Zur Beschreibung datenbasierter Parametrisierung von Softwarekomponenten

Jörg Ackermann

Universität Augsburg
Universitätsstraße 16
86135 Augsburg
joerg.ackermann.hd@t-online.de

Zusammenfassung. Softwarekomponenten können nur selten ohne Anpassung wiederverwendet werden. Ein bekanntes Anpassungsverfahren ist die datenbasierte Parametrisierung. Ein vorgesehener Parametrisierungsspielraum muss jedoch in der Spezifikation der Komponente berücksichtigt werden. Diese Problematik ist Gegenstand eines derzeit laufenden Forschungsvorhabens. Diese Arbeit beschäftigt sich mit der Frage, welche Sachverhalte in einer Spezifikation zu berücksichtigen sind. Die Beiträge dieser Arbeit sind eine neue Klassifikation verschiedener Parametrisierungsansätze, ein Parameter-Metamodell zur Beschreibung der Parameter selbst und ein erster Ansatz zur Beschreibung von Parameterauswirkungen mit Hilfe eines Klassifikationsschemas.

Ein schon lange verfolgtes Ziel ist, unternehmensindividuelle Anwendungssysteme aus wiederverwendbaren Softwarekomponenten zu erstellen [Mc68]. Durch eine solche Komponentenstrategie kann man die Effektivität und Qualität der Softwareentwicklung steigern und die Flexibilität der Anwendungssysteme erhöhen. Zwei wichtige Erfolgsfaktoren sind dabei standardisierte Spezifikation und Anpassbarkeit von Komponenten: Eine standardisierte Spezifikation ist Voraussetzung für Entwicklungsmethodologie und Werkzeugunterstützung [Ov04] sowie für die Wiederverwendung von Komponenten durch Dritte [Tu01]. Anpassung ist deshalb unerlässlich, da sich Komponenten kaum ohne Anpassung wiederverwenden lassen [Bo97].

Ein bekanntes Verfahren zur Anpassung ist die (datenbasierte) Parametrisierung. Da sich durch Parametrisierung Struktur und Verhalten einer Komponente ändern, muss ein vorhandener Parametrisierungsspielraum spezifiziert werden. Die Spezifikation datenbasierter Parametrisierung ist Thema eines derzeit laufenden Forschungsvorhabens. Bevor man Spezifikationsvorschriften erstellen kann, muss bekannt sein, was überhaupt zu spezifizieren ist. Dazu werden in der vorliegenden Arbeit – mit der Parametrisierung zusammenhängende - spezifikationsrelevante Sachverhalte identifiziert und beschrieben.

Die Arbeit gliedert sich in zwei Teile. Der erste Teil dient Überblick und Einordnung in die Thematik: Nach einer allgemeinen Diskussion zu Anpassungsverfahren für komponentenbasierte Anwendungen (Kapitel 1) beschäftigt sich Kapitel 2 detaillierter mit Parametrisierung. Es wird eine Unterteilung in datenbasierte und programmbasierte Parametrisierung vorgeschlagen, wodurch verschiedene unter *Parametrisierung* bekannte

Verfahren gemeinsam dargestellt werden können. Kapitel 3 diskutiert die Problematik der Spezifikation datenbasierter Parametrisierung und stellt den derzeitigen Stand der Technik vor. Der zweite Teil der Arbeit enthält Vorschläge zur Beschreibung datenbasierter Parametrisierung: Dazu wird für Parameter ein Parameter-Metamodell entwickelt (Kapitel 4) und für Parametrisierungsauswirkungen wird ein Ansatz vorgeschlagen, bei dem die Beschreibung mit Hilfe eines Klassifikationsschemas erfolgt (Kapitel 5). Die Arbeit schließt mit dem zusammenfassenden Kapitel 6 ab.

1 Anpassung von Komponenten und komponentenbasierten Anwendungen

Die Anpassung in komponentenbasierten Anwendungssystemen wird von vielen Autoren diskutiert, vgl. z. B. [Bo97], [Bo00], [SC8], [Be00], [Re01] und [BN03]. Bekannte Anpassungsverfahren sind: Kopieren von Code, Vererbung, Aggregation, Ummantelung, Superimposition, Adapterinterfaces, parametrisierte Verträge und verschiedene Formen der Parametrisierung. Ein Vergleich verschiedener Anpassungsverfahren findet sich z. B. in [Bo97] und [Re01].

Anpassungsverfahren für Komponenten lassen sich u. a. durch folgende Kriterien klassifizieren:

- Anpassung der Komponenten vs. Anpassung der Komponentenkomposition (vgl. [SC98]): Angepasst werden kann entweder die wiederzuverwendende Komponente selbst (z. B. durch Vererbung oder datenbasierte Parametrisierung) oder die Komposition mit anderen Komponenten bzw. die Komponentenarchitektur (z. B. bei Aggregation, Ummantelung oder Adapterinterfaces).
- Black-Box vs. White-Box Wiederverwendung (vgl. [Bo97] und [Sz98]): White-Box-Verfahren erfordern eine Einsicht in die Implementierung der Komponente – Black-Box-Verfahren dagegen nicht. Beispiele für White-Box-Verfahren sind Kopieren von Code und Vererbung - für Black-Box-Verfahren Aggregation, Ummantelung und Parametrisierung.
- Schnittstellen-basierte vs. Code-basierte Verfahren (vgl. [Re01]): Schnittstellenbasierte Verfahren betreffen lediglich die Schnittstelle der anzupassenden Komponente (z. B. Aggregation und Ummantelung), während bei Code-basierten Verfahren auch die Implementierung gelesen und verändert wird (z. B. bei Kopieren von Code und Vererbung). Nicht so bekannt ist, dass es auch Code-basierte Black-Box-Verfahren gibt (z. B. Anpassung durch Metaprogrammierung) – d. h. dieses Kriterium ist tatsächlich orthogonal zum vorherigen Kriterium.
- Geplante vs. ungeplante Anpassung (vgl. [AT03]): Bei geplanter Anpassung werden die Anpassungsmöglichkeiten vom Komponentenhersteller vorgesehen (z. B. bei Parametrisierung und Programmieren von vorgesehenen User Exits) – bei ungeplanter Anpassung dagegen nicht (z. B. bei Vererbung oder Ummantelung).

Bei betrieblichen Fachkomponenten wird darüber hinaus zwischen technischer und fachlicher Anpassung unterschieden [Tu01]. Die *technische Anpassung* dient dazu, imple-

mentierungsbedingte, technische Inkompatibilitäten zu beheben. Unter *fachlicher Anpassung* werden Einstellungen verstanden, die betriebswirtschaftlich und aufgabenbezogen sind. Als Techniken zur fachlichen Anpassung stehen z. B. verschiedene Varianten der datenbasierten Parametrisierung zur Verfügung: das Anlegen oder Ändern von Parameter- oder Initialisierungsdateien, die Pflege von Parametertabellen in Datenbanken, die Veränderung von Einstellungen über grafische Benutzeroberflächen.

2 Parametrisierung zur Anpassung von Softwarekomponenten

Der Begriff der *Parametrisierung* ist in der Literatur oft anzutreffen – beschreibt aber zum Teil verschiedene, wenn auch verwandte Konzepte.

In der Wirtschaftsinformatik wird *Parametrisierung* häufig synonym zu *Customizing* verwendet, worunter die Anpassung einer Standardsoftware an die Anforderungen eines Kunden verstanden wird [Gö97]. Solche Anpassungen erfolgen meist durch das Setzen von Parametern, die in Datenbanktabellen abgelegt werden. 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] und bei der generischen Programmierung [CE00]. Dabei gehört Parametrisierung (bzw. parametrischer Polymorphismus) neben dem Subtyp-Polymorphismus zu den zentralen Konzepten, um Variabilität in Software zu realisieren. Bei den Parametern kann es sich um Daten, Typen, Objekte oder ganze Komponenten handeln. Auf der Implementierungsebene wird Parametrisierung durch spezielle Konstrukte in Programmiersprachen unterstützt: *templates* in C++ oder *generics* in Ada und Eiffel. Ähnliche Konzepte werden in [MBL97] und [Br98] für Java vorgeschlagen.

Parametrisierung ist außerdem als Anpassungsverfahren für Softwarekomponenten bekannt ([Bo00], [Re01]) und tritt dabei in verschiedenen Varianten auf.

Im Folgenden beschäftigen wir uns mit der Parametrisierung zur Anpassung von Komponenten und diskutieren Gemeinsamkeiten und Unterschiede der auftretenden Varianten. Gemeinsam ist allen Varianten, dass es sich um Verfahren zur geplanten Anpassung handelt, die sich durch folgende Merkmale auszeichnen:

- Die Anpassungsmöglichkeiten werden vom Komponentenhersteller vorgedacht.
- Der Hersteller definiert Parameter inklusive deren Bedeutung und deren Auswirkungen auf Struktur und Verhalten der Komponente.
- Der Verwender bindet die Parameter so, dass die Komponente seinen Anforderungen entsprechend arbeitet. Um das Binding herzustellen, muss der Verwender – abhängig von der Art des Parameters – geeignete Daten, Programme bzw. Komponenten bereitstellen.

Die Unterschiede zwischen den verschiedenen Varianten werden im Klassifikationsschema in Abb. 1 dargestellt. Das Merkmal *Parametrisierungsart* dient zur Unterscheidung der zwei wichtigsten Formen der Parametrisierung: Wir sprechen von *programmbasierter Parametrisierung*, wenn für einen Parameter ein – im weitesten Sinne – ausführbares Programm erwartet wird. Im Gegensatz dazu handelt es sich um *datenbasierte Parametrisierung*, wenn für einen Parameter Daten erwartet werden.

MERKMAL	MERKMALSAUSPRÄGUNG				
Parametrisierungsart	Datenbasiert		Programmbasiert		
Technische Realisierung	Datenbanktabellen	Parameterdateien	Komponenten	Programme	Workflowschemata

Abbildung 1: Klassifikationsschema für die Parametrisierung von Softwarekomponenten

Das Merkmal *Technische Realisierung* beschreibt, auf welche Art das Parameterbinding realisiert bzw. abgelegt ist. Die Spielarten der programmbasierten Parametrisierung werden dabei in die Gruppen *Komponenten*, *Programme* und *Workflowschemata* unterteilt:

- Eine Parametrisierung durch eine Komponente liegt vor, wenn eine (in ihrer Funktionalität) generischere Komponente an sogenannten Plug-Points durch eine spezifischere Komponente parametrisiert wird [DW98]. Die anzupassende Komponente kann mehrere solcher Plug-Points haben und damit gleichzeitig durch mehrere andere Komponenten parametrisiert werden. Gängige Bezeichnungen für die in den Plug-Points verwendeten Schnittstellen sind *lower interfaces* [DW98] und *adaptation interfaces* [Be00]. Durch diese Technik wird streng genommen nicht die Komponente selbst angepasst, sondern die Anpassung erfolgt durch Komposition mit den spezifischen Komponenten.

- Wir sprechen von einer Parametrisierung durch ein Programm, wenn eine Komponente – an vom Hersteller vorgedachten User Exits - durch ein Programm des Verwenders angepasst wird. Solche Programme können z. B. in der Programmiersprache der Komponente oder in einer vom Komponenten-Framework bereitgestellten Programmiersprache erstellt sein. Details zum Umgang mit solchen Erweiterungen finden sich z. B. in [JGJ97]. Im Gegensatz zur Parametrisierung durch eine Komponente wird bei dieser Technik die Funktionalität der anzupassenden Komponente direkt verändert.

- Eine Parametrisierung durch Workflowschemata liegt vor, wenn bei der Ausprägung eines Parameters ein Workflowschema erwartet wird (zu Workflows allgemein und zur Verwendung von Workflowmanagementsystemen vgl. z. B. [Ho96] oder [JB96]). Die Besonderheiten dieser Variante sind, dass ein Workflowschema mit Hilfe spezieller Modellierungssprachen definiert wird und zur Ausführung ein Workflowmanagementsystem benötigt.

Für die Ablage der Daten bei der datenbasierten Parametrisierung kann man zwischen den Varianten *Datenbanktabellen* und *Parameterdateien* unterscheiden:

- Parameterwerte können in Datenbanktabellen abgelegt werden und steuern zur Laufzeit Struktur und Verhalten der Komponente. Parameter in Datenbanktabellen spielen vor allem für die fachliche Anpassung einer Komponente eine wichtige Rolle [Tu01].

- Alternativ können Parameterwerte in Dateien abgelegt werden. Hervorzuheben sind Konfigurations- und Initialisierungsdateien. Verschiedene Komponenten-Frameworks bieten die Möglichkeit, in einer Konfigurationsdatei vordefinierte Parameter mit Werten zu versehen, die dann beim Starten der Komponente geladen werden. (Bei Microsofts .NET sind solche Konfigurationsdateien z. B. im XML-Format [Pr02]). Initialisierungsdateien werden auch beim Start der Komponente geladen, stellen aber keine Funktionalität des Komponenten-Frameworks dar. Außerdem werden Einstellungen zur Personalisierung auch oft als Datei abgelegt.

Die Verwendung datenbasierter Parametrisierung bietet im Allgemeinen folgende Vorteile: Es ist keine Modifikation der Software notwendig, d. h. es sind keine Implementierungsarbeiten durchzuführen. Anpassungen können daher mit vergleichsweise geringem Aufwand vorgenommen werden und erfordern keine Programmierkenntnisse. Vorteilhaft ist außerdem, dass der Hersteller der Software sicherstellen kann, dass die getroffenen Einstellungen beim Upgrade auf eine neuere Softwareversion erhalten bleiben. Der bei einer Modifikation erforderliche – und meist sehr aufwändige - Abgleich zwischen bisheriger, modifizierter Version und neuer Version entfällt. Speziell für Komponenten der betrieblichen Anwendungsdomäne lassen sich folgende Vorteile ergänzen: Parametrisierung passt gut zum Leitbild einer einfachen Black-Box-Wiederverwendung, da sie keine Kenntnisse von Implementierungsdetails erfordert. Außerdem legt die extensive Verwendung der Parametrisierung bei betrieblicher Standardsoftware den Schluss nahe, dass sie für die fachliche Anpassung geeignet ist. Den angegebenen Vorteilen steht ein gewichtiger Nachteil gegenüber: Es sind nur solche Anpassungen möglich, die vom Hersteller explizit vorgesehen werden.

3 Problematik der Spezifikation datenbasierter Parametrisierung

Im weiteren Verlauf dieser Arbeit beschäftigen wir uns nur noch mit der datenbasierten Parametrisierung. Daher ist im Folgenden immer die datenbasierte Parametrisierung gemeint, auch wenn verkürzt von Parametrisierung gesprochen wird.

Unter der Spezifikation einer Softwarekomponente versteht man die vollständige, widerspruchsfreie und eindeutige Beschreibung der Außensicht der Komponente [Tu02]. Vorhandene Parameter beeinflussen Struktur und Verhalten einer Komponente und wirken sich damit auf deren Außensicht aus. Deshalb müssen Parameter und Auswirkungen einer Parametrisierung spezifiziert werden, wenn eine Komponente parametrisierbar ist. Ohne Berücksichtigung von Parametrisierungsaspekten ist die Spezifikation einer Softwarekomponente nicht vollständig und einem Verwender der Komponente fehlen wesentliche Informationen für ihren Einsatz.

Einen Überblick zum Stand der Technik und zum Stand der Praxis datenbasierter Parametrisierung wird in [Ac03b] gegeben. Zum Stand der Technik wird festgestellt, dass die

meisten Arbeiten zur Anpassung von Komponenten *Parametrisierung* zwar als Anpassungsverfahren identifizieren, aber nicht vertiefend behandeln. Eine Diskussion von Spezifikationsaspekten findet nicht statt. Auch in der Literatur zur Spezifikation von Softwarekomponenten wird auf die datenbasierte Parametrisierung nicht gezielt eingegangen.

Zum Stand der Praxis finden sich umfangreiche Erfahrungen für nicht komponentenbasierte, monolithische Standardsoftware (wie z. B. SAP R/3). Literatur zum Customizing behandelt meist allgemeines Vorgehen, Konfiguration von Business-Prozessen und Projektmanagementaspekte (für SAP R/3 vgl. z. B. [KT98] und [AR00]). Mit der Beschreibung von Parametern und Parameterauswirkungen beschäftigen sich z. B. [MWH91] und [LM93]. Dabei wird festgestellt, dass Parameter und deren Bedeutung meist nur unzureichend beschrieben werden - eine formale Spezifikation der Parameter findet nicht statt. Dadurch sind Wechselbeziehungen zwischen Parametern und Funktionsweise der Software nur durch Simulation oder Reengineering ermittelbar und Ursache-Wirkungs-Zusammenhänge oft nicht transparent. Auf die Verbesserung des Parametermanagements zielen Untersuchungen für verschiedene Softwarepakete im Bereich der Produktionsplanung und -steuerung (PPS) (z. B. [Pi94] und [DMH99]). Dabei wurden vorhandene PPS-Parameter und deren Abhängigkeiten ermittelt sowie Werkzeuge zur Konfigurationsunterstützung entwickelt. Die Untersuchungen beschränken sich auf den Bereich PPS sowie auf planerische und dispositive Datenfelder. Ein allgemeines – von der jeweiligen Standardsoftware unabhängiges - Modell für Parameter und Parameterauswirkungen wurde nicht erstellt. Zusammengefasst existiert für das Customizing betrieblicher Anwendungssysteme kein allgemeiner Ansatz, wie Parameter und ihre Auswirkungen spezifiziert werden können.

In einem derzeit laufenden Forschungsvorhaben wird untersucht, wie sich Parameter sowie funktionale und nicht-funktionale Auswirkungen einer Parametrisierung spezifizieren lassen. Bisherige Arbeiten untersuchten exemplarisch das Customizing von SAP R/3, identifizierten die spezifikationsrelevanten Sachverhalte bei Parametern und Parameterpflege und entwickelten einen ersten Ansatz, wie die ermittelten Sachverhalte mit Hilfe der UML spezifiziert werden können (vgl. [Ac02], [Ac03a] und [AT03]). Bisher erfolgte jedoch weder eine theoretische Fundierung noch die Betrachtung der Auswirkungen einer Parametrisierung.

Abb. 2 zeigt die für das Forschungsvorhaben zu lösenden Teilaufgaben [Ac02] und unterscheidet dabei zwei Dimensionen:

- Es müssen zunächst alle spezifikationsrelevanten Sachverhalte ermittelt und strukturiert werden - d.h. es wird untersucht, *was* zu spezifizieren ist. Danach müssen Spezifikationsvorschriften für diese Sachverhalte entwickelt werden - d. h. es wird untersucht, *wie* die Spezifikation erfolgen soll.

- Die zu beschreibenden und zu spezifizierenden Sachverhalte können gruppiert werden in die Beschreibung der Parameter selbst und die Beschreibung der Auswirkungen einer Parametrisierung.

Spezifikationsrelevante Sachverhalte zu Parametern ...	Spezifikationsrelevante Sachverhalte zu Parameterauswirkungen ...
... ermitteln (Kapitel 4)	... ermitteln (Kapitel 5)
... spezifizieren	... spezifizieren

Abbildung 2: Für Spezifikation datenbasierter Parametrisierung zu lösende Teilaufgaben

Ziel der folgenden Kapitel ist, die – mit Parametrisierung zusammenhängenden – spezifikationsrelevanten Sachverhalte zu ermitteln und zu beschreiben. Dabei beschäftigt sich Kapitel 4 mit der Beschreibung der Parameter und Kapitel 5 mit der Beschreibung von Parametrisierungsauswirkungen. Die Spezifikation der ermittelten Sachverhalte ist nicht Thema dieser Arbeit und Richtung weiterer Forschung.

4 Metamodell für Parameter

In diesem Kapitel stellen wir ein Parameter-Metamodell vor. Ausgangspunkt dazu waren Ergebnisse in [Ac02]. Dort wurde ein Teilbereich des Customizings von SAP R/3 analysiert und eine Liste aller Sachverhalte erstellt, die im Zusammenhang mit Parametern stehen und in einer Spezifikation zu berücksichtigen sind. Weitere Ausgangspunkte waren die Struktur von Konfigurationsdateien im XML-Format [GP02] sowie einfacher Initialisierungsdateien. Methodisch wurde so vorgegangen: die identifizierten Sachverhalte und deren Struktur wurden (für Datenbanktabellen, Konfigurations- und Initialisierungsdateien getrennt) analysiert, Gemeinsamkeiten identifiziert und dann in Form des Metamodells dargestellt. Im Ergebnis entstand ein für alle Spielarten der datenbasierten Parametrisierung gültiges Parameter-Metamodell, welches die spezifikationsrelevanten Sachverhalte zu Parametern strukturiert darstellt.

Der Forschungsbeitrag dieses Kapitels liegt darin, dass die in [Ac02] identifizierten Sachverhalte durch das Metamodell strukturiert dargestellt werden und dass ein für alle Spielarten der datenbasierten Parametrisierung vereinheitlichter Ansatz vorliegt.

Das Metamodell wird als UML-Typdiagramm [OMG04] dargestellt und findet sich in Abb. 3. Im Metamodell werden Typ- und Instanzebene unterschieden. Die Typebene wird vom Hersteller der Komponente vorgegeben und enthält die verfügbaren Parameter sowie deren Strukturierung – im Einzelnen enthält sie die folgenden Elemente:

- Zentrales Element des Metamodells ist der Parameter. Unter einem *Parameter* verstehen wir ein Datenfeld, welches für die Parametrisierung verwendet wird. 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] oder [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 (*Parameterabhän-*

gig). Darüber hinaus lassen sich Parameter nach ihrer Wirkungsart klassifizieren. Wir unterscheiden identifizierende, beschreibende, definierende, auswählende, einschränkende, berechnende und steuernde Parameter – nähere Erläuterungen dazu finden sich im Abschnitt 5.4. Bei datenbankbasierter Ablage entspricht ein Parameter einer Spalte einer Tabelle - bei XML-Dateien handelt es sich bei den Parametern um die Elemente (Knoten unterster Ebene) oder die Attribute.

- Parameter treten häufig nicht isoliert, sondern in Gruppen auf. Dieser Sachverhalt 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-Dateien sind 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 bei einfachen Initialisierungsdateien häufig vor.

- Parametergruppen können hierarchisch angeordnet sein. Dies wird im Metamodell durch die reflexive Beziehung bei *Parametergruppe* beschrieben.

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

- Eine Parametergruppe kann mit beliebig vielen Ausprägungen versehen werden, was in Abb. 3 durch den Typ *Parametergruppe-Ausprägung* repräsentiert wird. Bei datenbankbasierter Ablage entspricht dies gerade einem Datensatz in der Tabelle. In XML-Dateien handelt es sich um das Vorkommen eines Knotens inklusive seiner enthaltenen Attribute, Knoten, und Elemente.

- *Parameterwert* bezeichnet eine konkrete Belegung eines Parameters, d.h. der Inhalt eines Datenbankfeldes bzw. der Inhalt eines Elements oder Attributs in einer XML-Datei. Ein Parameterwert bezieht sich immer auf einen Parameter und gehört zu einer oder keiner Parametergruppe-Ausprägung. Parameterwerte müssen den Restriktionen entsprechen, die durch Datentyp und Wertebereich ihres Parameters vorgegeben werden.

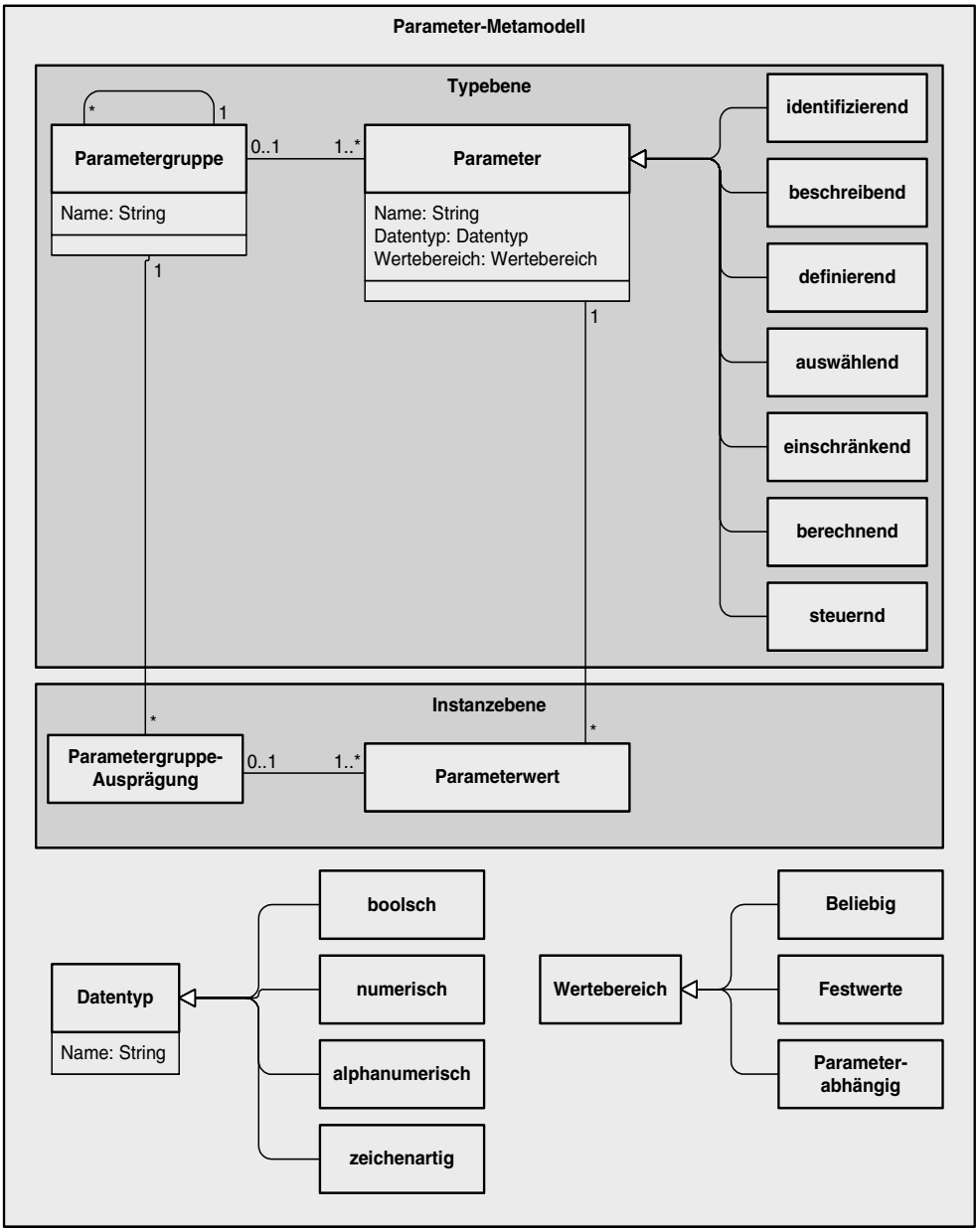

Abbildung 3: Metamodell zur Beschreibung von Parametern

Im Folgenden wird das Metamodell anhand eines Beispiels illustriert. Bei SAP R/3 werden zur Parameterpflege sogenannte Customizing-Aktivitäten (CA) verwendet [SAP02]. In Abb. 4 wird die CA dargestellt, mit welcher Torbelegungsprofile für die Anlieferung gepflegt werden. Auf dem ersten Bild werden die Parameter *Torbelegungsprofil (Schlüs-*

sel) und *Bezeichnung* erfasst – dabei können beliebig viele Torbelegungsprofile definiert werden. Pro Eintrag kann man auf ein Detailbild verzweigen. Dort werden weitere Parameter zu diesem Profil erfasst (z. B. *tägliche-Anlieferungszeit-von* oder *Zeitraster*). In einer folgenden CA wird jedem Lager ein Torbelegungsprofil zugewiesen, welches die Torbelegung für dieses Lager steuert.

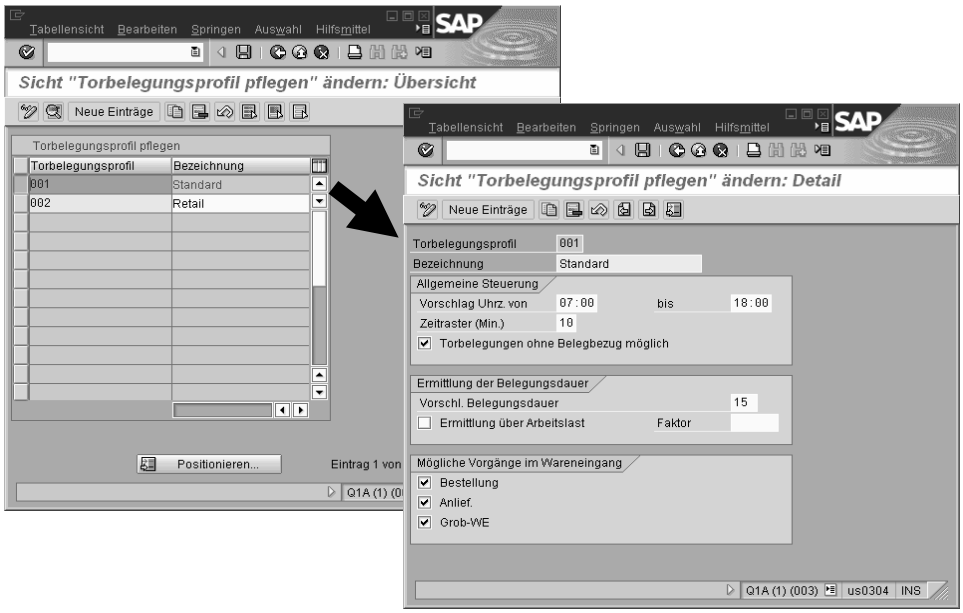

Abbildung 4: Customizing-Aktivität „Torbelegungsprofil pflegen“ (Quelle: SAP)

Die Daten zu den Torbelegungsprofilen werden in der Datenbanktabelle *TWAP1* abgelegt. Da die Pflegebilder der CA anschaulicher als die Tabelle sind, wird in Abb. 4 die CA dargestellt. Die Elemente der CA lassen sich folgendermaßen auf das Parameter-Metamodell abbilden: Bei dem *Torbelegungsprofil* handelt es sich um eine Parametergruppe. Zur Parametergruppe gehörende Parameter sind z. B. *Torbelegungsprofil (Schlüssel)*, *Bezeichnung*, *tägliche-Anlieferungszeit-ab* und *Zeitraster*. In Abb. 4 sind zwei Ausprägungen für die Parametergruppe definiert: *Standard* und *Retail*. Zur Ausprägung Standard hat der Parameter *tägliche-Anlieferungszeit-ab* den Parameterwert *07:00* und der Parameter *Zeitraster* den Parameterwert *10*. Bei der Ausprägung *Retail* könnten die Parameter mit anderen Werten belegt sein.

5 Klassifikation von Parameterauswirkungen

Dieses Kapitel enthält einen ersten Ansatz, welche Auswirkungen sich durch Parametrisierung ergeben können und was dazu in einer Komponentenspezifikation zu berücksichtigen ist. Der Forschungsbeitrag dieses Kapitels besteht darin, dass mögliche Parametrisierungsauswirkungen identifiziert und strukturiert werden.

5.1 Schema zur Beschreibung von Parameterauswirkungen

Um Auswirkungen einer Parametrisierung spezifizieren zu können, muss man solche Auswirkungen zunächst erfassen und beschreiben. Es wird vorgeschlagen, diese Beschreibung in Form einer tabellarischen Aufzählung vorzunehmen - Abb. 5 zeigt beispielhaft eine solche Tabelle. Dabei wird zu jedem Parameter erfasst, worauf und wie er sich auswirkt.

Um zu erfassen, worauf sich ein Parameter auswirkt, beginnen wir mit folgender Überlegung: Die in einer Spezifikation zu berücksichtigenden Auswirkungen eines Parameters sind gerade die Änderungen, welche sich (durch Setzen des Parameters) an spezifikationsrelevanten Sachverhalten der Komponente ergeben. Als *spezifikationsrelevanter Sachverhalt* wird dabei alles bezeichnet, was die Außensicht der Komponente beeinflusst. Dies können Objekte (z. B. Parameter), deren Eigenschaften (z. B. Wertebereich) und zugehörige Ausprägungen (z. B. Festwerte ‚A' und ‚B') sein. Um die Breite des Spektrums auszudrücken, wurde der allgemeine Begriff *Sachverhalt* gewählt. Beispiele solcher Sachverhalte sind die Signaturen von Methoden oder Zusicherungen beim Aufruf der Methoden. Die spezifikationsrelevanten Sachverhalte, auf die ein Parameter wirkt, dienen als erstes Ordnungskriterium bei der Beschreibung von Parameterauswirkungen.

Parameter	Betroffenes Objekt	Betroffener Sachverhalt	Wirkungs-art	Auswirkungen des Parameters (textuelle Beschreibung)
Auftrags-statuscodes	Auftrag (Typ)	Status (Attribut)	Definierend	Das Attribut *Status* kann nur solche Werte annehmen, die im Parameter *Auftragsstatuscodes* vordefiniert sind.
Auftragswert-höchstgrenze	Auftrag. Annehmen (Methode)	Vorbedingung	Steuernd	Auftrag wird nur angenommen, wenn Auftragswert <= Parameter *Auftrags-werthöchstgrenze*

Abbildung 5: Schema zur Beschreibung von Parameterauswirkungen

Die spezifikationsrelevanten Sachverhalte existieren nicht isoliert, sondern gehören (meist) zu einem übergeordneten Bezugsobjekt. (Beispielsweise gehört eine Methodensignatur oder eine Vorbedingung immer zu einer Methode.) Um die Übersichtlichkeit der Einträge zu erhöhen, wird das Bezugsobjekt ebenfalls als Ordnungskriterium verwendet (Spalte „Betroffenes Objekt" in Abb. 5). Welche Sachverhalte zu welchen Bezugsobjekten gehören, wird im Abschnitt 5.3 diskutiert.

Um zu erfassen, wie ein Parameter auf den betroffenen spezifikationsrelevanten Sachverhalt wirkt, wird eine textuelle Beschreibung erstellt. Die textuelle Beschreibung von Auswirkungen eignet sich jedoch nicht für eine Klassifikation. Eine solche Klassifikation wäre aber wünschenswert, da sie die Übersichtlichkeit und Verständlichkeit erhöhen würde. Daher wird vorgeschlagen, die Auswirkungen eines Parameters zusätzlich nach der *Wirkungsart* zu klassifizieren – siehe dazu Abschnitt 5.4.

5.2 Strukturierung spezifikationsrelevanter Sachverhalte

Im Abschnitt 5.3 werden für alle spezifikationsrelevanten Sachverhalte einer Komponente mögliche Auswirkungen einer Parametrisierung ermittelt. Dieser Abschnitt stellt eine mögliche Strukturierung spezifikationsrelevanter Sachverhalte vor, die ein systematisches Vorgehen ermöglicht.

Im Memorandum „Vereinheitlichte Spezifikation von Fachkomponenten" – ein im Rahmen des GI-Arbeitskreises WI-KobAS von Vertretern aus Forschung und Praxis erstellter Standardisierungsvorschlag – werden spezifikationsrelevante Sachverhalte identifiziert und mehreren Spezifikationsebenen zugeordnet [Tu02]. In [Ov04] wird der in Abb. 6 dargestellte Ordnungsrahmen für die Spezifikationsebenen vorgestellt.

	fachlich	**logisch**	**physisch**
statisch (Struktur)	Informationsobjekte (Datenmodell)	*Spezifikationsdatenmodell*, *Invarianten*, Datentypdeklarationen	Anwendbarkeit, Wartbarkeit, Übertragbarkeit
operational (Wirkungen)	Funktionen (Funktionsmodell)	Ereignisse, Methoden, Ausnahmen, Zusicherungen	Funktionalität, Zuverlässigkeit
dynamisch (Interaktionen)	Prozesse (Prozessmodell)	Interaktionsprotokolle	Effizienz

Abbildung 6: Ordnungsrahmen für spezifikationsrelevante Sachverhalte (in Anlehung an [Ov04])

Das Schema aus [Ov04] wurde um ein Spezifikationsdatenmodell erweitert: Um das Verhalten von Methoden einer Komponente zu beschreiben, ist es sinnvoll – und oft notwendig – auf die Daten Bezug zu nehmen, die von der Komponente verwaltet werden. Dies geschieht mit Hilfe eines Spezifikationsdatenmodells, welches die extern sichtbaren Daten einer Komponente auf logischer Ebene darstellt (und nicht mit der tatsächlichen Implementierung übereinstimmen muss). Die Verwendung eines solchen konzeptionellen Modells bei der Spezifikation von Fachkomponenten [Tu02] geht auf Vorschläge aus [Ac01] zurück. Ähnliche Konstrukte sind z. B. die Interface Information Models bei der Modellierung von Komponenten [CD01].

5.3 Parameterauswirkungen auf spezifikationsrelevante Sachverhalte

In diesem Abschnitt wird diskutiert, wie sich die spezifikationsrelevanten Sachverhalte einer Komponente durch Parametrisierung ändern können. In dieser Arbeit beschränken wir uns auf die Untersuchung der Objekte, die der logischen Ebene in Abb. 6 zugeordnet sind. Eine Parametrisierung erfolgt typischerweise mit dem Ziel, die genaue Funktionsweise der Komponente einzustellen – die meisten Auswirkungen einer Parametrisierung ergeben sich dadurch auf der logischen Ebene. Die Analyse der anderen Ebenen ist Richtung zukünftiger Forschung.

Der Untersuchung liegt folgendes Vorgehen zugrunde: Die Objekte der statisch-logischen, der operational-logischen und der dynamisch-logischen Ebene wurden getrennt betrachtet. Für jede Ebene wurden die spezifikationsrelevanten Sachverhalte aus

[Tu02] und [Ov04] ermittelt. Danach wurde überlegt, wie sich die Sachverhalte durch Parameter ändern können. Dabei konnte zum einen auf eigene Vorarbeiten zurückgegriffen werden: In [Ac03a] erfolgte die Analyse und Spezifikation einer parametrisierbaren Beispielkomponente und in [Ac02] wurde ein Teilbereich des Customizings von SAP R/3 analysiert. In beiden Fallstudien treten Parameterauswirkungen implizit auf und wurden im Rahmen dieser Arbeit explizit identifiziert. Zum anderen standen Vorarbeiten zur Verfügung, die sich mit Variabilität in Referenzmodellen [Sc98] und im PPS-Customizing [Di97] beschäftigen – diese Vorarbeiten wurden (soweit sinnvoll) auf Softwarekomponenten übertragen. Die Strukturierung der so gesammelten Auswirkungen erfolgt anhand eines Schemas, das aus Abb. 5 abgeleitet wurde.

5.3.1 Statisch-logische Ebene

Folgende spezifikationsrelevante Sachverhalte lassen sich der statisch-logischen Ebene zuordnen:

- Datentypdeklarationen
- Elemente des Spezifikationsdatenmodells: Typen (beschreiben die extern sichtbaren Daten einer Komponente, vergleichbar zu Entitätstypen), Attribute, Spezialisierungen, Beziehungen (Assoziationen)
- Invarianten (formulieren Bedingungen an die Modellelemente)

Änderungen an Datentypdeklarationen sind bei datenbasierter Parametrisierung nicht relevant - alle anderen spezifikationsrelevanten Sachverhalte können sich ändern.

Alle Sachverhalte auf der statisch-logischen Ebene lassen sich zu einem *Typ* (als Bezugsobjekt) zuordnen: *Attribute*, *Spezialisierungen* und *Invarianten* gehören immer zu einem Typ, woraus sich die Zuordnung kanonisch ergibt. Beziehungen bestehen immer zwischen mehreren Typen. In vielen Fällen gibt es bei einer Beziehung einen führenden Typ, dem die Beziehung zugeordnet werden kann. In allen anderen Fällen kann die Beziehung allen betroffenen Typen zugeordnet werden – im Beschreibungsschema (Abb. 5) wäre der entsprechende Eintrag dann mehrfach vorhanden.

Betroffenes Objekt	Betroffener Sachverhalt	Mögliche Auswirkungen eines Parameters
Typ	Typdefinition	Setzt Typ auf aktiv/inaktiv (auswählend)
Typ	Attribut	Setzt Attribut auf aktiv/inaktiv (auswählend)
Typ	Beziehung	Setzt gesamte Beziehung auf aktiv/inaktiv (auswählend) Schränkt Kardinalität der Beziehung ein (einschränkend)
Typ	Spezialisierung	Wählt erlaubte Subtypen aus (auswählend)
Typ	Invariante (für Typinstanzen)	Schränkt Anzahl möglicher Instanzen ein (einschränkend) Geht in Schlüssel eines Typs ein (identifizierend)

Typ	Invariante (für Attributwerte)	Legt Wertebereich eines Attributs fest (auswählend) Schränkt Wertebereich eines Attributs ein (einschränkend) Geht in Berechnung eines abgeleiteten Attributs ein (berechnend)
Typ	Invariante (für Beziehung)	Entscheidet auf Instanzebene über Vorkommen der Beziehung (einschränkend)
Typ	Invariante (weitere)	Erzwingt Konsistenzbedingung (einschränkend)

Abbildung 7: Mögliche Auswirkungen einer Parametrisierung auf statisch-logischer Ebene

Um mögliche Auswirkungen einer Parametrisierung zu ermitteln, können neben den Fallstudien [Ac03a] und [Ac02] Ergebnisse aus der Referenzmodellierung verwendet werden. Dort werden mögliche Variabilitäten in Datenmodellen identifiziert [Sc98], die sich hier auf die Struktur des Spezifikationsdatenmodells übertragen lassen.

Im Ergebnis ergibt sich eine Liste typischer Auswirkungen, die in Abb. 7 den spezifikationsrelevanten Sachverhalten zugeordnet wurden. Die Identifikation möglicher Auswirkungen ist unabhängig von einer später zu verwendenden Spezifikationstechnik. Man beachte jedoch, dass die Zuordnung der Auswirkungen zu Sachverhalten abhängig von der Spezifikationstechnik ist. So gehört die Bedingung „Das Attribut XY darf den Wert 100 nicht überschreiten.“ zwar logisch zum Attribut XY, wird aber in der UML über eine Invariante (für das Attribut) ausgedrückt. Wir lehnen uns an die UML [OMG04] an, da sie der de facto Standard bei der Modellierung von Komponenten ist. Die angegebene Bedingung betrifft also nicht die Attributdefinition an sich, sondern eine Invariante zum Attributwert. Um den semantischen Bezug nicht verloren gehen zu lassen, wurden die Invarianten (wie in Abb. 7 angegeben) unterteilt.

Zu beachten ist noch, dass die in Abb. 7 angegebenen Einträge zwar typische, häufig auftretende Auswirkungen darstellen, die Liste aber keineswegs abschließend ist. Gerade bei den Invarianten sind beliebig komplexe Bedingungen vorstellbar. Für eine bessere Strukturierung eignet sich eine Klassifikation nach der Wirkungsart (z. B. identifizierend, steuernd, berechnend) – vgl. dazu Abschnitt 5.4.

5.3.2 Operational-logische Ebene

Auf operational-logischer Ebene finden sich folgende spezifikationsrelevante Sachverhalte:

- Interfacedeklarationen
- Methoden-, Ausnahmen- und Ereignisdeklarationen
- Zusicherungen (Vor- und Nachbedingungen bei Methoden)

Die angegebenen Deklarationen kann man in dem Sinne als parameterabhängig betrachten, dass ganze Interfaces, Methoden, Ausnahmen und Ereignisse durch Parameter auf aktiv oder inaktiv gesetzt sein können. Eine parameterabhängige Veränderung der Struktur von Deklarationen betrachten wir nicht, da eine dynamische Veränderung statischer Schnittstellendefinitionen bei heutigen Komponententechnologien nicht üblich ist. (Aus

diesem Grund haben wir in 5.3.1 auch keine Veränderungen an Typdeklarationen betrachtet.) Parameter können jedoch sehr stark das Ergebnis von Methodenaufrufen oder das Auslösen von Ereignissen und Ausnahmen beeinflussen, weshalb bei den Vor- und Nachbedingungen vielfältige Parameterauswirkungen auftreten können.

Alle mit Methoden zusammenhängenden Sachverhalte (Aufrufparameter, Resultat, Vor- und Nachbedingungen) haben als Bezugsobjekt ihre zugehörige Methode – die Deklarationen von Interfaces, Ausnahmen und Ereignissen stehen für sich selbst.

Betroffenes Objekt	**Betroffener Sachverhalt**	**Mögliche Auswirkungen eines Parameters**
Interface		Setzt Interface komplett auf aktiv/inaktiv (auswählend)
Ereignis		Setzt Ereignis komplett auf aktiv/inaktiv (auswählend)
Ausnahme		Setzt Ausnahme komplett auf aktiv/inaktiv (auswählend)
Methode		Setzt Methode auf aktiv/inaktiv (auswählend)
Methode	Vorbedingung (allgemein)	Beeinflusst Voraussetzungen eines Methodenaufrufs (einschränkend)
Methode	Vorbedingung (für Inputparameter)	Legt Wertebereich fest (auswählend) Schränkt Wertebereich ein (einschränkend) Legt Defaultwert fest (auswählend) Legt fest, ob optional oder obligatorisch (auswählend)
Methode	Nachbedingung (für Ausnahme)	Beeinflusst Auslösen einer Ausnahme (steuernd)
Methode	Nachbedingung (für Ereignis)	Beeinflusst Auslösen eines Ereignisses (steuernd)
Methode	Nachbedingung (allgemein)	Legt Verfahrensauswahl fest (auswählend) Legt Struktur der abgelegten Daten fest (auswählend) Schränkt Werte der abgelegten Daten ein (einschränkend) Liefert Vergleichswert für Entscheidungen; z. B. Toleranzgrenzen oder Höchstwerte (steuernd)
Methode	Nachbedingung (für Output- bzw. Returnparameter)	Geht in Berechnung des Wertes ein (berechnend) Legt Wertebereich fest (auswählend) Schränkt Wertebereich ein (einschränkend)

Abbildung 8: Mögliche Auswirkungen einer Parametrisierung auf operational-logischer Ebene

Abb. 8 enthält eine Liste typischer Auswirkungen, die sich auf Sachverhalte der operational-logischen Ebene ergeben können. Diese Auswirkungen wurden auch hier aus bestehenden Vorarbeiten abgeleitet: Dies waren die Fallstudien [Ac03a] und [Ac02] sowie Ergebnisse aus [Di97], wo die Auswirkungen dispositiver PPS-Parameter auf SAP R/3-PPS analysiert wurden und ein besonderer Schwerpunkt auf Aspekten lag, die in Abb. 6 der operationalen Ebene zuzuordnen sind.

5.3.3 Dynamisch-logische Ebene

Auf dynamisch-logischer Ebene finden sich in [Ov04] Interaktionsprotokolle. Um Auswirkungen einer Parametrisierung beschreiben zu können, wäre es wünschenswert, (wie in den Abschnitten zuvor) detailliert die spezifikationsrelevanten Sachverhalte zu kennen. Diese hängen jedoch von der verwendeten Beschreibungstechnik ab. Für Interaktionsprotokolle werden in der Literatur verschiedene Techniken vorgeschlagen: endliche reguläre Automaten [Ni93], Petri-Netze [Pe62] oder temporale Logiken [CT00]. Für eine Diskussion zur Eignung dieser Techniken für Interaktionsprotokolle sei auf [Ov04] und [Ni93] verwiesen. Eine weitere Alternative zur Beschreibung dynamischer Aspekte sind Prozessmodelle und die dafür gängige Technik der erweiterten Ereignisgesteuerten Prozesskette (eEPK), vgl. z. B. [KNS92].

Für unsere Analyse gehen wir im Folgenden von den eEPK aus, da für diese Vorarbeiten existieren, die sich hier verwenden lassen: In der Referenzmodellierung werden Variabilitäten identifiziert, die in Prozessmodellen modellierungsrelevant sind [Sc98].

Das Bezugsobjekt ist immer der Prozess. Eine Prozessdefinition besteht aus Funktionen, Ereignissen, Verknüpfungen und Prozesssträngen. Überträgt man die in [Sc98] vorgeschlagenen Konstrukte zur Beschreibung von Variabilität auf unseren Fall, ergibt sich die in Abb. 9 dargestellte Liste möglicher Parameterauswirkungen.

Betroffenes Objekt	Betroffener Sachverhalt	Mögliche Auswirkungen eines Parameters
Prozess		Setzt Prozess komplett auf aktiv/inaktiv (auswählend)
Prozess	Ereignis	Beeinflusst Auslösen eines Ereignisses (steuernd) Bestimmt Entscheidungsalgorithmus (steuernd) Liefert Vergleichswert für Entscheidungen; z. B. Toleranzgrenzen oder Höchstwerte (steuernd)
Prozess	Verknüpfungen	Verändert Art des Junktors (steuernd)
Prozess	Prozessstrang	Eliminiert gesamten Prozessstrang (auswählend)

Abbildung 9: Mögliche Auswirkungen einer Parametrisierung auf dynamisch-logischer Ebene

Man beachte, dass das Eliminieren von Funktionen oder Ereignissen im Prozessablauf nicht vorgesehen ist, da dies eine weitergehende Anpassung des Modells erfordert. Soll ausgedrückt werden, dass abhängig von einem Parameter entweder Funktion A und Funktion B oder nur Funktion B ausgeführt wird, kann dies durch zwei parallele Prozessstränge mit einer geeigneten Bedingung bei der Ausgangsverknüpfung modelliert werden.

5.4 Wirkungsart zur Klassifikation von Parameterauswirkungen

Im Abschnitt 5.3 wurde eine Vielzahl von Möglichkeiten identifiziert, wie sich Parameter auf die spezifikationsrelevanten Sachverhalte einer Komponente auswirken können. Dabei wurde jedoch festgestellt, dass die angegebenen Möglichkeiten (vor allem bei den

Invarianten, Vor- und Nachbedingungen) nur exemplarisch und nicht vollständig sind. Aus diesem Grund eignen sich die aufgeführten Auswirkungen in dem Detailgrad nicht für eine Klassifikation. Andererseits ist eine Klassifikation, wie sich Parameter auswirken können, wünschenswert, da sie die Übersichtlichkeit und Verständlichkeit erhöht.

Deshalb wurde im Abschnitt 5.1 vorgeschlagen, die Auswirkungen eines Parameters nach der *Wirkungsart* zu klassifizieren. In Abb. 10 finden sich mögliche Ausprägungen für die Wirkungsart inklusive einer erklärenden Beschreibung. Teile dieser Klassifikation gehen auf [DMH99] zurück.

Wirkungsart	Beschreibung der Wirkungsart
Definierend	Definiert alle erlaubten Ausprägungen
Einschränkend	Schränkt erlaubte Ausprägungen ein
Auswählend	Wählt aus vorgegebenen Ausprägungen die erlaubten aus und deaktiviert die anderen - Auswahl erfolgt zur Konfigurationszeit und gilt für die gesamte Laufzeit
Steuernd	Beeinflusst Ablauf von Prozessen oder das Ergebnis von Entscheidungen bei Funktionsabarbeitung und Prozessablauf
Berechnend	Geht als Variable in Berechnung eines Datenfeldes ein – beeinflusst aber nicht das Berechnungsverfahren
Identifizierend	Dient zur Identifikation von Ausprägungen
Beschreibend	Beschreibt Ausprägungen – hat aber keine struktur- oder ablaufrelevanten Auswirkungen

Abbildung 10: Wirkungsarten zur Klassifikation von Parameterauswirkungen

Beispiele zu den Wirkungsarten finden sich in den Abb. 7 – 9 in Klammern zur textuellen Beschreibung. Man beachte, dass in Abb. 10 der Begriff *Ausprägung* für die Instanzen der jeweiligen spezifikationsrelevanten Sachverhalte steht.

6 Ausblick

Diese Arbeit beschäftigte sich mit der Frage, wie die – mit einer Parametrisierung zusammenhängenden - spezifikationsrelevanten Sachverhalte beschrieben werden können. Dazu wurde für die Parameter selbst ein Metamodell entwickelt. Für die Parameterauswirkungen liegt ein erster Ansatz vor, wie diese mit Hilfe eines Klassifikationsschemas beschrieben werden können.

In Zukunft muss das Klassifikationsschema der Auswirkungen auf die fachlichen und physischen Ebenen erweitert sowie anhand parametrisierbarer Komponenten verifiziert werden. Außerdem muss – auf den Ergebnissen dieser Arbeit aufbauend – ein Vorgehen entwickelt werden, wie der Parametrisierungsspielraum von Softwarekomponenten spezifiziert werden kann.

Literaturverzeichnis

[Ac01] *Ackermann, J.*: Fallstudie zur Spezifikation von Fachkomponenten. In: *K. Turowski (Hrsg.):* 2. Workshop Modellierung und Spezifikation von Fachkomponenten. Bamberg 2001, S. 1-66.

[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.

[Ac03a] *Ackermann, J.*: Zur Spezifikation der Parameter von Fachkomponenten. In: *K. Turowski (Hrsg.)*: 5. Workshop Komponentenorientierte betriebliche Anwendungssysteme (WKBA 5). Augsburg 2003, S. 47-154.

[Ac03b] *Ackermann, J.*: Specification Proposals for Customizable Business Components. In: *Overhage, S.; Turowski, K. (Hrsg.)*: Proceedings 1st International Workshop Component Engineering Methodology. Erfurt 2003, S. 51-62.

[AR00] *Appelrath, H.-J.; Ritter, J.*: SAP R/3 Implementation: Methods and Tools. Springer, Berlin, Heidelberg 2000.

[AT03] *Ackermann, J.; Turowski, K.*: Specification of Customizable Business Components. In: *Chroust, G.; Hofer, S. (Hrsg.)*: Euromicro Conference 2003. Belek-Antalya (Türkei) 2003, S. 391-394.

[Be00] *Bergner, K.; Rausch, A.; Sihling, M.; Vilbig, A.:* 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.

[Br98] *Bracha, G.; Odersky, M.; Stoutamire, D.; Wadler, P.*: Making the future safe for the past: Adding Genericity to the Java Programming Language. In: Proceedings of the Conference on Object-Oriented Programming, Systems, Languages and Applications (OOPSLA'98). Vancouver 1998.

[CD01] *Cheesman, J.; Daniels, J.:* UML Components. Addison-Wesley, Boston 2001.

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

[CT00] *Conrad, S.; Turowski, K.*: Vereinheitlichung der Spezifikation von Fachkomponenten auf der Basis eines Notationsstandards. In: *Ebert, J.; Frank, U. (Hrsg.)*: Modelle und Modellierungssprachen in Informatik und Wirtschaftsinformatik: Beiträge des Workshops "Modellierung 2000". St. Goar 2000, S. 179-194.

[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.

[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.

[GP02] *Goldfarb, C.F.; Prescod, P.*: XML Handbook. 4th Edition. Prentice Hall, 2002.

[Ho96] *Hollingsworth, D.*: Workflow Management Coalition: The Workflow Reference Model. Document TC00-1003, Version Draft 1.1. WfMC, Winchester 1996.

[JB96] *Jablonski, S.; Bussler, C.*: Workflow Management: Modeling Concepts, Architecture and Implementation. International Thomson Computer Press, London 1996.

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

[KNS92] *Keller, G.; Nüttgens, M.; Scheer, A.-W.*: Semantische Prozessmodellierung auf der Grundlage "Ereignisgesteuerter Prozessketten (EPK)". In: *Scheer, A.-W. (Hrsg.)*: Veröffentlichungen des Instituts für Wirtschaftsinformatik (IWi), Universität des Saarlandes, Heft 89. Saarbrücken 1992.

[KT98] *Keller, G.; Teufel, T.:* SAP R/3 Process-oriented Implementation : Iterative Process Prototyping. Addison Wesley Longman, Harlow 1998.

[LM93] *Ludwig, L.; Mertens, P.*: Die Einstellung der Parameter eines Materialwirtschaftssystems in einem Unternehmen der Hausgeräteindustrie. In: Wirtschaftsinformatik 35 (1993) 5, S. 446-454.

[MBL97] *Myers, A.C.; Bank, J.A.; Liskov, B.*: Parameterized Types for Java. In: 24th ACM Symposium on Principles of Programming Languages. New York 1997, S. 132-145.

[Mc68] *McIlroy, M.D.*: Mass Produced Software Components. In: *Naur, P.; Randell, B. (Hrsg.)*: Software Engineering: Report on a Conference by the NATO Science Committee. Brussels 1968, S. 138-150.

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

[Ni93] *Nierstrasz, O.*: Regular Types for Active Objects. In: Proceedings of the OOPSLA 93. ACM SIGPLAN Notices 28 (1993) 10, S. 1-15.

[OMG04] *OMG (Hrsg.):* Unified Modeling Language: UML 2.0 Superstructure. URL: http://www.omg.org/uml, Abruf am 2004-09-24.

[Ov04] *Overhage, S.*: Zur Spezifikation von Komponenten der Informationssystementwicklung mit Softwareverträgen. In: *M. Rebstock (Hrsg.)*: Tagungsband Modellierung betrieblicher Informationssysteme - MobIS 2004. GI-Edition. Essen 2004, S. 3-20.

[Pe62] *Petri, C.A.*: Fundamentals of a Theory of Asynchronous Information Flow. In: Information Processing 62, IFIP, 1962, S. 386-391.

[Pi94] *Pietsch, M.*: Beiträge zur Konfiguration von Standardsoftware am Beispiel der Geschäftsprozeßimplementierung und der Parameterinitialeinstellung eines großintegrierten PPS-Systems. Dissertation, Nürnberg 1994.

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

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

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

[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.

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

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

[Tu02] *Turowski, K. (Hrsg.):* Vereinheitlichte Spezifikation von Fachkomponenten: Memorandum des Arbeitskreises 5.10.3 Komponentenorientierte betriebliche Anwendungssysteme, Februar 2002. Universität Augsburg, Augsburg 2002.