung des Qualitätsmerkmals Efficiency – Fortsetzung auf der nächsten Seite

Pattern	Quelle	**Bewertung**	Efficiency	Time behavior	Resource behavior
Facade	(Khomh und Guéhéneuc, 2008a)				
Integration Facade	(Lutz, 2000)				
Service Interface	(Trowbridge, 2003)	**-**			-
Integration Messenger (broker)	(Lutz, 2000)				
Integration Messenger (message queuing)	(ebenda)				
Integration Messenger (publish subscribe)	(ebenda)				
Proxy	(Buschmann; Meunier u. a., 1996, S. 263ff.)	**+/-**	+/-		
Proxy	(Khomh und Guéhéneuc, 2008a)				
Mediator	(Gamma u. a., 1995, S. 277)				
Integration Mediator	(Lutz, 2000, S. 71)				
Mediator	(Buschmann; Henney und Schmidt, 2007a, S. 411)	**-**	-		
Mediator	(Khomh und Guéhéneuc, 2008a)				
Mediator Style	(Andersson und Johnson, 2001, S. 230)	+		+	
Message Router	(ebenda, S. 231)				
Message Router	(Hohpe und Woolf, 2003, S. 81)	**-**		-	-
Message Router	(Buschmann; Henney und Schmidt, 2007a, S. 232)	**-**	-		
Gateway	(Andersson und Johnson, 2001)	**-**		-	
Service Gateway	(Trowbridge, 2003, S. 293ff.)				
Broker	(ebenda, S. 207f.)	**-**		-	
Direct Broker	(Trowbridge u. a., 2004, S. 155,215ff.)				
Indirect Broker	(ebenda, S. 221)				
Message Broker	(ebenda, S. 240f.)	**-**	-		
Message Broker	(Hohpe und Woolf, 2003, S. 322ff.)				
Broker	(Buschmann; Meunier u. a., 1996, S. 119ff.)	**-**		-	
Broker Revisited	(Kircher; Voelter u. a., 2004, S. 581)	**-**			-
Broker	(Harrison und Avgeriou, 2007, S. 267f.)	**n**	n		
Pipes and Filter	(Hohpe und Woolf, 2003, S. 70ff., 322ff.)	**+/-**		+/-	-
Pipes and Filter	(Buschmann; Meunier u. a., 1996, S. 67ff.)	**+/-**		+/-	+/-
Pipes and Filter	(Trowbridge u. a., 2004, S. 292f.)	**+/-**	+/-		
Pipes and Filters	(Garlan und Shaw, 1993, S. 7f.)	**+/-**		+	-
Pipes and Filters	(Meunier, 1995, S. 437)	**+/-**	+/-		
Pipes and Filters	(Harrison und Avgeriou, 2007, S. 267f.)	**+/-**	+/-		
Observer	(Khomh und Guéhéneuc, 2008a)				
Observer	(Trowbridge, 2003, S. 129f.)	**-**		-	-
Publish and Subscribe	(Trowbridge u. a., 2004, S. 280f.)	**-**	-		
Process Automator	(Lutz, 2000)				
Database Federation	(Andersson und Johnson, 2001)				
Desktop Integration	(ebenda)				
Data Transfer Object	(Trowbridge, 2003, S. 229)	**+/-**		+	-
Singleton	(ebenda, S. 257)	**-**		-	-
Singleton	(Khomh und Guéhéneuc, 2008a)				
File Gateway	(Erl, 2009, S. 457)	**-**		-	
Process Centralizazion	(ebenda, S. 193)				
Protocol Bridging	(ebenda, S. 687)	**-**		-	
Legacy Wrapper	(ebenda, S. 441)	**-**		-	
Service Broker (COMP)	(ebenda, S. 707)				
Enterprise Service Bus (COMP)	(ebenda, S. 704)				
Proxy Capability	(ebenda, S. 497)				
Service Facade	(ebenda, S. 333)	**-**		-	
UI Mediator	(ebenda, S. 366)	**-**		-	-
Data Model Transformation	(ebenda, S. 671)	**-**		-	-

Bewertung des Qualitätsmerkmals Efficiency – Fortsetzung auf der nächsten Seite

Pattern	Quelle	**Bewertung**	Efficiency	Time behavior	Resource behavior
Data Format Transformation	(ebenda, S. 681)	**-**		-	
Asynchronous Queuing	(ebenda, S. 582)	**-**		-	
Intermediate Routing	(ebenda, S. 549)	**-**		-	

+: positiv; -: negativ; +/-: posititv/ negativ; n: neutral

Tabelle B.4: Bewertung des Qualitätsmerkmals Efficiency

Pattern	Quelle	**Bewertung**	Maintainability	Analyzability	Changeability	Stability	Testability
Adapter Style	(Andersson und Johnson, 2001, S. 229)	**-**			-		
Adapter static	(ebenda, S. 229)	**-**			-		
Adapter dynamic	(ebenda, S. 229)	**-**			-		
Adapter thin	(ebenda, S. 229)	**-**			-		
Adapter thick	(ebenda, S. 229)	**-**			-		
Adapter centralized	(ebenda, S. 229)	**-**			-		
Adapter distributed	(ebenda, S. 229)	**-**			-		
Adapter	(Khomh und Guéhéneuc, 2008a)	+			+		
(Integration) Adapter	(Lutz, 2000)						
Component Wrapper	(Goedicke und Zdun, 2002, S. 7ff.)	+			+		
Wrapper Facade	(Schmidt; Stal u. a., 2000, S. 47ff.)	+			+		
(Legacy) Wrapper	(Mularz, 1995)						
Facade	(Khomh und Guéhéneuc, 2008a)	+			+		
Integration Facade	(Lutz, 2000)						
Service Interface	(Trowbridge, 2003)	**+/-**			+/-		
Integration Messenger (broker)	(Lutz, 2000)						
Integration Messenger (message queuing)	(ebenda)						
Integration Messenger (publish subscribe)	(ebenda)						
Proxy	(Buschmann; Meunier u. a., 1996, S. 263ff.)	+				+	
Proxy	(Khomh und Guéhéneuc, 2008a)	**-**			-		
Mediator	(Gamma u. a., 1995, S. 277)	+	+		+		
Integration Mediator	(Lutz, 2000, S. 71)	+	+				
Mediator	(Buschmann; Henney und Schmidt, 2007a, S. 411)	**-**	-				
Mediator	(Khomh und Guéhéneuc, 2008a)	+			+		
Mediator Style	(Andersson und Johnson, 2001, S. 230)						
Message Router	(ebenda, S. 231)						
Message Router	(Hohpe und Woolf, 2003, S. 81)	**+/-**	+/-	-		+	-
Message Router	(Buschmann; Henney und Schmidt, 2007a, S. 232)	**+/-**	+/-	-	+		
Gateway	(Andersson und Johnson, 2001)	**+/-**			-(static) +(dynamic)		
Service Gateway	(Trowbridge, 2003, S. 293ff.)						
Broker	(ebenda, S. 207f.)						
Direct Broker	(Trowbridge u. a., 2004, S. 155,215ff.)						
Indirect Broker	(ebenda, S. 221)						
Message Broker	(ebenda, S. 240f.)	**+/-**	-	-			+
Message Broker	(Hohpe und Woolf, 2003, S. 322ff.)						
Broker	(Buschmann; Meunier u. a., 1996, S. 119ff.)	**+/-**			+		+/-

Bewertung des Qualitätsmerkmals Maintainability – Fortsetzung auf der nächsten Seite

Pattern	Quelle	**Bewertung**	Maintainability	Analyzability	Changeability	Stability	Testability
Broker Revisited	(Kircher; Voelter u. a., 2004, S. 581)						
Broker	(Harrison und Avgeriou, 2007, S. 267f.)	+	+				
Pipes and Filter	(Hohpe und Woolf, 2003, S. 70ff., 322ff.)	+			+		+
Pipes and Filter	(Buschmann; Meunier u. a., 1996, S. 67ff.)	+			+		
Pipes and Filter	(Trowbridge u. a., 2004, S. 292f.)	-	-	-			
Pipes and Filters	(Garlan und Shaw, 1993, S. 7f.)	+	+	+	+		
Pipes and Filters	(Meunier, 1995, S. 437)	+		+	+		+
Pipes and Filters	(Harrison und Avgeriou, 2007, S. 267f.)	+	+				
Observer	(Khomh und Guéhéneuc, 2008a)	+			+		
Observer	(Trowbridge, 2003, S. 129f.)	-					-
Publish and Subscribe	(Trowbridge u. a., 2004, S. 280f.)	+/-	-	-			+
Process Automator	(Lutz, 2000)	+		+	+		
Database Federation	(Andersson und Johnson, 2001)	+			+		
Desktop Integration	(ebenda)	-			-		
Data Transfer Object	(Trowbridge, 2003, S. 229)	+/-	-				+
Singleton	(ebenda, S. 257)	+			+		
Singleton	(Khomh und Guéhéneuc, 2008a)	-			-		
File Gateway	(Erl, 2009, S. 457)						
Process Centralizazion	(ebenda, S. 193)						
Protocol Bridging	(ebenda, S. 687)						
Legacy Wrapper	(ebenda, S. 441)						
Service Broker (COMP)	(ebenda, S. 707)						
Enterprise Service Bus (COMP)	(ebenda, S. 704)						
Proxy Capability	(ebenda, S. 497)					-	
Service Facade	(ebenda, S. 333)						
UI Mediator	(ebenda, S. 366)						
Data Model Transformation	(ebenda, S. 671)						
Data Format Transformation	(ebenda, S. 681)						
Asynchronous Queuing	(ebenda, S. 582)	+/-		+/-			
Intermediate Routing	(ebenda, S. 549)	+/-			+/-		-

+: positiv; -: negativ; +/-: posititv/ negativ; n: neutral

Tabelle B.5: Bewertung des Qualitätsmerkmals Maintainability

Pattern	Quelle	**Bewertung**	Portability	Adaptability	Installability	Replaceability	Co-existence
Adapter Style	(Andersson und Johnson, 2001, S. 229)						
Adapter static	(ebenda, S. 229)						
Adapter dynamic	(ebenda, S. 229)						
Adapter thin	(ebenda, S. 229)						
Adapter thick	(ebenda, S. 229)						
Adapter centralized	(ebenda, S. 229)						
Adapter distributed	(ebenda, S. 229)						
Adapter	(Khomh und Guéhéneuc, 2008a)	-		-			
(Integration) Adapter	(Lutz, 2000)	-					-(intrusive)
Component Wrapper	(Goedicke und Zdun, 2002, S. 7ff.)	+		+		+	
Wrapper Facade	(Schmidt; Stal u. a., 2000, S. 47ff.)	+	+			+	
(Legacy) Wrapper	(Mularz, 1995)						
Facade	(Khomh und Guéhéneuc, 2008a)	-		-			
Integration Facade	(Lutz, 2000)						

Bewertung des Qualitätsmerkmals Portability – Fortsetzung auf der nächsten Seite

Pattern	Quelle	**Bewertung**	Portability	Adaptability	Installability	Replaceability	Co-existence
Service Interface	(Trowbridge, 2003)	+				+	
Integration Messenger (broker)	(Lutz, 2000)						
Integration Messenger (message queuing)	(ebenda)						
Integration Messenger (publish subscribe)	(ebenda)						
Proxy	(Buschmann; Meunier u. a., 1996, S. 263ff.)	-		-			
Proxy	(Khomh und Guéhéneuc, 2008a)	-		-			
Mediator	(Gamma u. a., 1995, S. 277)	+		+		+	
Integration Mediator	(Lutz, 2000, S. 71)						
Mediator	(Buschmann; Henney und Schmidt, 2007a, S. 411)						
Mediator	(Khomh und Guéhéneuc, 2008a)	-		-			
Mediator Style	(Andersson und Johnson, 2001, S. 230)	+		+			
Message Router	(ebenda, S. 231)	+		+			
Message Router	(Hohpe und Woolf, 2003, S. 81)	+		+			
Message Router	(Buschmann; Henney und Schmidt, 2007a, S. 232)						
Gateway	(Andersson und Johnson, 2001)						
Service Gateway	(Trowbridge, 2003, S. 293ff.)	+/-				+	-
Broker	(ebenda, S. 207f.)	+		+		+	
Direct Broker	(Trowbridge u. a., 2004, S. 155,215ff.)						
Indirect Broker	(ebenda, S. 221)						
Message Broker	(ebenda, S. 240f.)	+		+		+	
Message Broker	(Hohpe und Woolf, 2003, S. 322ff.)						
Broker	(Buschmann; Meunier u. a., 1996, S. 119ff.)	+	+	+			
Broker Revisited	(Kircher; Voelter u. a., 2004, S. 581)	+	+				
Broker	(Harrison und Avgeriou, 2007, S. 267f.)	+	+				
Pipes and Filter	(Hohpe und Woolf, 2003, S. 70ff., 322ff.)						
Pipes and Filter	(Buschmann; Meunier u. a., 1996, S. 67ff.)	+		+			
Pipes and Filter	(Trowbridge u. a., 2004, S. 292f.)	+/-		+		+	-
Pipes and Filters	(Garlan und Shaw, 1993, S. 7f.)						
Pipes and Filters	(Meunier, 1995, S. 437)	+		+		+	
Pipes and Filters	(Harrison und Avgeriou, 2007, S. 267f.)	+	+				
Observer	(Khomh und Guéhéneuc, 2008a)	-		-			
Observer	(Trowbridge, 2003, S. 129f.)	+		+		+	
Publish and Subscribe	(Trowbridge u. a., 2004, S. 280f.)	+				+	
Process Automator	(Lutz, 2000)	+		+			
Database Federation	(Andersson und Johnson, 2001)						
Desktop Integration	(ebenda)						
Data Transfer Object	(Trowbridge, 2003, S. 229)						
Singleton	(ebenda, S. 257)						
Singleton	(Khomh und Guéhéneuc, 2008a)	-		-			
File Gateway	(Erl, 2009, S. 457)						
Process Centralizazion	(ebenda, S. 193)						
Protocol Bridging	(ebenda, S. 687)						
Legacy Wrapper	(ebenda, S. 441)						
Service Broker (COMP)	(ebenda, S. 707)						
Enterprise Service Bus (COMP)	(ebenda, S. 704)						
Proxy Capability	(ebenda, S. 497)						
Service Facade	(ebenda, S. 333)						
UI Mediator	(ebenda, S. 366)						

Bewertung des Qualitätsmerkmals Portability – Fortsetzung auf der nächsten Seite

Pattern	**Quelle**	**Bewertung**	Portability	Adaptability	Installability	Replaceability	Co-existence
Data Model Transformation	(Erl, 2009, S. 671)						
Data Format Transformation	(ebenda, S. 681)						
Asynchronous Queuing	(ebenda, S. 582)						
Intermediate Routing	(ebenda, S. 549)						
+: positiv; -: negativ; +/-: posititv/ negativ; n: neutral							

Tabelle B.6: Bewertung des Qualitätsmerkmals Portability

C Gesamtklassifikation

Pattern	**Quelle**	Subsystem BIS	Integrationsoperation	Integrationsarchetyp	Komponente der Referenzarchitketur	Heterogenitätskonflikt	Zweck der Kommunikation	Verbindungsorientierung	Kardinalität d. Komm	Interaktionsmechanismus	Interaktionslogik	Send	Receive	Time/Synchronität	Space	Functionality	Reliability	Usability	Efficiency	Maintainability	Portability	Reusability
Adapter Style	(Andersson und Johnson, 2001, S. 229)	AM	Ad	B	C/D	I		vo	1:1	PA	d			tc	sc	+				-		
Adapter static	(ebenda, S. 229)	AM	Ad	B	C/D	I		vo	1:1	PA	d			tc	sc	+				-		
Adapter dynamic	(ebenda, S. 229)	AM	Ad	B	C/D	I		vo	1:1	PA	d			tc	sd	+				-		
Adapter thin	(ebenda, S. 229)	AM	Ad	B	C/D	I		vo	1:1	PA	d			tc	sc	+				-		
Adapter thick	(ebenda, S. 229)	AM	Ad	B	C/D	kK		vo	1:1	PA	d			tc	sc	+				-		
Adapter centralized	(ebenda, S. 229)	AM	Ad	B	C/D	I		vo	1:1	PA	d			tc	sc	+				-		
Adapter distributed	(ebenda, S. 229)	AM	Ad	B	C/D	I		vo	1:1	PA	d			tc	sc	+	+			-		
Adapter	(Khomh und Guéhéneuc, 2008a)	EM														+	-	+/-		+	-	+
(Integration) Adapter	(Lutz, 2000)	AM	Ad	B	C/D	I		vo	m:1		d			tc	sd						-	+
Component Wrapper	(Goedicke und Zdun, 2002, S. 7ff.)	EM	Ad	B	C/D	I		vo	1:1	PA	d			tc	sc	+		-	-	+	+	+

Gesamtklassifikation – Fortsetzung auf der nächsten Seite

Pattern	**Quelle**	Subsystem BIS	Integrationsoperation	Integrationsarchetyp	Komponente der Referenzarchitketur	Heterogenitätskonflikt	Zweck der Kommunikation	Verbindungsorientierung	Kardinalität d. Komm	Interaktionsmechanismus	Interaktionslogik	Send	Receive	Time/Synchronität	Space	Functionality	Reliability	Usability	Efficiency	Maintainability	Portability	Reusability
Wrapper Facade	(Schmidt; Stal u. a., 2000, S. 47ff.)	EM	Ad					vo	1:1	PA	d			tc	sc	-		+	-	+	+	+
(Legacy) Wrapper	(Mularz, 1995)	AM	Ad	B	C/D	I			1:n		d			tc	sc							
Facade	(Khomh und Guéhéneuc, 2008a)															+	-	+		+	-	-
Integration Facade	(Lutz, 2000)	AM	iK	B	C/D	kK			m:n		z			tc	sd							
Service Interface	(Trowbridge, 2003)	AM	Ad	B	C/D	kK		vl	1:n	N	d	nbs	nbr	td	sd	-			-	+/-	+	
Integration Messenger (broker)	(Lutz, 2000)	AM	iK	B	L	Tr		vo	1:1	N	d	nbs	nbr	tc	sd							
Integration Messenger (message queuing)	(ebenda)	AM	iK	B	L	Tr		vl	1:1	N	d	nbs	nbr	td	sd							
Integration Messenger (publish subscribe)	(ebenda)	AM	iK	B	L	Tr		vl	1:n	N	d	nbs	nbr	td	sd							
Proxy	(Buschmann; Meunier u. a., 1996, S. 263ff.)	EM		B	C/D	Tr		vo	1:n	PA	d	bs	br	tc	sd				+/-	+	-	+
Proxy	(Khomh und Guéhéneuc, 2008a)															+	-	+/-		-	-	+

Gesamtklassifikation – Fortsetzung auf der nächsten Seite

Pattern	**Quelle**	Subsystem BIS	Integrationsoperation	Integrationsarchetyp	Komponente der Referenzarchitketur	Heterogenitätskonflikt	Zweck der Kommunikation	Verbindungsorientierung	Kardinalität d. Komm	Interaktionsmechanismus	Interaktionslogik	Send	Receive	Time/Synchronität	Space	Functionality	Reliability	Usability	Efficiency	Maintainability	Portability	Reusability
Mediator	(Gamma u. a., 1995, S. 277)	EM	iK	B	L	Tr		vo	1:n	PA	z	bs	br	tc	sd			+		+	+	+/-
Integration Mediator	(Lutz, 2000, S. 71)	AM	iK	B	L	kK		vl	m:n		z	nbs	nbr		sd					+		+
Mediator	(Buschmann; Henney und Schmidt, 2007a, S. 411)	EM	iK	B	L	Tr			1:n		z				sd		-		-	-		
Mediator	(Khomh und Guéhéneuc, 2008a)																-	+/-		+	-	-
Mediator Style	(Andersson und Johnson, 2001, S. 230)	AM	iK	B	L	kK		vl	m:n		z	nbs	nbr	td		+	-		+		+	
Message Router	(ebenda, S. 231)	AM	iK	B	L	Tr		vl	m:n	N	z	nbs	nbr	td	sd	+/-	-				+	
Message Router	(Hohpe und Woolf, 2003, S. 81)	AM	iK	B	L	Tr		vl	1:n	N	z	nbs	nbr	td	sd			-	-	+/-	+	
Message Router	(Buschmann; Henney und Schmidt, 2007a, S. 232)	AM	iK	B	L	Tr		vl	m:n	N	d	nbs	nbr	td	sd			-	-	+/-		
Gateway	(Andersson und Johnson, 2001)	AM	dK	B	C/D	I (kK thick)		vo	1:1	PA	d			tc	sc (sd dynamic)	n	n		-	+/-		

Gesamtklassifikation – Fortsetzung auf der nächsten Seite

Pattern	Quelle	Subsystem BIS	Integrationsoperation	Integrationsarchetyp	Komponente der Referenzarchitketur	Heterogenitätskonflikt	Zweck der Kommunikation	Verbindungsorientierung	Kardinalität d. Komm	Interaktionsmechanismus	Interaktionslogik	Send	Receive	Time/Synchronität	Space	Functionality	Reliability	Usability	Efficiency	Maintainability	Portability	Reusability
Service Gateway	(Trowbridge, 2003, S. 293ff.)	AM	iK	B	C/D	kK	D	vl	m:n	N	d			td	sd						+/-	+
Broker	(ebenda, S. 207f.)	AM	iK	B	L	Tr		vl	m:n	PA	d			tc	sd	+		+	-		+	
Direct Broker	(Trowbridge u. a., 2004, S. 155,215ff.)	AM	dK	B	L	kK			m:n		d			tc	sc							
Indirect Broker	(ebenda, S. 221)	AM	iK			kK			m:n		d			td	sd							
Message Broker	(ebenda, S. 240f.)	AM	iK	B	L	kK		vl	m:n	N	d	nbs	nbr	td	sd	+	-		-	+/-	+	
Message Broker	(Hohpe und Woolf, 2003, S. 322ff.)	AM	iK	B	L	Tr		vl	m:n	N	d	nbs	nbr	td	sd							
Broker	(Buschmann; Meunier u. a., 1996, S. 119ff.)	AM	iK	B	L	Tr			m:n	PA	d	nbs	nbr	td (indirekt) tc (direct)	sd	+	-		-	+/-	+	+
Broker Revisited	(Kircher; Voelter u. a., 2004, S. 581)	AM	iK	B	L	Tr			m:n		d	nbs	nbr	td	sd		-		-		+	
Broker	(Harrison und Avgeriou, 2007, S. 267f.)															+	n	+	n	+	+	

Gesamtklassifikation – Fortsetzung auf der nächsten Seite

Pattern	**Quelle**	Subsystem BIS	Integrationsoperation	Integrationsarchetyp	Komponente der Referenzarchitketur	Heterogenitätskonflikt	Zweck der Kommunikation	Verbindungsorientierung	Kardinalität d. Komm	Interaktionsmechanismus	Interaktionslogik	Send	Receive	Time/Synchronität	Space	Functionality	Reliability	Usability	Efficiency	Maintainability	Portability	Reusability
Pipes and Filter	(Hohpe und Woolf, 2003, S. 70ff., 322ff.)	AM	iK	B	tT	Tf		vl	1:1	N	d	nbs	nbr	td	sd				+/-	+		+
Pipes and Filter	(Buschmann; Meunier u. a., 1996, S. 67ff.)	AM	iK	B	tT	Tf		vl	1:1	N	d	nbs	nbr	td	sd				+/-	+	+	+
Pipes and Filter	(Trowbridge u. a., 2004, S. 292f.)	AM	iK	B	tT	Tf		vl	1:1	N	d	nbs	nbr	td	sd			-	+/-	-	+/-	+
Pipes and Filters	(Garlan und Shaw, 1993, S. 7f.)	AM	iK	B	tT	Tf		vl	1:1	N	d	nbs	nbr	td	sd			+	+/-	+		+
Pipes and Filters	(Meunier, 1995, S. 437)	AM	iK	B	tT	Tf		vl	1:1	N	d	nbs	nbr	tc	sd			+	+/-	+	+	+
Pipes and Filters	(Harrison und Avgeriou, 2007, S. 267f.)															-	-	-	+/-	+	+	
Observer	(Khomh und Guéhéneuc, 2008a)															+	-	+/-		+	-	+
Observer	(Trowbridge, 2003, S. 129f.)	EM	Se	B	C/D	Tr	Inf	vo	1:1	PA	d	bs?	br?	tc	sc			-	-	-	+	+
Publish and Subscribe	(Trowbridge u. a., 2004, S. 280f.)	AM	iK	B	L	Tr		vl	m:n	N	d	nbs		tc	sd	+			-	+/-	+	
Process Automator	(Lutz, 2000)	AM	iK	B	PE	V			m:n		z	nbs	nbr	td	sd		+			+	+	

Gesamtklassifikation – Fortsetzung auf der nächsten Seite

Pattern	**Quelle**	Subsystem BIS	Integrationsoperation	Integrationsarchetyp	Komponente der Referenzarchitketur	Heterogenitätskonflikt	Zweck der Kommunikation	Verbindungsorientierung	Kardinalität d. Komm	Interaktionsmechanismus	Interaktionslogik	Send	Receive	Time/Synchronität	Space	Functionality	Reliability	Usability	Efficiency	Maintainability	Portability	Reusability
Database Federation	(Andersson und Johnson, 2001)	AM	iK	B	MD	I	Inf	vo	1:n	PA	d			tc	sd	+	-			+		
Desktop Integration	(ebenda)	AM	dK	B	Spa	kK		vo	1:n	PA	z			tc	sc	+		-		-		
Data Transfer Object	(Trowbridge, 2003, S. 229)	EM	dK	B	L	Tr	Inf	vo	1:1	PA	d	bs	br	tc	sc			+	+/-	+/-		
Singleton	(ebenda, S. 257)	EM	dK	B	C/D	Tr	Inf	vo	1:n	PA	d	bs	br	tc	sc	+		-	-	+		
Singleton	(Khomh und Guéhéneuc, 2008a)															-	-	+		-	-	-
File Gateway	(Erl, 2009, S. 457)	AM	iK	B	C/D	kK	Inf	vl	1:1	N	d	nbs	nbr	td	sd				-			-
Process Centralizazion	(ebenda, S. 193)	AM	iK	B	PE	V	Inf		m:n	N	z	nbs	nbr	td	sd							
Protocol Bridging	(ebenda, S. 687)	AM	iK	B	tT	I		vl	m:n	N	d	nbs	nbr	td	sd		-		-			
Legacy Wrapper	(ebenda, S. 441)	AM	Ad	B	C/D	kK			1:n	N	d	nbs	nbr	td	sd				-			
Service Broker (COMP)	(ebenda, S. 707)	AM	Ad	B	tT	I		vl		N		nbs	nbr	td	sd							
Enterprise Service Bus (COMP)	(ebenda, S. 704)	AM	iK	B	L	kK		vl	m:n	N	d	nbs	nbr	td	sd							
Proxy Capability	(ebenda, S. 497)	AM	iK	B	L	Tr		vl	1:1	N	d	nbs	nbr	td	sd		-					
Service Facade	(ebenda, S. 333)	AM	iK	B	C/D	kK		vl	1:n	N	d			tc	sc				-			

Gesamtklassifikation – Fortsetzung auf der nächsten Seite

Pattern	**Quelle**	Subsystem BIS	Integrationsoperation	Integrationsarchetyp	Komponente der Referenzarchitketur	Heterogenitätskonflikt	Zweck der Kommunikation	Verbindungsorientierung	Kardinalität d. Komm	Interaktionsmechanismus	Interaktionslogik	Send	Receive	Time/Synchronität	Space	Functionality	Reliability	Usability	Efficiency	Maintainability	Portability	Reusability
UI Mediator	(ebenda, S. 366)	AM	iK	B	C/D	I		vl	m:n	N	d	nbs	nbr	tc	sd				-			+
Data Model Transformation	(ebenda, S. 671)	AM	iK	B	fT	I		vl	1:1	N	d	nbs	nbr	tc	sd				-			-
Data Format Transformation	(ebenda, S. 681)	AM	iK	B	tT	I		vl	1:1	N	d	nbs	nbr	tc	sd				-			
Asynchronous Queuing	(ebenda, S. 582)	AM	iK	B	L	Tr		vl	1:1	N	d	nbs	nbr	td	sd		+/-		-	+/-		
Intermediate Routing	(ebenda, S. 549)	AM	iK	B	L	Tr		vl	m:n	N	z	nbs	nbr	td	sd	-	-		-	+/-		

AM: Architekturmuster, EM: Entwurfsmuster, iK: indirekte Kopplung, dK: direkte Kopplung, Ad: Adaption, L: Lieferung, tT: technische Transformation, PE: Process Engine, C/D: Collection/Distribution, fT: fachliche Transformation, Spa: Steuerung von Portalarbeitsschritten, Tr: Transformation, kK: kombinierte Konflikte, V: Verhalten, I: Information, D: Dienste, Inf: Informationen

Tabelle C.1: Gesamtklassifikation

D Interviewleitfaden

In diesem Anhang finden Sie einen Abdruck des Interviewleitfadens aus Gebauer und Stefan.[1]

Leitfaden und Dokumentationsbogen Experteninterview

I. Kopfdaten	
Interviewpartner	
Gruppe	☐ Integrationsdienstleister ☐ Standardprodukthersteller ☐ Anwender ☐ Werkzeughersteller
Unternehmen	
Position	
Datum/Uhrzeit	
Ort	
Interviewer	Fred Stefan (FSt), Martin Gebauer (MG)
II. Kontext	
Unternehmensprofil	
Profil des Interview-partners	
Rahmenbedingungen	

[1] Vgl. (Gebauer und Stefan, 2011b, S. 239ff.).

III. Prolog	
Aufklärung zu Forschungszielen und Methodik	**Ziel:** Ziel der Arbeit soll sein, Aussagen über bestimmte Integrationskonzepte und -techniken zu treffen, sowie deren Gewichtung hinsichtlich bestimmter Kriterien zu untersuchen. Dabei sollen die folgende Fragenkreise beantwortet werden: • Wie erfolgen die Auswahl von Integrationslösungen und die Unterstützung der Entwicklung / Erstellung? • Anhand welcher Kriterien wird die Wirtschaftlichkeit von Integrationslösungen bewertet? • Welche Metriken existieren zur Evaluierung von Integrationsproblemen und –lösungen? Mehrwert für Befragten: • Ihre Praxiserfahrung fließt in unsere zukünftigen Forschungsbemühungen auf dem Gebiet der Integration ein. Sie haben die Möglichkeit damit Problemlösungen anzustoßen. • Ziel ist es bestehende Probleme aufzunehmen und in laufende Forschungsbemühungen zu integrieren. • In Form zweier Dissertationsprojekte werden die Ergebnisse dieses Experteninterviews aufgearbeitet und zu Lösungsvorschlägen weiterentwickelt. • Sie bekommen die Auswertung der Studie als kostenloses PDF-Dokument zugesandt. • Wir planen ein großes Forschungsvorhaben im Bereich der Integration für das wir ausgewiesene Experten suchen. Nach Abschluss der Befragung erläutern wir gern wie Sie sich beteiligen können. Befragte Unternehmen und Auswertung • Wir haben ca. 539 Unternehmen angeschrieben, die sich einer Firmendatenbank nach mit Systemintegration beschäftigen. • Davon werden wir nach Möglichkeit 25-30 analog zu diesem Gespräch hier befragen. • Im Anschluss daran wird eine Qualitative Inhaltsanalyse durchgeführt • Die Ergebnisse werden auf der GI Jahrestagung 2010 im Rahmen des Workshops Integration Engineering präsentiert
Aufklärung zum Interviewablauf und Verhaltensregeln	- Dauer des Interviews ca. 60 Minuten - Gliederung kurz vorstellen 1. Demografische Basisdaten 2. Verständnis zum Thema Integration 3. Integrationsproduke / -werkzeuge / -hilfsmittel 4. Integrationsmuster und Automatisierung 5. Gewichtigkeit/Invasivität 6. Integrationsobjekt Mainframe - Kein Antwortzwang - Einverständnis zum Mitschnitt des Interviews - Nur anonymisierte Verwendung - Haben Sie noch eine Frage, bevor es losgeht?

Hauptteil

Ein Modul + variabler Ergänzungsteil falls Produkthersteller oder reiner Dienstleister

1. Demografische Basisdaten	
1.1	Erläutern Sie bitte kurz Ihre Position im Unternehmen und Ihren beruflichen Werdegang.
1.2	Welche berufliche Erfahrung haben Sie im Themenfeld Systemintegration?
1.3	Seit wann beschäftigt sich Ihr Unternehmen mit Systemintegration?
1.4	Es lassen sich Integrationsdienstleister (DL), Anwender (AW), Standardproduktthersteller (SP), Werkzeug-/Hilfsmittelhersteller (WH) unterscheiden. Welcher der Gruppen würden Sie Ihr Unternehmen zuordnen?
	☐ Integrationsdienstleister ☐ Standardproduktthersteller ☐ Anwender ☐ Werkzeug-/Hilfsmittelhersteller Bemerkungen:
1.5	Ist Ihr Unternehmen auf eine oder mehrere Branche(n) spezialisiert? Erläutern Sie diese Spezialisierung bitte.
2. Verständnis zum Thema Integration	
2.1	Was verstehen Sie unter Integration?
2.2	Welche Ziele stehen bei der Auswahl von Integrationslösungen nach Ihrer Einschätzung im Vordergrund?
	Können Sie die von Ihnen genannten Ziele Gewichten?

2.3	Wie beurteilen Sie die Ausrichtung von Integrationslösungen auf die zukünftige Unternehmensentwicklung?
2.4	Erläutern Sie das methodische Vorgehen in Integrationsprojekten.
2.5	Nennen Sie die an einem Integrationsprojekt beteiligten Rollen.
2.6	Gibt es Anspruchsgruppen an eine Integrationslösung, die nicht oder nur marginal am Integrationsprojekt beteiligt sind? Welche sind dies?
	mögliche Unterscheidung Kunden- / Unternehmensseite Wartung und Betrieb Fachabteilung
3. Integrationsprodukte / -werkzeuge / -hilfsmittel	
	Es lassen sich Integrationswerkzeuge und Integrationshilfsmittel voneinander unterscheiden. Integrationswerkzeuge werden nur bei der Erstellung / Entwicklung der Integrationslösung eingesetzt. Integrationshilfsmittel hingegen gehen in die Integrationslösung ein. Middlewarelösungen, Enterprise Service Bus Produkte sind also als Integrationshilfsmittel einzustufen.
(Falls unter 1.4 Integrationsdienstleister)	
3.1	Welche Integrationswerkzeuge (z.B. Modellierungswerkzeuge, Entwicklungsumgebungen oder Konfiguratoren) setzen Sie ein?
3.2	Was sind die Alleinstellungsmerkmale dieser Werkzeuge?
3.3	Welche Funktionalitäten vermissen Sie bei den von Ihnen eingesetzten Werkzeugen?

3.4	Welche Integrationshilfsmittel (ESB, Middleware etc.) setzen Sie ein?
	Werden diese je nach Kundenanforderung ausgewählt oder haben Sie sich auf eine bestimmte Auswahl spezialisiert? Erläutern Sie diese bitte kurz und begründen Sie die Spezialisierung.
Falls unter 1.4 Werkzeug / Hilfsmittelhersteller (evtl. auch zusätzlich zu Dienstleister)	
3.5	Welches Integrationswerkzeug / Integrationshilfsmittel stellen Sie her / vertreiben Sie?
3.6	Welches Alleinstellungsmerkmal besitzt Ihr Produkt?.
	Ist Ihr Werkzeug besonders für eine spezielle Problemstellung z.B. Mainframeintegration geeignet?
3.7	Richtet sich Ihr Produkt an eine bestimmte Kundengruppe? Wenn ja, wodurch ist diese charakterisiert?
4. Integrationsmuster und Automatisierung	
4.1	Welche regelmäßig wiederkehrenden Tätigkeiten und Probleme in Integrationsprojekten sind Ihnen bekannt?
	Organisatorisch / technische Probleme/Tätigkeiten? Wie begegnen Sie diesen? Einsatz von Lösungsschemas, Best Practices? Welche Mechanismen der Wiederverwendung werden nach Ihrer Erfahrung eingesetzt? Wie ist die Weitergabe des erworbenen Lösungswissens organisiert?
4.2	Welche Integrationstopologien und welche Integrationsmuster kennen Sie?
	Integrationstopologien: Point-to-Point, Hub-and-Spoke , Bus-Topologien Integrationsmuster: Adapter, Beobachter oder Message Router Welche Ziele adressieren die von Ihnen genannten Muster?

4.3	(wenn unter 1.5 eine Spezialisierung auf bestimmte Branchen) Unter 1.5 gaben Sie an, dass sich Ihr Unternehmen auf die Branchen […] spezialisiert hat. Treten in diesen Branchen Domänenspezifische Objekte auf wie Systemklassen, Geschäftsobjekte und spezifische Prozesse auf? Erläutern Sie diese bitte.
4.4	Halten Sie diese Domänenspezifika für geeignet höhere Integrationsmuster zu bilden? Begründen Sie bitte Ihre Entscheidung.
4.5	Wie beurteilen Sie die Verfügbarkeit eines Werkzeuges welches folgende Funktionen bietet: • Erfassung IST und SOLL Unternehmensarchitektur • strukturierte Erfassung aller Integrationsinformationen • Hilfestellung bei der Auswahl der geeigneten Integrationsmethode (Entscheidungsunterstützung) • Erhöhung des Automatisierungsgrades durch Generierung
4.6	Welche Werkzeuge sind Ihnen bekannt, die diesen Funktionsumfang (evtl. auch teilweise) abdecken.
5. Gewichtigkeit / Invasivität	
5.1	Wie in der Chirurgie ist bei Integrationsprojekten häufig von Invasivität / Minimal-Invasivität die Rede. Wie würden Sie diesen Begriff in der Systemintegration definieren?
5.2	Welche Integrationskonzepte kennen Sie, die durch besonders geringe Invasivität gekennzeichnet sind?

5.3	In der Softwareentwicklung werden leichtgewichtige und schwergewichtige Vorgehensweisen voneinander unterschieden. Kenn sie eine ähnlich Unterscheidung im Bereich der Systemintegration? Beschreiben sie kurz die genannten Beispiele.
5.4	Worin sehen Sie den Nutzen von besonders geringer Invasivität und leichtgewichtigen Integrationskonzepten?
6. Integrationsobjekt Mainframe	
6.1	Wie häufig ist ihr Unternehmen mit Mainframeanwendungen konfrontiert?
→ Falls in noch keinem Projekt Mainframes vorgekommen sind – nur Frage 6.3 und 6.4 und Wegfall der Fragen 6.2, 6.5 und 6.6	
6.2	Welche Integrationsprobleme treten speziell bei Mainframeanwendungen auf, und wie gehen Sie damit um?
6.3	Legacy Anwendungen gelten oft als Altlasten, da für Integrationsaufgaben oft geeignete Schnittstellen fehlen. Wie ist Ihr Vorgehen in solchen Fällen?
6.4	In wie vielen Fällen war jedwede Veränderung der Legacy Anwendung ausgeschlossen? Wie wurde dies gegebenenfalls begründet und wie sind Sie damit umgegangen?
6.5	Setzen Sie spezielle Werkzeuge oder Hilfsmittel für die Integration von Mainframeanwendungen ein?
6.6	In wie vielen Fällen war eine Ablösung des Mainframes geplant? In welchen wurde diese tatsächlich umgesetzt?

IV. Epilog - Feedback zum Fragebogen und weiteres Vorgehen	
	Anmerkungen zu den Fragen?
	Fehlen Fragen?
	Möchten Sie uns weitere interessante Interviewpartner vorschlagen?
	Würden sie an einer folgenden empirischen Erhebung (Folgeinterview oder Fragebogen) teilnehmen?
	Falls Interesse besteht können wir Ihnen nun die Rahmenbedingungen eines Forschungsprojektes erläutern. Gern vereinbaren wir dazu – falls Interesse besteht – einen weiteren Termin.

E Ausgewählte Implementierungsartefakte

Quellcode E.1: OFBiz Komponente ERP-Showcase

```
<?xml version="1.0" encoding="UTF-8"?>
<ofbiz-component name="ERPshowcase"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:noNamespaceSchemaLocation="http://ofbiz.apache.org/dtds/ofbiz-component.xsd">
    <!-- define resource loaders; most common is to use the component resource loader -->
    <resource-loader name="main" type="component"/>

    <!-- place the config directory on the classpath to access configuration files -->
    <classpath type="dir" location="config"/>
    <classpath type="dir" location="dtd"/>

    <!-- load single or multiple external libraries -->
    <classpath type="jar" location="build/lib/*"/>
    <classpath type="jar" location="lib/*"/>

    <!-- entity resources: model(s), eca(s), group, and data definitions -->
    <entity-resource type="model" reader-name="main" loader="main" location="entitydef/
        entitymodel.xml"/>
    <!-- <entity-resource type="eca" reader-name="main" loader="main" location="entitydef
        /eecas.xml"/> -->
    <entity-resource type="data" reader-name="seed" loader="main" location="data/
        ERPShowcaseTypeData.xml"/>
    <entity-resource type="data" reader-name="seed" loader="main" location="data/
        ERPShowcaseSecurityData.xml"/>
    <entity-resource type="data" reader-name="demo" loader="main" location="data/
        ERPShowcaseDemoData.xml"/>

    <!-- service resources: model(s), eca(s) and group definitions -->
    <service-resource type="model" loader="main" location="servicedef/services.xml"/>
    <!--
    <service-resource type="eca" loader="main" location="servicedef/secas.xml"/>
    <service-resource type="group" loader="main" location="servicedef/groups.xml"/>
    -->

    <test-suite loader="main" location="testdef/ERPShowcaseTests.xml"/>

    <!-- web applications; will be mounted when using the embedded container -->
    <webapp name="ERPshowcase"
        title="ERPShowcase"
        server="default-server"
        location="webapp/ERPshowcase"
        base-permission="OFBTOOLS,ERPSHOWCASE"
        mount-point="/ERPshowcase"/>
</ofbiz-component>
```

Quellcode E.2: Realisierung der Serviceschnittstelle importERPProduct

```
<?xml version="1.0" encoding="UTF-8"?>
<services xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:noNamespaceSchemaLocation="http://ofbiz.apache.org/dtds/services.xsd">
    <description>ERPShowcase Services</description>
    <vendor></vendor>
    <version>1.0</version>

   <service name="importERPProduct" engine="java" export="true" location="de.unileipzig.
       tec.IntigrationServices" invoke="importERPProduct" auth="true">
        <description>Import a Product from ERP</description>
        <attribute name="productId" type="java.lang.String" mode="INOUT" optional="false"
            />    <!-- ERP Bspdaten - Artikel -->
        <attribute name="productName" type="java.lang.String" mode="IN" optional="false"/
            >    <!-- ERP Bspdaten - Bezeichnung -->
```

```
        <attribute name="description" type="java.lang.String" mode="IN" optional="true"/>
                <!-- ERP Bspdaten - Bezeichnung (Zeile2) -->
        <attribute name="longDescription" type="java.lang.String" mode="IN" optional="
            true"/>  <!-- ERP Bspdaten - WBZ -->
        <attribute name="categoryId" type="java.lang.String" mode="IN" optional="true"/>
    </service>

    <service name="getOrderIds" engine="java" export="true" location="de.unileipzig.tec.
        IntigrationServices" invoke="getOrderIds" auth="true">
        <description>Returns the orderid's of orders created in the given time interval
            .</description>
        <attribute name="minDate" type="java.sql.Timestamp" mode="IN" optional="true"/>
        <attribute name="maxDate" type="java.sql.Timestamp" mode="IN" optional="true"/>
        <attribute name="ids" type="java.util.ArrayList" mode="OUT"/>
    </service>

    <service name="getOrderItems" engine="java" export="true" location="de.unileipzig.tec
        .IntigrationServices" invoke="getOrderItems" auth="true">
        <description>Returns the order for the given orderId</description>
        <attribute name="orderId" type="java.lang.String" mode="IN" optional="false"/>
        <attribute name="orderItems" type="java.util.ArrayList" mode="OUT" />
    </service>

        <service name="getOrders" engine="java" export="true" location="de.unileipzig.tec
            .IntigrationServices" invoke="getOrders" auth="true">
        <description>Returns the orders for the given time interval.</description>
        <attribute name="minDate" type="java.sql.Timestamp" mode="IN" optional="true"/>
        <attribute name="maxDate" type="java.sql.Timestamp" mode="IN" optional="true"/>
        <attribute name="orderItems" type="java.util.ArrayList" mode="OUT" />
    </service>

</services>
```

Quellcode E.3: Relaisierung der Zugriffspunkte durch Java-Klassen

```
package de.unileipzig.tec;

import java.sql.Timestamp;
import java.util.ArrayList;
import java.util.Collections;
import java.util.Iterator;
import java.util.List;
import java.util.Map;
import java.util.Set;

import javolution.util.FastMap;
import javolution.util.FastSet;

import org.ofbiz.base.util.Debug;
import org.ofbiz.base.util.UtilDateTime;
import org.ofbiz.base.util.UtilMisc;
import org.ofbiz.entity.Delegator;
import org.ofbiz.entity.GenericValue;
import org.ofbiz.service.DispatchContext;
import org.ofbiz.service.ServiceUtil;


public class IntigrationServices {

    public static final String module = IntigrationServices.class.getName();
    public static final String resource = "ProductErrorUiLabels";

    public static Map<String, Object> importERPProduct(DispatchContext dctx, Map<String,
        ? extends Object> context) {
        final String STD_CATEGORY_ID = "ShowcaseCategory1";
        final String SEARCH_CATEGORY_ID = "ShowcaseSearch";

        Delegator delegator = dctx.getDelegator();
        Timestamp nowTimestamp = UtilDateTime.nowTimestamp();
```

```
Map<String, Object> successResult = ServiceUtil.returnSuccess();

String productId = (String) context.get("productId");
String productName = (String) context.get("productName");
String description = (String) context.get("description");
String longDescription = (String) context.get("longDescription");
String categoryId = (String) context.get("categoryId");

if (description == null)
        description = "";
if (longDescription == null)
        longDescription = "";
if (categoryId == null)
        categoryId = STD_CATEGORY_ID;

GenericValue product = null;
GenericValue price = null;
GenericValue category = null;
GenericValue categorySearch = null;

try {
    product = delegator.findByPrimaryKey("Product", UtilMisc.toMap("productId",
        productId));

    if(product == null){
        //Create new product
        product = delegator.makeValue("Product");
        product.set("productId", productId);
        product.set("internalName", productId);
        product.set("productName", productName);
        product.set("description", description);
        product.set("longDescription", longDescription);
        product.set("lastModifiedDate", nowTimestamp);

        product.set("createdDate", nowTimestamp);
        product.set("productTypeId", "GOOD");
        product.set("isVirtual", "N");
        product.set("isVariant", "N");
        product.set("returnable", "N");
        product.set("taxable", "N");
        product.set("chargeShipping", "N");
        product.set("autoCreateKeywords", "Y");
        product.set("includeInPromotions", "N");
        product.create();

        //set default price needed to complete a order
        price = delegator.makeValue("ProductPrice");
        price.set("productId", productId);
        price.set("productPriceTypeId", "DEFAULT_PRICE");
        price.set("productPricePurposeId", "PURCHASE");
        price.set("productStoreGroupId", "_NA_");
        price.set("fromDate", nowTimestamp);
        price.set("currencyUomId", "EUR");
        price.set("price", new java.math.BigDecimal(1));

        price.create();

        //add product to category
        category = delegator.makeValue("ProductCategoryMember");
        category.set("productCategoryId", categoryId);
        category.set("productId", productId);
        category.set("fromDate", nowTimestamp);
        category.create();

        //add product to search
        categorySearch = delegator.makeValue("ProductCategoryMember");
        categorySearch.set("productCategoryId", SEARCH_CATEGORY_ID);
        categorySearch.set("productId", productId);
        categorySearch.set("fromDate", nowTimestamp);
        categorySearch.create();
```

```
        }else{
            //Update product
            //TODO don't override if no change occurs
            product.set("productName", productName);
            product.set("description", description);
            product.set("longDescription", longDescription);
            product.set("lastModifiedDate", nowTimestamp);
            product.store();
        }

        successResult.put("productId", productId);

    } catch (Exception e) {
        String errMsg = "Error importing a product: " + e.toString();
        Debug.logError(e, errMsg, module);
        return ServiceUtil.returnError(errMsg);
    }

    return successResult;
}

public static Map<String, Object> getOrderIds(DispatchContext dctx, Map<String, ?
    extends Object> context) {
    Delegator delegator = dctx.getDelegator();
    //Timestamp nowTimestamp = UtilDateTime.nowTimestamp();

    Timestamp minDate = (Timestamp) context.get("minDate"); //optional
    Timestamp maxDate = (Timestamp) context.get("maxDate"); //optional

    if (minDate == null)
            minDate = UtilDateTime.getTimestamp(0);

    if (maxDate == null)
            maxDate = UtilDateTime.nowTimestamp();


    Map<String, Object> successResult = ServiceUtil.returnSuccess();
    List <GenericValue> orders = null;
    GenericValue order = null;
    List <String> ids = null;

    try{
            Set<String> fieldsToSelect = FastSet.newInstance();
            fieldsToSelect.add("createdStamp");
            fieldsToSelect.add("orderId");

            orders = delegator.findList("OrderItem", null, fieldsToSelect, null, null
                , true);

            Iterator<GenericValue> it = orders.iterator();
            while (it.hasNext()){
                    order = it.next();
                    if(((Timestamp)order.get("createdStamp")).after(minDate) && ((
                        Timestamp)order.get("createdStamp")).before(maxDate)){
                            if(ids == null){
                            ids = new ArrayList<String>();
                    }
                            ids.add((String) order.get("orderId"));
                    }
            }
            if(ids == null)
                    successResult.put("ids", Collections.emptyList());
            else
                    successResult.put("ids", ids);
        } catch (Exception e) {
            String errMsg = "Error exporting orders: " + e.toString();
            Debug.logError(e, errMsg, module);
            return ServiceUtil.returnError(errMsg);
        }
```

```
return successResult;
}

public static Map<String, Object> getOrderItems(DispatchContext dctx, Map<String, ?
    extends Object> context) {
    Delegator delegator = dctx.getDelegator();

    String orderId = (String) context.get("orderId");

    Map<String, Object> successResult = ServiceUtil.returnSuccess();
    List<GenericValue> searchResult = null;
    List<Map<String, Object>> orderItems = new ArrayList<Map<String, Object>>();

    try{
            searchResult = delegator.findByAnd("OrderItem", UtilMisc.toMap("orderId",
                 orderId));

            Iterator<GenericValue> it = searchResult.iterator();
            GenericValue value;
            Map<String, Object> item;
            while (it.hasNext()){
                    value = it.next();
                    item = FastMap.newInstance();
                    item.put("orderId", (String)value.get("orderId"));
                    item.put("productId", (String)value.get("productId"));
                    item.put("quantity", ((java.math.BigDecimal)value.get("quantity")
                        ).floatValue());
                    orderItems.add(item);
            }

            //Set result
            successResult.put("orderItems", orderItems);


        } catch (Exception e) {
            String errMsg = "Error exporting orders: " + e.toString();
            Debug.logError(e, errMsg, module);
            return ServiceUtil.returnError(errMsg);
        }

return successResult;
}

public static Map<String, Object> getOrders(DispatchContext dctx, Map<String, ?
    extends Object> context) {
    Delegator delegator = dctx.getDelegator();
    //Timestamp nowTimestamp = UtilDateTime.nowTimestamp();

    Timestamp minDate = (Timestamp) context.get("minDate"); //optional
    Timestamp maxDate = (Timestamp) context.get("maxDate"); //optional

    if (minDate == null)
            minDate = UtilDateTime.getTimestamp(0);

    if (maxDate == null)
            maxDate = UtilDateTime.nowTimestamp();

    Map<String, Object> successResult = ServiceUtil.returnSuccess();
    List <GenericValue> orders = null;
    GenericValue order = null;
    List<GenericValue> searchResult = null;
    List<Map<String, Object>> orderItems = new ArrayList<Map<String, Object>>();

    try{
            Set<String> fieldsToSelect = FastSet.newInstance();
            fieldsToSelect.add("createdStamp");
            fieldsToSelect.add("orderId");
```

```
                orders = delegator.findList("OrderItem", null, fieldsToSelect, null, null
                    , true);

                Iterator<GenericValue> oit = orders.iterator();
                while (oit.hasNext()){
                        order = oit.next();
                        if(((Timestamp)order.get("createdStamp")).after(minDate) && ((
                            Timestamp)order.get("createdStamp")).before(maxDate)){

                        searchResult = delegator.findByAnd("OrderItem", UtilMisc.toMap("
                            orderId", (String) order.get("orderId")));

                        Iterator<GenericValue> oiit = searchResult.iterator();
                        GenericValue value;
                        Map<String, Object> item;
                        while (oiit.hasNext()){
                                value = oiit.next();
                                item = FastMap.newInstance();
                                item.put("orderId", (String)value.get("orderId"));
                                item.put("productId", (String)value.get("productId"));
                                item.put("quantity", ((java.math.BigDecimal)value.get("
                                    quantity")).floatValue());
                                orderItems.add(item);
                        }
                        }
                }

                //Set result
                successResult.put("orderItems", orderItems);

            } catch (Exception e) {
                String errMsg = "Error exporting Orders: " + e.toString();
                Debug.logError(e, errMsg, module);
                return ServiceUtil.returnError(errMsg);
            }

            return successResult;
    }

}
```

Quellcode E.4: Quellcode des JBossESB Moduls für den Produktimport

```
<?xml version="1.0"?>
<jbossesb parameterReloadSecs="5"
 xmlns="http://anonsvn.labs.jboss.com/labs/jbossesb/trunk/product/etc/schemas/xml/
     jbossesb-1.0.1.xsd"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://anonsvn
     .labs.jboss.com/labs/jbossesb/trunk/product/etc/schemas/xml/jbossesb-1.0.1.xsd http:
     //anonsvn.jboss.org/repos/labs/labs/jbossesb/trunk/product/etc/schemas/xml/jbossesb
     -1.0.1.xsd">
 <providers>
  <schedule-provider name="ERPWSConsumeSchedule">
   <simple-schedule frequency="3600" frequencyUnits="seconds" scheduleid="3600-sec-
       Schedule"/>
  </schedule-provider>
 </providers>

 <services>
   <service category="MyServiceCategory"
   description="Imports Products from ERP into OfBiz" name="ProductImportESB">

    <listeners>
      <scheduled-listener
      event-processor="de.unileipzig.sit.efie.szenario.ScheduledActionMsgComposer"
      name="ERPWSConsumeScheduledListener" scheduleidref="3600-sec-Schedule"/>
    </listeners>

    <actions mep="RequestResponse">
```

```
      <action
        class="de.unileipzig.sit.efie.szenario.ERPProductExportParameterMapping" name="
            ERPProductExportParameterMapping">
        <property name="ERPProductIdFrom" value="0000000000000"/>
        <property name="ERPProductIdTo" value="ZZZZZZZZZZZZZ"/>
        <property name="ERPUsername"/>
        <property name="ERPPassword"/>
      </action>

      <action class="org.jboss.soa.esb.actions.soap.SOAPClient" name="ERPProductExport">
        <property name="wsdl" value="http://192.168.1.21:8080/axis/services/
            ERPProductWebService?wsdl"/>
        <property name="operation" value="getItemsByIdRange"/>
        <property name="SOAPAction" value="getItemsByIdRange"/>
        <property name="get-payload-location" value="ERPRequestParameterMap"/>
        <property name="set-payload-location" value="ERPResponseParameterMap"/>
      </action>

      <action class="org.jboss.soa.esb.smooks.SmooksAction" name="transformToMap">
        <property name="smooksConfig" value="smooks-res.xml"/>
        <property name="get-payload-location" value="ERPResponseParameterMap"/>
        <property name="set-payload-location" value="ProductList"/>
        <property name="resultType" value="JAVA"/>
      </action>

      <action class="de.unileipzig.sit.efie.szenario.OfbizProductImport" name="
          OfbizProductImport">
        <property name="OfbizURI" value="http://192.168.1.12:8080/webtools/control/
            SOAPService"/>
        <property name="OfbizSoapAction" value="importERPProduct"/>
        <property name="OfbizUsername" value="admin"/>
        <property name="OfbizPassword" value="ofbiz"/>
      </action>

    </actions>
  </service>

 </services>
</jbossesb>
```

Quellcode E.5: Java Realisierung des angepassten SOAP-Request

```
package de.unileipzig.sit.efie.szenario;

import java.io.IOException;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.Map;
import java.util.Properties;

import org.apache.commons.httpclient.HttpClient;
import org.apache.commons.httpclient.HttpStatus;
import org.apache.commons.httpclient.methods.PostMethod;
import org.apache.commons.httpclient.methods.StringRequestEntity;
import org.apache.log4j.Logger;
import org.jboss.soa.esb.ConfigurationException;
import org.jboss.soa.esb.http.HttpClientFactory;
import org.jboss.soa.esb.actions.AbstractActionLifecycle;
import org.jboss.soa.esb.actions.ActionProcessingException;
import org.jboss.soa.esb.helpers.ConfigTree;
import org.jboss.soa.esb.message.Message;

public class OfbizProductImport extends AbstractActionLifecycle {

        private Logger logger = Logger.getLogger(OfbizProductImport.class);
        protected ConfigTree _config;
        private String url;
        private String SOAPAction;
        private String ofbizUsername;
        private String ofbizPassword;
```

```
public OfbizProductImport(ConfigTree config)
{
        _config = config;
        try {
                url = config.getRequiredAttribute("OfbizURI");
                SOAPAction = config.getRequiredAttribute("OfbizSoapAction");
                ofbizUsername = config.getRequiredAttribute("OfbizUsername");
                ofbizPassword = config.getRequiredAttribute("OfbizPassword");
        } catch (ConfigurationException e) {
                System.out.println("OfbizURI and OfbizSoapAction are required
                    Attributes");
        }
}
public Message process(Message message) throws Exception
{

        HashMap items = (HashMap) message.getBody().get("ProductList");
        ArrayList products = (ArrayList) items.get("items");
        for (Object o : products)
        {
                Map product = (Map) o;
                if (product.containsKey("itemId"))
                {
                        if (product.get("valid").equals("true"))
                        {
                                invokeEndpoint(product.get("itemId").toString(),
                                    product.get("title").toString(), product.get(
                                    "description").toString());
                                System.out.println("
                                    ----------------------------------------");
                                System.out.println("Updated "+product.get("title"
                                    ).toString()+"("+product.get("itemId").
                                    toString()+")");
                                System.out.println("
                                    ----------------------------------------");
                        }
                }

        }

        return message;
}
protected String invokeEndpoint(String itemId, String title, String description)
{
PostMethod post = new PostMethod(url);

post.setRequestHeader("Content-Type", "text/xml;charset=UTF-8");
post.setRequestHeader("SOAPAction", SOAPAction);
String request = "<soapenv:Envelope xmlns:soapenv=\"http://schemas.xmlsoap.org/
    soap/envelope/\" xmlns:ser=\"http://ofbiz.apache.org/service/\"><soapenv:
    Header/><soapenv:Body>"+
        "<ser:"+SOAPAction+"><map-Map><ser:map-Entry>" +
        "<ser:map-Key><ser:std-String value=\"login.username\"/></ser:map-Key><
            ser:map-Value><ser:std-String value=\""+ofbizUsername+"\"/></ser:map-
            Value></ser:map-Entry>"+
        "<ser:map-Entry><ser:map-Key><ser:std-String value=\"productName\"/></ser
            :map-Key><ser:map-Value><ser:std-String value=\""+title+"\"/></ser:
            map-Value></ser:map-Entry>"+
        "<ser:map-Entry><ser:map-Key><ser:std-String value=\"productId\"/></ser:
            map-Key><ser:map-Value><ser:std-String value=\""+itemId+"\"/></ser:
            map-Value></ser:map-Entry>"+
        "<ser:map-Entry><ser:map-Key><ser:std-String value=\"description\"/></ser
            :map-Key><ser:map-Value><ser:std-String value=\""+description+"\"/></
            ser:map-Value></ser:map-Entry>"+
        "<ser:map-Entry><ser:map-Key><ser:std-String value=\"login.password\"/></
            ser:map-Key><ser:map-Value><ser:std-String value=\""+ofbizPassword+"
            \"/></ser:map-Value></ser:map-Entry>"+
        "</map-Map></ser:"+SOAPAction+"></soapenv:Body></soapenv:Envelope>";
post.setRequestEntity(new StringRequestEntity(request));
```

```
        HttpClient httpclient;
        try {
                Properties httpProps = new Properties();
                //139.18.2.219:8203
                httpProps.setProperty(HttpClientFactory.TARGET_HOST_URL, url);
                httpclient = HttpClientFactory.createHttpClient(httpProps);
          int result = httpclient.executeMethod(post);
            if (result != HttpStatus.SC_OK) {
                // TODO: We need to do more here!!
                logger.warn("Received status code '" + result
                        + "' on HTTP SOAP (POST) request to '"
                        + "'.");
            }
            return post.getResponseBodyAsString();
        } catch (IOException e) {
            try {
                                throw new ActionProcessingException(
                                        "Failed to invoke SOAP Endpoint: '"
                                                + " ' - '" + "'.", e);
                        } catch (ActionProcessingException e1) {
                                e1.printStackTrace();
                        }
        } catch (ConfigurationException e) {
                        e.printStackTrace();
                } finally {
            post.releaseConnection();
        }
                return "";


        }
}
```

F Details der Archimate Formalisierung und der Analysemuster

Dieser Anhang legt die Details der in Kapitel 8 vorgestellten Formalisierung von Archimate Modellen dar.

F.1 Formalisierung

Zunächst werden in Tabelle F.1 kurz die Symbole nach Šaša und Krisper eingeführt. Im Anschluss daran werden diese verwendet und eine vollständige Formalisierung von Archimate entwickelt.

Symbol	Beschreibung
PIA	alle Elemente einer Unternehmensarchitektur
BP	Menge aller Geschäftsprozesse
BF	Menge aller Business-Function
BI	Menge aller Business-Interaction
BE	Menge aller Business-Event
BBE, $BBE = BP \cup BF \cup BI \cup BE$	Menge aller Business-Behavior Elemente
BO	Menge aller Business-Object
DO	Menge aller Data-Object
R	Menge aller Repräsentationen
BR	Menge aller Business-Roles
AS	Menge aller Application-Service
AC	Menge aller Application-Component
$AnBP$,$AnBP : AnBP \subseteq BP$	Menge der für die Analyse ausgewählten Geschäftsprozesse
$AnBO$, $AnBO : AnBO \subseteq BO$	Menge der für die Analyse ausgewählten Business-Object
X'	transitive Relation zu einer Relation X
$(a, b) \in Realization$	a steht in Relation zu b durch eine *Realization* relation
$(a, b) \in Used\ by$	a steht in Relation zu b durch eine $Usedby$ relation
$(a, b) \in Assignment$	a steht in Relation zu b durch eine *Assignment* relation
$(a, b) \in Access$	a steht in Relation zu b durch eine *Access* relation
$(a, b) \in Composition$	a steht in Relation zu b durch eine *Composition* relation
$(a, b) \in Aggregation$	a steht in Relation zu b durch eine *Aggregation* relation

Tabelle F.1: Teilweise Formalisierung von Archimate
Quelle: (Šaša und Krisper, 2011)

Darüber hinaus beschreiben die folgenden Definitionen weitere Formalisierungen, welche von Sasa und Krisper eingeführt wurden. Diese werden wiedergegeben und im Detail analysiert, inwieweit diese – als Muster bezeichneten – Formalismen für eine Integrationsanalyse verwendet werden können. Wie bereits angedeutet, wurde diese Vorgehensweise gewählt um beide Arbeiten sinvoll miteinander verbinden zu können.

$$\begin{aligned} SP(bp) &= \{sp | sp \in BBE : (bp, sp) \in Composition' \vee (bp, sp) \in Aggregation'\} \\ SP_{AnBP} &= \{x | \exists y \in AnBP \wedge x \in SP(y)\} \\ SO(bo) &= \{so | so \in BO : (bo, so) \in Composition' \vee (bo, so) \in Aggregation'\} \\ SO_{AnBO} &= \{x | \exists y \in AnBO \wedge x \in SO(y)\} \end{aligned} \tag{F.1}$$

Ein Archimatemodell lässt sich grundsätzlich als gerichteter Graph mit Mehrfachkanten auffassen:

$$PIA = (N, X) \tag{F.2}$$

Die Menge der Knoten besteht aus 3 Partitionen:

$$N = N_B \cup N_A \cup N_T \; und \; N_A \cap N_B \cap N_T = \emptyset \tag{F.3}$$

N_B ist die Menge der Knoten des Businesslayers, N_A ist die Menge der Knoten des Applicationlayers und N_T ist die Menge der Knoten des Technology Layers. Archimate partitioniert seine Elmente in Struktur-, Verhaltens- und Informationselemente. Die Verhaltenselemente wurden durch Šaša und Krisper mit BBE bereits abgebildet. Darüber hinaus werden diese 3 Submengen für alle Layer definiert:

$$N_B = BBE \cup BSE \cup BIE \tag{F.4}$$
$$N_A = ABE \cup ASE \cup AIE \tag{F.5}$$
$$N_T = TBE \cup TSE \cup TIE \tag{F.6}$$
$$\tag{F.7}$$

Dabei steht $*BE$ für Behavior-Element, $*SE$ für Structural-Element, $*IE$ für Information-Element. Die Formalisierungen aus Tabelle F.1 bezüglich der Knoten ordnen sich wie folgt ein:

$$N_B = BBE \cup B* \tag{F.8}$$
$$BSE = BO \cup BR \cup BS* \tag{F.9}$$
$$BIE = R \cup BI* \tag{F.10}$$
$$ABE = AS \cup AB* \tag{F.11}$$
$$ASE = DO \cup AC \cup AS* \tag{F.12}$$
$$AIE = \emptyset \tag{F.13}$$
$$TBE = \emptyset \tag{F.14}$$
$$TSE = \emptyset \tag{F.15}$$
$$TIE = \emptyset \tag{F.16}$$

Die mit * gekennzeichneten Mengen dienen als Platzhalter, da für die adressierten Analysezwecke keine vollständige Formalisierung eines Archimatemodells benötigt wird. Diese Mengen subsummieren alle restlichen Elemente, die sich keiner der anderen Mengen zuordnen lassen und aus genanntem Grund nicht weiter spezifiziert werden.

Zusätzlich wird eine Funktion definiert, welche den den Typ eines Elementes für ein beliebiges Elemten $pia \in PIA$ zurückgibt. Damit lässt sich die Zugehörigkeit beliebiger Elemente zu den Untermengen von PIA prüfen, in dem der Rückgabewert von $Type(pia)$ ausgewertet wird.

$$\begin{aligned} Type(pia) = m :(&pia \in N \wedge m \in \{BusinessFunction, \ldots, ApplicationComponent, \ldots, Node\}) \\ &\vee (pia \in X \wedge m \in \{Assigment, Composition, \ldots\}) \end{aligned} \tag{F.17}$$

Über die bereits definierten Mengen in Tabelle F.1 hinaus werden weitere Mengen benötigt unter anderem die Menge aller *ApplikationInterface AI*:

$$AI : AI \subseteq ASE \wedge n \in AI \ g.d.w. \ type(n) = ApplicationInterface \tag{F.18}$$

Auch die Menge der Relationen ist in verschiedene Untermengen eingeteilt. Die Partitionierung der Relationen in verschiedene Submengen wird bereits im Hinblick auf eine spätere Implementierung vorgenommen. Für die Bestimmung – ob und welche Knoten mit einander verbunden sind – spielen die Relationsmengen eine wesentliche Rolle. Durch die Partitionierung können die Mengen – auf deren Basis die Analysen stattfinden – reduziert werden, was die Berechnungen weniger aufwändig macht.

$$X = X_B \cup X_{BA} \cup X_{AB} \cup X_A \cup X_{AT} \cup X_{TA} \cup X_T \tag{F.19}$$

X_B ist die Menge aller Kanten für die gilt, dass die durch diese verbundenen Knoten ausschließlich in N_B liegen:

$$X_B = \{a, b\} : a \in N_B \wedge b \in N_B. \tag{F.20}$$

Analog zu dieser Definition sind die weiteren Relationsmengen wie folgt definiert:

$$X_{AB} = \{a, b\} : a \in N_B \wedge b \in N_A \tag{F.21}$$
$$X_{BA} = \{a, b\} : a \in N_A \wedge b \in N_B \tag{F.22}$$
$$X_A = \{a, b\} : a, b \in N_A \tag{F.23}$$
$$X_{AT} = \{a, b\} : a \in N_A \wedge b \in N_T \tag{F.24}$$
$$X_{TA} = \{a, b\} : a \in N_T \wedge b \in N_A \tag{F.25}$$
$$X_T = \{a, b\} : a, b \in N_T \tag{F.26}$$

Damit wurde über die Definitionen von Šaša und Krisper hinausgegangen, welche die Relationsmenge nicht partitioniert haben.
Darüber hinaus zeigt die bei Šaša und Krisper demonstrierte Implementierung in den kommerziellen Werkzeugen Bizz Design Architect und ARIS Archimate Modeler[1], dass eine weite Übertragbarkeit der hier vorgestellten Ergebnisse existiert. In den Ansätzen wird die Einsetzbarkeit auf Basis von Open Source demonstriert.

F.2 Analysemuster

In der Arbeit von Šaša und Krisper wurden Analysemuster für die Analyse des Business-Process-Support vorgestellt[2]. Diese Muster dienen als Basis für die angestrebte Analyse mit der Zielrichtung Integration. In diesem Abschnitt werden die bekannten Pattern analysiert und auf ihre Eignung überprüft. In den dann folgenden Abschnitten werden die Erweiterungen und neue Analysemuster vorgestellt.

Fully automated business process pattern:

$$\begin{aligned} FABP_{AnBP} = \{a | a \in AnBP \wedge ((\exists b \in AC : (b, a) \in Assignment) \\ \vee (\forall sp \in SP(a) : \exists ac \in AC \\ \wedge (ac, sp) \in Assignment)\} \end{aligned} \tag{F.27}$$

[1] Vgl. (Šaša und Krisper, 2011, S. 21).
[2] Vgl. (ebenda).

$FABP$ enthält alle analysierten Prozesse ($\in AnBP$), welche direkt über eine Assignment Relation mit einer Application Component in Beziehung stehen oder bei denen alle Unterelemente eine solche Assigment Relation zu einer Application Component aufweisen. Dabei wird unterstellt, dass nur eine Assignment Relation für eine derartige Zuordnung relevant ist. Alle anderen Relationen sind bestandteil anderer Muster – z.b. die Used by Relation – und repräsentieren keine vollständige Automatisierung. Da bei vollautomatisierten Prozessen auch die notwendige Weitergabe von Informationen und das Anstoßen nachfolgender Prozesse – idealer Weise – vollautomatisch erfolgen sollte, ist dieses Muster relevant für die angestrebte Integrationsanalyse.

Das Pattern **Partially automated business process pattern** ($PABP$), ist in der Lage teilautomatisierte Prozesse auf zu zeigen und ist wie folgt definiert:

$$\begin{aligned} PABP_{AnBP} = \{a | a \in AnBP \setminus FABP_{AnBP} \\ \wedge ((\exists b \in AS : (b, a) \in UsedBy \\ \vee (\exists sp1 \in SP(a), \exists ac \in AC : (ac, sp1) \in Assignment) \\ \vee (\exists sp2 \in SP(a), \exists as \in AS : (as, sp2) \in UsedBy))\} \end{aligned} \tag{F.28}$$

Die Ausgangsbasis sind alle zu analysierenden Prozesse abzüglich der Vollautomatisierten im Sinne von $FABP$. Als Merkmal wird die Zuordnung des Prozesses zu einer Application Component über eine Used by Relation verwendet. Ist nicht der Prozess sondern seine Subelemente mit dem Application Layer verbunden so drückt sich dies über eine partielle Relation (nicht alle Subelemente sind zugeordnet) von Subelementen eines Prozesses zu Application Component über eine Assignment Relation oder zu Application Service über eine Used by Relation aus. Da partiell automatisierte Prozesse ebenfalls zu Integrationsanforderungen führen können, stellt dieses Muster ebenfalls eine gute Ausgangsbasis für die beabsichtigte Integrationsanalyse dar.

Unautomated business process pattern:

$$UABP_{AnBP} = \{a | a \in AnPB \setminus (FABP_{AnBP} \cup PABP_{AnBP})\} \tag{F.29}$$

Das erkennen von nicht automatisierten Prozessen kann dazu verwendet werden, Prozesse zu finden, welche Gegenstand von Automatisierungsbemühungen werden können. Diese führen dann zwar mittelbar zu potentiellen Integrationserfordernissen, sind aber nicht direkt relevant für eine Integrationsanalyse.

Fully manually supported business process pattern:

$$\begin{aligned} FMBP_{AnBP} = \{a | a \in AnBP \\ \wedge (((\exists r \in BR : (r, a) \in Assignment) \\ \wedge (\nexists as \in AS : (as, a) \in UsedBy)) \\ \vee (\forall sp \in SP(a) : (\exists r1 \in BR : (r1, sp) \in Assignment) \\ \wedge (\nexists as1 \in AS : (as1, sp) \in UsedBy)))\} \end{aligned} \tag{F.30}$$

Ein vollständig manuell realisierter Geschäftsprozess ist analog zu einem nicht automatisierten Prozess –zumal eine Schnittmenge beider Muster exisitert – zu sehen, er führt nicht zu direkten Integrationserfordernissen und kann deswegen aus den Betrachtungen ausgeklammert werden.

Partially manually supported business process pattern

$$\begin{aligned} PMBP_{AnBP} = \{a | a \in AnBP \setminus FMBP_{AnBP} \\ \wedge (((\exists r \in BR : (r, a) \in Assignment) \\ \wedge (\exists as \in AS : (as, a) \in UsedBy)) \\ \wedge (\exists sp1 \in SP(a) : ((\exists br1 \in BR : (br1, sp1) \in Assignment) \\ \wedge (\exists as1 \in AS : (as, sp1) \in UsedBy))) \\ \vee (\exists sp2, sp3 \in SP(a) : ((sp2 \neq sp3) \\ \wedge (\nexists br2 \in BR : (br2, sp2) \in Assignment) \\ \wedge (\exists br3 \in BR : (br3, sp3) \in Assignment)))) \end{aligned} \tag{F.31}$$

Ein partiell manuell unterstützter Prozess ist nur dann für eine Integrationsanalyse von Interesse, wenn der Teile des Prozesses – welcher nicht manuell unterstützt wird – automtisiert ist. Ist dies der Fall, so treffen auf einen derartigen Prozess andere Muster zu. Insbesondere das Muster Partially automated business process. Demnach kann das Muster Partially manually supported business process von den weiteren Betrachtungen zur Integrationsanalyse ausgeklammert werden.

Manually unsupported business process pattern

$$MUSBP_{AnBP} = \{a | a \in AnBP \setminus (FMBP_{AnBP} \cup PMBP_{AnBP}\} \tag{F.32}$$

Dieses Muster muss für die Belange einer Integrationsanalyse nicht weiter berücksichtigt werden. Mit diesem Muster werden alle Prozesse gefunden, die nicht manuell unterstützt werden. Die aus der Mengenoperation entstehende Menge $MUSBP_{AnBP}$ enthält alle vollautomatisierten Prozesse, alle partiell automatisierten Prozesse für die gilt, dass sie nicht gleichzeitig einen manuell unterstützen Teil haben und dadurch in $PMBP$ enthalten sind, sowie diejenigen für die kein Unterstützung spezifiziert wurde. Damit sind keine Prozesse enthalten, die für eine Integrationsanalyse unmittelbar interessant sind. Darüber hinaus, kann diese Menge auch durch alternative Mengenoperationen auf Basis betrachteter Mengen erzeugt werden.

Strictly unsupported business process pattern

$$USBP_{AnBP} = (UABP_{AnBP} \cap MUSBP_{AnBP}) \tag{F.33}$$

Prozesse für die keine Unterstützung spezifiziert wurde, können auf der einen Seite ein unterspezifiziertes Modell anzeigen oder auf der anderen Seite bewusst leer gelassen worden sein. Sie stellen einen Ansatzpunkt für eine erneute Nachfrage beim Kunden zum Zwecke einer weiteren Detaillierung des Modells dar, liefern aber keine unmittelbaren Hinweise auf Integrationskandidaten, da es sich auch um vollständig manuell unterstützte Prozesse handeln kann.

Partially supported business process pattern

$$\begin{aligned} PSBP_{AnBP} = \{a | a \in (AnBP \setminus (FABP_{AnBP} \cup FMBP_{AnBP} \cup USBP_{AnBP})) \\ : ((\exists as \in AS : (as, a) \in Used\ By) \\ \wedge (\nexists br \in BR : (br, a) \in Assignment)) \\ \vee (\exists sp \in SP(a) : (\nexists br1 \in BR : (br1, sp) \in Assignment) \\ \wedge (\nexists ac \in AC : (ac, sp) \in Assignment)))\} \end{aligned} \tag{F.34}$$

Partiell unterstütze Prozesse stellen ein Muster dar, welches für die Integrationsanalyse von relevanz sein kann. Handelt es sich um Prozesse, bei denen die partielle Unterstützung in einer Automatisierung durch eine Applikation besteht, so können sich daraus unmittelbare Integrationsanforderungen ergeben. Darüber hinaus ist bei einem Auftreten dieses Musters ebenfalls eine erneute Konsultation des Kunden angezeigt, da es sich bei dem nicht unterstützten Teile um eine Unterspezifizierung des Modells handeln kann, wenn die restlichen unterstützenden Elemente dem Modell lediglich noch nicht hinzugefügt wurden.

Fully supported business process pattern

$$FSBP_{AnBP} = AnBP \setminus (USBP_{AnBP} \cup PUSBP_{AnBP}) \tag{F.35}$$

Vollständig unterstützte Prozesse weisen die Stellen aus, an denen ein Modell in ausreichender Weise detailliert wurde. Da sie aus anderen Mengen bestehen, die bereits als in die Betrachtungen einzuschließen, gekennzeichnet wurden, ergibt sich durch dieses Muster kein neuer Beitrag zum Analyseziel.

Heterogeneous application support of a business process pattern

$$\begin{aligned}
ACU(bp) =& \{ac | ac \in AC \\
& \wedge (((ac, bp) \in Used\ By) \\
& \vee (\exists sp \in SP(bp) : (ac, sp) \in Used\ By))\} \\
ACA(bp) =& \{ac | ac \in AC \wedge ((ac, bp) \in Assignment)\} \\
ACSPA(bp) =& \{ac | ac \in AC \wedge (\exists sp \in SP(bp) : (ac, sp) \in Assignment)\} \\
HeASBP_{AnBP} =& a | a \in AnBP \wedge |ACU(a) \cup ACA(a) \cup ACSPA(a)| \geq 2
\end{aligned} \tag{F.36}$$

Dieses Muster detailliert die Art der Unterstützung, in dem es lediglich auf solche Prozesse fokussiert, welche eine heterogene Unterstützung aufweisen. Für unmittelbare Integrationsanforungen kommen jedoch nur die bereits automatisierten Teile eines Prozesses in Frage. Diese sind bereits in $FABP$ und $PABP$ enthalten.

homogeneous application support of a business process pattern

$$HoASBP_{AnBP} = \{a | a \in AnBP \wedge (|ACU(a) \cup ACA(a) \cup ACSPA(a)| = 1)\} \tag{F.37}$$

Auch dieses Muster liefert keinen neuen Beitrag im Sinne der Integrationsanalyse. Es werden Prozesse gefunden, die sich durch eine einheitliche Unterstützung auszeichen. Die automatisierten Teile, welche für eine Integration in Frage kommen, sind bereits in anderen Mengen enthalten. Darüber hinaus sind natürlich auch die Prozesse relevant, bei denen die Unterstützung nicht homogen ist, weshalb diese Einschränkung für das Analyseziel nicht relevant ist.

Full redundant business process automation pattern

$$\begin{aligned}
RFABP_{AnBP} = \{a | a \in & FABP_{AnBP} \\
& \wedge ((|ACA(a)| > 1) \vee ((|ACA(a)| > 0) \\
& \wedge (|ACU(a)| > 0)) \vee ((|ACA(a)| = 1) \\
& \wedge (\exists b1 \in SP(a) : (|ACA(a) \cup ACA(b1)| > 1) \\
& \vee (|ACU(b1)| > 0))) \vee ((|ACA(a)| = 0) \\
& \wedge (\exists b2 \in SP(a) : (|ACA(b2)| > 1) \\
& \vee ((|ACA(b2)| = 1) \wedge (|ACU(b2)| > 0))))\}
\end{aligned} \tag{F.38}$$

Eine redundante Unterstützung eines Prozesses besitzt für eine Integrationsanalyse Relevanz, wenn es sich – wie im vorliegenden Fall – um eine vollständig automatische Redundanz handelt. In diesem Fall kann die Gestaltung einer Integrationslösung aufwändiger werden, da es sich z.B. um einen Mechanismus handeln kann, bei dem das redundante System – im Sinne der Ausfallsicherheit – einen Ausfall des anderen Systems kompensieren soll. Dies muss dann in der Integrationslösung berücksichtigt werden.

Full non-redundant business process automation pattern

$$NRFABP_{AnBP} = FABP_{AnBP} \setminus RFABP_{AnBP} \tag{F.39}$$

Es wird bereits die Menge $FABP_{AnBP}$ berücksichtigt, weshalb das Muster vernachlässigt werden kann, da es nur eine weitere Einschränkung darstellt.

Fully manually supported business process with multiple role-assignments pattern

$$\begin{aligned} MFMBP_{AnBP} = \{a | a \in FMBP_{AnBP} \\ \wedge ((|BRA(a)| > 1) \vee ((|BRA(a)| = 1) \\ \wedge (\exists b1 \in SP(a) : (|BRA(a) \cup BRA(b1)| > 1))) \vee ((|BRA(a)| = 0) \\ \wedge (\exists b2 \in SP(a) : (|BRA(b2)| > 1))))\} \end{aligned} \tag{F.40}$$

Mit Hilfe dieses Musters wird die Aanalyse auf der Ebene des Business Layers verfeinert – in dem dieses Muster das bereits ausgeschlossene Muster – der vollständig manuell unterstützten Prozesse erweitert. Aus diesem Grund kann es ebenfalls ausgeschlossen werden. Die Identifikation von Kommunikationserfordernissen führt nicht zu Integrationserfordernissen auf der Applikationsebene.

Fully manually supported business process with single role-assignment pattern

$$SFMBP_{AnBP} = FMBP_{AnBP} \setminus MFMBP_{AnBP} \tag{F.41}$$

Auch dieses Muster erweitert die Analyse auf der Ebene des Business Layer um die Betrachtung der beteiligten Rollen. Es kann aus den selben Gründen wie das vorangegangene Muster ausgeklammert werden.

Fully redundantly supported business process pattern

$$\begin{aligned} BRA(bp) &= \{a | a \in BR : (a, bp) \in Assignment\} \\ RFSBP_{AnBP} &= MFMBP_{AnBP} \cup RFABP_{AnBP} \\ &\cup (FSBP_{AnBP} \cap PMBP_{AnBP} \cap PABP_{AnBP} \\ &\quad \cap \{a | a \in AnBP : \exists sp \in SP(a) \\ &\quad \wedge ((|BRA(a) \cup ACA(a) \cup BRA(sp) \cup ACA(sp)| > 1) \\ &\quad \vee (|ACA(sp) = 1 \wedge |ACU(sp)| \geq 1))\}) \end{aligned} \tag{F.42}$$

Bei dem vorliegenden Muster werden beide Arten der unterstützung gefunden, womit es sich nicht direkt für die Integrationsanalyse eignet, da bei dieser der Fokus auf die automatisierten Teile erfolg.

Fully non redundantly supported business process pattern

$$NRFSBP_{AnBP} = FSBP_{AnBP} \setminus RFSBP_{AnBP} \tag{F.43}$$

Dieses Muster entspricht einer Verfeinerung des Musters für vollständig automatisierte Prozesse. Es handelt sich um eine Mengensubtraktion und kann bei der Integrationsanalyse nicht direkt genutzt werden. Bei diesem Muster wird keine Einschränkung der Art der Unterstützung vorgenommen, dass heißt es sind auch vollständig manuell unterstützte Prozesse enthalten.

Business object realization analysis pattern:

Business object fully realized by application layer pattern

$$\begin{aligned} FABO_{AnBO} = \{a | a \in AnBO \\ \wedge ((\exists b \in DO : (b, a) \in Realization) \\ \vee (\exists bo \in BO : \\ (((bo, a) \in Composition) \vee ((bo, a) \in Aggregation)) \\ \wedge (\exists c \in DO : (c, bo) \in Realization)))\} \end{aligned} \tag{F.44}$$

Vollständig durch den Application Layer umgesetzte Geschäftsobjekte sind für die Integrationsanalyse von Interesse. Sie besitzen insbesondere für die Analyse von Datenintegration Fragestellungen eine Bedeutung und sind deshalb in weitere Untersuchungen zu integrieren.

Business object partially realized by application layer pattern

$$\begin{aligned} PABO_{AnBO} = \{a | (a \in AnBO \setminus FABO_{AnBO}) \\ \wedge (\exists b \in BO : ((a, b) \in Composition \vee (a, b) \in Aggregation) \\ \wedge (\exists c \in DO : (c, b) \in Realization))\} \end{aligned} \tag{F.45}$$

Partiell durch den Application Layer realisierte Geschäftsobjekte sind für eine Integrationsanalyse von Relevanz, da sich eine Integrationslösung unter Umständen auf genau den realisierten Teil beziehen kann. Dieses Muster wird in die Betrachtungen zur Integrationsanalyse mit einbezogen.

Business object without application layer realization pattern

$$UABO_{AnBO} = AnBO \setminus (FABO_{AnBO} \cup PABO_{AnBO}) \tag{F.46}$$

Geschäftsobjekte, welche keine Realisierung im Application Layer aufweisen sind ein Hinweis für manuelle Prozesse. Es besteht demnach keine Notwendigkeit einer Anwendungssystemintegration. Deshalb kann dieses Muster für die Integrationsanalyse ausgeklammert werden.

Full concrete business object realization pattern

$$\begin{aligned} FCBO_{AnBO} = \{a | \in AnBO \wedge ((\exists b \in R : (b, a) \in Realization) \\ \vee (\exists bo \in BO : (((bo, a) \in Composition) \vee ((bo, a) \in Aggregation)) \\ \wedge (\exists c \in R : (c, bo) \in Realization))\} \end{aligned} \tag{F.47}$$

Das Full concrete business object realization Pattern (FCBO), trifft auf alle Geschäftsobjekte zu, die direkt oder durch Unterelemente eine Realization Beziehung zu einer Repräsentation haben. Für eine Integrationsanalyse, welche sich auf Anwendungssysteme fokussiert ist diese Analyse nicht von Bedeutung, deshalb kann dieses Muster aus den Betrachtungen ausgeklammert werden.

Partial concrete business object realization pattern

$$\begin{aligned} PCBO_{AnBO} = \{a | (a \in AnBO \setminus FCBO_{AnBO}) \\ \wedge (\exists b \in BO : ((a,b) \in Composition \vee (a,b) \in Aggregation) \\ \wedge (\exists c \in R : (c,b) \in Realization))\} \end{aligned} \tag{F.48}$$

Dieses Muster ist eine Spezialisierung des vorangegangenen Musters und untersucht ob eine partielle Realisierung eines Geschäftsobjektes existiert. Dazu wird geprüft, ob Unterlemente –Geschäftsobjekte die zu dem Untersuchten in einer Composition oder Aggregation Relation stehen– eines Geschäftsobjektes über eine Realisierungs Reltation zu einer Representation stehen. Auch eine partielle Realisierung ist für eine Integrationsanalyse nicht relevant, da es sich bei Repräsentation um Modellierungselemente handelt, die nicht auf Anwendungssysteme verweisen.

Business object without concrete realization pattern

$$UCBO_{AnBO} = AnBO \setminus (FCBO_{AnBO} \cup PCBO_{AnBO}) \tag{F.49}$$

Die Abwesenheit einer Realsierung eines Geschäftsobjektes durch eine Repräsentation ergibt für eine Integrationsanalyse keine weiteren Erkenntnisbeitrag. Es kann sich um einer Unterspezifizierung des Modells handeln oder aber um einen Verweise darauf, dass bei einem auf das Pattern passendem Geschäftsobjekt eine andere Art der Realisierung vorliegt. Dennoch ist das Muster nicht direkt für eine Integrationsanalyse anwendbar, da die Art der Realsieriung keine Rolle spielt. Es wird eigentlich nur eine spezifische Realisierungsart ausgeschlossen.

Fully realized business object pattern

$$\begin{aligned} FRBO_{AnBO} = \{a | a \in AnBo \wedge ((\exists b \in (R \cup DO) : (b,a) \in Realization) \\ \vee (\exists bo \in BO : (((bo,a) \in Composition) \\ \vee ((bo,a) \in Aggregation)) \\ \wedge (\exists c \in (R \cup DO) : (c,bo) \in Realization)))\} \end{aligned} \tag{F.50}$$

Das Muster Fully realized business object pattern Die vollständige Realisierung eines Geschäftsobjektes ist – sofern es sich um eine Realisierung im Application Layer handelt – von Relevanz für die Integrationsanalyse. Das vorliegende Muster fasst jedoch die Relaisierungen durch physische Repräsentationen und durch Datenobjekte im Application Layer zusammen, weshalb es nicht direkt verwendet werden kann.

(Strictly) unrealized business object pattern

$$URBO_{AnBO} = AnBO \setminus (FCBO_{AnBO} \cup PCBO_{AnBO} \cup FABO_{AnBO} \cup PABO_{AnBO}) \tag{F.51}$$

Ein unrealized business object weist auf eine mangelhafte Modellierung hin und kann wertvolle Hilfe bei der Prüfung der Modelle auf Vollständigkeit leisten oder bei der Analyse auf Geschäftsprozessebene verwendet werden. Für eine Integrationsanalyse ist es jedoch nicht nutzbar.

Partially realized business object pattern

$$PRBO_{AnBO} = AnBO \setminus (FRBO_{AnBO} \cup URBO_{AnBO}) \tag{F.52}$$

Auch für eine partielle Realisierun gelten die bereits bei den vorangegangenen Pattern angeführten Ausschlusskriterien. Der reine Umstand dass ein Geschäftsobjekt nur partiell realisiert ist, besitzt für eine Integrationsanalyse noch nicht genügend Aussagekraft.

Multiple business object realization pattern

$$\begin{aligned} BOR(bo) =& \{rdo | (rdo \in R \cup DO) \wedge (((rdo, bo) \in Realization) \\ & \vee (\exists bo1 \in BO : (((bo1, bo) \in Composition) \\ & \vee ((bo1, bo) \in Aggregation)) \\ & \wedge (rdo, bo1) \in Realization))\} \\ MFRBO_{AnBO} =& \{a | a \in AnBO \wedge |BOR(a)| \geq 2\} \end{aligned} \tag{F.53}$$

Eine multiple Realisierung eines Geschäftsobjektes ist von hoher Relevanz für eine Integrationsanalyse. Aus diesem Muster lassen sich Hinweise auf mögliche Datenintegrationsproblematiken ableiten. Das vorliegende Muster kann jedoch nicht direkt verwendet werden, da auch hier die Realisierungen durch Represäntationen im Business Layer und durch Datenobjekte im Application Layer zusammengefasst werden. Für die Integrationsanalyse sind aber nur die Realisierungen durch Datenobjekte von Interesse.

Single business object realization pattern

$$SFRBO_{AnBO} = \{a | a \in AnBO \wedge |BOR(a)| = 1\} \tag{F.54}$$

Eine singuläre Relaisierung eines Geschäftsobjektes kann im Rahmen der Integrationanalyse bedingt verwendet werden. Handelt es sich um eine Realisierung durch ein Datenobjekt, so kann je nach dem an welchen Stellen die Daten benötigt werden Hinweise darauf abgeleitet werden, dass anderen Applikationen Zugriff auf dieses Datenobjekt gewährt werden muss. Darüberhinaus lassen sich bei vorliegen dieses Musters Datenintegrationsprobleme ausschließen. Auch müssen keine Fragen darüber, welche der Applikationen in Zukunft die Rolle des führenden Systems übernimmt, beantwortet werden. Das Muster fasst jedoch Realisierungen im Business Layer und Realisierungen im Application Layer zusammen, weshalb es nicht direkt verwendet werden kann.

Multiple application layer business object realization

$$\begin{aligned} BOAR(bo) =& \{do | (do \in DO) \wedge (((do, bo) \in Realization) \\ & \vee (\exists bo1 \in BO) : (((bo1, bo) \in Composition) \\ & \vee ((bo1, bo) \in Aggregation)) \wedge (do, bo1) \in Realization))\} \\ MFABO_{AnBO} =& \{a | a \in AnBO \wedge |BOAR(a)| \geq 2\} \end{aligned} \tag{F.55}$$

Das Muster Multiple application layer business object realization ist von hohem Interesse für die Integrationsanalyse. So müssen im Gegensatz zu dem vorangegangenen Muster beim vorliegen multipler Realisierungen im Application Layer nun Fragen nach dem führenden System und Datenintegrationsproblemen mit hoher Wahrscheinlichkeit beantwortet werden.

Single application layer business object realization pattern

$$SFABO_{AnBO} = \{a | a \in AnBO \wedge |BOAR(a)| = 1\} \tag{F.56}$$

Dieses Muster stellt aus der hier eingenommenen Sichtweise eine Spezialisierung des Musters Single business object realization dar. Hier werden nun nicht die Realisierungen durch Repräsentation und durch Datenobjekte zusammengefasst, sondern ausschließlich jene

im Application Layer betrachtet. Mit Hilfe dieses Musters lassen sich – wenn es auftritt – Datenintegrationsprobleme und Fragen nach dem führenden System ausschließen.

Multiple concrete business object realization pattern

$$\begin{aligned} BONAR(bo) =& \{r|(r \in R) \wedge (((r, bo) \in Realization) \\ & \vee (\exists bo1 \in BO : (((bo1, bo) \in Composition) \\ & \vee ((bo1, bo) \in Aggregation)) \\ & \wedge (r, bo1) \in Realization))\} \\ MFCBO_{AnBO} =& \{a|a \in AnBO \wedge |BONAR(a)| \geq 2\} \end{aligned} \tag{F.57}$$

Multiple concrete business object realization schränkt die Menge der betrachteten Realisierungen auf die jenigen im Business Layer – also physische Repräsentationen – ein. Aus diesem Grund fällt es für die Integrationsanalyse weg.

Single concrete business object realization pattern

$$SFCBO_{AnBo} = \{a|a \in AnBo \wedge |BONAR(a)| = 1\} \tag{F.58}$$

Auch in diesem Fall muss das Muster von den weiteren Betrachtungen ausgeklammert werden, da ebenfalls eine Einschränkung auf die Realisierungen im Business Layer erfolgt.

Die Tabelle 8.1 fasst noch einmal alle bekannten Muster mit Ihrerer Eignung für die Integrationsanalyse zusammen. Dabei steht (++) für eine uneingeschränkte Verwendbarkeit bei der Integrationsanalyse, (+) für eine bedingte Verwendbarkeit (das Muster muss angepasst werden) und (-) dafür, dass die Verwendung ausgeschlossen ist.

Literaturverzeichnis

123JavaShop
123JavaShop. URL: http://sourceforge.net/projects/p123javashop/ (zuletzt geprüft am 16. 08. 2012).

Adams, Jonathan u. a. (2001)
Patterns for e-business: A Strategy for Reuse. IBM Press. ISBN: 1931182027. URL: http://www.ibm.com/developerworks/patterns/ (zuletzt geprüft am 23. 05. 2012).

Agt, Henning u. a. (2009)
Model based semantic conflict analysis for software and data integration scenarios. Berlin: TU Berlin, Professoren der Fak. IV. URL: http://www.dima.tu-berlin.de/fileadmin/fg131/Publikation/Papers/henning-agt-bizycle-semantic-annotation-report-2009.pdf (zuletzt geprüft am 16. 08. 2012).

Aldred, Lachlan u. a. (2009)
»Dimensions of coupling in middleware«. In: *Concurrency and Computation: Practice and Experience* 21.18, Seiten 2233–2269.

Alexander, Christopher (1979)
The timeless way of building. Oxford University Press, USA.

Alexander, Christopher u. a. (1977)
A pattern language: towns, buildings, construction. Center for Environmental Structure: Oxford University Press New York.

Alt, Rainer (1997)
Interorganisationssysteme in der Logistik: Interaktionsorientierte Gestaltung von Koordinationsinstrumenten. Dt. Univ.-Verl. Zugl.: St. Gallen Univ. Dissertation 1997. ISBN: 382440334X.

– **(2008)**
Überbetriebliches Prozessmanagement: Gestaltungsalternativen und Vorgehen am Beispiel integrierter Prozessportale. Logos Berlin, Zugl.: Sankt Gallen, Univ., Habil.-Schr. 2004. ISBN: 3832519084.

Amsterdamska, Olga (1987)
Schools of thought: The development of linguistics from Bopp to Saussure. Band 6. Sociology of the Sciences Monographs. Springer Verlag.

Andersson, Jonas und Johnson, Pontus (2001)
»Architectural Integration Styles for Large-Scale Enterprise Software Systems«. In: *Proceedings of the 5th IEEE International Conference on Enterprise Distributed Object Computing*. EDOC '01. Washington, DC, USA: IEEE Computer Society, Seiten 224–236. ISBN: 0-7695-1345-X. DOI: http://doi.ieeecomputersociety.org/10.1109/EDOC.2001.950442.

Apache Software Foundation
Ofbiz The Apache Open for Business Project. URL: http://ofbiz.apache.org/ (zuletzt geprüft am 17. 08. 2012).

Avgeriou, Paris und Zdun, Uwe (2005)
»Architectural patterns revisited–a pattern language«. In: *In 10th European Conference on Pattern Languages of Programs (EuroPlop 2005)*. Irsee Germany.

Bajgoric, Nijaz und Moon, Young B. (2009)
»Enhancing systems integration by incorporating business continuity drivers«. In: *Industrial Management & Data Systems* 109.

Balzert, Heide (2001)
Objektorientierung in 7 Tagen. Lehrbücher der Informatik. Heidelberg, Berlin: Spektrum Akademischer Verlag.

Balzert, Helmut (1998)
Lehrbuch der Software-Technik, Bd. 2: Software-Management, Software-Qualitätssicherung, Unternehmensmodellierung. Lehrbücher der Informatik. Heidelberg: Spektrum Akademischer Verlag.

Barbier, Frank und Henderson-Sellers, Brian (2001)
»The whole-part relationship in object modelling: A definition in cOlOr«. In: *Information and Software Technology* 43 (1), Seiten 19–39. DOI: http://dx.doi.org/10.1016/S0950-5849(00)00133-6.

Bardmann, Manfred (2010)
Grundlagen der Allgemeinen Betriebswirtschaftslehre. 1., neue Ausg. Wiesbaden: Betriebswirtschaftlicher Verlag Gabler. ISBN: 3834923958.

Barnard, Chester I. (1938)
The functions of the executive. Cambridge (Mass.): Havard University Press.

– **(1968)**
The functions of the executive. Cambridge, Mass. [u.a.]: Harvard University Press. ISBN: 0-674-32803-5.

Bass, Len; Clements, Paul und Kazman, Rick (2003)
Software architecture in practice. Boston [u.a.]: Addison-Wesley. ISBN: 0321154959.

Bauch, Clea (2004)
Planung und Steuerung der Post Merger Integration. Deutscher Universitäts-Verlag, Zugl.: Fribourg Univ. Diss. 2004.

Becker, Jörg u. a. (2007)
»Forschungsmethodik einer Integrationsdisziplin Eine Fortführung und Ergänzung zu Lutz Heinrichs "Beitrag zur Geschichte der Wirtschaftsinformatik"aus gestaltungsorientierter Perspektive«. In: *Wissenschaftstheorie und gestaltungsorientierte Wirtschaftsinformatik*. Herausgegeben von Jörg Becker; Helmut Krcmar und Björn Niehaves. Physica-Verlag, Seiten 1 –19.

bedirect GmbH & Co. KG
bedirect Firmenkurzprofile. URL: www.bedirect.de (zuletzt geprüft am 24. 05. 2012).

Berger, Ulrike und Bernhard-Mehlich, Isolde (1999)
»Die Verhaltenswissenschaftliche Entscheidungstheorie«. In: *Organisationstheorien*. Herausgegeben von Alfred Kieser. Band 3., überarbeitete und erweiterte Auflage. Kohlhammer, Seiten 133–168.

Bezivin, Jean (2005)
»On the Unification Power of Models«. In: *Software and System Modeling*. Band 4. 2, Seiten 171–188.

Bieger, Thomas (2007)
Dienstleistungs-Management: Einführung in Strategien und Prozesse bei persönlichen Dienstleistungen. Band 2974. Bern: UTB/BRO, Haupt Verlag/BRO. ISBN: 978-3-8385-2974-5.

Biro, John Ivan und Kotatko, Peter, Herausgeber (1995)
Frege, sense and reference one hundred years later. Band 65. Philosophical Studies Series. Kluwer Academic Publishers.

Bleicher, Knut (2004)
Das Konzept Integriertes Management Visionen, Missionen, Programme. Band 7., überarb. u. erw. Aufl. Frankfurt am Main [u.a.]: Campus.

Booth, Wayne C.; Colomb, Gregory G. und Williams, Joseph M. (2003)
The craft of research. 2. Chicago guides to writing, editing, and publishing. Chicago [u.a.]: Univ. of Chicago Press.

Böttcher, Martin (2008)
»Architektur integrierter Dienstleistungssysteme: Konzeption, Metamodell und technikraumspezifische Konkretisierung«. Dissertation. Universität Leipzig.

Böttcher, Martin und Klingner, Stephan (2011)
»The Basics and Applications of Service Modeling«. In: *Proceedings of the 2011 Annual SRII Global Conference*. San Jose, CA, Seiten 636–645. DOI: 10.1109/SRII.2011.104.

Braga, Rosana T. Vaccare; Germano, Fernão S. R. und Masiero, Paulo Cesar (1999)
»A pattern language for business resource management«. In: *Proceedings of PLoP'99*. Band 1999.

Bretschneider, Mike und Gebauer, Martin (2011)
ArchiLe an EMF, GMF based Editor for Archimate 1.0. URL: http://sourceforge.net/projects/archile/ (zuletzt geprüft am 18. 08. 2012).

Bronner, Albert (1992)
Handbuch der Rationalisierung. Kontakt & Studium Bd. 331. Ehningen bei Böblingen: expert verlag.

Brown, Randall W. (2007)
»Implementation Of Enterprise Information Systems: A Comparative Study Of Enterpise Application Integration (EAI) vs Enterprise Resource Planning (ERP)«. Dissertation. University of Texas at Arlington.

Bullinger, Hans-Jörg und Fähnrich, Klaus-Peter (1997)
Betriebliche Informationssysteme. Grundlagen und Werkzeuge der methodischen Softwareentwicklung. Berlin [u.a.]: Springer. ISBN: 3-540-61274-2.

Bundesministerium für Bildung und Forschung (2007)
Forschungsrahmenprogramm IKT2020. URL: http://www.bmbf.de/pub/itk2020.pdf (zuletzt geprüft am 23. 05. 2012).

Buschmann, Frank; Henney, Kevlin und Schmidt, Douglas (2007a)
Pattern-Oriented Software Architecture. Band 4: *A Pattern Language for Distributed Computing*. John Wiley & Sons.

Buschmann, Frank; Henney, Kevlin und Schmidt, Douglas C. (2007b)
Pattern-Oriented Software Architecture. Band 5: *On patterns and pattern languages*. Wiley series in software design patterns. Chichester [u.a.]: John Wiley & Sons Inc.

Buschmann, Frank; Meunier, Regine u. a. (1996)
Pattern-Oriented Software Architecture. Band 1: *A System of Patterns*. Chichester [u.a.]: Wiley.

Cardoso, Jorge; Aalst, Wil M.P. van der u. a. (2008)
»Inter-enterprise System and Application Integration: A Reality Check«. In: *Enterprise Information Systems: 9th International Conference, Iceis 2007, Funchal, Madeira, June 12-16, 2007, Revised Selected Papers*. Springer, Seite 3.

Cardoso, Jorge; Barros, Alistair u. a. (2010)
»Towards a unified service description language for the internet of services: Requirements and first developments«. In: *Services Computing (SCC), 2010 IEEE International Conference on*. IEEE, Seiten 602–609.

Chen, David; Doumeingts, Guy und Vernadat, Francois (2008)
»Architectures for enterprise integration and interoperability: Past, present and future«. In: *Computers in Industry* 59.7, Seiten 647–659.

Chen, Si (2005)
Open For Business in a Nutshell Open Source Foundations for Enterprise Applications. URL: http://www.opensourcestrategies.com/downloads/ofbiz_nutshell.pdf (zuletzt geprüft am 30. 08. 2012).

Clements, Paul und Northrop, Linda (2002)
Software Product Lines Practices and Pattern. Addison-Wesley.

Conrad, Stefan (2002)
»Schemaintegration Integrationskonflikte, Lösungsansätze, aktuelle Herausforderungen«. In: *Informatik-Forschung und Entwicklung* 17.3, Seiten 101–111.

Conrad, Stefan u. a. (2006)
Enterprise Application Integration: Grundlagen, Konzepte, Entwurfsmuster, Praxisbeispiele. Elsevier, Spektrum, Akad. Verl. ISBN: 3827415721.

Conte, Agnés u. a. (2002)
»A tool and a formalism to design and apply patterns«. In: *Object-Oriented Information Systems*, Seiten 135–146.

Coplien, James O. und Schmidt, D.C., Herausgeber (1995)
Pattern languages of program design. Addison-Wesley Reading, MA.

Daniel, Rene de und Steinrötter, Hermann (2008)
Enterprise Integration Patterns für SAP NetWeaver PI. Galileo Press, SAP Press.

Davis, L.; Flagg, D. u. a. (2003)
»Classifying interoperability conflicts«. In: *COTS-Based Software Systems*, Seiten 62–71.

Davis, L. und Gamble, R. (2000)
Conflict patterns: Toward identifying suitable middleware. URL: http://www.sei.cmu.edu/library/assets/conflictpatterns.pdf (zuletzt geprüft am 18. 08. 2012).

– (2002)
»Identifying evolvability for integration«. In: *COTS-Based Software Systems*, Seiten 65–75.

Day, John (2008)
Patterns in Network Architecture: A Return to Fundamentals. Pearson Education.

D&B
D&B Datenbank. URL: http://www.dnbgermany.de (zuletzt geprüft am 10. 05. 2011).

Delessy, Nelly; Fernandez, Eduardo B. und Larrondo-Petrie, Maria M. (2007)
»A Pattern Language for Identity Management«. In: ICCGI '07, Seiten 31–36. DOI: 10.1109/ICCGI.2007.5. URL: http://dx.doi.org/10.1109/ICCGI.2007.5.

Derntl, Michael und Motschnig-Pitrik, Renate (2004)
»Patterns for blended, Person-Centered learning: strategy, concepts, experiences, and evaluation«. In: *Proceedings of the 2004 ACM symposium on Applied computing*. ACM, Seiten 916–923.

Dey, Prasanta Kumar; Clegg, Ben und Bennett, David (2010)
»Handbook of Enterprise Integration«. In: Herausgegeben von Mostafa Hashem Sherif. CRC Press. Kapitel Managing Implementation Issues in Enterprise Resource Planning Projects, Seiten 563–578.

Dous, Malte (2007)
Kundenbeziehungsmanagement für interne IT-Dienstleister: Strategischer Rahmen, Prozessgestaltung und Optionen für die Systemunterstützung. Duv, zugl. Dissertation Universität St. Gallen.

Duden (2007)
Deutsches Universalwörterbuch. Band 6., überarbeitete Auflage. Dudenverlag.

Dustdar, Schahram; Gall, Harald und Hauswirth, Manfred (2003)
Software-Architekturen für Verteilte Systeme. Springer Verlag. ISBN: 3540430881.

Dyson, Paul (2001)
Proceedings of the 4th European Conference on Pattern Languages of Programs, 1999. Konstanz: UVK, Univ.-Verl. Konstanz. ISBN: 9783879407743. URL: http://www.worldcat.org/oclc/248380519.

Eco, Umberto (1994)
Einführung in die Semiotik. Autorisierte dt. Ausg., 8., unveränd. Aufl. / von Jürgen Trabant. München : Fink.

Eilebrecht, Kurt und Starke, Gernot (2010)
Patterns kompakt: Entwurfsmuster für effektive Software-Entwicklung. Springer. ISBN: 3827425255.

Engels, Gregor; Hess, Aandreas u. a. (2008)
Quasar Enterprise: Anwendungslandschaften serviceorientiert gestalten. dpunkt. Verlag.

Engels, Gregor und Voß, Marcus (2008)
»Quasar Enterprise«. In: *Informatik-Spektrum* 31.6, Seiten 548–555.

Erasala, Naveen; Yen, David C. und Rajkumar, T.M. (2003)
»Enterprise Application Integration in the electronic commerce world«. In: *Computer Standards & Interfaces* 25.2, Seiten 69–82.

Erl, Thomas (2009)
SOA Design Patterns. 1. Auflage. Prentice Hall International.

Ernst, Alexander M. (2008)
»Enterprise architecture management patterns«. In: *Proceedings of the 15th Conference on Pattern Languages of Programs*. PLoP '08, 7:1–7:20. DOI: 10.1145/1753196.1753205. URL: http://doi.acm.org/10.1145/1753196.1753205 (zuletzt geprüft am 19. 08. 2012).

Eugster, Patrick Th. u. a. (Juni 2003)
»The many faces of publish/subscribe«. In: *ACM Comput. Surv.* 35.2, Seiten 114–131. ISSN: 0360-0300. DOI: 10.1145/857076.857078. URL: http://doi.acm.org/10.1145/857076.857078 (zuletzt geprüft am 19. 08. 2012).

Europäische Gemeinschaften (2006)
Die neue KMU-Definition Benutzerhandbuch und Mustererklärung. German. URL: http://europa.eu.int/comm/enterprise/enterprise_policy/sme_definition/index_de.htm.

Evans, Eric (2004)
Domain-Driven Design: Tackling Complexity in the Heart of Software. Addison-Wesley.

Fernandez, Eduardo B. u. a. (2008)
»Classifying security patterns«. In: *Proceedings of the 10th Asia-Pacific web conference on Progress in WWW research and development*. Springer-Verlag, Seiten 342–347.

Ferstl, Otto K. und Sinz, Elmar J. (2008)
Grundlagen der Wirtschaftsinformatik. 6., überarb. und erw. Aufl. Oldenbourg Wissenschaftsverlag. ISBN: 978-3-486-58755-5.

Fettke, Peter und Loos, Peter (2001)
Zur Klassifikation von Patterns. URL: http://nbn-resolving.de/urn:nbn:de:bsz:ch1-200100766 (zuletzt geprüft am 19. 08. 2012).

Fielding, Roy Thomas (2000)
»Architectural styles and the design of network-based software architectures«. Dissertation. Irvine: University of California.

Fincham, Martin (1999)
»Application integration: to invade ... or not to invade?« In: *MiddlewareSpectra.* Spectrum Reports Limited.

Fischer, Daniel. (2008)
Unternehmensübergreifende Integration von Informationssystemen: Bestimmung des Integrationsgrades auf elektronischen Marktplätzen. Gabler Verlag, zugl. Dissertation TU Ilmenau.

Fowler, Martin (2003)
Patterns of enterprise application architecture. mit Beiträgen von Rice, David; Foemmel, Matthew; Hieatt, Edward; Mee, Robert und Stafford, Randy. Boston [u.a.]: Addison-Wesley Longman Publishing Co. ISBN: 0321127420.

Frank, Ulrich (2003)
»Ebenen der Abstraktion und ihre Abbildung auf konzeptionelle Modelle - oder: Anmerkungen zur Semantik von Spezialisierungs- und Instanziierungsbeziehungen«. In: *EMISA Forum.* Band 23. 2.

Frantz, Frederick K. (1995)
»A taxonomy of model abstraction techniques«. In: *Proceedings of the 27th conference on Winter simulation.* WSC '95. IEEE Computer Society, Seiten 1413–1420. ISBN: 0-7803-3018-8.

Gabriel, Roland (2011)
»Informationssysteme«. In: *Enzyklopädie der Wirtschaftsinformatik Online-Lexikon.* Herausgegeben von Karl Kurbel u. a. Oldenbourg Wissenschaftsverlag. URL: http://www.enzyklopaedie-der-wirtschaftsinformatik.de/wi-enzyklopaedie/lexikon/uebergreifendes/Kontext-und-Grundlagen/Informationssystem/index.html.

Gamma, Erich u. a. (1993)
»Design patterns: Abstraction and reuse of object-oriented design«. In: *ECOOP 93 Object-Oriented Programming*, Seiten 406–431.

– **(1995)**
Design Patterns. Boston, MA: Addison-Wesley.

Garlan, David; Allen, Robert und Ockerbloom, John (1995)
»Architectural mismatch or why it's hard to build systems out of existing parts«. In: *Proceedings of the 17th International Conference on Software Engineering, ICSE 1995.* ISSN: 0270-5257.

Garlan, David und Shaw, Mary (1993)
»An introduction to software architecture«. In: *Advances in software engineering and knowledge engineering* 1, Seiten 1–40.

GBI-Genios Deutsche Wirtschaftsdatenbank GmbH
WISO Datenbank. URL: http://www.wiso-net.de (zuletzt geprüft am 24. 05. 2012).

Gebauer, Martin (2007)
»Wiederverwendung im Integration Engineering«. In: *Integration Engineering Motivation, Begriffe, Methoden, Anwendungsfälle.* Herausgegeben von Klaus-Peter Fähnrich; Maik Thränert und Peter Wetzel. Leipziger Informatik-Verbund (LIV), Seiten 259–266.

Gebauer, Martin und Ngonga Ngomo, Axel-Cyrille (2008)
»SMORE – A Semantic Model Repository«. In: *1st Workshop of Knowledge Reuse at the International Conference on Software Reuse (ICSR)*, Seiten 29–37. ISBN: 978-84-691-3166-4.

Gebauer, Martin und Schmidt, Johannes (2012)
»Problemadäquate Klassifikation und Systematisierung von Integrationsmustern«. In: *Integration betrieblicher Informationssysteme und modellgetriebene Entwicklung.* Herausgegeben von Heiko Kern und Stefan Kühne. Band XXX. Leipziger Beiträge zur Informatik. Leipziger Informatik Verbund, Seiten 21–41.

Gebauer, Martin und Stefan, Fred (2011a)
»Integration Services Today: A Qualitative Study–Design and Preliminary Results«. In: *2011 Ninth IEEE European Conference on Web Services (ECOWS).* Herausgegeben von Gianluigi Zavattaro; Ulf Schreier und Cesare Pautasso. IEEE, Seiten 169–176.

Gebauer, Martin und Stefan, Fred (2011b)
Systemintegration, Eine qualitative Erhebung aus der Sicht von Integrationsdienstleistern. Band XXVIII. Leipziger Beiträge zur Informatik. Leipziger Informatik Verbund. ISBN: 978-3-941608-15-3. URL: http://nbn-resolving.de/urn:nbn:de:bsz:15-qucosa-78607 (zuletzt geprüft am 19. 08. 2012).

Gebauer, Martin und Thränert, Maik (2008)
»Ein Framework für die modellgetriebene Integration im E-Business.« In: *Model-Driven Integration Engineering - Modellierung, Validierung und Transformation zur Integration betrieblicher Anwendungssysteme*. Herausgegeben von Klaus-Peter Fähnrich; Stefan Kühne und Maik Thränert. Eigenverlag Leipziger Informatik-Verbund (LIV), Seiten 277–290.

Gebauer, Martin; Thränert, Maik u. a. (2008)
»Model-Driven Integration Engineering im E-Procurement - Das Beispiel eines Integrationsdienstleisters.« In: *Model-Driven Integration Engineering - Modellierung, Validierung und Transformation zur Integration betrieblicher Anwendungssysteme*. Herausgegeben von Klaus-Peter Fähnrich; Stefan Kühne und Maik Thränert. Eigenverlag Leipziger Informatik-Verbund (LIV), Seiten 217–231.

Geib, Malte (2006)
Kooperatives Customer-Relationship-Management Fallstudien und Informationssystemarchitektur in Finanzdienstleistungsnetzwerken. Springer, zugl. Dissertation Universität St. Gallen.

GI Fachgruppe Software Architektur (2009)
Glossar. URL: http://sdqweb.ipd.kit.edu/mediawiki-fg/index.php/Kategorie:Glossar (zuletzt geprüft am 16. 08. 2012).

Gläser, Jochen und Laudel, Grit (2009)
Experteninterviews und qualitative Inhaltsanalyse: Als Instrumente rekonstruierender Untersuchungen. VS Verlag.

Glinz, Martin (2005)
Vorlesunsgsskript Informatik II: Modellierung Kapitel 12 Abstraktion. Technischer Bericht. Universität Zürich. URL: http://www.ifi.uzh.ch/rerg/fileadmin/downloads/teaching/courses/infII_ss06/inf_II_kapitel_12.pdf.

Goedicke, Michael und Zdun, Uwe (2002)
»Piecemeal legacy migrating with an architectural pattern language: a case study«. In: *Journal of Software Maintenance and Evolution: Research and Practice* 14.1, Seiten 1–30.

Goh, Cheng Hian (1997)
»Representing and Reasoning about Semantic Conflicts in Heterogeneous Information Systems«. Dissertation. Sloan School of Management at the Massachusetts Institute of Technology.

Goldammer, Gerd; Huhn, Günter und Picht, Jochen (1988)
Wirtschaftsinformatik - Grundlagen. Verlag Die Wirtschaft Berlin.

Grochla, Erwin (1974)
»Modelle und betriebliche Informationssysteme«. In: *Integrierte Gesamtmodelle der Datenverarbeitung-Entwicklung und Anwendung des Kölner Integrationsmodells (KIM)*. Seiten 19–33.

Gutenberg, Erich (1983)
Grundlagen der Betriebswirtschaftslehre. Band 1: *Die Produktion*. Berlin [u.a.]: Springer.

Hagen, Claus und Schwinn, Alexander (2006)
»Measured Integration – Metriken für die Integrationsarchitektur«. In: *Integrationsmanagement. Planung, Bewertung und Steuerung von Applikationslandschaften*, Seiten 267–292.

Hahn, Axel (2011)
»Integration von Informationssystemen«. In: *Enzyklopädie der Wirtschaftsinformatik Online-Lexikon*. Herausgegeben von Karl Kurbel u. a. Oldenbourg Wissenschaftsverlag. URL: http://www.enzyklopaedie-der-wirtschaftsinformatik.de/wi-enzyklopaedie/lexikon/daten-wissen/Informationsmanagement/Informationsmanagement--Aufgaben-des/Informationssystem--Integration-von- (zuletzt geprüft am 19. 08. 2012).

Hansen, Hans Robert und Neumann, Gustaf (2002)
Grundlagen betrieblicher Informationsverarbeitung. 8., völlig neubearb. und erw. Aufl., durchges. Nachdr. Lucius & Lucius.

Harrison, Neil und Avgeriou, Paris (2007)
»Leveraging Architecture Patterns to Satisfy Quality Attributes«. In: *Software Architecture*. Herausgegeben von Flavio Oquendo. Band 4758. Lecture Notes in Computer Science. Groningen The Netherlands: Springer Berlin / Heidelberg, Seiten 263–270. ISBN: 978-3-540-75131-1. URL: http://dx.doi.org/10.1007/978-3-540-75132-8_21.

Harrison, Neil; Foote, Brian und Rohnert, Hans (2000)
Pattern Languages of Program Design 4. Boston, MA: Addison-Wesley. ISBN: 0201433044.

Hartmann, Ingo (2001)
»Integration akquirierter Unternehmen in den neuen Bundesländern«. Dissertation. Universität Duisburg Essen.

Hasselbring, Wilhelm (2006)
»Software-Architektur«. In: *Informatik-Spektrum* 29.1, Seiten 48–52.

Hasselbring, Wilhelm u. a. (2004)
»The dublo architecture pattern for smooth migration of business information systems: An experience report«. In: *Proceedings of the 26th international Conference on Software Engineering*. IEEE Computer Society, Seiten 117–126.

Heer, Jeffrey und Agrawala, Maneesh (2006)
»Software Design Patterns for Information Visualization«. In: *IEEE Transactions on Visualization and Computer Graphics* 12, Seiten 853–860. ISSN: 1077-2626. URL: http://doi.ieeecomputersociety.org/10.1109/TVCG.2006.178 (zuletzt geprüft am 19. 08. 2012).

Heinrich, Lutz J. (1990)
»Der Prozeß der Systemplanung und -entwicklung«. In: *Handbuch Wirtschaftsinformatik*. Herausgegeben von Karl Kurbel und Horst Strunz. Stuttgart: Schäffer-Poeschel, Seiten 199–214.

– (2005)
»Forschungsmethodik einer Integrationsdisziplin: Ein Beitrag zur Geschichte der Wirtschaftsinformatik«. In: *NTM Zeitschrift für Geschichte der Wissenschaften, Technik und Medizin* 13.2, Seiten 104–117.

Henderson-Sellers, Brian und Barbier, Franck (1999)
»What is this thing called aggregation?« In: *Proceedings of the Technology of Object-Oriented Languages and Systems*. IEEE, Seiten 236–250.

Hennies, Manfred O.E. (2003)
Allgemeine Volkswirtschaftslehre für Betriebswirte. Band 1: *Grundlagen, Wirtschaftsordnungen, Wirtschaftskreislauf, Agrarwirtschaft*. Berlin: BWV Berliner-Wissenschafts-Verlag.

Henninger, Scott und Ashokkumar, Padmapriya (2006)
An Ontology-Based Metamodel for Software Patterns. CSE Technical reports. University of Nebraska.

Hentrich, Carsten und Zdun, Uwe (2006)
»Patterns for process-oriented integration in service-oriented architectures«. In: *Proceedings of 11th European Conference on Pattern Languages of Programs (EuroPlop 2006)*.

Hepner, M. u. a. (2006)
»Patterns of conflict among software components«. In: *Journal of Systems and Software* 79.4, Seiten 537–551. ISSN: 0164-1212. URL: http://dx.doi.org/10.1016/j.jss.2005.11.211 (zuletzt geprüft am 19. 08. 2012).

Herden, S. u. a. (2006)
Software-Architekturen für das E-Business. Springer.

Hergula, Klaudia (2003)
»Daten- und Funktionsintegration durch Föderierte Datenbanksysteme«. Dissertation. Kaiserslautern: Technische Universität Kaiserslautern.

Hesse, Wolfgang (2006)
»More matters on (meta-) modelling: remarks on Thomas Kühne's "matters"«. In: *Software and Systems Modeling* 5.4, Seiten 387–394. ISSN: 1619-1366.

Heutschi, Roger (2007)
Serviceorientierte Architektur: Architekturprinzipien und Umsetzung in die Praxis. Berlin, Heidelberg: Springer.

Hofmeister, Helge (2008)
»On the Application of the Service-Oriented Architectural Style to Heterogeneous Application Landscapes«. Dissertation. Universität Bamberg.

Hohpe, Gregor und Woolf, Bobby (2003)
Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc. ISBN: 0321200683.

Hoppenstedt
Hoppenstedt Firmeninformationen. URL: http://www.hoppenstedt.de/.

Hruby, Pavel (2006)
Model-driven design using business patterns. Springer-Verlag New York.

IFAC TC 5.3 International Federation of Automatic Control
IFAC TC 5.3. Enterprise Integration and Networking. URL: http://www.ifac-tc53.org/ (zuletzt geprüft am 29. 05. 2012).

Jonge, Merin de (2003)
»To Reuse or to be Reused - Techniques for Component Composition and Construction«. Dissertation. Universiteit van Amsterdam.

Joos, Stefan (2000)
»ADORA-L – Eine Modellierungssprache zur Spezifikation von Software-Anforderungen«. Dissertation. University of Zurich.

Jung, Reinhard (2006)
Architekturen zur Datenintegration: Gestaltungsempfehlungen auf der Basis fachkonzeptueller Anforderungen. Band Zugl.: St. Gallen, Univ., Habil.-Schr., 2005. Wiesbaden: Deutscher Universitäts-Verlag, zugl. Habilitationsschrift (habilitation treatise).

Juric, Matjaz B. u. a. (2007)
SOA Approach to Integration - XML, Web services, ESB, and BPEL in real-world SOA projects. Birmingham - Mumbai: Packt Publishing.

Kapici, Senol (2005)
»Ein stochastisches Risikomodell für komplexe Projekte«. Dissertation. Magdeburg: Otto-von-Guericke-Universität Magdeburg.

Kashyap, Vipul und Sheth, Amit (1996a)
Semantic and schematic similarities between database objects: a context-based approach. Technischer Bericht. Berlin, Heidelberg, Seiten 276–304.

– **(1996b)**
»Semantic Heterogeneity in Global Information Systems: The Role of Metadata, Context and Ontologies«. In: *Cooperative Information Systems: Current Trends and Directions*, Seiten 139–178.

Keller, Wolfgang (2002)
Enterprise Application Integration. Erfahrungen aus der Praxis. Dpunkt Verlag.

Keshav, R. und Gamble, R. (1998)
»Towards a taxonomy of architecture integration strategies«. In: *ISAW '98: Proceedings of the third international workshop on Software architecture*. ACM, Seiten 89–92. ISBN: 1-58113-081-3. DOI: http://doi.acm.org/10.1145/288408.288431.

Khomh, Foutse und Guéhéneuc, Yann-Gaël (Jan. 2008a)
An empirical study of design patterns and software quality. Technischer Bericht 1315. Montreal, Quebec, Canada. URL: http://www.ptidej.net/course/inf6306/fall09/notes/course5/Technical%20Report1315-static%20%28shortened%20for%20INF6306%29.pdf (zuletzt geprüft am 20. 08. 2012).

– **(2008b)**
»Do design patterns impact software quality positively?« In: *Software Maintenance and Reengineering, 2008. CSMR 2008. 12th European Conference on*. IEEE, Seiten 274–278.

Kieser, Alfred, Herausgeber (1999)
Organisationsthorien. Kohlhammer.

Kim, Won (1995)
Modern database systems: the object model, interoperability, and beyond. New York, NY, USA: ACM Press/Addison-Wesley Publishing Co. ISBN: 0-201-59098-0.

Kim, Won und Seo, Jungyun (1991)
»Classifying Schematic and Data Heterogeneity in Multidatabase Systems«. In: *Computer* 24.12, Seiten 12–18. ISSN: 0018-9162. DOI: http://doi.ieeecomputersociety.org/10.1109/2.116884.

Kircher, Michael und Jain, Prashant (2004)
Pattern-Oriented Software Architecture. Band 3: *Patterns for Resource Management*. John Wiley & Sons Inc. ISBN: 978-0-470-845252-7.

Kircher, Michael; Voelter, Marcus u. a. (2004)
»Broker Revisited«. In: *Proceedings of the 9th European Conference on Pattern Languages of Programs*. URL: http://hillside.net/europlop/europlop2004/Papers/wwd/D5.pdf (zuletzt geprüft am 20. 08. 2012).

Klettke, Meike u. a. (2001)
»GETESS - Ontologien, objektrelationale Datenbanken und Textanalyse als Bausteine einer Semantischen Suchmaschine«. In: *Datenbank-Spektrum* 1.1, Seiten 14–24.

Klose, Martina (2002)
»Design Patterns für digitale Produkte im digitalen Wirtschaftsraum«. Dissertation. Universität St. Gallen Hochschule für Wirtschafts-, Rechts-und Sozialwissenschaften (HSG).

Kodituwakku, Saluka R. und Bertok, Peter (2003)
»Pattern categories: a mathematical approach for organizing design patterns«. In: *Proceedings of the 2002 conference on Pattern languages of programs - Volume 13*. CRPIT '02. Melbourne, Australia: Australian Computer Society, Inc., Seiten 63–73. ISBN: 0-909-92591-7. URL: http://dl.acm.org/citation.cfm?id=1151669.1151675 (zuletzt geprüft am 20. 08. 2012).

Kolp, Manuel; Giorgini, Paolo und Mylopoulos, John (2003)
»Organizational Patterns for Early Requirements Analysis«. In: *Advanced Information Systems Engineering*. Herausgegeben von Johann Eder und Michele Missikoff. Band 2681. Lecture Notes in Computer Science. Springer Berlin / Heidelberg, Seiten 1030–1030. URL: http://dx.doi.org/10.1007/3-540-45017-3_41.

Kornek, Christoph (2004)
»Konzeption und Realisierung einer Supply-Chain-Management-orientierten Anwendungsintegration im mittelständischen Automotive-Umfeld«. Dissertation. Universitätsbibliothek Duisburg-Essen.

Kosiol, Erich (1962)
Organisation der Unternehmung. Gabler.

Krcmar, Helmut (2003)
Informationsmanagement. 3., neu überarb. und erw. Aufl. Springer. ISBN: 3-540-43886-6.

Kruchten, Philippe (1995)
»Architectural blueprints – The 4+ 1 view model of software architecture«. In: *IEEE Software* 12.6, Seiten 42–50.

Krueger, Charles W. (1992)
»Software reuse«. In: *ACM Comput. Surv.* 24.2, Seiten 131–183. ISSN: 0360-0300. URL: http://doi.acm.org/10.1145/130844.130856.

Krüger, Wolfgang (Nov. 1988)
»Ein früher Sozialliberaler Der Kathedersozialist Werner Sombart und der Sündenfall des Kapitalismus«. In: *Die Zeit* 46. URL: http://www.zeit.de/1988/46/ein-frueher-sozialliberaler/seite-1.

Kühne, Thomas (2006)
»Matters of (meta-) modeling«. In: *Software and Systems Modeling* 5.4, Seiten 369–385.

Kumar, Kiran und Prabhakar, TV (2010)
»Design Decision Topology Model for Pattern Relationship Analysis«. In:

Kurpjuweit, Stephan (2009)
Stakeholder-orientierte Modellierung und Analyse der Unternehmensarchitektur unter besonderer Berücksichtigung der Geschäfts-und IT-Architektur. Logos Verlag Berlin GmbH.

Lam, Wing (2005)
»Investigating success factors in enterprise application integration: a case-driven analysis«. In: *European journal of information systems* 14.2, Seiten 175–187.

Lam, Wing und Shankararaman, Venky (2004)
»An Enterprise Integration Methodology«. In: *IT Professional* 06.2, Seiten 40–48. ISSN: 1520-9202.

Larsen, Grant (2006)
»Model-driven development: Assets and reuse«. In: *IBM Systems Journal* 45.03, Seiten 541 –554. URL: http://www.research.ibm.com/journal/sj/453/larsen.pdf.

Lee, Kangsun und Fishwick, Paul A. (1996)
»Dynamic model abstraction«. In: *Proceedings of the 28th conference on Winter simulation*. WSC '96. IEEE Computer Society. Coronado, California, United States, Seiten 764–771. URL: http://dx.doi.org/10.1145/256562.256806 (zuletzt geprüft am 20. 08. 2012).

Lehmann, Helmut (1980)
»Integration«. In: *Handwörterbuch der Organisation*. Herausgegeben von Erwin Grochla. Band 2. Auflage. Schäffer-Poeschel, Seiten 976 –984.

Leicher, Andreas (2004)
A framework for identifying compositional conflicts in component-based systems. Technischer Bericht 2004-23, ISSN 1436-9915. TU Berlin.

– **(2005)**
»Analysis of Computational Conflicts in Component-Based Systems«. Dissertation. Technische Universität Berlin.

Leicher, Andreas; Busse, Susanne und Süß, Jörn (2005)
»Analysis of Compositional Conflicts in Component-Based Systems«. In: *Software Composition*. Herausgegeben von Thomas Gschwind; Uwe Aßmann und Oscar Nierstrasz. Band 3628. Lecture Notes in Computer Science. Springer Berlin / Heidelberg, Seiten 67–82. ISBN: 978-3-540-28748-3. URL: http://dx.doi.org/10.1007/11550679_6.

Leppänen, Mauri (2007)
»Towards an Abstraction Ontology«. In: *Proceeding of the 2007 conference on Information Modelling and Knowledge Bases XVIII*. IOS Press. Amsterdam: IOS Press, Seiten 166–185. ISBN: 978-1-58603-710-9.

Liebhart, Daniel u. a. (2008)
Integration Architecture Blueprint: Leitfaden zur Konstruktion von Integrationslösungen. Hanser Verlag.

Lim, Wayne C. (1998)
Managing Software Reuse A Comprehensive Guide to Strategically Reengineering the Organization for Reuseable Components. Prentice Hall PTR.

Lind, Mikael und Goldkuhl, Göran (2001)
»Generic layered patterns for business modelling«. In: *Proceedings of the Sixth International Workshop on the Language-Action Perspective on Communication Modelling (LAP 2001)*. Herausgegeben von M. Schoop und J. Taylor.

Longshaw, Andy und Zdun, Uwe (2006)
Proceedings of the 10th European Conference on Pattern Languages of Programs, 2005. Konstanz: UVK, Universitätsverlag Konstanz. ISBN: 9783879408054. URL: http://www.worldcat.org/oclc/71255715.

Lorenz, Oliver u. a. (2007)
»eBusiness Jahrbuch der deutschen Wirtschaft 2007/2008, Wegweiser GmbH«. In: Herausgegeben von Oliver Lorenz. Wegweiser GmbH, Fraunhofer IAO, Bitkom e.V. Kapitel eBusiness-Barometer 2007/2008, Seiten 8–61.

Lublinsky, Boris und Farell, Michael Jr. (2002)
»Top 10 Reasons why EAI fails«. In: *EAI Journal*.

Lucredio, Daniel u. a. (2006)
»The Draco Approach Revisited: Model-Driven Software Reuse«. In: *6th Brazilian Workshop on Component-Based Software Development (WDBC2006)*. URL: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.94.3298&rep=rep1&type=pdf (zuletzt geprüft am 20. 08. 2012).

Luhmann, Niklas (1984)
Soziale Systeme: Grundriss einer allgemeinen Theorie. 1. Auflage. Frankfurt am Main: Suhrkamp. ISBN: 351857700X.

Lünendonk GmbH (Mai 2012)
TOP 25 IT-Beratungs- und Systemintegrations-Unternehmen in Deutschland 2011. German. URL: http://luenendonk-shop.de/out/pictures/0/lue_listepi_2012_it_beratung_160512%281%29_fl.pdf (zuletzt geprüft am 15. 08. 2012).

Lutz, Jeffrey C. (2000)
»EAI Architecture Patterns«. In: *EAI Journal*.

Majidi, Emad; Alemi, Mahdieh und Rashidi, Hassan (2010)
»Software Architecture: A Survey and Classification«. In: *Second International Conference on Communication Software and Networks, ICCSN'10.* IEEE, Seiten 454–460.

Malich, Stefan (2008)
Qualität von Softwaresystemen: Ein pattern-basiertes Wissensmodell zur Unterstützung des Entwurfs und der Bewertung von Softwarearchitekturen. Gabler, zugl. Dissertation Universität Duisburg-Essen 2007.

Mandl, Peter (2009)
Masterkurs Verteilte betriebliche Informationssysteme: Prinzipien, Architekturen und Technologien. Vieweg+ Teubner. ISBN: 3834805181.

Manolescu, Dragos; Voelter, Marcus und Noble, James, Herausgeber (2006)
Pattern languages of program design 5. Addison-Wesley Professional.

Marquardt, Klaus, Herausgeber (2005)
Proceedings of the 9th European Conference on Pattern Languages of Programs, 2004. Konstanz: UVK, Univ.-Verl. Konstanz. ISBN: 9783879407965. URL: http://www.worldcat.org/oclc/255006525 (zuletzt geprüft am 20. 08. 2012).

Martin, Robert C.; Riehle, Dirk und Buschmann, Frank, Herausgeber (1998)
Pattern Languages of Program Design 3. The Software Patterns Series. Addison-Wesley.

Al-Mashari, Majed; Al-Mudimigh, AAbdullah und Zairi, Mohamed (2003)
»Enterprise resource planning: A taxonomy of critical factors«. In: *European journal of operational research* 146.2, Seiten 352–364.

Matros, Norbert (2006)
Gedanken zu Abstraktion und Modellbildung. Technischer Bericht. URL: http://www.schulabakus.de/Mathematik_und_Bildung/Abstraktion_und_Modellbildung.pdf (zuletzt geprüft am 20. 08. 2012).

Mayring, Phillipp (2002)
Einführung in die qualitative Sozialforschung: eine Anleitung zu qualitativem Denken. Beltz. ISBN: 978-3-407-25252-4.

McIlroy, M. D. (1968)
»Mass Produced Software Components«. In: *Software Engineering*. Herausgegeben von P. Naur und B. Randell. NATO Scientific Affairs Division. Garmisch, Germany, Seiten 138–155.

Mellor, Stephen J. u. a. (2004)
MDA Distilled: Principles of Model-Driven Architecture. Boston: Addison-Wesley.

Metsker, Steven John und Wake, William C. (2006)
Design patterns in Java. Addison-Wesley Professional. ISBN: 0321333020.

Meunier, Regine (1995)
»The Pipes and Filters Architecture«. In: *Pattern Languages of Program Design*. Herausgegeben von James O. Coplien und Douglas C. Schmidt. Band 1. Addison-Wesley. Kapitel 22, Seiten 427–440. ISBN: 0-201-60734-4.

Molina, Arturo u. a. (2007)
»Enterprise Integration and Networking: challenges and trends«. In: 16.4, Seiten 353–368.

Moll, J.H. und Ammerlaan, R.W.M. (2008)
»Identifying Pitfalls of System Integration – An Exploratory Study«. In: *Software Testing Verification and Validation Workshop, IEEE International Conference on*, Seiten 332–338. DOI: http://doi.ieeecomputersociety.org/10.1109/ICSTW.2008.22.

Motschnig-Pitrik, Renate und Kaasbøll, Jens (1999)
»Part-whole relationship categories and their application in object-oriented analysis«. In: *IEEE Transactions on Knowledge and Data Engineering* 11.5, Seiten 779–797.

Mularz, Diane E. (1995)
»Pattern Languages of Program Design«. In: *Pattern languages of program design*. Herausgegeben von James O. Coplien und Douglas C. Schmidt. Addison-Wesley. Kapitel Pattern-based integration architectures, Seiten 441 –452.

Neubauer, Marten (2007)
»Modell einer Integration informationstechnischer Systeme zur Dokumentation im Operationssaal«. Dissertation. Technische Universität Berlin.

Nilsson, Erik G.; Nordhagen, Else K. und Oftedal, Gro (1990)
»Aspects of systems integration«. In: *Proceedings of the First International Conference on Systems Integration, Systems Integration'90*. IEEE, Seiten 434–443.

Noble, James (1998)
»Classifying relationships between object-oriented design patterns«. In: Seiten 98–107.

Noble, James und Biddle, Robert (2006)
»Patterns as signs«. In: *ECOOP 2002–Object-Oriented Programming*, Seiten 1599–1635.

Nomina GmbH Informations- und Marketing-Services
nomina IT-Firmenprofile. URL: www.nomina.de (zuletzt geprüft am 20. 08. 2012).

Northrop, Linda u. a. (2006)
Ultra-large-scale systems: The software challenge of the future. Carnegie Mellon University.

Object Management Group, OMG (2001)
OMG Unified Modeling Language specification. Version 1.4. [Needham, Mass.]: Object Management Group, Inc. URL: http://www.worldcat.org/oclc/223501382 (zuletzt geprüft am 20. 08. 2012).

Object Management Group (OMG) (2003)
UML 2.0 Superstructure Specification, final adopted specification. Technischer Bericht. URL: http://www.omg.org/cgi-bin/doc?ptc/03-08-02 (zuletzt geprüft am 20. 08. 2012).

Object Management Group, OMG (2006)
Reuseable Asset Specification (RAS) Final Adopted Specification. URL: http://www.omg.org/cgi-bin/doc?ptc/04-06-06.pdf (zuletzt geprüft am 20. 08. 2012).

Ogden, Charles K. und Richards, Ivor A. (1974)
Die Bedeutung der Bedeutung: Eine Untersuchung über den Einfluss der Sprache auf das Denken und über die Wissenschaft des Symbolismus. Frankfurt am Main: Suhrkamp.

Ouksel, Aris M. und Sheth, Amit (1999)
»Semantic interoperability in global information systems«. In: *ACM Sigmod Record* 28.1, Seiten 5–12. ISSN: 0163-5808.

o.V. (2012)
Stichwort Rationalisierungsinvestition in Gabler Wirtschaftslexikon. Gabler Verlag. URL: http://wirtschaftslexikon.gabler.de/Archiv/119042/rationalisierungsinvestition-v3.html (zuletzt geprüft am 20. 08. 2012).

Panetto, Hervé und Molina, Arturo (2008)
»Enterprise integration and interoperability in manufacturing systems: Trends and issues«. In: *Computers in Industry* 59.7. Enterprise Integration and Interoperability in Manufacturing Systems, Seiten 641 –646. ISSN: 0166-3615. DOI: DOI:10.1016/j.compind.2007.12.010. URL: http://www.sciencedirect.com/science/article/B6V2D-4SJ9504-1/2/bc0c65595577f763181899d3874c16bb.

Parnas, David Lorge (1972)
»On the criteria to be used in decomposing systems into modules«. In: *Communications of the ACM* 15.12, Seiten 1053–1058. ISSN: 0001-0782.

Patig, Susanne (2001)
»Überlegungen zur theoretischen Fundierung der Disziplin Wirtschaftsinformatik, ausgehend von der allgemeinen Systemtheorie«. In: *Journal for General Philosophy Science* 32, Seiten 39 –64.

Payton, J. u. a. (2000)
»The opportunity for formal models of integration«. In: *International Conference on Information Reuse and Integration, Hawaii, USA.[Links].*

Porter, Ronald; Coplien, James O. und Winn, Tiffany (2005)
»Sequences as a basis for pattern language composition«. In: *Science of Computer Programming* 56.1-2, Seiten 231–249.

Prieto-Diaz, Ruben (1991)
»Implementing faceted classification for software reuse«. In: *Communications of the ACM* 34.5, Seiten 88–97.

Purao, Sandeep; Paul, Sharoda und Smith, Steven (2007)
»Understanding enterprise integration project risks: A focus group study«. In: *18th International Conference on Database and Expert Systems Applications,DEXA'07*, Seiten 850–854.

Rauecker, Bruno (1925)
»Die Bedeutung der Rationalisierung«. In: *Die Arbeit* 11, Seite 685. URL: http://opus.kobv.de/fes/volltexte/2007/244/.

Rautenstrauch, Claus (1993)
Integration Engineering. Addison-Wesley.

Reussner, Ralf und Hasselbring, Wilhelm (2009)
Handbuch der Software-Architektur. 2. Auflage. Heidelberg: dpunkt. verlag GmbH. ISBN: 3898645592.

Riehle, Dirk (1997)
»Composite design patterns«. In: *ACM SIGPLAN Notices* 32.10, Seiten 218–228.

Rising, Linda (2000)
The pattern almanac 2000. Addison-Wesley. ISBN: 0201615673.

Rizzi, Stefano u. a. (2003)
»Towards a logical model for patterns«. In: Lecture Notes in Computer Science. Herausgegeben von Il-Yeol Song u. a., Seiten 77–90.

Robra, Christian (2003)
»Ein komponentenbasiertes Software-Architekturmodell zur Entwicklung betrieblicher Anwendungssysteme«. Dissertation. Universität Bamberg.

– **(2007)**
Modellierung komponentenbasierter Software-Architekturen Grundlagen, Konzepte und Methoden. VDM Verlag Dr. Müller.

Rohnert, Hans (1996)
»The proxy design pattern revisited«. In: *Pattern languages of program design.* Band 2. Addison-Wesley Longman Publishing Co., Inc., Seiten 105–118. ISBN: 0-201-895277.

Rossak, Wilhelm und Ng, Peter A. (1991)
»Some thoughts on systems integration: A conceptual framework«. In: *Journal of Systems integration* 1.1, Seiten 97–114.

Salingaros, Nikos A. (2000)
»The structure of pattern languages«. In: *arq: Architectural Research Quarterly* 4.02, Seiten 149–162.

Šaša, Ana und Krisper, Marjan (2011)
»Enterprise architecture patterns for business process support analysis«. In: *Journal of Systems and Software* to appear.

Sauter, Günther (1998)
Interoperabilität von Datenbanksystemen bei struktureller Heterogenität. Architektur, Beschreibungs- und Ausführungsmodell zur Unterstützung der Integration und Migration. Infix.

Schäfer, Torsten F. (2008)
Stakeholderorientiertes Integrationsmanagement bei Fusionen und Akquisitionen. Springer, zugl. Dissertation Universität Zürich 2007. ISBN: 3835009850.

Schatten, Alexander u. a. (2010)
Best Practice Software-Engineering: Eine praxiserprobte Zusammenstellung von komponentenorientierten Konzepten, Methoden und Werkzeugen. Heidelberg: Spektrum Akademischer Verlag. ISBN: 3827424860.

Scheibler, Thomas (2010)
»Ausführbare Integrationsmuster«. Dissertation. Universität Stuttgart.

Schenderlein, Jens u. a. (2009)
»Applying Model-Driven Integration Engineering to e-business - striving towards a framework concept«. In: *GI Jahrestagung.* Herausgegeben von Stefan Fischer; Erik Maehle und Rüdiger Reischuk. Band 154. LNI. GI, Seiten 3308–3320. ISBN: 978-3-88579-248-2.

Schmidt, Alexander; Otto, Boris und Österle, Hubert (2010)
»Integrating information systems: case studies on current challenges«. In: *Electronic Markets*, Seiten 1–14. ISSN: 1019-6781.

Schmidt, Douglas C. und Buschmann, Frank (2003)
»Patterns, frameworks, and middleware: their synergistic relationships«. In: *ICSE03: Proceedings of the 25th International Conference on Software Engineering.* IEEE Computer Society, Seiten 694–704.

Schmidt, Douglas; Stal, Michael u. a. (2000)
Pattern-Oriented Software Architecture. Band 2: *Patterns for Concurrent and Networked Objects.* Wiley.

Schmidt, Johannes (2010)
»Integrationsmuster: Übersicht und Vorschlag einer übergreifenden Systematisierung«. Masterarbeit. Universität Leipzig, Insitut für Informatik.

Schumacher, Markus (2001)
»Merging security patterns«. In: *EuroPLoP-Focus Group Merging Pattern Languages.*

Schwarzer, Bettina und Krcmar, Helmut (1999)
Wirtschaftsinformatik: Grundzüge der betrieblichen Datenverarbeitung. Schäffer-Poeschel.

Schwinn, Alexander (2005)
»Entwicklung einer Methode zur Gestaltung von Integrationsarchitekturen für Informationssysteme«. Dissertation. Universität St. Gallen Hochschule für Wirtschafts-, Rechts- und Sozialwissenschaften (HSG) Difo-Druck GmbH Bamberg.

– **(2006)**
»Integrationsmanagement – Planung, Bewertung und Steruerung von Applikationslandschaften«. In: Herausgegeben von Joachim Schelp und Robert Winter. Springer. Kapitel Entwurfsmusterbasierter Ansatz zur Systematisierung von Applikationsbeziehungen im Business Engineering, Seiten 31 –59.

Schwinn, Alexander und Schelp, Joachim (2005)
»Design patterns for data integration«. In: *Journal of Enterprise Information Management* 18.4, Seiten 471–482. ISSN: 1741-0398.

Schwinn, Alexander und Winter, Robert (2007)
»Success Factors and Performance Indicators for Enterprise Application Integration«. In: *Enterprise architecture and integration: methods, implementation, and technologies*. Herausgegeben von Wing Lam. Information Science Reference.

Scott, Judy E. und Vessey, Iris (2002)
»Managing risks in enterprise systems implementations«. In: *Communications of the ACM* 45.4, Seiten 74–81.

Selhofer, Hannes u. a. (2008)
The European e-Business Report 2008 6th Synthesis Report of the Sectoral e-Business Watch. Technischer Bericht. European Comission.

Shaw, Mary und Garlan, David (1996)
Software architecture: perspectives on an emerging discipline. Prentice Hall.

Sheth, Amid P. (1999)
»Interoperating Geographic Information Systems«. In: Herausgegeben von Michael Googchild u. a. Kluwer Academic Publishers. Kapitel Changing Focus on Interoperability in information systems: from system, syntax, structure to semantics, Seiten 5–31.

Shiva, Sajjan G. und Shala, Lubna Abou (2007)
»Software Reuse: Research and Practice«. In: *ITNG* 00, Seiten 603–609. DOI: http://doi.ieeecomputersociety.org/10.1109/ITNG.2007.182.

Sink, D. Scott; Poirier, David F. und Smith, George L. (2001)
»Full Potential Utilization of Industrial and Systems Engineering in Organizations«. In: *Handbook of industrial engineering: Technology and operations management*. Herausgegeben von Gavriel Salvendy. 3rd ed. John Wiley & Sons, Seiten 1–25.

Sombart, Werner (1913)
Der Bourgeois. Zur Geistesgeschichte des modernen Wirtschaftsmenschen. Duncker & Humblot.

Spinellis, D. (2001)
»Notable design patterns for domain-specific languages«. In: *Journal of Systems and Software* 56.1, Seiten 91–99.

Spohrer, Jim u. a. (2007)
»Steps toward a science of service systems«. In: *Computer* 40.1, Seiten 71–77.

Stachowiak, Herbert (1973)
Allgemeine Modelltheorie. Wien - New York: Springer-Verlag.

Stadlbauer, Florian (2007)
Zwischenbetriebliche Anwendungsintegration. Deutscher Universitäts-Verlag, zugl. Dissertation Universität München. ISBN: 3835007513.

Stahlknecht, Peter und Hasenkamp, Ulrich (2005)
Einführung in die Wirtschaftsinformatik. Springer.

Stahl, Thomas und Völter, Markus (2006)
Model-Driven Software Development. John Wiley & Sons, Ltd.

Stützle, Rupert (2002)
»Wiederverwendung ohne Mythos: Empirisch fundierte Leitlinien für die Entwicklung wiederverwendbarer Software«. Dissertation. Technische Universität München, Institut für Informatik.

Sumner, Mary (2000)
»Risk factors in enterprise wide information management systems projects«. In: *SIGCPR '00: Proceedings of the 2000 ACM SIGCPR conference on Computer personnel research*. ACM, Seiten 180–187. ISBN: 1-58113-212-X. DOI: http://doi.acm.org/10.1145/333334.333392. (Zuletzt geprüft am 20. 08. 2012).

Szyperski, Clemens (1998)
Component software: Beyond object-oriented programming. ACM Press und Addison-Wesley. ISBN: 9780201178883.

Taylor, Frederick Winslow (1993)
The principles of scientific management. Repr. of the 1911 and 1903 editions. Band 3. The development of management science. Routledge-Thoemmes Pr.

Tešanović, Aleksandra (2004)
What is a pattern? URL: http://st.inf.tu-dresden.de/Lehre/dpf/IntroductoryPapers/tesanovic-WhatIsAPattern.pdf (zuletzt geprüft am 20. 08. 2012).

Tetzel, Kathrin (2006)
»Rationalisierung: Definitionen, Vorstellungen und Zitate von gestern und heute«. In: *RKW Magazin* Nr. 2, Seiten 30–31.

Thalheim, Bernhard (2003)
Informationssystem-Entwicklung Die integrierte Entwicklung der Strukturierung, Funktionalität, Verteilung und Interaktivität von großen Informationssystemen. Technischer Bericht I-15-2003. Postfach 101344, D-03013 Cottbus, Germany: Institut für Informatik, Brandenburgische Technische Universität Cottbus. Vorabdruck.

The Open Group (2009)
Archimate 1.0 Specification. Technischer Bericht. URL: https://www.opengroup.org/archimate/doc/ts_archimate/ (zuletzt geprüft am 20. 08. 2012).

Thränert, Maik (2005)
»Integration - eine Begriffsbestimmung«. In: *Umsetzung von kooperativen Geschäftsprozessen auf eine internetbasierte IT-Struktur: Arbeiten aus dem Forschungsvorhaben Integration Engineering*. Herausgegeben von K.-P. Fähnrich; M. Thränert und P. Wetzel. Universität Leipzig, S. 11 –22.

– (2008)
»Integration-Engineering – Grundlagen, Vorgehen, Fallstudien«. Dissertation. Universität Leipzig.

Tichy, Walter F. (1997)
»A catalogue of general-purpose software design patterns«. In: *Technology of Object-Oriented Languages and Systems, 1997. TOOLS 23. Proceedings*. IEEE, Seiten 330–339. ISBN: 081868383X.

Tiedemann, Marcus (2006)
»Konzepte und Technologien zur Anwendungs-Integration«. Dissertation. Universität Lübeck.

Trotta, Gian (2003)
Dancing Around EAI Bear Traps. URL: http://www.ebizq.net/topics/int_sbp/features/3463.html (zuletzt geprüft am 02. 06. 2012).

Trowbridge, David (2003)
Enterprise Solution Patterns Using Microsoft. NET. Microsoft. ISBN: 0735618399.

Trowbridge, David u. a. (2004)
Integration Patterns. Patterns & Practices. Microsoft Press.

Ulrich, Hans (2001)
Die Unternehmung als produktives soziales System : Grundlagen der allgemeinen Unternehmungslehre. Bern, Stuttgart, Wien: Haupt.

Umapathy, Karthikeyan; Purao, Sandeep und Barton, Russel R. (2008)
»Designing enterprise integration solutions: effectively«. In: *European Journal of Information Systems* 17.5, Seiten 518–527.

Uzquiano, José-Luis (2010)
»Anwendungslandschaften von Versicherungsunternehmen«. In: *Informationsverarbeitung in Versicherungsunternehmen*. Herausgegeben von M. Aschenbrenner u. a. Springer-Verlag New York Inc, Seiten 151 –162.

Vargo, Stephen L. und Lusch, Robert F. (2004)
»Evolving to a new dominant logic for marketing«. In: *Journal of marketing* 68.January, Seiten 1–17.

Vernadat, Francois B. (2009)
»Enterprise Integration and Interoperability«. In: *Handbook of Automation*. Herausgegeben von Shimon Y. Nof. Springer, Seiten 1529–1538.

Vlissides, John M.; Coplien, James O. und Kerth, Norman L., Herausgeber (1996)
Pattern languages of program design 2. Addison-Wesley Longman Publishing Co. Inc.

Vogel, Oliver u. a. (2009)
Software-Architektur Grundlagen - Konzepte - Praxis. Spektrum Akademischer Verlag.

Vogler, Petra (2006)
Prozess-und Systemintegration: Umsetzung des organisatorischen Wandels in Prozessen und Informationssystemen. Deutscher Universitätsverlag, zugl. Habilitation, Universität St. Gallen 2004.

Voigt, Kai-Ingo (2012)
Stichwort Rationalisierung in Gabler Wirtschaftslexikon. Gabler Verlag. URL: http://wirtschaftslexikon.gabler.de/Archiv/57344/rationalisierung-v4.html (zuletzt geprüft am 20. 08. 2012).

Völter, Marcus; Kircher, Michael und Zdun, Uwe (2005)
Remoting patterns: foundations of enterprise, Internet and realtime distributed object middleware. Wiley. ISBN: 0470856629.

Völter, Marcus; Schmid, Alexander und Wolff, Eberhard (2002)
Server Component Patterns: Component Infrastructures Illustrated with EJB. Wiley. ISBN: 0470843195.

Wache, Holger (2003)
Semantische Mediation für heterogene Informationsquellen. Band 261. Dissertationen zur künstlichen Intelligenz. Akademische Verlagsgesellschaft Aka GmbH, zugl. Dissertation Universität Bremen.

Wagner, Holger (2006)
»Der Entwicklungsaufwand der Anwendungssystemintegration: Eine empirische Untersuchung der Einflussfaktoren«. Dissertation. Universität zu Köln.

Wegner, Peter (1996)
»Interoperability«. In: *ACM Computing Surveys (CSUR)* 28.1, Seiten 285–287.

Weichelt, Christian (2009)
»Ein Software-Framework für die Entwicklung betrieblicher Anwendungssysteme auf der Basis von Geschäftsprozessmodellen«. Dissertation. Otto-Friedrich-Universität Bamberg.

Weigand, Hans und Van den Heuvel, Willem-Jan (1998)
»Meta-patterns for Electronic Commerce Transactions based on FLBC«. In: *Proceedings of the the Thirty-First Hawaii International Conference on Systems Sciences (HICSS'98)*. IEEE Computer Society, Seiten 261–270.

Weinert, Peter (2002)
Organisation: Organisationsgestaltung, Organisationsmethodik Fallklausuren. München: Verlag Vahlen.

Weiss, Michael (2008)
»Patterns for Designing Agent-Based E-Business Systems«. In: *Electronic Business: Concepts Methodologies, Tools, and Applications*. Herausgegeben von In Lee. Hershey, PA: Information Science Reference - Imprint of: IGI Publishing.

Wilde, Thomas und Hess, Thomas (2007)
»Forschungsmethoden der Wirtschaftsinformatik«. In: *Wirtschaftsinformatik* 49.4, Seiten 280–287. ISSN: 0937-6429.

Winn, Tiffany und Calder, Paul (2002)
»Is this a pattern?« In: *Software, IEEE* 19.1, Seiten 59–66.

Winter, Robert (2008)
Metamodellbasierte Taxonomie von Integrationsprojekten. Institut für Wirtschaftsinformatik, Universität St. Gallen, St. Gallen.

– **(2009a)**
»Integrationsarchetypen–Herleitung und Begründung«. In: *Management von Integrationsprojekten*. Springer, Seiten 17–51.

– **(2009b)**
»Patterns in der Wirtschaftsinformatik«. In: *Wirtschaftsinformatik* 51.6, Seiten 535–542.

– **(2009c)**
»Was ist eigentlich Grundlagenforschung in der Wirtschaftsinformatik?« In: *Wirtschaftsinformatik* 51.2, Seiten 223–231. ISSN: 0937-6429 0937-6429.

Winter, Robert und Aier, Stephan (2011)
»Informationssystem-Architektur«. In: *Enzyklopädie der Wirtschaftsinformatik Online-Lexikon*. Herausgegeben von Karl Kurbel u. a. Oldenbourg Wissenschaftsverlag. URL: http://www.enzyklopaedie-der-wirtschaftsinformatik.de/wi-enzyklopaedie/lexikon/daten-wissen/Informationsmanagement/Information-/Informationssystem-Architektur/index.html (zuletzt geprüft am 20. 08. 2012).

Winter, Robert; Brocke, Jan vom u. a. (2009)
»Patterns in der Wirtschaftsinformatik«. In: *Wirtschaftsinformatik* 51.6, Seiten 535–542.

Zarnekow, Rüdiger (2007)
Produktionsmanagement von IT-Dienstleistungen: Grundlagen, Aufgaben und Prozesse. Berlin, Heidelberg: Springer.

Zdun, Uwe (2004)
»Some patterns of component and language integration«. In: *Proceedings of 9th European Conference on Pattern Languages of Programs (EuroPlop 2004)*. URL: http://hillside.net/europlop/HillsideEurope/Papers/EuroPLoP2004/2004_Zdun_SomePatternsOfComponent.pdf (zuletzt geprüft am 20. 08. 2012).

Zdun, Uwe und Hvatum, Lise B., Herausgeber (2007)
Proceedings of the 11th European Conference on Pattern Languages of Programs, 2006. Konstanz: UVK Universitätsverlag Konstanz. ISBN: 3879408130. URL: http://hillside.net/europlop/ (zuletzt geprüft am 20. 08. 2012).

Zhao, Liping u. a. (2008)
»A pattern language for designing e-business architecture«. In: *The Journal of Systems & Software* 81.8, Seiten 1272–1287.

Zimmermann, Olaf; Koehler, Jana u. a. (2009)
»Managing architectural decision models with dependency relations, integrity constraints, and production rules«. In: *The Journal of Systems & Software* 82.8, Seiten 1249–1267.

Zimmermann, Olaf; Zdun, Uwe u. a. (2008)
»Combining Pattern Languages and Reusable Architectural Decision Models into a Comprehensive and Comprehensible Design Method«. In: *Proceedings of the Seventh Working IEEE/IFIP Conference on Software Architecture (WICSA 2008)*. IEEE Computer Society Washington, DC, USA, Seiten 157–166.

Zimmer, Walter (1995)
»Relationships between design patterns«. In: *Pattern languages of program design*. Herausgegeben von James O. Coplien und Douglas C. Schmidt. Band 1, Seiten 345–364.

Zwanziger, André und Herden, Sebastian (2004)
Modeling Business Applications with Patterns. URL: http://www.arch-it-ecture.com/download/2004_CICE_Patterns.pdf (zuletzt geprüft am 20. 08. 2012).

Abkürzungsverzeichnis

Abkürzung	Erklärung
ADM	Architectural-Decision-Model
API	Application-Programming-Interface, Programmierschnittstelle
B2B	Business-to-Business
BIS	betriebliches Informationssystem
BMBF	Bundesministerium für Bildung und Forschung
BWL	Betriebswirtschaftslehre
CIM	Computation Independent Model
COMP	Compound Pattern, zusammengesetztes Muster
DMZ	demilitarisierte Zone
EAI	Enterprise Application Integration
EAM	Enterprise-Architecture-Management
EIP	Enterprise Integration Pattern
EMF	Eclipse-Modeling-Framework
ERP	Enterprise-Ressource-Planning
GDV	Gesamtverband der deutschen Versicherungswirtschaft
GMF	Graphical-Modeling-Framework
IC	Integrationskomponente
IE	Integration Engineering
IFAC	International Federation of Automatic Control
IO	Integrationsobjekt
ISZ	Integrationsszenario
IT	Informationstechnologie
IT-System	System der Informationstechnologie
JDBC	Java-Database-Connectivity
KMU	klein und mittlere Unternehmen
MDA	Model-Driven Architecture
MDE	Model-Driven-Engineering
MDIE	Model-Driven Integration Engineering
MDSD	modellgetriebene Softwareentwicklung
MT	Mann-Tage
n.a.	nicht angebbar
ODBC	Open-Database-Connectivity
OMG	Object Management Group
PIM	Platform-Independent-Model
PL/SQL	Procedural Language/SQL
POS	Point-of-Sale
POSA	Pattern-Oriented-Software-Architecture
PPS	Produktionsplanung und -steuerung
PSM	Platform-Specific-Model
RCP	Rich-Client-Platform
RMI	Remote-Method-Invocation
RPC	Remote-Procedure-Call, entfernter Prozeduraufruf
SOA	serviceorientierte Architektur
SPOF	Single-Point-Of-Failure
SQL	Structured Query Language
TCO	Total-Cost-of-Ownership
UA	Unternehmensarchitektur
UI	User Interface
USDL	Unified-Service-Description-Language
VB	Visual Basic
XML	Extensible-Markup-Language

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 sämtliche 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, den 31. August 2012

_____

Martin Gebauer