Use Cases vs. Geschäftsprozesse

Das Requirements Engineering als Gewinner klarer Abgrenzung

Hartmut Umbach · Pierre Metz

Historie und Definitionen

Wir stellen zunächst die unterschiedliche Herkunft von Use Case und Geschäftsprozess dar, bevor wir dann die hier relevanten Aspekte der Systemtheorie kurz erläutern und schließlich darauf basierend unsere Lösung präsentieren.

Geschäftsprozess

Ein Geschäftsprozess ist eine logisch zusammenhängende Kette von Aktivitäten, die bestimmte Einsatzgüter nach vorgegebenen Regeln in Arbeitsergebnisse (Produkte/Erkenntnisse) transformiert [28]. Hammer und Champy [15] rückten mit dem Konzept des „Business Reengineering" Geschäftsprozesse in den Mittelpunkt der Unternehmensgestaltung. Hammer [14] beschreibt den Geschäftsprozess als eine „Gruppe verwandter Aufgaben, die zusammen für den Kunden ein Ergebnis von Wert ergeben". In diesem Sinne ist ein Geschäftsprozess also eine durchgängige Folge von Aktivitäten entlang einer Wertschöpfungskette innerhalb eines Unternehmens zwischen einem internen oder externen Lieferanten und einem internen oder externen Kunden mit einem festgelegten Ergebnis von Wert für den Kunden und wirtschaftlichem Mehrwert für das Unternehmen.

Als Notationsformen für dieses Konzept der Geschäftsprozesse sind z.B. eEPK (erweiterte Ereignisgesteuerte Prozessketten), LoV (Line of Visibility) Charts und auch UML-Aktivitätsdiagramme verbreitet (Abb. 1).

Use Case

Use Cases sind ursprünglich zur Ermittlung funktionaler Anforderungen im Rahmen der Anforderungsanalyse für Softwaresysteme vorgeschlagen worden [17, 19]. Mittlerweile werden Use Cases jedoch als Grundlage für die Modellierung der Außenbeziehungen jeder Art von „System" diskutiert, so auch für Unternehmen [7, 9, 16, 20, 24] und enthaltene Organisationseinheiten.

Ein Use Case ist die Artikulation des Wunsches eines Aktors (eingenommen durch einen Menschen, eine Software, eine Hardware oder einen Zeitstempel), der die Unterstützung eines Systems für seine operativen Aufgaben wünscht. Das Use Case Ziel leitet sich daher von den Bedürfnissen des Aktors ab. Alle von diesem Ziel getriebenen Interaktionen zwischen Aktor und System führen zu einem messbaren Ergebnis, welches einem Aktor oder Stakeholder einen angestrebten Mehrwert (Wertschöpfung) bringen muss.

Use Cases konzentrieren sich somit auf die Art und Weise wie ein Aktor die *Benutzung* des Systems aus *Außensicht* wahrnimmt. Dabei kommt es nicht darauf an, *wie* das System intern die durch den Use Case spezifizierten Anforderungen erfüllt, sondern nur dass dieses geschieht [1, 3, 7, 9, 10, 18, 19, 21] (Abb. 2). Use Cases werden durch das Use Case Template und UML Use Case Diagramme beschrieben (Abb. 3).

DOI 10.1007/s00287-006-0106-8

Hartmut Umbach
Habichthorst 11, 31275 Lehrte
E-Mail: hartmut@umbach.biz

Pierre Metz
Synspace Deutschland GmbH,
Brucknerstraße 50a, 64291 Darmstadt
E-Mail: pierre.metz@synspace.com

Use Case Modelle

Zunehmend gewinnen Use Case Modelle in Ergänzung zu Geschäftsprozessmodellen Bedeutung für die Softwareentwicklung und das Business Process (Re-)Engineering. Diese pragmatische Vorgehensweise, die Vorzüge der Use Case-Technik mit der Methode der Geschäftsprozessanalyse zu kombinieren, führt jedoch seit längerem zu erheblicher Konfusion in Literatur und industrieller Praxis: Beide Konzepte sind nur ungenügend voneinander abgegrenzt und werden als überlappend und teilweise sogar als identisch erklärt. Wir zeigen jedoch die klare konzeptuelle Verschiedenheit, indem wir auf Basis der Systemtheorie beiden Konzepte einen eindeutigen Platz sowohl im Bezugsrahmen „Organisation“ wie auch im Bezugsrahmen „Software“ zuweisen, was darüber hinaus dazu geeignet ist, den Aussagewert des (auch nicht UML-basierten) Requirements und Business (Re-)Engineering erheblich anzureichern. Dadurch wird klar, wie beide Konzepte isoliert und kombiniert sowohl im Business (Re-)Engineering als auch im SW/HW Requirements Engineering zu verwenden sind.

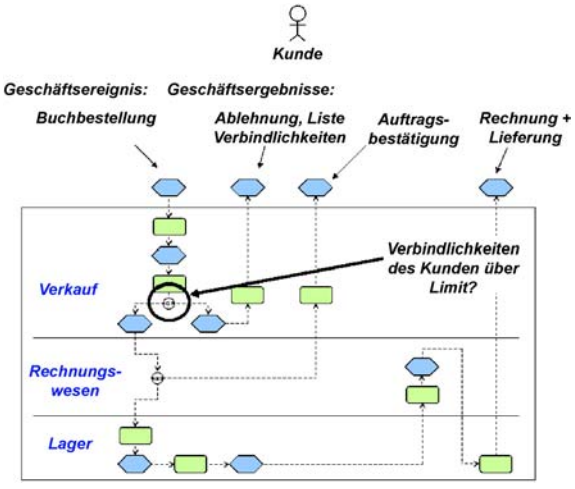

Abb. 1 Beispiel und Skizze eines Geschäftsprozesses für einen Buchhandel, dargestellt durch eine Kombination der Notationen eEPK (erweiterte Ergebnisgesteuerte Prozessketten) und LoV (Line of Visibility) Charts

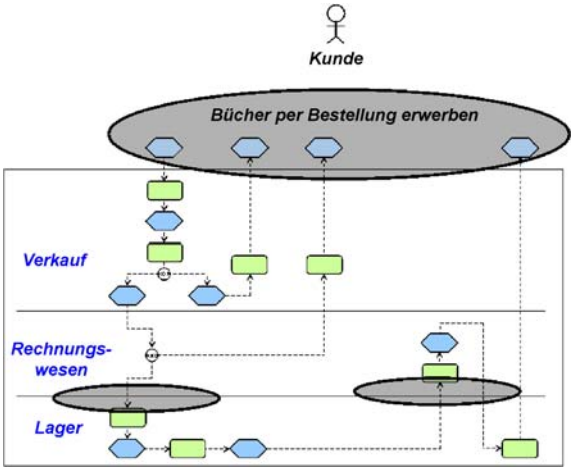

Abb. 2 Abbildung 1 ergänzt um Use Cases (Ellipsen) an der Grenze und im Inneren des Unternehmens: Innen links: Aktor = Mitarbeiter Verkauf, Anbieter = Lager; Innen rechts: Aktor = Mitarbeiter Lager, Anbieter = Rechnungswesen

Unsicherheit in der Literatur

Die Definitionen und Abbildungen in Kapitel „Historie und Definitionen“ zeigen auf den ersten Blick Gemeinsamkeiten

- Use Cases und Prozesse enthalten zeitlich geordnete Abläufe sowie deren Auslöser und Ergebnisse

wie auch Unterschiede

- Use Cases beschreiben das Verhalten des Unternehmens nach außen, Geschäftsprozesse geben eine Sicht auf Unternehmensinterna.

Diese Erkenntnisstufe reicht jedoch nicht für eine scharfe, allgemeingültige Abgrenzung aus, wie die unterschiedlichen Herangehensweisen in der Literatur belegen, in der wir folgende Richtungen erkennen:

Use Case und Geschäftsprozess bestehen aus jeweils dem anderen

In [25] und [26] werden zunächst „Geschäftsanwendungsfälle“ (i.e. Use Cases) identifiziert, um diese anschließend zu Geschäftsprozessen zusammenzusetzen. Demnach sind Geschäftsanwendungsfälle Bestandteile von Geschäftsprozessen.

In [20] wird der Anspruch erhoben, mit Use Cases sowohl die Unternehmensumwelt (durch externe

Use case technology
The pragmatic approach of combining the use case technology with business process analysis and modelling has become more and more relevant to today's software development and business process (re-)engineering. However, since the differences between both concepts have not yet been fully understood, this has caused major confusion in both industrial practice and literature: though treated as distinct approaches, these concepts are explained as overlapping or even being entirely the same thing. Therefore, in this contribution we show the clear conceptual differences by making use of Systems Theory, thereby clearly assigning use cases and business processes distinct but complementary roles in the system contexts "organisation" and "software". As a result, definitions are provided, and the prevailing confusion is replaced with guidance on how to, in isolation or as a combination, exploit both concepts for modern software requirements engineering and business process (re-)engineering.

Aktoren wie Kunden, Lieferanten usw.) wie auch die mit diesen Aktoren abzuwickelnden Geschäftsprozesse beschreiben zu können („The use-case model ... will describe the business and its environment. The business is made up of all the relevant business processes ..." [20]).

Use Case und Geschäftsprozess sind das Gleiche oder dasselbe

In [13] wird die Organisationssicht (d.h. Ablauf- und Aufbauorganisation) der ARIS-Architektur nach Scheer [27] mit dem „Benutzermodell" nach Stary verglichen und festgestellt, dass „der Begriff des Geschäftsprozesses mit Anwendungsfall beschrieben werden" [13] kann.

Ebenso stellt der Autor auch Zusammenhänge zwischen Use Case, Geschäftsprozess und Aufgabe (im Aufgabenmodell nach Stary [30]) fest. Die drei Begriffe können „… praktisch als Synonyme angesehen werden" [12].

In [9] wird explizit erlaubt, dass ein Use Case für ein Unternehmen (*Business Use Case*) sowohl eine reine Black-Box-Darstellung (vollständige Ausblendung von Interna) als auch eine White-Box-Beschreibung sein kann (Systeminterna liegen ganz oder teilweise offen), wie auch ähnlich in [1, 3, 10, 21] gesehen. Sobald also beim „Business Process Modeling" ein White-Box Business Use Case benutzt wird („…showing people and departments and perhaps computers interacting to deliver the organization's externally visible behaviour" [9]) ist ein konzeptueller Unterschied zu einem Geschäftsprozess nicht mehr vorhanden.

Ziel:	**Bücher per Bestellung erwerben**
Zielebene (goal level):	White/Cloud
Trigger:	Kunde tritt mit Bestellgesuch an den Verkäufer heran
Vorbedingungen:	Keine

Hauptablauf:

1. Kunde wählt ein Buch.
2. Verkäufer merkt das Buch für die Bestellung vor.
   Schritt 1 bis 2 wiederholbar
3. Kunde bestätigt die Bestellung.
4. Verkäufer übernimmt die Bestellung und händigt die Auftragsbestätigung aus.
5. Das Unternehmen versendet die Ware und Rechnung unter postalischer Empfangsgarantie.

Nebenabläufe:

4a. Kunde ist Erstkunde:
- 4a1. Der Kunde liefert die Kundendaten und Einverständnis für Kreditauskunft
- 4a2. Das Unternehmen holt erfolgreich die Kreditinformation ein.
- 4a3. Der Verkäufer speichert die Kundendaten.

Weiter bei 4.

4b. Kunde bereits registriert und seine Verbindlichkeiten über Limit:
- 4b1. Der Verkäufer lehnt die Bestellung ab und übergibt Kontoauszug.

Use Case bricht erfolglos ab.

Erfolgsergebnis:

Die Bestellung und Abarbeitung ist im Unternehmen registriert. Der Kunde hat eine Auftragsbestätigung per Ausdruck, die Ware und die Rechung empfangen.

Misserfolgsergebnis:

Die Bestellung wurde abgelehnt, dem Kunden ist eine Liste seiner Verbindlichkeiten und Fristen per Ausdruck ausgehändigt worden.

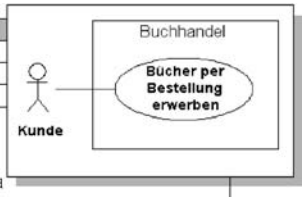

Abb. 3 Spezifikation des Use Case auf Unternehmensebene in Abb. 2 (nach [1, 9, 24]) (Zur Kreditinformation (Nebenablauf 4a) siehe auch Kapitel „Zusammenfassung")

In [6] wird für die Geschäftsprozessmodellierung die UML-Notation für Use Cases verwendet, jedoch wird der Begriff „Use Case“ nicht benutzt. Stattdessen wird eine Ellipse (das UML-Symbol für Use Case) explizit als „Geschäftsprozess“ erklärt und im Text auch so weitergeführt.

In [5] wird zwischen *Business Use Cases* und *Use Cases* unterschieden. Ein Use Case wird als funktionale Anforderung an ein *Software*system bezeichnet, während ein Business Use Case als ein *Synonym* für einen Geschäftsprozess gesehen wird. Hier werden also beide Konzepte auf der Ebene der Unternehmensmodellierung gleichgesetzt.

Unsicherheit in der Praxis

Unklare Handlungsanleitungen

In [16] werden Use Cases ebenfalls als ein Mittel zur Geschäftsprozess-Analyse vorgeschlagen. Das Wie bleibt jedoch unklar: Zum einen wird auf den Zusammenhang zwischen Use Cases und von außen auf das Unternehmen wirkenden Ereignissen hingewiesen, zum anderen wird geraten, in Use Case Beschreibungen „…Ausnahmeabläufe kürzer zu halten als den Hauptablauf, damit das Wesentliche im Geschäftsprozess im Vordergrund steht“ [16].

Missbrauch der Use Case Syntax

Die noch immer unpräzise Semantik der “include” und “extend” Relationen der Use Case-Diagramm-Notation seit UML v1.0 führt immer wieder zu der Missinterpretation, es würde sich hier um dynamische Beziehungen handeln. Folglich wird die Use Case Notation oft fälschlicherweise zur Modellierung von Prozessen oder Workflows vorgeschlagen (z.B. in [2, 6, 20]). Tatsächlich handelt es sich aber um semantisch rein *statische* Beziehungen zur *strukturellen* Reorganisation von Use Case Beschreibungen (z.B. Redundanz-Eliminierung) [1, 3, 7, 18, 19, 21, 23, 24, 34].

Vermischung von Syntax und Konzept in Hinsicht auf Use Cases

Ganz im Gegensatz zu Geschäftsprozessen werden Use Cases oft nicht als ein eigenes Konzept, sondern als bloße Notation wahrgenommen. Der Grund ist vermutlich, dass es für sie im Gegensatz zu Geschäftsprozessen gegenwärtig nur *eine* relevante Darstellungsweise gibt (siehe Kapitel „Use Case“). Warum ist dies ein Problem? – *Konzepte* können nicht mit *Notationen* verglichen werden. Solange Use Cases also als eine *Notation* wahrgenommen werden, können notwendigerweise (wie oben gezeigt) auch keine Abgrenzungskriterien zu dem *Konzept* des Geschäftsprozesses aufgefunden werden.

Systemtheorie – Grundlage unserer Lösung

Die in den Kapiteln „Unsicherheit in der Literatur“ und „Unsicherheit in der Praxis“ aufgezeigten Differenzen zeigen, dass es an einem allgemein akzeptierten theoretischen Überbau fehlt. Die Suche nach einem solchen Überbau führt fast zwangsläufig auf das Gebiet der Systemtheorie, beschäftigt man sich doch im Software Engineering mit *System*anforderungen, *System*eigenschaften, *System*architektur, Software*systemen*, Organisations*systemen* usw.

Organisation und evolutionäre Eigenschaften eines Systems (Systemfunktionen)

Mit den Begriffen der Systemtheorie sind Grundphänomene und Problemmuster in Natur-, Wirtschafts- und Sozialwissenschaften übergreifend interpretierbar. Auch in der Organisations- und Managementlehre hat sich die systemische Sichtweise durchgesetzt [4, 11, 29, 31, 32]. Mit dem durch Luhmann [22] in die allgemeine Systemtheorie eingeführten Konzept der *Selbstreferenz* wurde eine Erklärung der *Autonomie* von Systemen möglich, die einerseits Innensicht und Außensicht strikt voneinander abgrenzt, andererseits jedoch Wechselwirkungen zwischen Innen und Außen postuliert, nach denen die evolutionäre Weiterentwicklung eines Systems abläuft. Die Entstehung und die Weiterentwicklung von Systemen wird über „Systemfunktionen“ erklärt, die aufeinander aufbauen, einen *geschlossenen Regelkreis* bilden und somit ein *evolutionäres Bedingungsgefüge* ergeben [33].

Systemfunktion 1: Grenzbildung

Alle Systemeigenschaften besitzen ihren Ursprung in der Entscheidung darüber, was zum System und was zu seiner Umwelt gehört, d.h., die wichtigste Voraussetzung für die Existenz eines Systems ist die exakte Trennung von „Innen“ und „Außen“. Mit der Festlegung der Systemgrenze entscheidet sich der Zweck des Systems, d.h. welche Leistung das System seiner Umwelt bringt, wie es von außen gesehen und genutzt wird.

Systemfunktion 2: Ressourcengewinnung

Systeme konkurrieren mit ihrer Umwelt um Ressourcen als Grundlage zur Bestandssicherung und Weiterentwicklung. Ressourcen sind Rohstoffe, Produkte, Informationen, Ideen ebenso wie Menschen, die vom Systemzweck abgeleitete Rollen übernehmen können.

Systemfunktion 3: Strukturbildung

Die Struktur eines Systems bildet sich aus der Menge aller Relationen, die Systemelemente miteinander eingehen können, und der Menge aller Fähigkeiten, mit denen Systemelemente auf andere Systemelemente entlang dieser Relationen einwirken können. Das System beschränkt sich bei der Erfüllung seines Zwecks jedoch auf diejenigen Relationen und Fähigkeiten, die nicht widersprüchlich sind und von denen es eine möglichst gute Unterstützung der Bestandssicherung und Weiterentwicklung erwartet (funktionale Differenzierung), z.B. Geschäftsbereiche, Linien- u. Matrixorganisation, aber auch „verdeckte" Strukturen wie informale Netzwerke und „graue Eminenzen". Das System schafft sich also eine Ordnung.

Systemfunktion 4: Prozesssteuerung

Auf der Grundlage des systemeigenen Verständnisses der Abgrenzung zur Außenwelt (Systemfunktion 1), der vorhandenen Ressourcen (Systemfunktion 2) sowie der funktionalen Differenzierung (Systemfunktion 3) kombiniert das System nun bestimmte Fähigkeiten ausgewählter Elemente in der Zeitdimension und generiert damit in freier Entscheidung seine Menge von Handlungsabläufen. Diese stellen eine auf den Systemzweck ausgerichtete, aus Innensicht optimierte Konfiguration dar. Die Möglichkeit der internen Selbstbestimmung führt also unter Berücksichtigung der in Systemfunktion 1 festgelegten relevanten Umwelt(anforderungen) zu autonom erarbeiteten Zielvorstellungen und Steuerungsmechanismen.

Systemfunktion 5: Reflexion

In diesem Stadium beobachtet das System sich selbst in Bezug zu seiner Umwelt. Es analysiert, ob zur Erhöhung des eigenen Nutzens seine Stellung in der und für die Umwelt verändert werden muss und reflektiert entsprechende Strategien und Umsetzungsmöglichkeiten. Vergleichbares erfolgt auch innerhalb des Systems: Mit der Zeit (d.h. emergent über die Systemfunktionen 1 bis 4 hinweg) bilden sich Teilsysteme mit eigenen Identitäten aus, die mit der Identität und den Zielen des Gesamtsystems kompatibel bleiben, aber ebenso wie das Gesamtsystem ihre Position verteidigen oder verbessern müssen. Diese Sichtweise erklärt Organisationskonzepte zur Bildung von selbstständigen Einheiten durch Dezentralisation oder Bildung von Profitcentern.

Systemfunktion 6: Genese

Die in der Reflexion erkannten notwendigen Strategien werden durch die Bildung neuer Systemelemente (Stellen, Ziele, Kommunikationsformen, neue Niederlassungen o.ä.) und Verknüpfungen operativ realisiert.

Diese neuen Elemente und Verknüpfungen bedingen – deshalb wurde ihre Erzeugung in der Reflexion angestoßen – eine veränderte Wahrnehmung des Systems für/durch seine Umwelt und damit die Überprüfung und eventuelle Veränderung der Systemgrenze (z.B. bei Unternehmen Veräußerungen von Unternehmensteilen oder Übernahme anderer Unternehmen). Jegliche Veränderung der Systemgrenze stößt jedoch wiederum eine Modifikation aller nachfolgenden Systemfunktionen an (Ressourcengewinnung, weitere Strukturbildung und Prozesssteuerung, erneute Reflexion und Genese). Damit ist der evolutionäre Regelkreis geschlossen.

Beeinflussbarkeit von Systemen

Die ausschließlich von innen heraus (selbstreferenziell) motivierte Evolution eines Systems liefert aufschlussreiche Hinweise auf dessen Veränderungsbereitschaft. Diese ist gekapselt und ergibt sich ausschließlich dadurch, ob das System für sich Vorteile in einer Veränderung sieht *und* ob es dies mit seiner Kultur vereinbaren kann.

Von außen „aufgezwungene" Veränderungen (z.B. durch Technologieeinführung oder Reorganisation) haben nur dann Aussicht auf Akzeptanz, wenn sie von Argumenten begleitet werden, die eine Chance haben, in der Reflexion als vorteilhafte Handlungsalternative erkannt zu werden. Dies könnte eine Erklärung für manche fehlgeschlagenen Organisations- und Softwareprojekte sein.

Was ist nun die konzeptuelle Abgrenzung Geschäftsprozess – Use Case?

Use Cases explizit als Konzept begreifen

Der Begriff des Use Case muss als ein eigenständiges Konzept verstanden werden: Ein Use Case ist das Beziehen einer Systemleistung durch die Außenwelt, i.e. eine durchgängige, zielgetriebene und von diesem Ziel abgeleitete ergebnisorientierte Menge an Interaktion.

Demgegenüber sind Notationen wie UML Use Case *Diagramme* oder Text*templates* (deren Interaktionsbeschreibung ggf. wiederum grafisch z.B. mittels UML Aktivitäts*diagrammen* dargestellt werden kann) rein syntaktische Beschreibungsalternativen für das Konzept des Use Case.

Adressierung verschiedener Systemfunktionen

Durch die Akzeptanz des Use Case als Konzept wird überhaupt erst eine Abbildung auf Systemfunktionen möglich. Sucht man eine solche Zuordnung, so wird ganz unmittelbar klar, dass Use Cases der „Systemfunktion 1 – Grenzbildung" und Geschäftsprozesse der „Systemfunktion 4 – Prozesssteuerung" zuzuordnen sind:

- **Die Menge aller Use Cases** begründet die Systemgrenze und damit den Zweck des Systems. Denn: Use Cases beschreiben den dynamischen Leistungs- und Nachrichtenaustausch des Systems mit seiner Umwelt aus Außensicht. In der Sprache der Systemtheorie formuliert beschreiben Use Cases, was das System von seiner Umwelt unterscheidet. In die Sprache der Softwareentwicklung übertragen heißt das: Was die Umwelt von dem System erwartet. Die Grenze (die Use Cases) und somit das System bestehen *nur so lange* und *in dieser Form*, wie das System seiner Umwelt Nutzen bringt.
- **Die Menge aller Geschäftsprozesse** ist eine vollständige Beschreibung, wie das System zu einem *bestimmten Zeitpunkt* seine internen Einzelteile dynamisch miteinander verknüpft hat, um die Anforderungen, die sich aus der Differenz zur Umwelt ergeben (die Use Cases) zu erfüllen. Die Systemleistung muss nach außen hin zielbefriedigend sein, um die Existenz des Systems sichern zu können.

Komplementäre Sichten der Wertschöpfung

Es scheint immer noch eine konzeptionelle Überlappung zu geben: Beide Konzepte beschreiben **Dynamik** und **Wertschöpfung**. Bei genauerem Hinsehen handelt es sich hierbei jedoch nicht um eine Überlappung, sondern um einen weiteren Unterschied:

- **Use Cases beschreiben die Wertschöpfung der Außenwelt** während des dynamischen Interagierens mit dem System.
- **Prozesse beschreiben die Wertschöpfung des Systems** während der dynamischen Abarbeitung von Anforderungen der Außenwelt.

Was heißt das für die Praxis?

Sowohl die Analyse als auch die Dokumentation von Use Cases und Geschäftsprozessen sind streng getrennt zu halten und nicht zu integrieren. Warum? Die Systemtheorie liefert folgende Hinweise:

(a) Ohne den Vorgang der Grenzbildung können Prozesse nicht existieren.

(b) Aus Sicht der Umwelt gibt es keine direkte Ursache-Wirkungs-Beziehung zwischen Umweltanforderungen und der internen Organisation des Systems: Trotz gleichbleibender Umweltanforderungen kann das System eigenmotiviert seine internen Prozesse verändern (Optimierung) oder – umgekehrt – trotz sich verändernder Umweltanforderungen seine Prozesse beibehalten (Beharrung). Beispiele:

- **Optimierung:** Das Rechnungswesen (Abb. 1) wartet nicht, bis die Ware kommissioniert ist, sondern erstellt und übermittelt die Rechnung dem Lager parallel zur Kommissionierung.
- **Beharrung:** Kunden verlangen zunehmend Bestellmöglichkeiten via Internet, das Unternehmen hält aber am realen Filialkonzept fest.

(c) Wegen der systemimmanenten Evolution gilt die Ausprägung der Systemfunktionen ausschließlich nur für einen bestimmten Zeitpunkt oder ein Intervall in der Zeit.

Reihenfolge der Analyse (a)

Da Systeme autonom nach internen Bewertungsmaßstäben Entscheidungen treffen (siehe Kapitel „Beeinflussbarkeit von Systemen"), bleiben Sichten und Verbesserungsvorschläge von außen solange hochgradig spekulativ, wie sie nicht von einem tiefgehenden Verständnis für die Motivation des Systems begleitet sind, sich so und nicht anders organisiert zu haben.

Diese Motivation ergibt sich aber aus der Systemgrenze. Diese ist demnach im ersten Schritt durch die Use Cases zu beschreiben. Erst danach ist ausreichend Basis vorhanden, die Geschäftsprozesse zu finden und zu **verstehen**, die die **zu diesem Zeitpunkt** bestehenden Anforderungen aus den Use Cases realisieren (sollen).

Nur wenn man verstanden hat, warum ein (Teil-)System seine Umwelt so und nicht anders wahrnimmt und als Antwort die vorgefundenen Prozesse betreibt, kann man für das System überzeugende Argumente für Veränderung finden.

Dokumentabgrenzung (b) und (c)

Integrierte Dokumentationen von Use Cases und Geschäftsprozessen widersprechen der fehlenden direkten Ursache-Wirkungsbeziehung zwischen Anforderungen der Außenwelt und den Systemreaktionen. Daher sollten beide Dokumentationen/Modelle voneinander getrennt gehalten werden, um diese parallel zu Umweltveränderungen und Systemevolution unabhängig voneinander iterativ revidieren zu können. Erst die Erkenntnis einer Differenz zwischen Umweltanforderungen und Systemverhalten liefert das Ziel für effektives Reengineering, nämlich die (punktuelle) Beseitigung dieser Differenz.

Das System „Unternehmen" und seine Teilsysteme

Auf der Basis des oben gewonnenen Verständnisses vermitteln Use Cases im Business (Re-)Engineering das Gesamtverständnis über die existenziellen Grundlagen des Unternehmens. Sie stellen damit die operationalisierten Unternehmensziele und -strategien dar, wie z.B.

- welche Produkte das System erzeugt und welche Kunden diese kaufen sollen,
- von welchen Lieferanten das System Ressourcen bezieht,
- welche Einflüsse aus der Umwelt wirken (Gesetzgeber, Tarifpartner, ...).

Diesen Zielen ist die interne Organisation verpflichtet (Systemfunktion 5 – Reflexion). Eine gute (d.h. zunächst einmal vom System selbst akzeptierte) Anpassung der Geschäftsprozesse und einzuführender Technologien (wie z.B. Software) kann ausschließlich vor diesem Hintergrund erfolgen. Denn erst jetzt liegen die Determinanten für die Geschäftsprozesse vor, wie z.B.

- zu welchem Preis und mit welchen Produktionsverfahren die Produkte erzeugt werden sollen,
- welche Logistik erforderlich ist,
- welche Personalpolitik angemessen ist.

Was haben nun diese Erkenntnisse für das System „Unternehmen" mit dem Requirement Engineering einer Softwareeinführung in einem Unternehmensausschnitt zu tun?

Einführung von Software ist stets der Versuch, steuernden Einfluss auf ein oder mehrere Teilsysteme des Unternehmens auszuüben. Um die Anforderungen an eine geplante Software für ein oder mehrere Teilsysteme einer Organisation angemessen feststellen zu können, ist es notwendig, als erstes ein genaues Bild des Teilsystems und seiner Umwelt (siehe Kapitel „Dokumentabgrenzung (b) und (c)") zu entwerfen, dessen Funktionieren durch die Software verbessert werden soll.

Aus den Anforderungen an das Teilsystem (dargestellt durch die Use Cases) entstehen Handlungsweisen des Teilsystems (dargestellt durch die Geschäftsprozesse). Bestimmte Teile hiervon werden als automatisierungswürdig eingestuft und daher u.a. in wiederum aus Use Cases bestehenden Software-Anforderungsspezifikationen überführt, die dann schließlich in eine ausführbare Applika-

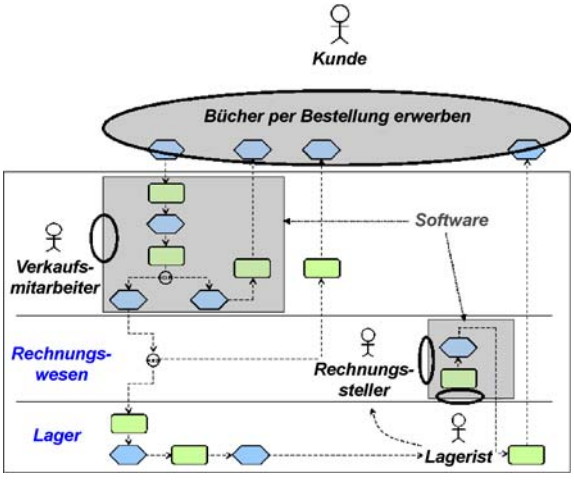

Abb. 4 Sowohl die Bestellregistrierung durch einen Verkaufsmitarbeiter wie auch die Fakturierung durch das Rechnungswesen ist SW-unterstützt, während das Kommissionieren der Bücher manuell bleibt

tion umgesetzt werden (Abb. 4). Diese offensichtlich scheinende Reihenfolge wird sogar in Use Case betonten Vorgehensmodellen nicht erkannt, wie z.B. im Rational Unified Process (RUP) [8].

Damit zeigt die systemische Betrachtungsweise, dass Use Cases und Geschäftsprozesse **auf jeder der Stufen** Unternehmen, Unternehmensausschnitt und Software **notwendig aber disjunkt** sind.

Zusammenfassung

Ohne Widerspruch zu den ursprünglichen Definitionen (Kapitel „Historie und Definitionen") konnten wir unter Zuhilfenahme der Systemtheorie den Use Case als ein explizit eigenständiges Konzept erklären (anstatt als Synonym für eine Notation), welches

- die Grenzziehung,
- die Operationalisierung der Systemziele,
- die zielgetriebene Wertschöpfung der Außenwelt ...

... darstellt, und Geschäftsprozesse als das Konzept der

- Prozesssteuerung hinter der Systemgrenze,
- Umsetzung der Systemziele,
- zielgetriebenen Wertschöpfung des Systeminneren,

was beide Konzepte als scharf abgegrenzte, jedoch komplementäre und notwendige Bestandteile von sowohl Requirements- als auch Business (Re-) Engineering bestätigt. Diese integralen Bestandteile liefern zugleich auf jeder Systemstufe eine Anleitung zur methodischen Modellierungsreihenfolge mit. Man gewinnt zudem hohe dynamische Anpassbarkeit durch eine natürliche getrennte Dokumentation bei gleichzeitiger Garantie durchgängiger Modelle mit Rückverfolgbarkeit von Anforderungen.

Als Konsequenz sind die in der Literatur enthaltenen Darstellungen von Use Cases als „bloßes Hilfsmittel" zur Geschäftsprozessanalyse oder -darstellung und die impliziten oder expliziten Aussagen, beide Konzepte seien äquivalent oder sogar „weitgehend" oder „gänzlich" dasselbe, nicht mehr kompatibel mit unserer Lösung.

In diesem Zusammenhang ergibt sich klar, dass das in Praxis und Literatur vielfach bemühte Unterscheidungskriterium Black-Box vs. White-Box keines darstellen kann, denn solange es sich tatsächlich um eines für die Wertschöpfung der Umweltsicht relevantes operationalisierbares Systemziel handelt, können Use Cases durchaus White-Box-Aussagen enthalten: In Abb. 3 z.B. ist das Einholen von Kreditinformationen eine Aussage über Interna. Da der Aktor einen Rechtsanspruch darauf hat zu wissen, was mit seinen personenbezogenen Daten passiert, muss sich das System tatsächlich an dieser Stelle Einsichtnahme in sein Inneres gefallen lassen. Dies lässt jedoch nach wie vor spezifische Realisierungslösungen (Geschäftsprozesse, Softwareunterstützung) offen.

Besondere Begriffsbildungen wie *Business Use Case* und *System Use Case* sind unnötig. Diese Begriffe beschreiben keine verschiedenen Konzepte, sondern sind lediglich Namensgebungen bestimmter Systemgrenzen.

Die Systemtheorie zeigt weiterhin die Bedeutung von zwei bislang im Requirements und Business (Re-)Engineering zu wenig beachteten Gesichtspunkten: Der Vorgang der Grenzbildung (siehe Kapitel „Organisation und evolutionäre Eigenschaften eines Systems (Systemfunktionen)") sowie die Unmöglichkeit direkter Beeinflussung (siehe Kapitel „Beeinflussbarkeit von Systemen") sind essenziell für das Verständnis von Systemen.

Während das initiale Finden von Use Cases bei einer Unternehmensgründung „auf der grünen Wiese" einigermaßen naheliegend erscheint, ist es tatsächlich ein Schwachpunkt der gegenwärtigen Praxis in *bereits lebenden* Unternehmen, dass hauptsächlich Prozesse im Fokus stehen („sofort in Prozessen denken"). Die Systemtheorie und unsere Ergebnisse jedoch zeigen auf, dass für den Erfolg viel mehr nötig ist als nur eine möglichst exakte Aufnahme der bestehenden Prozesse und deren vermeintliche Optimierung.

Literatur

1. Adolph et al.: Patterns for Effective Use Cases. Addison-Wesley (2003)
2. Ambler, S.: Extending the RUP with the Enterprise Business Modeling Discipline (2004–2006) http://www.enterpriseunifiedprocess.com/essays/enterpriseBusinessModeling.html#ModelTheDomain
3. Armour, F., Miller, G.: Advanced Use Case Modeling. Addison-Wesley (2001)
4. Baecker, D.: Organisation als System. Suhrkamp Akademischer Verlag (1999)
5. Balzert, H.: Objektorientierung in 7 Tagen. Spektrum Akademischer Verlag (2000)
6. Balzert, H.: Lehrbuch der Softwaretechnik. Spektrum Akademischer Verlag (2000)
7. Bittner, K., Spence I.: Use Case Modeling. Addison-Wesley (2003)
8. Booch, G., Jacobson,W., Rumbaugh, J.: Rational Unified Process. Rational Software Corporation
9. Cockburn, A.: Writing Effective Use Cases. Addison-Wesley (2001)
10. Constantine, L., Lockwood, L.: Software For Use. Addison-Wesley (1999)
11. Flämig, M.: Naturwissenschaftliche Weltbilder in Managementtheorien. Campus (1998)
12. Forbrig, P.: Beziehungen zwischen Aufgabenmodellierung und objektorientierter Softwareentwicklung. http://wwwcs.uni-paderborn.de/cs/ag-szwillus/mci/mci2001/forbrig.pdf
13. Forbrig, P.: Objektorientierte Softwareentwicklung mit UML. Hanser (2002)

14. Hammer, M.: Das prozessorientierte Unternehmen. Campus (1997)
15. Hammer, M., Champy, J.: Business Reengineering: Die Radikalkur für das Unternehmen. Campus (1994)
16. Hruschka, P., Josuttis, N., Kocher, H., Oesterreich, B., Krasemann, H., Reinhold, M.: Erfolgreich mit Objektorientierung. Oldenbourg (1999)
17. Jacobson, I.: Object-Oriented Development in an Industrial Environment, Proceedings of OOPSLA'87, special issue of SIGPLAN Notices, Vol.22, No.12 (1987)
18. Jacobson, I.: Use Cases – Yesterday, Today, and Tomorrow, Rational Software Corporation, IBM Software Group, 2004, http://www.therationaledge.com/content/mar_03/f_useCases_ij.jsp
19. Jacobson et al.: Use Case Driven Software Engineering. Addison Wesley (1992, 1994)
20. Jacobson et al.: The Object Advantage. Addison-Wesley (1994)
21. Kulak, D., Guiney, R.: Use Cases – Requirements in Context. Addison-Wesley (2000)
22. Luhmann, N.: Soziale Systeme. Grundriß einer allgemeinen Theorie. Suhrkamp (1984)
23. Metz, P., O'Brien, J., Weber, W.: Against Use Case Interleaving, 4th International IEEE UML Conference, LNCS, Springer (2001)
24. Metz, P.: Revising and Unifying the Use Case Textual Worlds. Ph.D. thesis, Computing Dept. Cork Institute of Technology, Ireland (2005)
25. Oestereich, B. et al.: Objektorientierte Geschäftsprozessmodellierung mit der UML. dpunkt (2003)
26. Oestereich, B.: Analyse und Design mit UML 2. Oldenbourg (2004)
27. Scheer, A.-W.: Architektur integrierter Informationssysteme. Springer (1992)
28. Schwickert, A.C., Fischer, K.: Der Geschäftsprozess als formaler Prozess – Definition, Eigenschaften, Arten. Arbeitspapiere WI Nr. 4/1996, Lehrstuhl für Allg. BWL und Wirtschaftsinformatik, Johannes-Gutenberg-Universität Mainz
29. Senge, P.M.: Die Fünfte Disziplin. Klett-Cotta (1996)
30. Stary, C.: Interaktive Systeme: Softwareentwicklung und Software-Ergonomie. Vieweg (1994)
31. Ulrich, H., Probst, G.: Anleitung zum ganzheitlichen Denken und Handeln: Ein Brevier für Führungskräfte. Haupt (1990)
32. Warnecke, H.: Revolution der Unternehmenskultur – Das fraktale Unternehmen. Springer (1993)
33. Willke, H.: Systemtheorie I: Grundlagen. Lucius&Lucius (2000)
34. Winter, M. „Methodische objektorientierte Softwareentwicklung". dpunkt (2005)