Zentrale Gegenstände der Parametrisierung bei betrieblichen Softwarekomponenten

Jörg Ackermann, Klaus Turowski

Lehrstuhl für Wirtschaftsinformatik und Systems Engineering
Universität Augsburg
Universitätsstraße 16, 86135 Augsburg
{joerg.ackermann | klaus.turowski}@wiwi.uni-augsburg.de

Zusammenfassung. Mit dem Einsatz der Komponentenorientierung wird die Hoffnung verbunden, flexiblere und einfacher anzupassende betriebliche Anwendungssysteme erstellen zu können. Die Praxis zeigt, dass die dabei verwendeten Softwarekomponenten selbst auch anpassbar sein müssen. Parametrisierung ist ein bekanntes Verfahren, welches vor allem zur Anpassung der fachlichen Funktionalität eingesetzt wird. Diese Arbeit beschäftigt sich mit den Grundlagen parametrisierbarer betrieblicher Softwarekomponenten. Dazu werden die zentralen Gegenstände der Parametrisierung (Parameter, Parametergruppen, Parametrisierungsaufgaben und -auswirkungen) bestimmt, präzise definiert und deren Eigenschaften diskutiert. Darüber hinaus wird ein Parameter-Metamodell vorgestellt.

1 Einleitung

Erfolg und Misserfolg von Unternehmen werden zunehmend von deren Fähigkeit bestimmt, schnell auf veränderte Marktsituationen reagieren zu können. Bei den von Unternehmen derzeit eingesetzten betrieblichen Informationssystemen handelt es sich jedoch häufig um große, integrierte Anwendungen, die nur schwierig zu warten [Tu03:1] und nur schwer an veränderte Anforderungen anzupassen sind [Sz02:5].

Komponenten- und serviceorientierte betriebliche Anwendungssysteme haben demgegenüber den Vorteil, dass ein Teil der notwendigen Flexibilität durch Auswahl und Austausch von Komponenten bzw. Services erreicht werden kann. Allerdings kann ein Austausch allein nicht die gesamte notwendige Variabilität zur Verfügung stellen und ist bei häufigen, kleineren Änderungen auch nicht effizient, weil dieser mit einem nicht unerheblichen Aufwand verbunden sein kann. Eine Alternative dazu besteht im Einsatz parametrisierbarer Softwarekomponenten. So lässt sich eine notwendige Variabilität bei komponentenorientierten betrieblichen Anwendungen dann gut durch Parametrisierung realisieren, wenn eine lokal begrenzte Funktionalität anzupassen ist, sich häufige Änderungen ergeben oder mehrere Varianten gleichzeitig zum Einsatz kommen sollen [AT06:357]. Daraus ergibt sich die Motivation zur näheren Untersuchung parametrisierbarer betrieblicher Softwarekomponenten.

Zur Parametrisierung von Komponenten existieren verschiedene Ansätze, die sich vor allem in der Art der Ablage der Parameterwerte unterscheiden: Eingesetzte Ablagemechanismen sind Datenbanktabellen, einfache Initialisierungsdateien sowie die von den gängigen Komponententechnologien vorgesehenen XML-Konfigurationsdateien (Property-Files beim OMG CORBA Component Model [OMG02:1-47], Deployment-Descriptor bei Suns Enterprise Java Beans (EJB) [Mo00:234], Datei Web.config bei Microsofts .NET [Pr02:377]). Die eingesetzten Parametrisierungskonzepte unterscheiden sich jedoch zum Teil deutlich in Bezug auf Funktionsumfang und verwendeter Terminologie – eine einheitliche, technologieübergreifende Betrachtung von Parametrisierung fehlt bisher.

Ziel dieser Arbeit sind daher die Bestimmung und die umfassende Darstellung der für Parametrisierung relevanten Eigenschaften betrieblicher Softwarekomponenten. Dazu wird zunächst Parametrisierung als ein mögliches Anpassungsverfahren in komponentenorientierten Anwendungssystemen vorgestellt (Kapitel 2) und eine Beispielkomponente zur weiteren Illustration eingeführt (Kapitel 3). Danach wird ein Schema hergeleitet, das die Bestimmung der zentralen Gegenstände der Parametrisierung erlaubt (Kapitel 4). Die so identifizierten Gegenstände werden in den nachfolgenden Kapiteln ausführlicher diskutiert: es werden Parameter und Parametergruppen (in Kapitel 5), Parametrisierungsaufgaben (in Kapitel 6) und Parametrisierungsauswirkungen (in Kapitel 7) definiert und deren zentrale Eigenschaften anhand von Klassifikationsschemata herausgearbeitet. Außerdem wird ein Parameter-Metamodell vorgestellt. Die Arbeit schließt mit dem zusammenfassenden Kapitel 8.

Die originären Beiträge dieser Arbeit liegen in der systematischen Bestimmung der zentralen Gegenstände der Parametrisierung sowie in der Definition und der ausführlichen Charakterisierung dieser Gegenstände. Nach Wissen der Autoren wurden die zentralen Parametrisierungseigenschaften bei betrieblichen Fachkomponenten noch nie und allgemein bei betrieblichen Anwendungssystemen noch nicht so umfassend und systematisch herausgearbeitet. Der hier entwickelte Begriffsrahmen kann damit helfen, weitergehende Untersuchungen zur Parametrisierung zu strukturieren und wurde in diesem Sinne schon bei der Erarbeitung einer Spezifikationsmethodik für parametrisierbare Fachkomponenten eingesetzt [AT07].

2 Parametrisierung zur Anpassung von Softwarekomponenten

Anpassung spielt eine wichtige Rolle in komponentenorientierten Anwendungssystemen, da sich in der Praxis gezeigt hat, dass Komponenten nur selten ohne Anpassung wiederverwendet werden können [Bo97:13]. Dabei kann es sich entweder um eine Anpassung der Komponentenarchitektur sowie Auswahl und Zusammenspiel der einzelnen Komponenten oder um eine Anpassung der Komponenten selbst handeln [SC98:305]. Gängige Anpassungsmechanismen sind: Kopieren von Code, Vererbung, Aggregation, Verwendung von Wrapperkomponenten, Superimposition, Adapterinterfaces, parametrisierte Verträge und Parametrisierung – für einen Vergleich siehe z. B. [Bo97] oder [Re01:48-55]. Für einen allgemeinen Überblick zur Anpassung in komponentenorientierten Anwendungssystemen siehe z. B. [Re01].

Bei der Anpassung einzelner Komponenten kann man zwischen geplanter und ungeplanter Anpassung unterscheiden [Ac04:132]. *Geplante Anpassung* bedeutet dabei, dass die Anpassungsmöglichkeiten vom Komponentenhersteller vorgesehen werden. Bei betrieblichen Fachkomponenten wird darüber hinaus zwischen technischer und fachlicher Anpassung unterschieden [Tu03:44]. Die *technische Anpassung* dient dazu, implementierungsbedingte, technische Inkompatibilitäten zu beheben. Unter *fachlicher Anpassung* werden Tätigkeiten verstanden, die die betriebswirtschaftlichen, aufgabenbezogenen Eigenschaften einer Komponente ändern.

Diese Arbeit beschäftigt sich mit Anpassung durch Parametrisierung. Unter *Parametrisierung* (im engeren Sinne) wird ein Verfahren zur Anpassung von Softwarekomponenten (oder anderen Softwareeinheiten) verstanden, bei dem der Komponentenhersteller Parameter vordefiniert und der Komponentenanwender die Parameter mit geeigneten Werten belegt (vgl. auch Kapitel 5). Für die Parameter wird dabei von einer Datenfeldpragmatik ausgegangen, d. h. die Parameter werden mit nicht-ausführbaren Daten belegt. Unter nicht-ausführbaren Daten werden solche Daten verstanden, die kein Programm darstellen (wie z. B. Skripte) und von keiner anderen Seite als Programm interpretiert werden (wie z. B. Workflowschemata). Man kann in diesem Zusammenhang auch von *datenbasierter Parametrisierung* sprechen [Ac04:134]. Zur Abspeicherung der gewählten Parameterwerte können Datenbanktabellen oder die von den gängigen Komponententechnologien vorgesehenen Konfigurationsdateien (z. B. Property Files beim OMG CORBA Component Model [OMG02:1–47] oder Deployment Descriptors bei Suns Enterprise Java Beans (EJB) [Mo00:234]) verwendet werden. Es bleibt zu erwähnen, dass datenbasierte Parametrisierung – außerhalb komponentenorientierter Systeme – vielfach in kommerziell vertriebener Software (wie z. B. in betriebswirtschaftlicher Standardsoftware oder Office-Produkten) zum Einsatz kommt.

Der Begriff *Parametrisierung* wird auch außerhalb des Bereichs Softwarekomponenten verwendet, bezeichnet dabei aber zum Teil Konzepte, die sich bei aller Ähnlichkeit in einigen Punkten unterscheiden: In der Wirtschaftsinformatik wird *Parametrisierung* häufig synonym zu *Customizing* verwendet, worunter die Anpassung einer Standardsoftware an die Anforderungen eines Kunden verstanden wird [Gö97:101]. Ein prominentes Beispiel dafür ist das komplexe, parametergetriebene Customizing von SAP R/3 [SAP02]. Im Software Engineering ist *Parametrisierung* als eine Implementierungstechnik bekannt, die im Zusammenhang mit Wiederverwendung und Variabilität eingesetzt wird. Anwendungsgebiete finden sich im Reuse-driven Software Engineering Business (RSEB) [JGJ97], bei der generischen Programmierung [CE00] und bei Produktlinienansätzen [SB00].

3 Beispielkomponente *Lagermanagement*

Zur Illustration der weiteren Ausführungen wird in diesem Abschnitt eine Beispielkomponente *Lagermanagement* eingeführt. Um ein möglichst realistisches Beispiel zu erhalten, orientieren sich Design und Terminologie der Komponente an der betriebswirtschaftlichen Standardsoftware SAP R/3 Enterprise [SAP02] sowie an einem anerkannten Referenzdatenmodell [BS04]. Um die Verständlichkeit des Beispiels zu unterstützen,

wurde die Komponente jedoch deutlich vereinfacht – reale Anwendungen sind typischerweise komplexer. Die betriebliche Aufgabe der Komponente besteht darin, einen einfachen Lagerkomplex zu verwalten. Die von der Komponente verwalteten Daten werden in Abb. 1 durch ein Datenmodell dargestellt. Die Komponente erlaubt die Definition mehrerer *Lager*. Jedes Lager besteht aus einer Anzahl von *Lagerplätzen*, auf denen die Materialien physisch gelagert werden. Eine Instanz von *Lagerbestand* repräsentiert eine Einheit eines Materials, welche auf einem bestimmten Lagerplatz gelagert wird. Der Typ *Material* enthält die lagerspezifischen Eigenschaften eines Materials. Zur Vereinfachung wird angenommen, dass jedes Material in genau einem Lager gelagert wird – auf die Verwendung komplexer Lagerfindungsstrategien wird verzichtet. Außerdem werden im Beispiel keine Lagereinheitentypen verwendet und es wird stattdessen angenommen, dass innerhalb eines Lagers jeder Lagerplatz für ein Material geeignet ist.

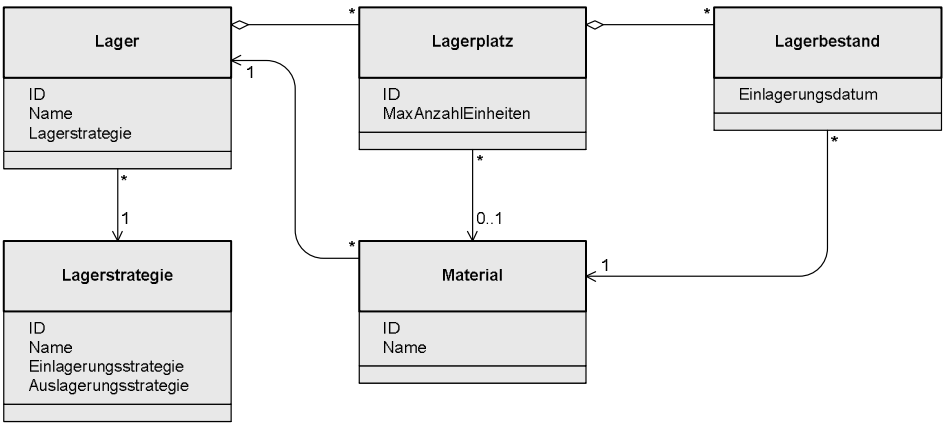

Abbildung 1: Von der Komponente *Lagermanagement* verwaltete Daten

Außerdem erlaubt die Komponente mehrere Varianten der Lagersteuerung einzusetzen (z. B. Fixplatzlager oder Hochregallager). Dazu enthält der Typ *Lagerstrategie* neben einer ID und einem Namen die zwei Steuerungsparameter *Einlagerungsstrategie* und *Auslagerungsstrategie*, mit denen die anzuwendenden Verfahren für Einlagerung (z. B. *dynamisch* oder *statisch*) und Auslagerung (z. B. *First-In-First-Out*) auszuwählen sind. Jedem Lager ist die für das Lager geeignete Lagerstrategie zuzuordnen.

Die Komponente bietet Schnittstellen zur Parametrisierung (z. B. *ILagerstrategie*) und zum Aufruf der betrieblichen Dienste der Komponente (z. B. *ILagermanagement*) an – auf detailliertere Ausführungen dazu wird an dieser Stelle verzichtet.

4 Ermittlung der zentralen Gegenstände der Parametrisierung

Zum besseren Verständnis der parametrisierungsrelevanten Eigenschaften betrieblicher Softwarekomponenten werden in diesem Beitrag die zentralen Gegenstände der Parametrisierung vorgestellt und beschrieben. Unter dem Begriff *zentrale Gegenstände der Parametrisierung* werden dabei alle wesentlichen Konzepte und Begriffe verstanden, die

im Zusammenhang mit Parametrisierung stehen. In diesem Kapitel werden dazu zunächst die zentralen Gegenstände bestimmt und überblicksartig vorgestellt.

Bei der Untersuchung von Parametrisierung ist es hilfreich, nach Sachverhalten zu unterscheiden, die zur Anpassungszeit und zur Laufzeit von Bedeutung sind [Ac04:136]:

- Der Begriff *Parametrisierungsoptionen* bezeichnet die zur Anpassungszeit relevanten Eigenschaften zur Parametrisierung und fasst dabei die funktionalen und nicht-funktionalen Eigenschaften der vom Hersteller vorgesehenen Parameter sowie die Möglichkeiten zur Parameterpflege zusammen.
- Unter *Parametrisierungsauswirkung* wird eine Variabilität an Datenstruktur oder Verhalten der Software verstanden, welche durch das Setzen von Parameterwerten erreicht werden kann und damit zur Laufzeit relevant ist.

Beispiele für Parametrisierungsoptionen und -auswirkungen finden sich in der Beispielkomponente *Lagermanagement* (vgl. Kapitel 3): *Einlagerungsstrategie* ist ein obligatorischer Parameter, der durch die Schnittstelle *ILagerstrategie* gesetzt wird und dessen Pflege aus fachlicher Sicht durch die Aufgabe *Einlagerungsstrategie festlegen* beschrieben wird. Eine Auswirkung des Parameters *Einlagerungsstrategie* besteht darin, dass er festlegt, mit welcher Strategie die betriebliche Aufgabe der Einlagerung ausgeführt wird.

Anzumerken bleibt, dass die Trennung in Parametrisierungsoptionen und -auswirkungen in der Praxis so nicht möglich ist, da für einen Komponentenanwender die Parametrisierungsoptionen immer nur im Zusammenhang mit den daraus resultierenden Auswirkungen interessant sind. Diese Trennung ist jedoch für die weiteren Untersuchungen sinnvoll, da sie den Gesamtkomplex der Eigenschaften zur Parametrisierung unterteilt und dadurch ein strukturiertes Vorgehen ermöglicht.

Zur Bestimmung der zentralen Gegenstände der Parametrisierung spielen zwei Aspekte eine wesentliche Rolle:

- Es ist (wie zuvor diskutiert) nach Parametrisierungsoptionen und Parametrisierungsauswirkungen zu unterscheiden.
- Zur Strukturierung von Anwendungssystem(teil)en wird häufig zwischen den Sichten für Daten, Funktionen und Prozessen unterschieden: Eine solche Strukturierung (mit teils abweichenden Begriffen) findet sich z. B. bei der Modellierung betrieblicher Anwendungssysteme [Sc98:37; SAP97], bei der Entwicklung komponentenorientierter (und anderer) Anwendungssysteme [DW98:43], zur Strukturierung der Außensicht von Softwarekomponenten [Ov06:126] sowie als Modellperspektiven bei der Systemtheorie [Ro99:75-81].

	Datensicht	**Funktionssicht**	**Prozesssicht**
Parametrisierungs-optionen	Parameter und Parametergruppen	Parametrisierungs-aufgaben	Reihenfolge von Parametrisierung-saufgaben
Parametrisierungs-auswirkungen	Auswirkungen auf betriebliche Begrif-fe und Daten	Auswirkungen auf betriebliche Aufga-ben und Funktionen	Auswirkungen auf Reihenfolge von Aufgaben und Funktionen

Abbildung 2: Bestimmung der zentralen Gegenstände der Parametrisierung

Durch diese zwei Aspekte ergibt sich die Struktur der Matrix in Abb. 2, welche die zentralen Gegenstände der Parametrisierung enthält. Die so identifizierten Gegenstände werden ebenfalls in Abb. 2 überblicksartig aufgeführt, allerdings erst in späteren Kapiteln hergeleitet und detaillierter beschrieben: Für die Parametrisierungsoptionen werden zunächst in Kapitel 5 Parameter und Parametergruppen (Datensicht) und dann in Kapitel 6 Parametrisierungsaufgaben und deren Reihenfolge (Funktions- und Prozesssicht) untersucht. Danach werden die laufzeitrelevanten Parametrisierungsauswirkungen im Kapitel 7 thematisiert.

5 Parameter und Parametergruppen

Die zentralen Gegenstände bei den Parametrisierungsoptionen aus Datensicht sind naturgemäß gerade die Parameter und deren Struktur. In den folgenden Unterabschnitten werden die Parameter selbst sowie ihre strukturelle Zusammenfassung zu Parametergruppen näher untersucht. Die so gewonnenen Erkenntnisse werden danach in einem Parameter-Metamodell strukturiert abgebildet.

5.1 Parameter

Es gibt derzeit keine einheitliche Definition des Begriffs *Parameter* für die Anpassung von Softwarekomponenten. Ausgehend von bestehender Literatur (vgl. [AT06:343f.] für eine ausführliche Diskussion) werden die Begriffe *Parameter* und *Parameterwert* wie folgt definiert:

> „Ein *(datenbasierter) Parameter* ist ein Datenfeld, welches zu einer Softwarekomponente (oder zu einem anderen Softwarebaustein) gehört und sich durch folgende Merkmale auszeichnet:

- Das Datenfeld dient zur geplanten Anpassung der Software und wird dazu vom Hersteller der Software zur Entwicklungszeit vordefiniert. Der Hersteller definiert neben dem Datenfeld auch dessen Bedeutung und dessen Auswirkungen auf Struktur und Verhalten der Software.
- Der Verwender der Software belegt das Datenfeld zur Anpassungszeit so mit Werten, dass die Software seinen Anforderungen entsprechend arbeitet.
- Die vom Verwender ausgewählten Werte sind nicht-ausführbar, werden (z. B. in Datenbanktabellen oder Dateien) abgespeichert und zur Laufzeit ausgewertet."

„*Parameterwert* bezeichnet einen Wert, der einem Parameter aktuell zugewiesen ist."

Es sei darauf hingewiesen, dass ein Parameter nach dieser Definition ein nicht weiter strukturiertes Datenfeld ist. Komplexere Parameterstrukturen sind natürlich bei der Parametrisierung möglich, werden aber als Parametergruppe und nicht als Parameter bezeichnet. Der Grund dafür liegt – neben der Anlehnung an die Literatur – darin, dass damit eine begriffliche Trennung erfolgt zwischen (den für die Steuerung einer Komponente verantwortlichen) elementaren Parametern und der Struktur der Parameter.

Es bleibt anzumerken, dass sich Parameter zur fachlichen Anpassung betrieblicher Anwendungssysteme teilweise nur schwer von Anwendungsdaten (insbesondere Stammdaten) abgrenzen lassen [Gö97:102, AR00:65]. Ansatzpunkte für eine Unterscheidung ergeben sich dadurch, dass Parameter im Gegensatz zu Anwendungsdaten

- einem Anwender einen Entscheidungsspielraum im Sinne alternativer Eingabewerte bieten (im Gegensatz zu nicht beeinflussbaren Ist-Werten) [DMH99:3],
- typischerweise von fachlichen Systemverantwortlichen (Superuser) und nicht vom Endanwender gesetzt werden [Gr98:481],
- zur Konfigurationszeit zu pflegen sind (spätere Änderungen sind meist möglich) [Ri00:258],
- bei Softwarekomponenten oft über eigene Schnittstellen zu ändern sind, die nur für Konfigurationszwecke und nicht durch andere Softwarekomponenten des Anwendungssystems aufrufbar sein sollten [OMG02:1-45].

Im Sinne dieser Unterscheidung handelt es sich (bei der Beispielkomponente aus Kapitel 3) bei Lagersteuerung um Parameterdaten und bei Lagerplätzen und Materialien um Stammdaten. Zu beachten ist, dass damit (in Übereinstimmung mit anderen Autoren [Gö97:101, SGR98:152, DFH00:535]) organisatorische Einheiten (wie z. B. Lager) ebenfalls zu den Parameterdaten gezählt werden.

Zum besseren Verständnis werden wichtige Eigenschaften von Parametern (bei betrieblichen Softwarekomponenten) mit Hilfe des Klassifikationsschemas in Abb. 3 dargestellt. Die im Schema enthaltenen Merkmale stammen aus der Literatur zur Parametrisierung [Pi93; Gö97; DMH99] sowie aus einer Analyse des Customizings von SAP R/3 [Ac02].

Merkmal	**Merkmalsausprägung**			
Datentyp	Boolesch	Numerisch	Alpha-numerisch	Zeichenartig
Wertebereich	Beliebig	Eingeschränkt	Festwerte	Parameterabhängig
Notwendigkeit	Optional	Bedingt optional	Obligatorisch	
Defaultwert	Ja		Nein	
Charakter	Identifizierend	Beschreibend	Referenzierend	Steuernd

Abbildung 3: Klassifikationsschema für Parameter

Das Klassifikationsschema in Abb. 3 enthält folgende Merkmale:

- Das Merkmal *Datentyp* erfasst qualitativ, welche Art von Datentyp der Parameter hat: Nach [DMH99:4] oder [Ac04:137] sind Parameter immer formatiert und treten in den Zeichenarten *boolesch*, *numerisch*, *alphanumerisch* und *zeichenartig* auf.
- Das Merkmal *Wertebereich* beschreibt die Freiheiten bei der Wahl von Parameterwerten [Ac02:28; Pi93:439]. Zu unterscheiden ist zwischen folgenden Ausprägungen: im Rahmen seines Datentyps nicht eingeschränkt (*Beliebig*), mit Einschränkungen versehener Standarddatentyp (*Eingeschränkt*), vom Hersteller fest vorgegeben (*Festwerte*) oder durch die Werte anderer Parameter vorgegeben (*Parameterabhängig*).
- Das Merkmal *Notwendigkeit* gibt an, ob ein Parameter mit einem Wert versehen werden muss [Ac02:28]. Mögliche Ausprägungen sind *Obligatorisch*, *Optional* und *Bedingt optional*. Letzteres bedeutet, dass der Parameter nur unter bestimmten Bedingungen (meist andere Parametereinstellungen) optional ist.
- Das Merkmal *Defaultwert* gibt an, ob für den Parameter eine Voreinstellung vorgesehen ist, die bei fehlender Wertzuweisung verwendet wird – ein initialer Wert ohne semantische Bedeutung wird dabei nicht als Defaultwert betrachtet.
- Das Merkmal *Charakter* unterscheidet Parameter nach ihrer Rolle bei der Anpassung (einzelne Merkmalsausprägungen gehen auf [Ac02:28f.], [Ac04:147] und [Gö97:102] zurück): Von zentraler Bedeutung sind *steuernde* Parameter, welche unmittelbar Datenstrukturen, Funktionsabarbeitung oder Prozessabläufe beeinflussen. So definiert z. B. der Parameter *Einlagerungsstrategie* die bei der

Einlagerung anzuwendende Strategie. Mit Hilfe von *identifizierenden* Parametern kann zwischen verschiedenen Parametrisierungsvarianten unterschieden werden – ein Beispiel dafür ist der Parameter *Id* der Parametergruppe *Lagersteuerung*. *Beschreibende* Parameter legen z. B. Mengeneinheiten oder Texte fest, haben aber keine steuernde Funktion – so erlaubt der Parameter *Name* der Parametergruppe *Lager* eine nähere Beschreibung eines Lagers. Ein Parameter einer Parametergruppe hat *referenzierenden* Charakter, wenn er auf eine Instanz einer anderen Parametergruppe verweist und damit Eigenschaften der ersten Gruppe festlegt. Der Parameter *Strategie* von *Lager* weist einem Lager eine Lagerstrategie zu, welche die konkreten Steuerungseigenschaften für dieses Lager festlegt.

Einlagerungsstrategie ist ein Beispiel für einen Parameter bei der Komponente *Lagermanagement*. Der Datentyp des Parameters ist zeichenartig und sein Wertebereich ist durch Festwerte vorgegeben (*manuell*, *statisch*, *dynamisch*). Die Pflege des Parameters ist obligatorisch, es gibt keinen Defaultwert und sein Charakter ist steuernd.

5.2 Parametergruppen

Die Parameter einer Komponente treten häufig nicht isoliert auf, sondern liegen in strukturierter Form vor – für solche Struktureinheiten wird ein eigener Begriff eingeführt: Eine *Parametergruppe* ist eine Struktur semantisch zusammengehörender Parameter, zu welcher mehrere Instanzen (Datensätze) angelegt werden können. Ein wesentlicher Aspekt dieser Definition ist die Forderung, dass eine Parametergruppe mehrere Instanzen enthalten und damit als Träger verschiedener Anpassungsvarianten dienen kann.

Ein Beispiel für eine Parametergruppe ist *Lagersteuerung*, welche z. B. die Parameter *Einlagerungsstrategie* und *Auslagerungsstrategie* enthält. Es ist möglich, verschiedene Instanzen von *Lagersteuerung* zu definieren, um unterschiedliche Lager in verschiedener Weise steuern zu können.

Welche Parameter zu einer Parametergruppe zusammengefasst werden, ist nicht immer eindeutig zu beantworten und liegt daher teilweise im Ermessensspielraum des Komponentenherstellers. Ein Kriterium dafür liegt vor, wenn die Instanzen einer Parametergruppe durch einen semantischen Schlüssel eindeutig identifiziert werden: Zur Parametergruppe können nur solche Parameter gehören, welche dem Entitätstyp zugeordnet sind, der durch den Schlüssel beschrieben wird. Werden durch den Schlüssel z. B. Lager identifiziert, dann können zur Parametergruppe nur Parameter gehören, die eine Steuerung auf Lagerebene erlauben und nicht solche, die eine Steuerung auf Lagerbereichsebene intendieren.

Merkmal	**Merkmalsausprägung**			
Bwl. Zweck	Organisatorische Einheiten	Strategische Steuerdaten	Administrative Steuerdaten	Benutzersteuesteuerung/Darstellung
Max. Anzahl Instanzen	Eine	Feste Anzahl	Parameterabhängig	Beliebig

Abbildung 4: Klassifikationsschema für Parametergruppen

Wichtige Eigenschaften von Parametergruppen finden sich in Abb. 4:

- Das Merkmal *Betriebswirtschaftlicher Zweck* beschreibt die Funktion der Parametergruppe auf betriebswirtschaftlicher Ebene und ihren Einfluss auf die Systemsteuerung. Dieses Merkmal wird nicht in eine formale Spezifikation eingehen, da insbesondere die Abgrenzung zwischen den Merkmalsausprägungen nicht eindeutig ist. Als Zusatzinformation für den menschlichen Leser ist dieses Merkmal jedoch eine sinnvolle Ergänzung. Bei diesem Merkmal werden folgende (nicht notwendig disjunkte) Merkmalsausprägungen unterschieden [Gö97:101f.; Ac02:26]:
  - Bei der Parametrisierung werden *organisatorische Einheiten* (z. B. Werke, Lager) definiert, welche einerseits Datenhaltungsebenen sind [DFH00:535] und andererseits häufig mittelbaren Einfluss auf die Ablaufsteuerung haben, indem sie die Träger verschiedener Ablaufvarianten sind.
  - Steuerdaten steuern den Ablauf von Prozessen und Geschäftsvorfällen. Dabei definieren *strategische Steuerdaten* abstrakte Regeln und Strukturen für Prozessabläufe. *Administrative Steuerdaten* steuern konkrete Prozessabläufe und sind oft Ausprägungen der strategischen Steuerdaten.
  - Unter *Benutzersteuerung/Darstellung* werden Parametergruppen zusammengefasst, welche Darstellung und Bildschirmdetails sowie Benutzerbesonderheiten steuern, aber keinen Einfluss auf den Ablauf von Prozessen haben.

- Das Merkmal *Maximale Anzahl Instanzen* beschreibt, wie viele Datensätze für die Parametergruppe maximal definiert werden können [Ac02:27]. *Feste Anzahl* heißt, die maximale Anzahl ist vom Komponentenhersteller (z. B. durch Festwerte) vorgegeben. *Parameterabhängig* heißt, die maximal erlaubte Anzahl wird durch Parameterwerte oder durch die Anzahl von Instanzen anderer Parametergruppen festgelegt.

Lagersteuerung ist ein Beispiel für eine Parametergruppe bei der Beispielkomponente *Lagermanagement* – diese enthält z. B. den Parameter *Einlagerungsstrategie*. Diese Parametergruppe gehört zu den administrativen Steuerdaten, welche die Lagerprozesse steuern. Die Komponente ermöglicht, beliebig viele Instanzen von *Lagersteuerung* zu definieren.

5.3 Parameter-Metamodell

In diesem Abschnitt wird ein Parameter-Metamodell vorgestellt, welches eine Weiterentwicklung des Modells aus [Ac04:139] ist. Ein solches Metamodell erlaubt, die zuvor definierten Begriffe rund um Parameter weiter zu präzisieren, indem sie strukturiert und mit ihren Abhängigkeiten untereinander dargestellt werden. Bei der Erstellung des Metamodells wurde die Struktur von Parametern in Datenbanktabellen, in einfachen Initialisierungsdateien sowie in XML-Konfigurationsdateien (Property-Files beim OMG CCM [OMG02:1–47], Deployment-Descriptor bei Suns EJB [Mo00:234], Datei Web.config bei Microsofts .NET [Pr02:377]) berücksichtigt. Im Ergebnis entstand ein Metamodell, welches für alle Spielarten der datenbasierten Parametrisierung gültig ist. Zu beachten ist dabei, dass das Metamodell in dem Sinne ein Maximalmodell ist, dass nicht alle dargestellten Konzepte in allen Spielarten der Parametrisierung vorkommen müssen.

Das Metamodell wird als UML-Klassendiagramm [OMG05:21] dargestellt und findet sich in Abb. 5. Im Metamodell werden Typ- und Instanzebene unterschieden. Die Elemente der Typebene werden vom Komponentenhersteller vorgegeben und umfassen die verfügbaren Parameter sowie deren Strukturierung in Parametergruppen:

- Zentrales Element des Metamodells ist der Parameter, worunter ein Datenfeld verstanden wird, welches bei der Parametrisierung zum Einsatz kommt. Ein Parameter wird durch seinen Namen sowie seine Eigenschaften Datentyp, Wertebereich, Notwendigkeit und Defaultwert näher beschrieben (vgl. dazu Abb. 3). Bei datenbankbasierter Ablage entspricht ein Parameter einer Spalte einer Tabelle – bei XML-Konfigurationsdateien handelt es sich bei den Parametern um die Elemente (Knoten unterster Ebene) oder die Attribute.
- Parametergruppen werden im Metamodell durch den Typ *Parametergruppe* repräsentiert. Eine Parametergruppe wird durch ihren Namen näher beschrieben und kann beliebig viele Parameter umfassen. Bei datenbankbasierter Ablage entspricht eine Parametergruppe einer Tabelle. Bei XML-Konfigurationsdateien bilden alle die Knoten eine Parametergruppe, die andere Knoten oder Elemente enthalten. Das Metamodell erlaubt auch Parameter ohne Parametergruppe. Während dies bei datenbankbasierter Ablage unüblich ist, tritt dieser Fall bei

Dateien dann auf, wenn ein Element auf höchster Ebene definiert wird. Dieser Fall kommt z. B. bei einfachen Initialisierungsdateien häufig vor.

– Parametergruppen können hierarchisch angeordnet sein. Dies wird im Metamodell durch die reflexive Beziehung bei *Parametergruppe* beschrieben. Hierarchien von Parametergruppen entstehen insbesondere bei XML-Konfigurationsdateien dann, wenn mehrstufige Knotenhierarchien verwendet werden.

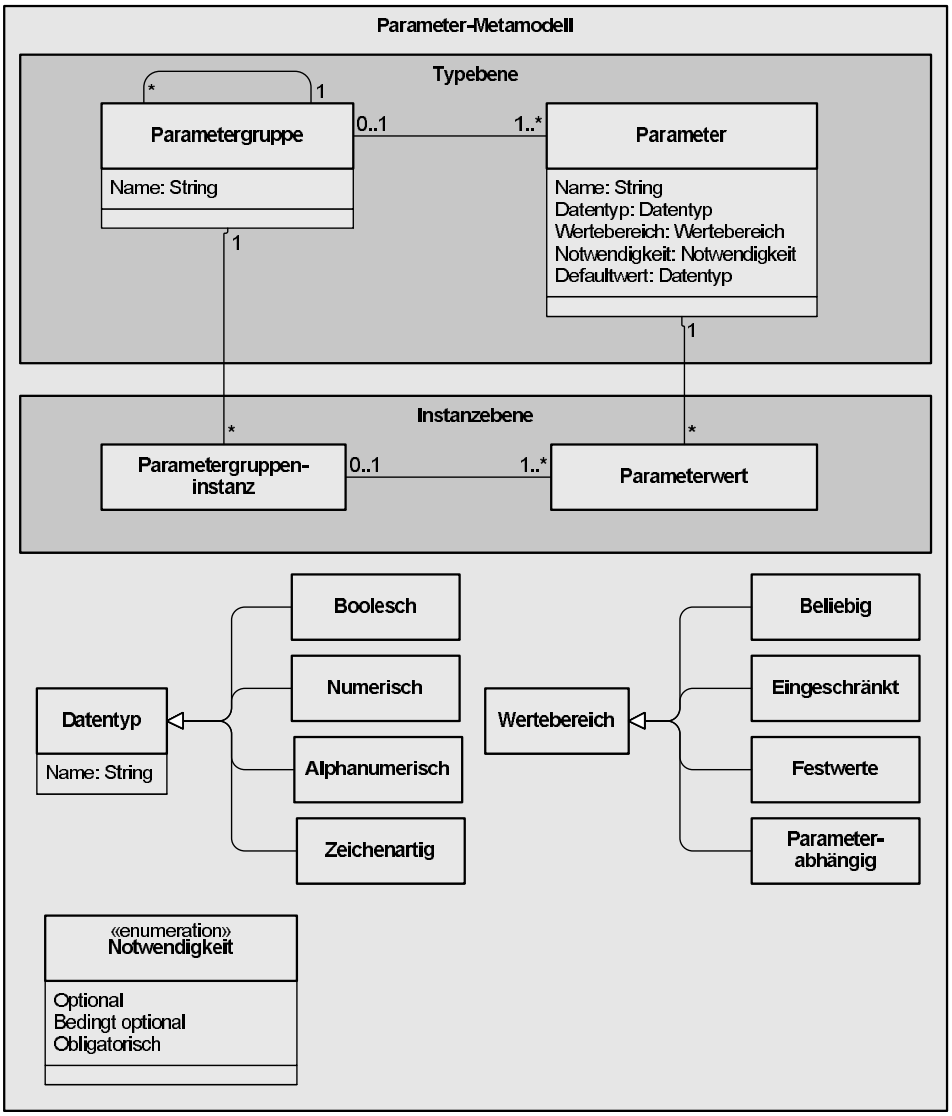

Abbildung 5: Metamodell zur Beschreibung von Parametern

Auf der Instanzebene finden sich die Daten, welche ein Komponentenanwender den Parametern zuweist. In Entsprechung zur Typebene werden dabei unterschieden:

- *Parameterwert* ist ein einem Parameter zugewiesener Wert, d. h. der Inhalt eines Datenbankfeldes bzw. der Inhalt eines Elements oder Attributs in einer XML-Konfigurationsdatei. Ein Parameterwert bezieht sich immer auf einen Parameter und gehört zu einer oder keiner Parametergruppeninstanz. Parameterwerte müssen den Restriktionen entsprechen, die durch Datentyp und Wertebereich ihres Parameters vorgegeben werden.
- Eine Parametergruppe kann mit beliebig vielen Ausprägungen versehen werden, was durch die *Parametergruppeninstanz* repräsentiert wird. Eine Parametergruppeninstanz (auf unterster Hierarchiestufe) ist dabei ein Tuple von Werten, welches aus genau einem Parameterwert für jeden Parameter der Parametergruppe besteht. Bei datenbankbasierter Ablage entspricht dies gerade einem Datensatz in der Tabelle. In XML-Konfigurationsdateien handelt es sich um das Vorkommen eines Knotens inklusive seiner enthaltenen Attribute, Knoten, und Elemente. Das Anlegen mehrerer Instanzen zu einer Parametergruppe ermöglicht die Definition verschiedener Parametrisierungsvarianten, die z. B. für unterschiedliche organisatorische Einheiten gelten können.

Ein Parameter bei der Beispielkomponente *Lagermanagement* ist die für ein Lager zu definierende *Einlagerungsstrategie*, welche den Festwert *dynamisch* als möglichen Parameterwert hat. Dieser Parameter ist Teil der Parametergruppe *Lagersteuerung*. Eine Parametergruppeninstanz ist z. B. eine Lagersteuerungsinstanz mit der Id *02*, für welche als Einlagerungsstrategie *dynamisch* und als Auslagerungsstrategie *First-In-First-Out (FIFO)* festgelegt wird.

6 Parametrisierungsaufgaben

6.1 Definition und Beschreibung

In Abb. 2 sind der Funktionssicht bei den Parametrisierungsoptionen solche Gegenstände zuzuordnen, mit deren Hilfe das Erfassen von Parameterwerten durch einen Komponentenanwender ermöglicht wird. Auf technischer Ebene sind das z. B. spezielle Schnittstellen zur Pflege von Parameterwerten. Auf konzeptioneller Ebene gibt es dafür bisher keinen eindeutigen Begriff – in der Literatur wird meist nur der gesamte Prozess der Anpassung bezeichnet (z. B. als Customizing [Gö97:101; DMH99:1; SGR98:152], Parametr(is)ierung [AR00:64; Me05:165], Adaption [Hu94]) oder allgemein von Parametereinstellungen [Me05:167] gesprochen.

Um die Funktionen zur Parametererfassung auf konzeptioneller Ebene zu repräsentieren, wird – analog zum Begriff der betrieblichen Aufgabe – der Begriff der *Parametrisierungsaufgabe* eingeführt:

- Eine *elementare Parametrisierungsaufgabe* repräsentiert eine Handlung, bei der eine Menge semantisch zusammengehörender Parameter mit einem eindeutig abgegrenzten, nicht mehr unterteilbaren Parametrisierungsziel, eingestellt werden.
- Eine *zusammengesetzte Parametrisierungsaufgabe* beschreibt ein übergeordnetes Parametrisierungsziel und ist dazu aus einer Menge anderer Parametrisierungsaufgaben zusammengesetzt.

Ein wesentlicher Aspekt bei diesen Definitionen ist die Unterscheidung in elementare und zusammengesetzte Parametrisierungsaufgaben: Um ein Parametrisierungsziel einfach beschreiben zu können und Abhängigkeiten zwischen Parametrisierungszielen klar erfassen zu können, müssen Parametrisierungsaufgaben so feingranular wie möglich sein – dafür gibt es die elementaren Parametrisierungsaufgaben. Zusammengesetzte Parametrisierungsaufgaben dagegen ermöglichen eine Strukturierung von elementaren und anderen zusammengesetzten Parametrisierungsaufgaben, die einem übergeordneten Parametrisierungsziel dienen.

Für Beispiele wird wieder auf die Komponente *Lagermanagement* zurückgegriffen: Die bei der betrieblichen Aufgabe *Einlagerung* einzusetzende Strategie wird durch den Parameter *Einlagerungsstrategie* der Parametergruppe *Lagersteuerung* festgelegt – die Pflege dieses Parameters erfolgt mit Hilfe der elementaren Parametrisierungsaufgabe *Einlagerungsstrategie festlegen*. Analog dazu bestimmt die Parametrisierungsaufgabe *Auslagerungsstrategie festlegen* die bei der Auslagerung einzusetzende Strategie. Eine Zusammenfassung der Parameter *Ein-* und *Auslagerungsstrategie* zu einer elementaren Parametrisierungsaufgabe ist nicht sinnvoll, da beide Entscheidungen unabhängig voneinander sind und verschiedene betriebliche Aufgaben betreffen. Die o. g. Parametrisierungsaufgaben lassen sich mit der Aufgabe *Lagersteuerung definieren* zur zusammengesetzten Parametrisierungsaufgabe *Lagersteuerung einrichten* zusammenfassen.

Durch das Konzept der elementaren Parametrisierungsaufgabe erfolgt eine andere Sichtweise auf die in der Datensicht eingeführten Parameter und Parametergruppen. Eine Parametrisierungsaufgabe kann zur Pflege eines einzelnen oder mehrerer Parameter dienen und erzeugt dadurch eine Gruppierung von Parametern. Diese zusammen zu pflegenden Parameter können zu verschiedenen Parametergruppen gehören und müssen auch nicht alle Parameter einer Parametergruppe umfassen. Es ist außerdem denkbar, dass ein Parameter von verschiedenen Parametrisierungsaufgaben eingestellt wird. Es besteht also keine eineindeutige Zuordnung einer Parametrisierungsaufgabe zu einem einzelnen Parameter oder zu einer Parametergruppe. Dies ist dadurch begründet, dass die Zusammenfassung von Parametern zu Parametergruppen strukturorientiert und zu Parametrisierungsaufgaben zielorientiert (nach Wirkungsweise) erfolgt.

Merkmal	**Merkmalsausprägung**			
Parametrisierungskontext	Aufbauorganisation	Daten	Funktionen	Abläufe (Prozesse)
Notwendigkeit	Optional	Bedingt optional	Obligatorisch	
Abhängigkeiten	Keine	Komponentenlokal	Komponentenübergreifend	

Abbildung 6: Klassifikationsschema für Parametrisierungsaufgaben

Wichtige Eigenschaften von Parametrisierungsaufgaben werden wieder in einem Klassifikationsschema aufgeführt (vgl. Abb. 6). Das Schema enthält folgende Merkmale:

- Das Merkmal *Parametrisierungskontext* beschreibt, welche Aspekte eines Anwendungssystems durch eine Parametrisierungsaufgabe angepasst werden (Merkmalsausprägungen *Aufbauorganisation*, *Daten*, *Funktionen* und *Abläufe*) [AT06:352].
- Das Merkmal *Notwendigkeit* gibt an, ob eine Parametrisierungsaufgabe ausgeführt werden muss oder nicht. Obligatorische Aufgaben müssen in jedem Fall vor Einsatz der Komponente durchgeführt werden – optionale dagegen nicht unbedingt. *Bedingt optional* bedeutet, dass die Aufgabe unter bestimmten Voraussetzungen (häufig aufgrund anderer Parametereinstellungen) obligatorisch wird.
- Das Merkmal *Abhängigkeiten* erfasst, ob zur Durchführung der Aufgabe die vorhergehende Ausführung anderer Parametrisierungsaufgaben notwendig ist [Ac02:26]. Bei bestehenden Abhängigkeiten wird dabei unterschieden, ob alle vorausgesetzten Parametrisierungsaufgaben zur selben Komponente gehören (*komponentenlokal*) oder teilweise anderen Komponenten zugeordnet sind (*komponentenübergreifend*).

Zur Erläuterung der Klassifikation wird die Parametrisierungsaufgabe *Einlagerungsstrategie festlegen* betrachtet: Anhand dieser Aufgabe wird festgelegt, nach welchem Verfahren die betriebliche Aufgabe der Einlagerung ausgeführt wird – Funktionen sind daher der Parametrisierungskontext. Die Aufgabe ist obligatorisch, da Einlagerung immer eingesetzt wird und damit immer einzustellen ist. Um die Aufgabe zu bearbeiten, muss zunächst die Parametrisierungsaufgabe *Lagersteuerung definieren* ausgeführt werden – es bestehen also komponentenlokale Abhängigkeiten.

6.2 Reihenfolgebeziehungen

In der Prozesssicht von Parametrisierungsoptionen werden Abläufe und Reihenfolgebeziehungen von Funktionen (bzw. Aufgaben) beschrieben. Die dafür relevanten Gegenstände der Parametrisierung zur Anpassungszeit sind einzuhaltende Abläufe und bestehende Reihenfolgerestriktionen bei der Abarbeitung von Parametrisierungsaufgaben. Ein Beispiel dafür findet sich bei der Beispielkomponente: Die Parametrisierungsaufgabe *Lagersteuerung definieren* ist Voraussetzung für die Aufgaben *Einlagerungsstrategie festlegen* und *Auslagerungsstrategie festlegen.*

Im Vergleich zu Geschäftsprozessen weisen die Abläufe zur Parametrisierung jedoch einen geringeren Umfang und eine geringere Komplexität auf. Dies erkennt man z. B. daran, dass bei SAP R/3 zwar die Geschäftsprozesse mittels Ereignisgesteuerter Prozessketten (EPK) modelliert werden, die Abläufe zur Parametrisierung jedoch nicht. Eine explizite Modellierung der Reihenfolgebeziehungen zwischen Parametrisierungsaufgaben in Form von Prozessen erscheint – unter Berücksichtigung des Grundsatzes der Wirtschaftlichkeit der Grundsätze ordnungsmäßiger Modellierung [BRS95:434] – nicht sinnvoll. Aus diesem Grund wird an dieser Stelle auch auf die explizite Einführung und Erläuterung eines eigenen Begriffs (wie z. B. *Parametrisierungsprozess*) verzichtet. Aber auch ohne eine Modellierung in Form von Prozessen bleibt zu betonen, dass Reihenfolgebeziehungen zwischen Parametrisierungsaufgaben bestehen und dass diese bei der Spezifikation einer betrieblichen Softwarekomponente zu berücksichtigen sind.

7 Parametrisierungsauswirkungen

Das Ziel von Parametrisierung ist, Struktur und Verhalten einer Softwarekomponente an die individuellen Anforderungen des Komponentenanwenders anzupassen. Die zur Anpassungszeit gesetzten Parameterwerte beeinflussen dabei zur Laufzeit die Datenstruktur sowie die Eigenschaften der ausgeführten Funktionen und Prozesse. Alle solche Änderungen werden unter dem Begriff *Parametrisierungsauswirkungen* zusammengefasst (vgl. Kapitel 4). Beispielsweise entscheidet der Parameter *Einlagerungsstrategie* (der Komponente *Lagermanagement*), mit welcher Strategie die betriebliche Aufgabe der Einlagerung ausgeführt wird (Auswirkung auf Funktionssicht).

Wirkungsart	Beschreibung der Wirkungsart
Varianten-bestimmend	Identifiziert eine anzuwendende Parametrisierungsvariante
Definierend	Definiert alle erlaubten Ausprägungen
Auswählend	Wählt aus vorgegebenen Ausprägungen die erlaubten aus
Einschränkend	Schränkt erlaubte Ausprägungen ein
Entscheidend	Trifft Entscheidungen bei Prozessablauf oder Funktionsabarbeitung
Entscheidungs-unterstützend	Unterstützt Entscheidungsfindung bei Prozessablauf oder Funktionsabarbeitung
Berechnend	Geht als Variable in eine Berechnung ein – beeinflusst aber nicht das Berechnungsverfahren

Abbildung 7: Wirkungsarten zur Klassifikation von Parametrisierungsauswirkungen

Um die konkreten Auswirkungen von Parametern zu erfassen und explizit darzustellen, kann man sich (in Anlehnung an [Ac04:141]) einer tabellarischen Darstellung bedienen. Dabei sind in jeder Tabellenzeile der Parametername, ein spezifikationsrelevanter Sachverhalt und dessen Kontext, eine textuelle Beschreibung und eine Wirkungsart anzugeben. Dadurch wird sowohl erfasst, worauf (Sachverhalt, Kontext) und wie (Beschreibung, Wirkungsart) sich der Parameter auswirkt. Bei den spezifikationsrelevanten Sachverhalten, auf die sich ein Parameter auswirkt, könnte es sich z. B. um eine Vorbedingung oder um ein Qualitätsprofil handeln. Zusätzlich wird der Kontext angegeben, zu dem dieser Sachverhalt gehört – beispielsweise könnte sich die Vorbedingung auf die Methode *ILagermanagement.Einlagern* oder das Qualitätsprofil auf die Schnittstelle *ILagermanagement* beziehen.

Um detailliert zu erfassen, wie ein Parameter auf den betroffenen spezifikationsrelevanten Sachverhalt wirkt, wird eine textuelle Beschreibung erstellt. Neben der ausführlichen textuellen Beschreibung ist eine Klassifikation der Auswirkungen wünschenswert, da eine solche die Übersichtlichkeit und Verständlichkeit erhöht. Dafür lässt sich das Konzept der *Wirkungsart* einsetzen. Die *Wirkungsart* klassifiziert, auf welche Art sich ein Parameter auf die Funktionalität einer Komponente auswirkt. Mögliche Ausprägungen für die Wirkungsart inklusive einer erklärenden Beschreibung werden in Abb. 7 aufgeführt. (Teile dieser Klassifikation gehen auf [DMH99:4] zurück.) Man beachte, dass in

Abb. 7 der Begriff *Ausprägung* für die Instanzen der jeweils betroffenen Sachverhalte steht.

Merkmal	**Merkmalsausprägung**						
Auswirkungen auf	Datensicht		Funktionssicht		Prozesssicht		
Charakter	Identifizierend	Beschreibend	Referenzierend	Steuernd			
Reichweite der Auswirkungen	Komponentenlokal		Komponentenübergreifend				
Wirkungsart	variantenbestimmend	definierend	auswählend	einschränkend	entscheidend	entscheidungsunterstützend	berechnend

Abbildung 8: Klassifikation von Parametern nach ihren Auswirkungen

Abschließend fasst Abb. 8 nochmal solche Eigenschaften von Parametern zusammen, die beschreiben, auf welche Art ein Parameter Struktur und Verhalten einer Komponente beeinflusst. Das Merkmal *Auswirkungen auf* gibt an, welche Sichten einer Komponente durch Wahl eines Parameterwertes beeinflusst werden (vgl. Abb. 2). Das Merkmal *Charakter* unterscheidet Parameter nach ihrer Rolle bei der Anpassung (vgl. Abb. 3). Das Merkmal *Reichweite der Auswirkungen* beschreibt, ob die durch einen Parameter erzeugten Auswirkungen nur die eigene Komponente betreffen oder aber sich über Komponentengrenzen hinweg erstrecken. Das Merkmal *Wirkungsart* klassifiziert, auf welche Art sich Parameter auf die Funktionalität der Komponente auswirken (vgl. Abb. 7).

8 Zusammenfassung und Ausblick

Diese Arbeit beschäftigte sich mit den Grundlagen parametrisierbarer betrieblicher Softwarekomponenten. Dazu wurden Parameter, Parametergruppen, Parametrisierungsaufgaben und -auswirkungen als die zentralen Gegenstände der Parametrisierung bestimmt, präzise definiert und deren Eigenschaften diskutiert. Außerdem wurde ein Parameter-Metamodell vorgestellt. Als Ergebnis entstand ein besseres und systematisches Verständnis der mit Parametrisierung zusammenhängenden Konzepte betrieblicher Softwarekomponenten. Interessant für die Zukunft wäre zu untersuchen, ob sich die

Ergebnisse auf nicht-komponentenorientierte betriebliche Anwendungssysteme verallgemeinern lassen und welche Rolle Parametrisierung zur Anpassung von Services spielt.

9 Literaturverzeichnis

[Ac02] Ackermann, J.: Spezifikation des Parametrisierungsspielraums von Fachkomponenten– Erste Überlegungen. In: K. Turowski (Hrsg.): 3. Workshop Modellierung und Spezifikation von Fachkomponenten. Nürnberg 2002, S. 17-68.

[Ac04] Ackermann, J.: Zur Beschreibung datenbasierter Parametrisierung von Softwarekomponenten. In: Turowski, K. (Hrsg.): Architekturen, Komponenten, Anwendungen – Proceedings zur 1. Verbundtagung Architekturen, Komponenten, Anwendungen (AKA 2004). LNI Bd. P-57. Augsburg 2004, S. 131-149.

[AR00] Appelrath, H.-J.; Ritter, J.: R/3-Einführung: Methoden und Werkzeuge. Springer, Berlin, Heidelberg 2000.

[AT06] Ackermann, J.; Turowski, K.: Zur Rolle von Parametrisierung bei der fachlichen Anpassung betrieblicher Softwarekomponenten. In: Schelp, J. et al. (Hrsg.): Integration, Informationslogistik und Architektur – Proceedings zur DW2006. LNI Band P-90. Friedrichshafen 2006, S. 341-359.

[AT07] Ackermann, J.; Turowski, K.: On the Specification of Parameterizable Business Components. In: Draheim, D.; Weber, G. (Hrsg.): Trends in Enterprise Application Architecture (TEAA 2006). Springer-Verlag LNCS 4473. Berlin 2007, S. 25-39.

[Bo97] Bosch, J.: Adapting Object-Oriented Components. In: Proceedings of the 2nd International Workshop on Component-Oriented Programming (WCOP'97). Turku, Finland, 1997.

[BRS95] Becker, J.; Rosemann, M.; Schütte, R.: Grundsätze ordnungsmäßiger Modellierung. In: Wirtschaftsinformatik 37 (1995) 5, S. 435-445.

[BS04] Becker, J.; Schütte, R.: Handelsinformationssysteme. Domänenorientierte Einführung in die Wirtschaftsinformatik. 2. Aufl. Redline Wirtschaft, Frankfurt 2004.

[CE00] Czarnecki, K.; Eisenecker, U. W.: Generative Programming: Methods, Tools, and Applications. Addison-Wesley, Boston 2000.

[DFH00] Disterer, G.; Fels, F.; Hausotter, A.: Taschenbuch der Wirtschaftsinformatik. Fachbuchverlag Leipzig. Leipzig 2000.

[DMH99] Dittrich, J.; Mertens, P.; Hau, M.: Dispositionsparameter von SAP R/3-PP : Einstellungshinweise, Wirkungen, Nebenwirkungen. Vieweg, Wiesbaden 1999.

[DW98] D'Souza, D.F.; Wills, A.C.: Objects, Components, and Frameworks with UML: The Catalysis Approach. Addison-Wesley. Reading 1998.

[Gö97] Görk, M.: Customizing. In: P. Mertens (Hrsg.): Lexikon der Wirtschaftsinformatik. 3. Auflage. Springer-Verlag, Berlin Heidelberg 1997, S. 101-102.

[Gr98] Griffel, F.: Componentware. Konzepte und Techniken eines Softwareparadigmas. dpunkt Verlag. Heidelberg 1998.

[Hu94] Hufgard, A.: Adaption betriebswirtschaftlicher Softwarebibliotheken. Dissertation. Würzburg 1994.

[JGJ97] Jacobson, I.; Griss, M.; Jonsson, P.: Software Reuse. ACM Press /Addison Wesley Longman. New York, 1997.

[Me05] Mertens, P.; Bodendorf, F.; König, W.; Picot, A.; Schumann, M.; Hess, X.: Grundzüge der Wirtschaftsinformatik. 9. Aufl. Springer-Verlag. Berlin/Heidelberg 2005.

[Mo00] Monson-Haefel, R.: Enterprise JavaBeans™. 2nd Edition. O'Reilly & Associates. Sebastopol, California, 2000.

[OMG02] OMG (Hrsg.): CORBA Components Specification. Version 3.0, June 2002. URL: http://www.omg.org/technology/documents, Abruf am 2007-09-19.

[OMG05] OMG (Hrsg.): Unified Modeling Language: Superstructure. Version 2.0, formal/05-07-04. URL: http://www.omg.org/technology/documents, Abruf am 2007-02-02.

[Ov06] Overhage, S.: Vereinheitlichte Spezifikation von Komponenten – Grundlagen, UnSCom Spezifikationsrahmen und Anwendung. Dissertation. Augsburg 2006.

[Pi93] Pietsch, M.: PAREUS-RM – ein Tool zur Unterstützung der Konfiguration von PPS-Parametern im SAP-System R/2. In: Wirtschaftsinformatik 35 (1993) 5, S. 434-445.

[Pr02] Prosise, J.: Microsoft .NET – Das Entwicklerbuch. Microsoft Press, 2002.

[Re01] Reussner, R.: Parametrisierte Verträge zur Protokolladaption bei Software-Komponenten. Logos Verlag, Berlin 2001.

[Ri00] Ritter, J.: Prozessorientierte Konfiguration komponentenbasierter Anwendungssysteme. Dissertation. Oldenburg 2000

[Ro99] Ropohl, G.: Allgemeine Technologie. Hanser Verlag. München/Wien 1999.

[SAP02] SAP (Hrsg.): SAP Implementation Guide (IMG). In: Online-Dokumentation für SAP R/3 enterprise, Release 4.70. Walldorf 2002.

[SAP97] SAP (Hrsg.): R/3-Referenz(prozeß)modell 4.0 im R/3 Business Engineer – Zielsetzung, Inhalte, Vorgehensweise. Walldorf 1997.

[SB00] Svahnberg, M.; Bosch, J.: Issues Concerning Variability in Software Product Lines. In: van der Linden, F. (Hrsg.): Software Architectures for Product Families. Proceedings of the Third International Workshop on Software Architectures for Product Families. Springer LNCS 1951. Las Palmas de Gran Canaria 2000, S. 146-157.

[Sc98] Schütte, R.: Grundsätze ordnungsmäßiger Referenzmodellierung. Gabler Verlag. Wiesbaden 1998.

[SC98] Stiemerling, O.; Cremers, A. B.: Tailorable Component Architectures for CSCW-Systems. In: Proceedings of the 6th Euromicro Workshop on Parallel and Distributed Programming. IEEE Press, Madrid 1998, S. 302-308.

[SGR98] Stickel, E.; Groffmann, H.-D.; Rau, K.-H.: Gabler Wirtschaftsinformatiklexikon. Gabler Verlag, Wiesbaden 1998.

[Sz02] Szyperski, C.; Gruntz, D.; Murer, S.: Component Software: Beyond Object-Oriented Programming. 2. Aufl. Addison-Wesley. Harlow 2002.

[Tu03] Turowski, K.: Fachkomponenten: Komponentenbasierte betriebliche Anwendungssysteme. Shaker Verlag,.Aachen 2003.