Zur Rolle von Parametrisierung bei der fachlichen Anpassung betrieblicher 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

Abstract: Komponentenbasierte betriebliche Anwendungssysteme bieten den Vorteil, dass ein Teil der notwendigen Flexibilität durch Auswahl und Austausch von Komponenten erreicht werden kann. Allerdings ist es weder möglich noch effizient, die gesamte notwendige Variabilität allein durch Komponentenaustausch zu realisieren. Eine Alternative dazu besteht vor allem bei kleineren, fachlichen Anpassungen im Einsatz von Parametrisierung. Diese Arbeit beschäftigt sich mit der Rolle, die Parametrisierung zur fachlichen Anpassung in komponentenbasierten betrieblichen Anwendungssystemen spielen kann, und untersucht insbesondere, in welchen Anpassungssituationen ihr Einsatz geeignet ist.

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 [Tu01, 1] und nur schwer an veränderte Anforderungen anzupassen sind [Sz98, 5].

Komponentenbasierte betriebliche Anwendungssysteme haben demgegenüber den Vorteil, dass ein Teil der notwendigen Flexibilität durch Auswahl und Austausch von Komponenten erreicht werden kann. Allerdings kann Komponentenaustausch allein nicht die gesamte notwendige Variabilität zur Verfügung stellen und ist bei häufigen, kleineren Änderungen auch nicht effizient, weil ein Komponentenaustausch mit einem nicht unerheblichen Aufwand verbunden sein kann. Eine Alternative dazu besteht vor allem bei kleineren, fachlichen Anpassungen in der (datenbasierten) Parametrisierung der eingesetzten Komponenten.

Parametrisierung wird in der Literatur zur Anpassung von Softwarekomponenten zwar häufig genannt, aber kaum vertiefend behandelt – nähere Betrachtungen insbesondere zur Eignung von Parametrisierung liegen nicht vor. Parametrisierung wird auf der anderen Seite bei (nicht-komponentenbasierter) betriebswirtschaftlicher Standardsoftware wie SAP R/3 so extensiv eingesetzt, dass aufgrund der großen Komplexität die Auswahl optimaler Parameterwerte in der Praxis fast unmöglich ist [MWH91].

Diese Arbeit thematisiert die Parametrisierung zur fachlichen Anpassung in komponentenbasierten betrieblichen Anwendungssystemen und untersucht insbesondere, in welchen Anpassungssituationen ihr Einsatz geeignet ist. Dazu gliedert sich die Arbeit in zwei Teile. Der erste Teil widmet sich den Grundlagen der Parametrisierung: Kapitel 2 gibt einen Überblick über gängige Anpassungsmechanismen bei Komponenten und stellt verschiedene Spielarten der Parametrisierung vor. In Kapitel 3 erfolgt eine Definition der zentralen Begriffe *Parameter* und *Parametrisierung*. Danach wird ein für alle Spielarten der datenbasierten Parametrisierung gültiges Parameter-Metamodell vorgestellt (Kapitel 4) und anhand einer Beispielkomponente illustriert (Kapitel 5). Der zweite Teil der Arbeit beschäftigt sich mit der Eignung von Parametrisierung zur Anpassung von Komponenten: Nach einer Diskussion von Vor- und Nachteilen der Parametrisierung (Kapitel 6) wird ein Klassifikationsschema für Parametrisierungseinstellungen vorgestellt, welches eine Entscheidung für oder gegen Parametrisierung in einer Anpassungssituation unterstützen kann (Kapitel 7). Die Arbeit schließt mit dem zusammenfassenden Kapitel 8. Die originären Beiträge dieser Arbeit liegen im Erarbeiten der Definitionen, im Zusammenstellen der Vor- und Nachteile sowie im Entwickeln des Klassifikationsschemas zur Bewertung des Einsatzes von Parametrisierung.

2 Parametrisierung zur Anpassung von Softwarekomponenten

Anpassung spielt eine wichtige Rolle in komponentenbasierten Anwendungssystemen, da sich in der Praxis gezeigt hat, dass Komponenten nur selten ohne Anpassung wieder verwendet werden können [Bo97] – sie wird entsprechend von vielen Autoren diskutiert, vgl. z.B. [Bo97; SC98; Bo00; Be00b; Re01; Tu01; BN03; KM05]. Man kann dabei zwischen zwei generellen Anpassungsstrategien unterscheiden: zum einen die Anpassung der Komponentenarchitektur sowie der Auswahl und des Zusammenspiels der einzelnen Komponenten, zum anderen die Anpassung der Komponenten selbst. Gängige Anpassungsmechanismen sind: Kopieren von Code, Vererbung, Aggregation, Verwendung von Wrapperkomponenten, Superimposition [Bo97], Adapterinterfaces [Be00b], parametrisierte Verträge [Re01] und Parametrisierung – für einen Vergleich siehe z.B. [Bo97] oder [Re01].

Bei der Anpassung einzelner Komponenten kann man zwischen geplanter und ungeplanter Anpassung unterscheiden [Ac04]. *Geplante Anpassung* bedeutet dabei, dass die Anpassungsmöglichkeiten vom Komponentenhersteller vorgesehen werden. Bei betrieblichen Fachkomponenten wird darüber hinaus zwischen technischer und fachlicher Anpassung unterschieden [Tu99]. Die *technische Anpassung* dient dazu, implementierungsbedingte, technische Inkompatibilität 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 3). Für die Parameter wird dabei von einer Datenfeld-

pragmatik 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]. 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] oder Deployment Descriptors bei Suns Enterprise Java Beans (EJB) [Mo00]) verwendet werden. Es bleibt zu erwähnen, dass datenbasierte Parametrisierung – außerhalb komponentenbasierter Systeme – vielfach in kommerziell vertriebener Software (wie z.B. betriebswirtschaftliche Standardsoftware, Office-Produkten oder Datenbank-Managementsystemen) zum Einsatz kommt.

In einem weiteren Sinne kann man auch Anpassungsverfahren für Softwarekomponenten zur Parametrisierung zählen, bei denen ausführbare Parameter verwendet werden. Dabei wird für einen vom Hersteller vordefinierten Parameter (auch Plug-Point, Hot Spot oder User Exit genannt) z.B. eine ganze Komponente, ein Programm oder ein Workflowschema erwartet. Allen diesen Verfahren ist gemeinsam, dass für einen Parameter ein – im weitesten Sinne – ausführbares Programm erwartet wird. Diese Verfahren können deshalb auch unter dem Begriff *programmbasierte Parametrisierung* zusammengefasst werden [Ac04]. Diese Arbeit beschäftigt sich ausschließlich mit datenbasierter Parametrisierung – diese ist im Folgenden immer gemeint, wenn von *Parametrisierung* gesprochen wird.

Der Begriff *Parametrisierung* wird nicht nur bei der Anpassung von Softwarekomponenten verwendet und bezeichnet dabei zum Teil ähnliche, zum Teil unterschiedliche Konzepte: In der Wirtschaftsinformatik wird *Parametrisierung* häufig synonym zu *Customizing* verwendet, worunter die Anpassung einer Standardsoftware an die Anforderungen eines Kunden verstanden wird [Gö97]. 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 Definitionen und Begriffsklärung

Es gibt derzeit keine genaue und einheitliche Definition der Begriffe *Parameter* und *(datenbasierte) Parametrisierung* für die Anpassung von Softwarekomponenten. Ausgehend von bestehender Literatur werden deshalb in diesem Abschnitt die im Weiteren verwendeten Begriffe definiert.

In der Literatur zur Einführung standardisierter betrieblicher Anwendungssysteme versteht man unter Parametrisierung (bzw. Customizing), dass „zur *Definitionszeit* Parameter auf ausgewählte Werte gesetzt werden, die dann zur Ausführungszeit (*Laufzeit*) das Verhalten der Software bestimmen“ [AR00] (Ähnliche Definitionen finden sich z.B. in

[Hu94; Gö97; Ri00].) Unter einem Parameter kann man allgemein „eine unbestimmte Konstante in einer Funktion, einem Rechenverfahren oder einem Programm [verstehen], die erst während der Abarbeitung konkretisiert wird“ [SGR98]. Eine Definition des Begriffs *Parameter* im Kontext betrieblicher Anwendungssysteme findet sich in [Ri00]: „Parameter sind Datenfelder, die helfen, das Verhalten des Systems zur Ausführungszeit (*Laufzeit*) entsprechend der Intention des Anwenders anzupassen.“ Bei Arbeiten zur Produktionsplanung und -steuerung (PPS) wird ein Schwerpunkt auf Planungsaspekte gelegt (vgl. z.B. [DMH99; Di97] und die dort zitierte Literatur): Dabei handelt es sich dann um einen Parameter, „wenn ein Datenfeld vorliegt, das den planerischen Willen des Anwenders repräsentiert und von dessen Eintrag eine substantielle Wirkung auf das Verhalten des Systems bezüglich der Planungsergebnisse ausgeht“ [Di97]. In der Literatur zur Anpassung von Softwarekomponenten wird Parametrisierung als Anpassungsmechanismus identifiziert, die Begriffe *Parameter* und *Parametrisierung* werden aber nicht genauer definiert und beschrieben (vgl. z.B. [SC98; Be00b; Tu01; We01; OMG02]).

Basierend auf der angegeben Literatur werden die Begriffe *Parameter*, *Parameterwert* und *Parametrisierung* 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 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 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.“

„Ein *Parameterwert* ist ein einem Parameter zugewiesener Wert.“

„*(Datenbasierte) Parametrisierung* ist ein Verfahren zur Anpassung von Komponenten (oder anderen Softwarebausteinen), welches auf der Verwendung von Parametern basiert.“

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 (vgl. auch das Parameter-Metamodell im nächsten Kapitel). Der Grund dafür liegt – neben der Anlehnung an die oben zitierte 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].

4 Metamodell für Parameter

In diesem Abschnitt wird ein Parameter-Metamodell vorgestellt, welches auf [Ac04] zurückgeht. Die Definition eines solchen Metamodells erlaubt es, 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 [Ac02], in XML-Konfigurationsdateien (Property Files beim OMG CORBA Component Model [OMG2002], Deployment Descriptor bei Suns Enterprise Java Beans (EJB) [Mo00], Datei web.config bei Microsofts .NET [Pr02]) sowie in einfachen Initialisierungsdateien 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-Typdiagramm [OMG05] dargestellt und findet sich in Abb. 1. Im Metamodell werden Typ- und Instanzebene unterschieden. Die Elemente der Typebene werden vom Hersteller der Komponente vorgegeben und enthält die verfügbaren Parameter sowie deren Strukturierung – im Einzelnen besteht sie aus folgenden Elementen:

- 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, seinen Datentypen und seinen Wertebereich näher beschrieben. Parameter sind immer formatiert und treten in den Zeichenarten *boolsch*, *numerisch*, *alphanumerisch* und *zeichenartig* auf (vgl. [Ac02; DMH99]). Diese Unterscheidung wird im unteren Teil von Abb. 3 durch die Spezialisierung von *Datentyp* dargestellt. Analog dazu lassen sich wichtige Erscheinungsformen von Wertebereich unterscheiden [Ac02]: Wertebereich nicht (bzw. nur durch Datentyp) eingeschränkt (*Beliebig*); Wertebereich vom Hersteller fest vorgegeben (*Festwerte*); Wertebereich durch andere Parameter vorgegeben (*Parameterabhängig*). 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.
- Die Parameter einer Softwarekomponente sind häufig nicht isoliert, sondern liegen strukturiert vor (teilweise mit mehreren Hierarchiestufen). Eine Zusammenfassung von Parametern zu einer Struktureinheit wird im Metamodell durch den Typ *Parametergruppe* dargestellt. Eine Parametergruppe wird durch seinen 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.

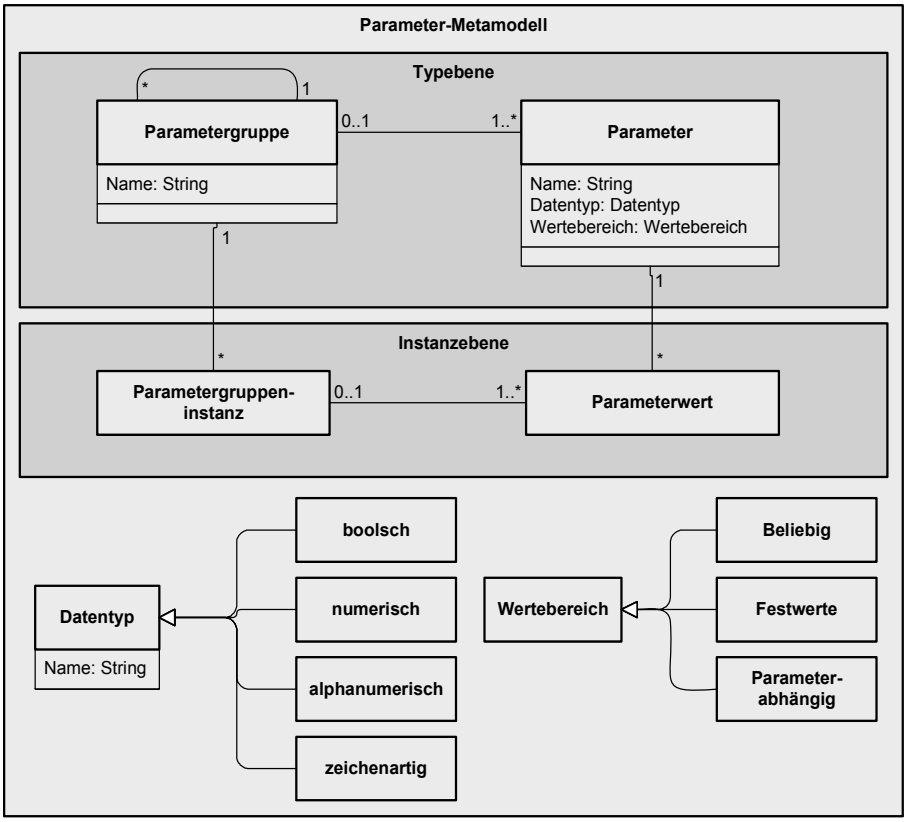

Abb. 1: Metamodell zur Beschreibung von Parametern

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 in Abb. 1 durch den Typ *Parametergruppeninstanz* repräsentiert wird. Eine Parametergruppeninstanz (auf unterster Hierarchiestufe) ist dabei eine Zusammenfassung von Werten, welche 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 Parametergrup-

pe ermöglicht die Definition verschiedener Parametrisierungsvarianten, die z.B. für unterschiedliche organisatorische Einheiten gelten können.

Die im Metamodell abgebildeten Sachverhalte werden im nächsten Kapitel anhand einer Beispielkomponente veranschaulicht.

5 Beispielkomponente *Lagermanagement*

Zur Illustration der bisher vorgestellten Konzepte wird in diesem Abschnitt eine Beispielkomponente *Lagermanagement* vorgestellt. 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. 2 durch ein Spezifikationsdatenmodell [Ac06] dargestellt. Die Komponente erlaubt die Definition mehrerer *Lager*, welche sich in der Lagersteuerung unterscheiden können (z.B. Fixplatzlager oder Hochregallager). 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.

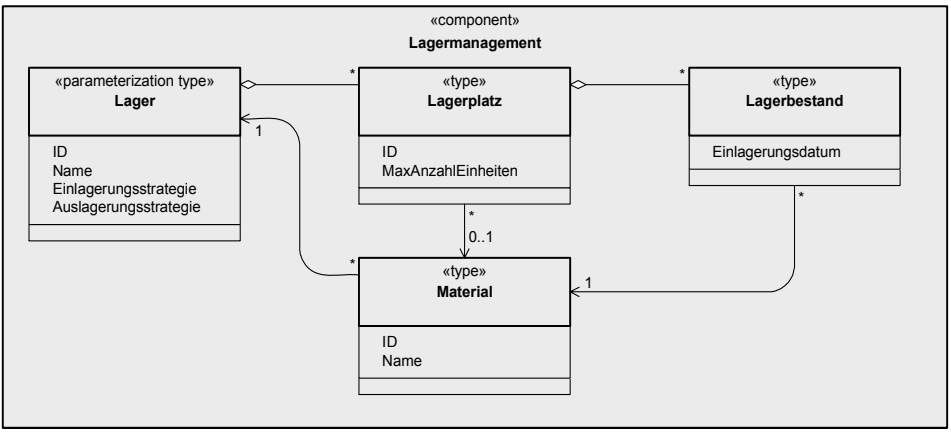

Abb. 2: Von der Komponente *Lagermanagement* verwaltete Daten

Die Komponente bietet die zwei Steuerungsparameter *Einlagerungsstrategie* und *Auslagerungsstrategie* an, mit denen für jedes Lager jeweils ein Verfahren für Einlagerung und Auslagerung auszuwählen ist. Darüber hinaus enthält ein Lager noch ein identifizierendes Attribut *ID* und ein beschreibendes Attribut *Name* – beide werden ebenfalls als Parameter betrachtet, da sie die organisatorische Einheit Lager näher beschreiben und zusammen mit den Steuerungsparametern zur Konfigurationszeit festgelegt werden.

Anhand dieser Beispielkomponente können die im Parameter-Metamodell (vgl. Abb. 1) dargestellten Konzepte veranschaulicht werden: Ein Beispiel für einen Parameter ist die für ein Lager zu definierende *Einlagerungsstrategie*. Ein möglicher Parameterwert für diesen Parameter ist der Wert 'F', welcher die Strategie *Festplatzsuche* repräsentiert. *Lager* ist ein Beispiel für eine Parametergruppe, welche aus den Parametern *ID*, *Name*, *Einlagerungsstrategie* und *Auslagerungsstrategie* besteht. Eine Parametergruppeninstanz ist die Lagerinstanz mit den Parameterwerten '001' (ID), 'Hochregallager' (Name), 'L' (Einlagerungsstrategie *Nächster Leerplatz*) und 'F' (Auslagerungsstrategie *First-In-First-Out*).

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

6 Vor- und Nachteile von Parametrisierung

Parametrisierung ist – wie jeder andere Anpassungsmechanismus auch – mit spezifischen Vor- und Nachteilen verbunden. Um in einer konkreten Anpassungssituation eine fundierte Entscheidung für einen Anpassungsmechanismus treffen zu können, ist es notwendig, die Vor- und Nachteile zu kennen.

Der Einsatz von Parametrisierung zur Anpassung von Software allgemein (unabhängig von Softwarekomponenten) ist mit folgenden Vorteilen verbunden:

- Anpassungen können mit vergleichsweise geringem Aufwand vorgenommen werden.
- Es ist keine Modifikation der Software notwendig und es sind keine Implementierungsarbeiten durchzuführen.
- Anpassungen erfordern keine Programmierkenntnisse und können daher auch von Mitarbeitern einer Fachabteilung vorgenommen werden.
- Anpassungen lassen sich zu einem späteren Zeitpunkt vergleichsweise einfach ändern, da erneut kein Implementierungsaufwand anfällt.
- Der Hersteller der Software kann die Upgradefähigkeit der Anpassungen sicherstellen, d. h. die getroffenen Einstellungen bleiben beim Upgrade auf eine neuere Softwareversion erhalten. Der bei einer Modifikation erforderliche – und meist sehr aufwändige – Abgleich zwischen bisheriger, modifizierter Version und neuer Version entfällt.

Für die Parametrisierung betrieblicher Fachkomponenten lassen sich darüber hinaus weitere Vorteile identifizieren:

- Parametrisierung ist kompatibel zum Leitbild der Black-Box-Wiederverwendung, da sie keine Kenntnisse von Implementierungsdetails erfordert.
- Parametrisierung erlaubt es vergleichsweise einfach, verschiedene Parametrisierungsvarianten gleichzeitig zu hinterlegen, von denen zur Laufzeit (abhängig vom Kontext) eine Variante dynamisch ausgewählt wird. Beispielsweise kann eine Lagerverwaltungssoftware den gleichzeitigen Einsatz mehrerer Lager erlauben, die sich in ihrer Einlagerungsstrategie unterscheiden können.
- Das Aktualisieren einer Anpassung zu einem späteren Zeitpunkt erfordert – im Unterschied zu den meisten anderen Anpassungsmechanismen – keinen Austausch von Komponenten, keine Änderung der Komponentenkonfiguration und bei Einsatz von Datenbanktabellen nicht einmal ein neues Deployment der Komponente.

Den angegebenen Vorteilen stehen folgende Nachteile gegenüber:

- Die möglichen Anpassungen sind auf solche beschränkt, die vom Hersteller explizit vorgesehen werden.
- Bei der derzeitigen Verwendung der Parametrisierung bei betrieblicher Standardsoftware liegt ein weiterer großer Nachteil in der Kombination von Komplexität und mangelnder Spezifikation [HW99]. Die genauen Wechselbeziehungen zwischen Parametern und Funktionsweise der Software sind in der Regel nur durch Simulation oder Reengineering ermittelbar und Ursache-Wirkungs-Zusammenhänge oft nicht transparent [MWH91].

7 Sinnvolle Einsatzszenarien für Parametrisierung

In einer konkreten Anpassungssituation stellt sich häufig die Frage, ob Parametrisierung eingesetzt werden kann oder ob ein anderer Anpassungsmechanismus vorzuziehen ist. Die zuvor diskutierten Vor- und Nachteile von Parametrisierung geben zwar erste Hinweise darauf, bieten aber allein noch keine wirksame Unterstützung bei der Entscheidungsfindung. Aus diesem Grund wird in diesem Abschnitt ein Ansatz entwickelt, welcher einen Komponentenentwickler bei einer Entscheidung für oder gegen Parametrisierung unterstützt.

Ausgangspunkt unserer Überlegungen bildet der derzeitige Einsatz von Parametrisierung in betrieblicher Standardsoftware. Dies begründet sich dadurch, dass es bisher kaum Erkenntnisse zur Parametrisierung betrieblicher Softwarekomponenten gibt, dafür aber umfangreiche Erfahrungen für herkömmliche Standardsoftware vorliegen. Parametrisierung wird bei betrieblicher Standardsoftware so umfassend und extensiv eingesetzt, dass wir die Hypothese treffen, dass es sich bei den sinnvollen Einsatzszenarien für Parametrisierung von Softwarekomponenten um eine Teilmenge der in betrieblicher Standardsoftware derzeit realisierten Einsatzszenarien handelt. Diese Teilmenge zu identifizieren ist Gegenstand der weiteren Untersuchungen.

In Abb. 3 wird ein Klassifikationsschema vorgestellt, welches die bei betrieblicher Standardsoftware gängigen Parametrisierungseinstellungen nach verschiedenen Kriterien klassifiziert. Anhand dieses Schemas erfolgt eine Bewertung, für welche der angegebenen Merkmalsausprägungen ein Einsatz von Parametrisierung auch bei komponentenbasierten Anwendungssystemen sinnvoll ist.

Gegenstand des Klassifikationsschemas bilden ganze Parametrisierungseinstellungen. Die Betrachtung einzelner Parameter wäre zu feingranular, da häufig mehrere Parameter gepflegt werden müssen, um in einer Anpassungssituation eine Anpassungsvariante auszuwählen und mit Details zu versehen. Unter einer *Parametrisierungseinstellung* wird dabei in Anlehnung an [Gö97, 101] und [Me01, 180] das gemeinsame Setzen solcher Parameter verstanden, die zusammen eine Anpassungsvariante beschreiben und deshalb gemeinsam zu pflegen sind.

Bei der Erarbeitung des Klassifikationsschemas wurde sowohl die relevante Literatur zur Parametrisierung (z.B. [MWH91; Pi93; Gö97; SGR98; DMH99; AR00]), welche Hinweise auf mögliche Klassifikationen enthält, als auch eine Fallstudie zum Customizing von SAP R/3 [Ac02] ausgewertet. Darüber hinaus wurden im Schema Merkmale aufgenommen, welche die Vor- und Nachteile eines Einsatzes von Parametrisierung besonders berücksichtigen. Zu beachten ist dabei, dass das Klassifikationsschema nicht alle möglichen Kriterien enthält, nach denen Parametrisierungseinstellungen klassifiziert werden können, sondern nur solche, die geeignet sind zu entscheiden, ob ein Einsatz von Parametrisierung in komponentenbasierten Anwendungssystemen sinnvoll ist. Eine Verifikation der Merkmale und Merkmalsausprägungen erfolgte anhand der in [Ac02] untersuchten Parametrisierungseinstellungen.

<table>
<tr><td colspan="2">MERKMAL</td><td colspan="6">MERKMALSAUSPRÄGUNG</td></tr>
<tr><td rowspan="5">Funktionalität</td><td rowspan="2">Parametrisierungskontext</td><td>Aufbauorganisation</td><td>Daten</td><td>Funktionen</td><td colspan="3">Abläufe (Prozesse)</td></tr>
<tr><td rowspan="2">Parametrisierungshandlung</td><td colspan="2">Vordefinierte Varianten</td><td colspan="3">Selbstdefinierte Varianten</td></tr>
<tr><td>Auswählen</td><td>Ausprägen</td><td>Definieren</td><td>Auswählen</td><td>Ausprägen</td></tr>
<tr><td>Parametrisierungscharakter</td><td colspan="2">Statisch</td><td colspan="3">Dynamisch</td></tr>
<tr><td>Gleichzeitig nutzbare Varianten</td><td colspan="2">Eine</td><td colspan="3">Mehrere</td></tr>
<tr><td rowspan="2">Lokalität</td><td>Zuordnung zu Subsystem</td><td colspan="2">Eindeutig</td><td colspan="3">Nicht eindeutig</td></tr>
<tr><td>Reichweite der Auswirkungen</td><td colspan="2">Subsystemlokal</td><td colspan="3">Subsystemübergreifend</td></tr>
<tr><td colspan="2">Änderungshäufigkeit</td><td colspan="2">Selten</td><td colspan="3">Häufig</td></tr>
<tr><td rowspan="2">Komplexität</td><td>Anzahl abhängiger Parametrisierungseinst.</td><td colspan="2">Niedrig</td><td colspan="3">Hoch</td></tr>
<tr><td>Anzahl betroffener Aufgaben</td><td colspan="2">Niedrig</td><td colspan="3">Hoch</td></tr>
</table>

Abb. 3: Klassifikationsschema für Parametrisierungseinstellungen zur Bewertung von Parametrisierung

Das Klassifikationsschema enthält 8 Merkmale mit ihren jeweiligen Merkmalsausprägungen. Zum besseren Verständnis wurden die Merkmale in verschiedene Merkmalsgruppen unterteilt: Funktionalität, Lokalität, Änderungshäufigkeit und Komplexität. In den folgenden Abschnitten werden die Merkmalsgruppen und Merkmale detailliert vorgestellt. Dazu wird jedes Merkmal mit seinen Merkmalsausprägungen beschrieben und durch Beispiele illustriert. Danach erfolgt eine Bewertung, für welche Merkmalsausprägungen der Einsatz von Parametrisierung in komponentenbasierten Anwendungssystemen geeignet ist und für welche Merkmalsausprägungen andere Anpassungsmechanismen vorteilhaft sind. Es ist zu beachten, dass bei der Darstellung in einigen Abschnitten zwei Merkmale zusammen behandelt werden, wenn z.B. die Bewertung sehr ähnlich ist oder die Bewertung für beide Merkmale im Zusammenhang erfolgen soll.

7.1 Merkmal Parametrisierungskontext

Das Merkmal *Parametrisierungskontext* beschreibt, welche Aspekte eines Anwendungssystems angepasst werden (Merkmalsausprägungen *Aufbauorganisation*, *Daten*, *Funktionen* und *Abläufe*). Dieses Merkmal findet sich (mit diesen oder ähnlichen Ausprägungen) häufig in der Literatur – vgl. z.B. [SGR98; AR00; Gö97]. Beispiele für die Klassi-

fikation sind: Definition von Buchungskreisen und die Zuordnung von Werken zu Buchungskreisen (Aufbauorganisation); Zuordnung einer oder keiner Preissteuerung zu einer Materialart (Datenstruktur); Auswahl einer Einlagerungsstrategie bei der Lagerhaltung, welche die Funktion der Einlagerung beeinflusst (Funktionen); Festlegung bei der Einkaufsabwicklung, dass einer Bestellung immer eine Bestellanforderung vorausgehen muss (Abläufe).

Dieses Merkmal spielt für die Klassifikation eine zentrale Rolle und liefert (im Zusammenspiel mit den anderen Merkmalen) erste Indizien für die Eignung von Parametrisierung. Eine Anpassung einzelner Daten und Funktionen ist häufig mit einer begrenzten Komplexität verbunden, wofür sich der Einsatz von Parametrisierung anbietet. Die Aufbauorganisation eines Unternehmens ist oft für größere Teile eines Anwendungssystems (und damit mehrere Komponenten) relevant – die Verwendung von Parametrisierung ist daher eher kritisch. Zentrale organisatorische Einheiten können in einem komponentenbasierten System stattdessen durch getrennte Komponenteninstanzen (derselben oder verschiedener Komponenten) realisiert werden. Wie die Aufbauorganisation betreffen auch Prozessabläufe typischerweise mehrere Komponenten, weshalb auch hier der Einsatz von Parametrisierung wohlüberlegt sein sollte. Mögliche Alternativen stehen mit Techniken zur Realisierung der zeitlichen Abstimmung [Tu01] (wie Workflowmanagementsysteme) zur Verfügung.

7.2 Merkmal Parametrisierungshandlung

Das Merkmal *Parametrisierungshandlung* berücksichtigt, dass man bei der Parametrisierung zwischen den Tätigkeiten der Auswahl einer Struktur- oder Verhaltensvariante und des Festlegens der Eigenschaften (Ausprägen) der ausgewählten Variante unterscheiden kann [Di97, 5 f.; Me01, 179]. Darüber hinaus wird in dieser Arbeit zwischen zwei Arten von Varianten unterschieden: Vordefinierte Varianten sind vom Hersteller der Software vorgegeben (Merkmalsausprägungen *Auswählen* und *Ausprägen*), während bei selbstdefinierten Varianten der Komponentenanwender (im Rahmen eines vom Herstellers vorgedachten Schemas) eigene Varianten definieren kann (Merkmalsausprägungen *Definieren*, *Auswählen* und *Ausprägen*). Bei der Lagerhaltung kann man einem Lager eine von mehreren vorgegebenen Einlagerungsstrategien zuordnen (Ausprägung *Auswählen*) und dann festlegen, ob jeder Einlagerungsvorgang quittierungspflichtig ist (Ausprägung *Ausprägen*). Ein Beispiel für selbstdefinierte Varianten ist die Definition von Torbelegungsprofilen, welche die Eigenschaften von Toren bei der Anlieferung an einem Lager festlegen. Parametrisierungshandlungen sind dabei das Anlegen eines Torbelegungsprofils (Ausprägung *Definieren*), das Festlegen der Eigenschaften des Profils wie z.B. der täglichen Anlieferungszeiten (Ausprägung *Ausprägen*) und die Auswahl eines Profils für ein Lager (Ausprägung *Auswählen*).

Bei komplexeren vordefinierten Varianten kann die Komplexität dadurch reduziert werden, dass neben einer generischen Komponente (für das gemeinsame Verhalten) für jede Variante eine eigenständige Komponenten implementiert wird, welche mit der generischen Komponente verknüpft wird [Be00a]. Selbstdefinierte Varianten sind von Natur aus weniger komplex und unterscheiden sich nur in einzelnen Attributen, da alle Varianten auf ein gemeinsames, vorab definiertes Schema zurückzuführen sind. Der Einsatz

von Parametrisierung ist daher in vielen Fällen sinnvoll. Dies gilt ebenso für das Ausprägen von Varianten (vordefinierte und selbstdefinierte), da damit im Allgemeinen keine große Komplexität verbunden ist.

7.3 Merkmale Parametrisierungscharakter und Gleichzeitig nutzbare Varianten

Das Merkmal *Parametrisierungscharakter* erfasst, ob eine Auswahl schon zum Konfigurationszeitpunkt (*statisch*) durch einen festen Wert oder erst zur Laufzeit (*dynamisch*) z.B. durch eine Entscheidungsregel erfolgt (vgl. [SGR98, 152]). So sind bei der Lagerhaltung für die Lagertypzuordnung zwei Varianten denkbar: entweder wird jedem Material ein fester Lagertyp zugeordnet (*statisch*) oder es wird eine Lagertypfindung definiert, die bei jeder Einlagerung in Abhängigkeit von Material und anderen Faktoren einen Lagertyp bestimmt (*dynamisch*).

Das Merkmal *Gleichzeitig nutzbare Varianten* beschreibt, ob zur Laufzeit nur eine Variante aktiv ist oder mehrere Varianten gleichzeitig verwendet werden (Merkmalsausprägungen *Eine* und *Mehrere*). So könnte bei einer Lagerhaltungskomponente genau eine Einlagerungsstrategie zu wählen sein oder aber die Komponente unterstützt den parallelen Einsatz mehrerer Lager, die sich in ihrer Einlagerungsstrategie unterscheiden dürfen.

Wird zur Konfigurationszeit eine feste Auswahl getroffen und wird zur Laufzeit nur diese eine Variante verwendet, lässt sich die Komplexität dadurch reduzieren, dass die verschiedenen Varianten in getrennten Komponenten implementiert werden und nur die benötigte Komponente eingesetzt wird. Erfolgt dagegen die endgültige Auswahl erst zur Laufzeit oder werden mehrere Varianten gleichzeitig eingesetzt, müssen alle benötigten Varianten im Anwendungssystem implementiert sein – für eine Auswahl zwischen den Varianten kann z.B. Parametrisierung eingesetzt werden.

7.4 Merkmalsgruppe Lokalität

Die Merkmale in der Gruppe *Lokalität* beschreiben, ob eine Parametrisierungseinstellung nur einen klar definierten Teilbereich des Gesamtsystems betrifft oder für mehrere Teilbereiche bzw. das gesamte System relevant ist. Für die Bewertung dieses Merkmals wird eine Unterteilung des Gesamtsystems in Subsysteme vorausgesetzt: bei SAP R/3 könnten dies z.B. die Module sein und bei einem komponentenbasierten Anwendungssystems wären dies gerade die betrieblichen Fachkomponenten.

Das Merkmal *Zuordnung zu Subsystem* beschreibt, ob ein eindeutiges Subsystem gefunden werden kann, dem die Parametrisierungseinstellung semantisch zuzuordnen ist (Merkmalsausprägungen *Eindeutig* und *Nicht eindeutig*). Die Klassifikation nach Zugehörigkeit geht auf [Pi93, 438] zurück, erfolgt dort aber als Zuordnung zu Teilfunktionen und Bezugsobjekten. Ein Extrembeispiel für eine nicht zuordenbare Einstellung ist das Setzen des so genannten *Retailschalters* in SAP R/3, mit dem sich ein Anwender entweder für die Standardversion von R/3 oder für die Branchenlösung Retail entscheidet (SAP R/3 Enterprise 4.70 und früher). Diese Einstellung verändert die Funktionsweise der Software an verschiedenen Stellen quer durch das System und lässt sich daher keinem speziellen Subsystem zuordnen.

Das Merkmal *Reichweite der Auswirkungen* klassifiziert, ob die Auswirkungen einer Parametrisierungseinstellung nur ein Subsystem betreffen (dem die Einstellung zugeordnet ist) oder aber sich über Subsystemgrenzen hinweg erstrecken (Merkmalsausprägungen *subsystemlokal* und *subsystemübergreifend*). Die Auswirkungen einer Parametrisierung betreffen ein anderes Subsystem dann, wenn das Verhalten eines externen Servicenutzers von den Auswirkungen abhängt. Eine subsystemübergreifende Auswirkung liegt z.B. vor, wenn bei der Einkaufsbelegabwicklung durch einen Parameter festgelegt wird, dass einer Bestellung immer eine Bestellanforderung vorausgehen muss – ein Servicenutzer muss dann zuerst eine Bestellanforderung anlegen. Von subsystemübergreifender Abhängigkeit wird dagegen nicht gesprochen, wenn die Auswirkungen einer Parametrisierung zwar von außerhalb des Subsystems sichtbar sind, aber nicht das Verhalten eines Servicenutzers verändern (sollten). Beispiele dafür sind bei der Produktionsplanung ein anderes Planungsergebnis in Abhängigkeit vom gewählten Planungsverfahren oder im Einkauf der Aufbau einer Einkaufsbelegnummer in Abhängigkeit von den festgelegten Nummernkreisen.

Die Lokalität einer Einstellung spielt eine wichtige Rolle bei der Bewertung, da gerade die übergreifenden Abhängigkeiten zur hohen Komplexität der Parametrisierung betriebswirtschaftlicher Standardsoftware beitragen. Eine Einstellung, die keiner Komponente zuzuordnen ist und damit mehrere Komponenten parametrisieren soll, muss als äußerst kritisch eingestuft werden – ein Einsatz von Parametrisierung sollte nicht erfolgen. Die Verwendung von Parametrisierung für eine Einstellung, die komponentenübergreifende Auswirkungen hat, ist aus Komplexitätsgründen als kritisch einzustufen. Abhängig von der Art und Weise sowie Umfang der Auswirkungen (vgl. auch Merkmalsgruppe *Komplexität*) ist abzuwägen, ob ein anderer Anpassungsmechanismus geeigneter ist. So könnten komponentenübergreifende Aspekte schon zur Designzeit aus den einzelnen Komponenten herausgelöst werden und stattdessen in extra Komponenten oder im Framework implementiert werden. Gerade für die Steuerung der Reihenfolge, in der Dienste ausgeführt werden (z.B. *Bestellanforderung anlegen* vor *Bestellung anlegen*), sind beispielsweise Workflowmanagementsysteme vorzuziehen.

7.5 Merkmal Änderungshäufigkeit

Das Merkmal *Änderungshäufigkeit* beschreibt, wie häufig sich eine getroffene Einstellung ändert [Pi93, 439]. Als Merkmalsausprägungen stehen *häufig* und *selten* zur Verfügung – auf eine genauere Beschreibung (z.B. als quantitative Aussage) wird verzichtet, da der Änderungsaufwand vom Einzelfall abhängt und eine genaue Bewertung, bis zu wie vielen Änderungen pro Zeitabschnitt Parametrisierung zu empfehlen ist, so allgemein nicht getroffen werden kann. Ein sich häufig ändernder Parameter ist der Beitragssatz von gesetzlichen Krankenversicherungen in der Lohnabrechnung: bei über 200 Krankenversicherungen und einer Beitragsänderung aller 1-2 Jahre sind (im Durchschnitt) monatlich mehrere Beitragssätze anzupassen. Im Gegensatz dazu wird sich die Einlagerungsstrategie in einem Warenlager nur selten ändern, da damit häufig eine komplette Reorganisation des Lagers verbunden wäre.

Durch dieses Kriterium wird berücksichtigt, dass bei Parametrisierung Änderungen während des Systembetriebs vergleichsweise einfach durchzuführen sind. Änderungen

erfordern keine Programmierarbeiten und keine Rekonfiguration der Komponentenzusammensetzung und sind auch von Mitarbeitern aus Fachabteilungen durchzuführen. Daher ist der Einsatz von Parametrisierung vorteilhaft bei Einstellungen, die sich häufig ändern.

7.6 Merkmalsgruppe Komplexität

Die Merkmale in dieser Merkmalsgruppe erlauben eine erste Einschätzung, wie groß die mit einer Parametrisierungseinstellung assoziierte Komplexität ist. Das Merkmal *Anzahl abhängiger Parametrisierungseinstellungen* erfasst, wie viele andere Parametrisierungseinstellungen von der betrachteten Einstellung abhängen (Merkmalsausprägungen *niedrig* und *hoch*). Eine betriebliche Aufgabe ist von einer Parametrisierungseinstellung betroffen, wenn die gewählten Parameterwerte Voraussetzungen oder Ergebnis der Dienstdurchführung beeinflussen. In diesem Sinne erfasst das Merkmal *Anzahl betroffener Aufgaben* den Einfluss einer Einstellung auf die betrieblichen Aufgaben einer Komponente (Merkmalsausprägungen *niedrig* und *hoch*).

Die Wahl einer Einlagerungsstrategie bei der Lagerhaltung hat typischerweise keine oder nur wenige Abhängigkeiten zu anderen Parametrisierungseinstellungen und beeinflusst nur die betriebliche Aufgabe der Einlagerung. Ein Extrembeispiel für eine Einstellung mit vielen Abhängigkeiten ist der Retailschalter in SAP R/3, welcher sowohl eine große Anzahl anderer Parameter als auch eine große Anzahl von Transaktionen beeinflusst.

Da einer der Nachteile beim derzeitigen Einsatz von Parametrisierung in der durch Parameter erzeugten Komplexität liegt, ist eine hohe Zahl von Abhängigkeiten als kritisch einzustufen. Dies gilt umso mehr, wenn komponentenübergreifende Abhängigkeiten vorausgesetzt werden, obwohl die anderen Komponenten eines Systems zum Entwicklungszeitpunkt der Komponente häufig noch gar nicht bekannt sind (wie z.B. bei der Definition komponentenübergreifend benötigter organisatorischer Einheiten). Anzumerken bleibt, dass die reine Anzahl von Abhängigkeiten nur ein Indiz für die Komplexität sein kann – im Einzelfall muss natürlich auch die Komplexität der Abhängigkeiten berücksichtigt werden.

7.7 Zusammenfassende Betrachtungen

Zweck des in Abb. 3 dargestellten Klassifikationsschemas ist es, einen Komponentenentwickler bei der Entscheidung zu unterstützen, für welche Anpassungssituationen ein Einsatz von Parametrisierung in komponentenbasierten Anwendungssystemen sinnvoll ist. Dazu kann der Entwickler eine konkrete Anpassungssituation bezüglich der angegebenen Merkmale untersuchen und darauf aufbauend eine Gesamtbewertung durchführen. Es wird naturgemäß vorkommen, dass die Bewertung mancher Merkmale für Parametrisierung und anderer Merkmale dagegen spricht – in solchen Fällen sind die angeführten Nachteile sowie die vorgeschlagenen Anpassungsalternativen gegeneinander abzuwägen. Zu berücksichtigen ist außerdem, dass die angegebenen Bewertungen für die Merkmalsausprägungen nicht in jedem Einzelfall zutreffen müssen – so ist es z. B. denk-

bar, dass mehrere Abhängigkeiten mit geringer Komplexität weniger problematisch sind als eine Abhängigkeit mit hoher Komplexität.

Zusammenfassend lässt sich feststellen, dass Parametrisierung dann sinnvoll eingesetzt werden kann, wenn lokal begrenzte Funktionalität anzupassen ist, sich häufige Änderungen ergeben oder mehrere Varianten gleichzeitig zum Einsatz kommen. Eher kritisch zu bewerten ist der Einsatz von Parametrisierung bei komponentenübergreifenden Abhängigkeiten und Auswirkungen sowie bei der Auswahl komplexer, vordefinierter Funktionsvarianten.

8 Zusammenfassung und Ausblick

Diese Arbeit beschäftigte sich mit der Bedeutung von Parametrisierung bei der fachlichen Anpassung in komponentenbasierten betrieblichen Anwendungssystemen. Dazu wurden zunächst Parametrisierung als Anpassungsmechanismus vorgestellt, die zentralen Begriffe *Parameter* und *Parametrisierung* definiert, ein Parameter-Metamodell entwickelt und anhand einer Beispielkomponente illustriert. Darüber hinaus wurde auf der Grundlage der Vor- und Nachteile von Parametrisierung ein Klassifikationsschema für Parametrisierungseinstellungen entwickelt, welches die Entscheidung unterstützen kann, ob für eine konkrete Anpassungssituation der Einsatz von Parametrisierung geeignet ist. Im Ergebnis zeigt sich, dass Parametrisierung eine durchaus wichtige Rolle bei der Anpassung in komponentenbasierten betrieblichen Anwendungssystemen zukommt. Die vorgestellten Ergebnisse bilden die Grundlage für zukünftige Forschungsarbeiten, bei denen ein Spezifikationsrahmen für parametrisierbare betriebliche Softwarekomponente entwickelt werden soll.

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.

[Ac06] Ackermann, J.: Using a Specification Data Model for Specification of Black-Box Software Components. In: Enterprise Modelling and Information Systems Architectures – An International Journal. (wird veröffentlicht).

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

[Be00a] Becker, M.: Generic Components: A Symbiosis of Paradigms. In (Butler, G.; Jarzabek, S.) (Hrsg.): Generative and Component-Based Software Engineering, Second International Symposium (GCSE 2000). LNCS 2177. Erfurt, 2000; S. 100-113.

[Be00b] Bergner, K. et. al.: Adaptation Strategies in Componentware. In: Proceedings 2000 Australian Software Engineering Conference. IEEE Computer Society 2000; S. 87-95.

[BN03] Bobett, G.; Noye, J.: Molding Components using Program Specialization Techniques. In (Bosch, J.; Szyperski, C.; Weck, W.) (Hrsg.): Proceedings of the Eighth International Workshop on Component-Oriented Programming (WCOP 2003). Darmstadt, 2003.

[Bo00] Bosch, J.: Design and Use of Software Architectures – Adopting and evolving a product-line approach. Addison-Wesley, Reading 2000.

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

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

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

[Di97] Dittrich, J.: Simulationsgestützte Analyse und Konfiguration von PPS-Stellgrößen am Beispiel ausgewählter Dispositionsparameter des Systems SAP R/3-PP. Dissertation, Nürnberg, 1997.

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

[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.

[HW99] Hufgard, A.; Wenzel-Däfler, H.: Reverse Business Engineering – Modelle aus produktiven R/3-Systemen ableiten. In: Tagungsband Wirtschaftsinformatik 1999. Electronic Business Engineering. Heidelberg, 1999; S. 426-441.

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

[KM05] Kim, S.D.; Min, H.G.: A Systematic Methodology for Adapting Software Components. In (Turowski, K.; Zaha, J.M.) (Hrsg.): Component-Oriented Enterprise Applications - Proceedings of the Conference on Component-Oriented Enterprise Applications (COEA 2005). LNI Bd. P-70. Erfurt, 2005; S. 9-23.

[Me01] Mertens, P. et. al.: Grundzüge der Wirtschaftsinformatik. 7. Auflage. Springer-Verlag, Berlin et al., 2001.

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

[MWH91] Mertens, P.; Wedel, T.; Hartinger, M.: Management by Parameters? In: Zeitschrift für Betriebswirtschaft 61 (1991) 5/6; S. 569-588.

[OMG02] OMG (Hrsg.): CORBA Components Specification. Version 3.0, June 2002. URL: http://www.omg.org/, Abruf am 2006-02-02.

[OMG05] OMG (Hrsg.): Unified Modeling Language: UML 2.0 Superstructure Specification (Formal version, 2005-07-04). URL: http://www.omg.org/technology/documents, Abruf am 2005-09-09.

[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

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

[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] 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.

[Sz98] Szyperski, C.: Component Software: Beyond Object-Oriented Programming. 2nd Edition., Addison-Wesley, Harlow, 1998.

[Tu01] Turowski, K.: Fachkomponenten: Komponentenbasierte betriebliche Anwendungssysteme. Habilitationsschrift, Universität Magdeburg. Magdeburg, 2001.

[Tu99] Turowski, K.: Ordnungsrahmen für komponentenbasierte betriebliche Anwendungssysteme. In (K. Turowski) (Hrsg.): 1. Workshop Komponentenorientierte betriebliche Anwendungssysteme (WKBA 1). Magdeburg, 1999; S. 3-14.

[We01] Weis, T.: Component Customization. In: Proceedings of the 6th International Workshop on Component-Oriented Programming (WCOP 2001). At ECOOP 2001. Budapest, Hungary, 2001.