Formale Modellierung von Komponenten und Abhängigkeiten zur Konfiguration von Product-Service-Systems

Abstract Aus der zunehmenden Verflechtung von Produkten und Dienstleistungen zu gemeinsamen Angeboten – sogenannten Product-Service-Systems (PSSs) – folgen hinsichtlich der Modellierung dieser Systeme verschiedene Anforderungen. Mit steigender Komplexität der PSSs ist beispielsweise die softwaretechnische Unterstützung zur Zusammenstellung individueller Angebote, sogenannter Konfiguratoren, von großer Bedeutung. Voraussetzung für den Einsatz von Konfiguratoren ist zunächst die Beschreibung der Struktur der Produkte und Dienstleistungen. Darüber hinaus sind die komplexen logischen Verflechtungen zwischen diesen beiden Strukturen zu definieren. Der nachfolgende Beitrag entwickelt ausgehend von einer existierenden Modellierungsmethode für Dienstleistungen eine holistische Beschreibungssprache für PSSs. Ausgehend von Untersuchungen zu Abhängigkeiten innerhalb von PSSs werden formal spezifizierte Regeln aufgestellt, mit denen sich diese Abhängigkeiten beschreiben lassen.

1 Einleitung

Die in den vergangenen Jahren stark gestiegene wirtschaftliche Bedeutung von Dienstleistungen (Hildebrandt & Klostermann, 2007) begünstigt auch eine zunehmende Konvergenz von Produkten und Dienstleistungen. Der darauf basierende Ansatz, Produkte und Dienstleistungen in sogenannten Product-Service-Systems (PSSs) zu verbinden (Morelli, 2002) und gebündelt anzubieten, hat daher in der Vergangenheit verstärkt an Relevanz gewonnen (Mont, 2002; Stille, 2003; Knackstedt, et al., 2008). Treiber dafür sind beispielsweise eine Fokusverschiebung vom Erwerb einzelner Produkte bzw. Dienstleistungen hin zum Erwerb von Lösungen bzw. Funktionalitäten (Baines, et al., 2007; Isaksson, et al., 2009) und die Nutzung von Dienstleistungen zur Abgrenzung des Angebotsportfolios gegenüber Wettbewerbern (Knackstedt, et al., 2008).

Obgleich die Idee des PSSs bereits ihren Platz in der wissenschaftlichen Diskussion gefunden hat, ist ihr Durchdringungsgrad in der Industrie bislang begrenzt (Baines, et al., 2007). Dies liegt an einer Vielzahl von Herausforderungen, welche sich aufgrund einer stärkeren holistischen Betrachtungsweise von Produkten und Dienstleistungen ergeben.

Die Abhängigkeiten zwischen Produkten und Dienstleistungen resultieren bei PSSs in neuen Herausforderungen hinsichtlich der Gestaltung von Geschäftsprozessen, da die Einführung neuer Abteilungen erforderlich bzw. die Zusammenar-

beit zwischen existierenden Abteilungen zu intensivieren ist (Mont, 2002). Aufgrund der Abhängigkeiten ergibt sich weiterhin eine gesteigerte Komplexität, da so die Gestaltung von PSSs deutlich aufwändiger ist als die abgegrenzte Betrachtung von Produkten und Dienstleistungen (Mont, 2002; Isaksson, et al., 2009). So führt dies z.B. zu erweiterten Anforderungen an das Portfoliomanagement, da ggf. Änderungen im Produktportfolio Modifikationen im Dienstleistungsportfolio nach sich ziehen und umgekehrt. Ebenso ist der Anspruch der Kunden nach individuellen Angeboten auch im Bereich PSS immanent, wodurch sich die Problematik der kundenindividuellen Angebotskonfiguration auf die beiden Felder Produkte und Dienstleistungen erweitert.

Das Ziel, trotz der hohen Komplexität PSSs handhabbar zu gestalten, erfordert geeignete softwaretechnische Unterstützung (Dietze, 2008). Besonderes Augenmerk ist dabei neben prozessualen Aspekten der Beschreibung von Abhängigkeiten zwischen den Produkt- und Dienstleistungsbereichen zu widmen (Becker, et al., 2008). Der Fokus der nachfolgenden Untersuchungen liegt demzufolge auf Fragestellungen der Konfiguration und damit der Individualisierung von PSSs. Um diese IT-gestützt konfigurieren zu können, bedarf es der vollständigen und holistischen Modellierung des PSSs, sowohl das Produktmodell als auch das Dienstleistungsmodell umfassend. Um eine holistische Betrachtungsweise inklusive der Beschreibung der Abhängigkeiten innerhalb des PSSs zu ermöglichen, können drei Herangehensweisen gewählt werden:

1. Die Nutzung von jeweils innerhalb der Domänen von Produktion und Dienstleistung gebräuchlichen Modellierungsmethoden. Abhängigkeiten zwischen den Modellen sind dabei zusätzlich zu beschreiben.
2. Die holistische Modellierung von Produkten, Dienstleistungen sowie deren Abhängigkeiten in einem Modell.
3. Die Transformation existierender Modelle (entweder zur Beschreibung von Produkten oder von Dienstleistungen) in eine holistische Modellierungsmethode.

Umfangreiche Abhängigkeiten zwischen und Produkten und Dienstleistungen sowie der Wunsch nach einer möglichst homogenen Modellierungslandschaft (Weber, et al., 2004) sprechen gegen den ersten Ansatz.

Nachfolgend wird zunächst ausgehend von existierenden Arbeiten eine Methode zur Spezifikation von Dienstleistungen vorgestellt. Diese basiert auf der komponentenbasierten Modellierung von Teilleistungen. Der konzeptionelle Ursprung der Komponentisierung in Teilleistungen respektive Bestandteile liegt im Bereich der Produktbeschreibungen. Folglich wird die komponentenbasierte Beschreibung auch auf die Modellierung von PSSs angewendet. Ausgehend von der integrierten Darstellung in einem PSS werden verschiedene Abhängigkeiten zwischen Produkt- und Dienstleistungskomponenten aus der Literatur abgeleitet und formalisiert. Der Fokus liegt dabei auf der leichtgewichtigen Darstellung der Zusammen-

hänge zwischen den Komponenten. Zur Verdeutlichung der praktischen Anwendbarkeit werden weiterhin Transformationen zwischen existierenden Methoden zur Beschreibung von Produkten in die holistische PSS-Notation vorgestellt. Dadurch wird gezeigt, dass sich die zahlreich existierenden und bereits etablierten Modellierungsverfahren (Isaksson, et al., 2009) der Sachgüterdomäne weiterverwenden lassen.

2 Ein Metamodell zur Dienstleistungsmodellierung

Mit der Modellierung von Dienstleistungen können verschiedene Ziele verfolgt werden. So können Prozessabläufe beschrieben, Ressourcen allokiert oder kundenspezifische Konfigurationen von Dienstleistungsangeboten auf Basis eines Modells erstellt werden. Aufgrund des Anstiegs der wirtschaftlichen Relevanz kundenindividueller Angebote wurden in vorangegangenen Arbeiten Methoden, Modelle und Tools zur Konfiguration von Dienstleistungen entwickelt (Böttcher & Fähnrich, 2009; Böttcher & Klingner, 2011; Becker, et al., 2011). Durch die Unterstützung von Kennzahlen, sogenannter Key Performance Indicators (KPIs), wird gleichzeitig die Bewertung der Konfiguration hinsichtlich Produktivitäts- bzw. Wirtschaftlichkeitsaspekten möglich (Böttcher, et al., 2011). Unterstützt durch einen vergleichbaren Bedarf der Individualisierung von Angeboten im Bereich von PSSs, stellt sich die Frage, inwieweit die vorgeschlagene Methode zur Dienstleistungsmodellierung zur Beschreibung von PSSs geeignet ist und um welche Aspekte diese gegebenenfalls zu erweitern ist. Dazu werden im Folgenden zunächst die bisherigen Arbeiten in aggregierter Form dargestellt.

2.1 Konzepte

Die Definition des Metamodells zur Beschreibung von Dienstleistungen erfordert es, eine Reihe von Konzepten zu definieren und in eine Ordnung zueinander zu bringen. Ziel des hier vorgestellten Metamodells ist die Abdeckung der in (Böttcher & Fähnrich, 2009) vorgestellten vier Dimensionen zur Beschreibung von Dienstleistungen. Diese umfassen ein Komponentenmodell zur Darstellung der Funktionalität einzelner Bestandteile einer Dienstleistung, ein Ressourcenmodell zur Beschreibung von Ressourcen und deren Abhängigkeiten voneinander, ein Produktmodell, um hierarchische Abhängigkeiten zwischen den Komponenten zu definieren sowie ein Prozessmodell zur Spezifikation der Reihenfolge der Abarbeitung von Komponenten.

Das hier vorgestellte Metamodell fokussiert die Beschreibung von Komponenten und deren logische Abhängigkeiten untereinander. Deshalb wird im Folgenden nicht weiter auf das Ressourcenmodell eingegangen. Die notwendigen Konzepte

werden mit Hilfe der Aussagen- und Prädikatenlogik formalisiert. Obwohl dies anfangs einen Mehraufwand bedeutet, ergeben sich daraus zwei wesentliche Vorteile. Einerseits ermöglicht die Darstellung basierend auf formaler Logik eine eindeutige und unmissverständliche Beschreibung der verwendeten Konzepte, da die Logik im Gegensatz zu natürlicher Sprache eine formal definierte Semantik besitzt. Gleichzeitig ergeben sich daraus aber auch einfache Möglichkeiten zur Erweiterung und Anpassung des Metamodells an spezifische Gegebenheiten. Dies ist insbesondere im Hinblick auf die in dieser Arbeit angestrebte Erweiterung des Modells zur Verwendung im Kontext von PSSs von Bedeutung.

Die Konzepte zur komponentenbasierten Beschreibung von Dienstleistungen lassen sich wie folgt zusammenfassen:

- Komponenten werden genutzt, um die Funktionalität einzelner Schritte bei der Erbringung einer Dienstleistung zu definieren. Die Menge aller Komponenten ergibt das Komponentenmodell.
- Hierarchische und nichthierarchische Abhängigkeiten zeigen Zusammenhänge zwischen Komponenten sowie deren zeitliche und logische Abhängigkeiten. Mit diesen Beziehungen lassen sich sowohl das Produkt- als auch das Prozessmodell repräsentieren.
- Kardinalitäten ermöglichen eine genauere Beschreibung der logischen Beziehungen zwischen Komponenten. Dadurch lässt sich das Produktmodell um zusätzliche Semantik anreichern.
- Konfigurationen ermöglichen die Zusammensetzung einzelner standardisierter Komponenten zu kundenspezifischen Gesamtleistungen.

Die strukturierte Zusammenstellung verschiedener Dienstleistungskomponenten ergibt das Dienstleistungsportfolio, welches als Tupel DP = (C, G) mit Komponenten C und einem Konfigurationsgraphen G zur Definition hierarchischer Abhängigkeiten dargestellt werden kann.

2.2 Komponenten

Eine Dienstleistungskomponente stellt eine wohldefinierte, abgrenzbare Funktionalität dar, die Ressourcen verbraucht und verändert. Die Funktionalität wird über präzise spezifizierte Schnittstellen angeboten. Komponenten sind somit nach (Böttcher & Fähnrich, 2009) Teilprozesse einer Gesamtleistung. Analog zu Softwarekomponenten sollten Dienstleistungskomponenten logisch und funktional zusammengehörige Aktivitäten beschreiben und damit eine hohe Kohäsion und geringe Kopplung besitzen, um effizient einsetzbar zu sein (Brocke, et al., 2010).

Eine kundenspezifische Konfiguration von Komponenten wird ermöglicht, indem eine Menge von Komponenten zu einer neuen Komponente (bzw. einer Gesamtleistung) zusammengesetzt wird. Zur Strukturierung eines existierenden Portfolios ist es weiterhin möglich, Komponenten in kleinere Einheiten zu zerlegen, die getrennt voneinander betrachtet werden können. Die Funktionalität einer

Komponente, die aus mehreren Teilkomponenten besteht, ist die Summe der Funktionalitäten der zusammengesetzten Komponenten (Böttcher & Klingner, 2011). Die Komponenten eines Portfolios sind in der Menge *C* enthalten.

Für einzelne Komponenten lassen sich Produktivitätskennzahlen (KPIs) definieren. Diese sind Indikatoren zur Messung der Produktivität einer Leistung. Durch die Zerlegung einer Gesamtleistung in Teilleistungen lassen sich KPIs spezifischer an die eigentliche Funktionalität einer Komponente anpassen. Detaillierte Ausführungen zur Definition der KPIs finden sich in (Böttcher & Klingner, 2011). An dieser Stelle ist nur von Relevanz, dass jeder Komponente eine KPI (identifiziert durch einen Namen aus der Menge *KPIName*) zugewiesen werden kann, die sich durch eine Formel berechnet (evtl. mittels Referenzierung anderer KPIs) oder durch eine Konstante gebildet ist. Dazu wird die Abbildung *KPI* verwendet

$$KPI: C \times KPIName \rightarrow Formula$$

2.3 Hierarchische Beziehungen zwischen Komponenten

Ziel der Komponentisierung von Dienstleistungen ist es u.a., komplexe Gesamtleistungen anhand des Zusammenspiels einzelner, weniger komplexer Teilleistungen (der Komponenten) zu beschreiben, um besser auf spezifische Kundenbedürfnisse eingehen zu können und dem gestiegenen Wettbewerbsdruck im Dienstleistungssektor zu begegnen (Sundbo, 1994). Um dieses Zusammenspiel zu spezifizieren, ist es notwendig, Beziehungen zwischen den Komponenten aufzustellen. Die Darstellung hierarchischer Beziehungen zwischen Komponenten mit Gozintographen (Vazsonyi, 1954) wird bereits im Industrial Engineering und in Feature-Modellen (Mendonca, et al., 2009) im Software Engineering erfolgreich verwendet.

Im Gegensatz zu Software- bzw. Produktkomponenten werden Dienstleistungskomponenten als Prozessbestandteile angesehen, deren Aktivitäten dazu dienen, eine wohldefinierte Funktionalität umzusetzen (Geum, et al., 2012). Demzufolge wird die Zerlegung von Dienstleistungskomponenten in Teilkomponenten als Verfeinerung von Prozessen angesehen. Formal lässt sich die Hierarchiebeziehung innerhalb einer Dienstleistung als gerichteter, azyklischer Graph *G*, im Folgenden Konfigurationsgraph genannt, beschreiben. Die Menge der Knoten *V* des Graphen enthält dabei neben den eigentlichen Komponenten *C* auch sogenannte Konnektoren *Co*. Diese Konnektoren dienen als Hilfskonstrukte, um die Verbindung zwischen Komponenten mit zusätzlicher Semantik anzureichern. Dies wird im nächsten Abschnitt im Rahmen der Definition von Kardinalitäten genauer gezeigt.

Der Konfigurationsgraph ist azyklisch, da eine Komponente sich nicht selber enthalten darf (übersetzt in die Interpretation als Prozess würde dies eine Schachtelung eines Prozesses in sich selber ergeben). Um weitere notwendige Bedingun-

gen an den Graphen zu stellen, sind die folgenden Mengen und Abbildungen notwendig:

- $G = (V, E)$ … Graph besteht aus Knoten *V* und Kanten *E*
- $V = C \cup Co$ … Knoten bestehen aus Komponenten *C* und Konnektoren *Co*
- $E \subseteq \{(v_1, v_2) | v_1, v_2 \in V\}$ … Kanten verbinden Knoten miteinander
- $prenodes: V \rightarrow P(V)$ … Vorgänger eines Knotens
$$prenodes(v_1) = \{v_2 \in V: \exists e \in E: e = (v_2, v_1)\}$$
- $postnodes: V \rightarrow P(V)$ … Nachfolger eines Knotens
$$postnodes(v_1) = \{v_2 \in V: \exists e \in E: e = (v_1, v_2)\}$$

Im Konfigurationsgraphen dürfen Komponenten nur über Konnektoren miteinander verbunden werden. Dies ist notwendig, um die Semantik der Verbindung zweier Komponenten beschreiben zu können und lässt sich wie folgt formalisieren.

- $\forall (v_1, v_2) \in E: \{v_1, v_2\} \nsubseteq C$

Ist es notwendig, identische Teilleistungen mehrfach auszuführen, so werden diese Leistungen durch Komponenten mit mehreren eingehenden Kanten dargestellt und als Klone bezeichnet. Allerdings dürfen aus Gründen der Konfiguration Konnektoren im Graphen maximal einen Oberknoten besitzen. Dies ist notwendig, um im Rahmen der Konfiguration nachvollziehen zu können, welche Oberknoten gewählt wurden.

- $\forall c \in Co: |prenodes(c)| = 1$

2.4 Kardinalitäten

Um komplexe Zusammenhänge zwischen Komponenten (z.B. alternative oder optionale Auswahlmöglichkeiten) zu definieren, sind die bisherigen einfachen hierarchischen Beziehungen nicht ausreichend. Eine weitergehende Semantik wird durch Kardinalitäten ermöglicht. Dadurch lässt sich genau spezifizieren, wie viele nachfolgende Knoten eines gewählten Knotens im Rahmen der Konfiguration gewählt werden müssen.

Einem Konnektor lässt sich durch die Abbildung *card* eine beliebige Menge verschiedener Kardinalitäten zuweisen.

- $card: Co \rightarrow P(\mathbb{N} \times \mathbb{N})$

Jede Kardinalität wird als Tupel *(min, max)* dargestellt, wobei *min* festlegt, wie viele nachfolgende Knoten mindestens gewählt werden müssen und *max* wie viele maximal gewählt werden dürfen. Daraus ergeben sich die Anforderungen, dass der Mindestwert einer Kardinalität kleiner als ihr Maximalwert sein muss, und dass der maximale Wert nicht größer als die Anzahl nachfolgender Knoten sein darf.

- $\forall c \in Co, \forall (m, n) \in card(c)$:
$$(m \leq n) \wedge (n \leq |postnodes(c)|)$$

Werden einem Konnektor mehrere Kardinalitäten zugewiesen, werden diese mit einer Disjunktion verknüpft. Bei der Konfiguration muss daher eine der angegebenen Kardinalitäten erfüllt sein aber nicht alle. Um Redundanzen und Inkonsistenzen zu vermeiden, dürfen Kardinalitäten eines Konnektors sich nicht überlappen.

- $\forall c \in Co \forall (m_1, n_1) \in card(c)$:
  $\nexists (m_2, n_2) \in card(c): (m_1 \leq m_2 \leq n_1) \vee (m_1 \leq n_2 \leq n_1)$

Um die Modellierung in der Praxis zu vereinfachen, sind in Tabelle 1 verschiedene gebräuchliche Konnektortypen aufgeführt.

Tabelle 1: Vordefinierte Konnektortypen

Konnektor	Beschreibung	Formalisierung
Konjunktiv-Konnektor (AND)	Bei der Konfiguration müssen alle nachfolgenden Knoten gewählt werden.	$\forall c \in Co_{AND}: card(c) = (\lvert postnodes(c)\rvert, \lvert postnodes(c)\rvert)\}$
Exklusiv-Oder-Konnektor (XOR)	Bei der Konfiguration muss genau ein nachfolgender Knoten gewählt werden.	$\forall c \in Co_{XOR}: card(c) = \{(1,1)\}$
Disjunktiv-Konnektor (OR)	Bei der Konfiguration kann eine beliebige Menge nachfolgender Knoten gewählt werden.	$\forall c \in Co_{OR}: card(c) = \{(0, \lvert postnodes(c)\rvert\}$

2.5 Erweiterungsmöglichkeiten

Durch die formalisierte Darstellung der Dienstleistungskomponenten und ihrer Abhängigkeiten untereinander, ergibt sich die Möglichkeit, das hier vorgestellte Grundmodell um weitere Konzepte zu erweitern. Darüber hinaus dient die Formalisierung auch als Grundlage für die in den folgenden Abschnitten vorgestellte Erweiterung zur Modellierung von PSSs.

Diese Erweiterungsmöglichkeit wurde ausgehend von Anforderungen aus der Praxis in (Becker, et al., 2011) genutzt, um das Modell um Attribute und Variablen zu erweitern. Variablen dienen dazu, die Umgebung zu beschreiben, in welcher die Dienstleistungen ausgeführt werden. So ist z.B. in einer Dienstleistung für Call Center die Anzahl der erwarteten Anrufe pro Tag von großer Bedeutung. Diese Eigenschaft basiert auf kundenspezifischen Annahmen und müsste daher

bei jeder Anwendung des Modells auf neue Kunden angepasst werden. Die Integration von Variablen ermöglicht es, individuelle und kundenspezifische Eigenschaften direkt in einem Modell darzustellen und Werte erst während der Konfiguration zu setzen. Im Gegensatz zu fest definierten Anpassungspunkten im Modell ermöglicht die flexible Integration von Variablen eine größere Abstraktion von der zu modellierenden Domäne.

Daneben lassen sich Komponenten auch mit nichtfunktionalen Eigenschaften genauer beschreiben. Ein Call Center hat demnach eine maximale Kapazität an Anrufern, die pro Stunde verarbeitet werden können. Bei der Auswahl verschiedener Komponenten spielen nichtfunktionale Eigenschaften daher oftmals eine große Rolle. Diese Eigenschaften können in Form unveränderlicher Attribute in Komponenten hinterlegt werden.

Weitere Erweiterungen, um z.B. den Ressourcenverbrauch von Komponenten zu beschreiben, sind denkbar. Diese können aufbauend auf dem existierenden Modell definiert werden und lassen sich aufgrund der formalen Struktur einfach integrieren.

2.6 Konfiguration

Ausgehend von den bisher beschriebenen Konzepten lassen sich durch die Auswahl geeigneter Komponenten kundenindividuelle Konfigurationen erstellen. Dazu ordnet die Abbildung *Chosen* jeder Komponente des Konfigurationsgraphen zu, ob sie ausgewählt wurde oder nicht. Die Konfiguration ist somit die Menge aller ausgewählten Komponenten.

$$Chosen: C \rightarrow \{0,1\}$$

$$Configuration = \{c | \forall c: c \in C \wedge Chosen(c) = 1\}$$

Um ungültige Konfigurationen zu verbieten, werden im Folgenden Anforderungen an die Menge *Configuration* formuliert. Zunächst sind alle nachfolgenden Knoten eines nicht gewählten Knotens ebenfalls nicht gewählt. Dadurch wird verhindert, dass Unterkomponenten ohne die entsprechenden Oberkomponenten gewählt werden.

$$\forall n \in V \forall n' \in postnodes(n): Chosen(n) = 0 \rightarrow Chosen(n') = 0$$

Weiterhin müssen alle nachfolgenden Konnektoren einer gewählten Komponente ebenfalls gewählt werden. Dadurch wird sichergestellt, dass auch notwendige, nachfolgende Unterkomponenten gewählt werden müssen. An dieser Stelle kann die Abbildung *postnodes* ohne weitere Einschränkungen verwendet werden, da aufgrund der Definition des Graphen V Nachfolger von Komponenten immer Konnektoren sind.

$$\forall c \in C, \forall n \in postnodes(c): Chosen(c) = 1 \rightarrow Chosen(n) = 1$$

Auf der Grundlage dieser Bedingungen lässt sich nun die Semantik von Konnektoren während der Konfiguration definieren. Dazu wird die Menge *ChosenPostNodes* verwendet, die alle nachfolgenden gewählten Knoten eines

Konnektors enthält. Die Anzahl gewählter nachfolgender Knoten muss dann eine der Kardinalitäten des Konnektors erfüllen.

$\forall c \in Co: \exists (m,n) \in card(c): m \leq |ChosenPostNodes(c)| \leq n$ mit
$\forall c_1 \in Co \forall c_2 \in postnodes(c_1)$:
$NodeChosen(c_2) = 1 \rightarrow c_2 \in ChosenPostNodes(c_1)$

3 Erweiterung des Metamodells zur Repräsentation von PSS

Das oben vorgestellte Metamodell lässt sich nicht nur zur Darstellung von Dienstleistungen verwenden. Da es auf dem generischen und insbesondere im Produktbereich weit verbreiteten Ansatz der Modularisierung durch Komponenten basiert, liegt die Übertragbarkeit des vormals dienstleistungsspezifischen Ansatzes auch auf die Modellierung von Produkten nahe. Durch diese semantische Erweiterung wird es möglich, neben Dienstleistungskomponenten auch Produktkomponenten zu spezifizieren, wobei sich diese beiden Typen nur in der zugrunde liegenden Interpretation unterscheiden. Während Dienstleistungskomponenten Prozesse darstellen, entsprechen Produktkomponenten materiellen Bestandteilen von Produkten (Schrauben, Zylinder, Motoren etc.). Dies hat allerdings keine Auswirkungen auf die grundlegende formale Spezifikation der Komponenten. Um mit diesen Mitteln PSSs darzustellen, werden im Folgenden Dienstleistungen und Produkte in zwei separaten Bäumen beschrieben, die jeweils Produkt- bzw. Dienstleistungskomponenten enthalten. Dazu wird ein Produktportfolio *PP* analog zum oben definierten Dienstleistungsportfolio *DP* aufgestellt. Diese beiden Portfolios werden in einem PSS *P=(DP, PP, A)* kombiniert und mit Abhängigkeiten *A* untereinander versehen. Die Abhängigkeiten zwischen Produkten und Dienstleistungen werden im folgenden Abschnitt genauer definiert. Zur Illustration wird als Praxisbeispiel ein PSS im Kontext einer Photovoltaik-Anlage (PV-Anlage) verwendet.

3.1 Produktportfolio

Die PV-Anlage besteht aus vier PV-Modulen und einem Wechselrichter, welcher optional eine LAN-Schnittstelle zur Fernwartung sowie eine RS485-Schnittstelle zum Datenmonitoring haben kann. Zur Montage der Unterkonstruktion werden acht Dachhaken sowie vier Montageprofile benötigt. Die mehrfach vorhandenen, identischen Komponenten sind dabei als Klone modelliert.

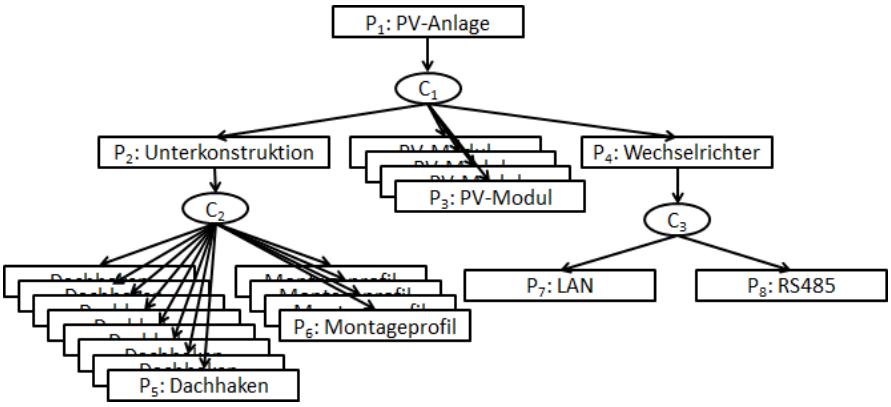

Abbildung 1: Produktportfolio des PSSs „PV-Anlage“

Abbildung 1 zeigt die graphische Darstellung des Produktportfolios der PV-Anlage. Klone werden dabei zur Erhöhung der Übersichtlichkeit als mehrfach vorhandene Knoten angezeigt, die überlappend angeordnet sind. Das Portfolio PP lässt sich formal wie folgt spezifizieren, wobei zur Verkürzung der Darstellung nur Komponentenkürzel verwendet werden.

$$PP = (C_P, G_P)$$
$$\mathrm{C_P} = \{\mathrm{P_1}, \mathrm{P_2}, \mathrm{P_3}, \mathrm{P_4}, \mathrm{P_5}, \mathrm{P_6}, \mathrm{P_7}, \mathrm{P_8}\}$$
$$G_P = (V_P, E_P)$$
$$V_P = C_P \cup \{C_1, C_2, C_3\}$$
$$E_P = \{(P_1, C_1),$$
$$(C_1, P_2), (C_1, P_3), (C_1, P_3), (C_1, P_3), (C_1, P_3), (C_1, P_4),$$
$$(P_2, C_2), (P_4, C_3),$$
$$(C_2, P_5), (C_2, P_5), (C_2, P_5), (C_2, P_5),$$
$$(C_2, P_5), (C_2, P_5), (C_2, P_5), (C_2, P_5),$$
$$(C_2, P_6), (C_2, P_6), (C_2, P_6), (C_2, P_6),$$
$$(C_3, P_7), (C_3, P_8)\},$$
$$card_P = \{\big(C_1, (6{,}6)\big), \big(C_2, (12{,}12)\big), (C_3, (0{,}2))\}$$

3.2 Dienstleistungsportfolio

Dienstleistungen im Umfeld von PV-Anlagen gliedern sich in die drei Aufgabenfelder „Installation“, „Wartung“ und „Monitoring“.[1] Die Installation umfasst dabei entweder nur die Anlieferung (der Kunde installiert selbst) oder Anlieferung

[1] Die Komponenten des Dienstleistungsportfolio sind inspiriert von der Beschreibung möglicher PV-Wartungsleistungen unter http://www.photovoltaikforum.com/pv-news-f25/wartung-von-photovoltaikanlagen-t46.html

und Aufbau. Die Wartungskomponente gliedert sich in die drei Teilaspekte der Fernwartung, welche vorrangig beim Wechselrichter vorgenommen wird, einer Reinigung der Anlage und der Vor-Ort-Wartung, welche beispielsweise eine Sichtkontrolle, eine Prüfung der Verschattung oder eine manuelle Wechselrichterprüfung umfasst. Abgerundet wird das Angebot für individuelle PV-Anlagen durch das Monitoring, bei welchem die Daten gespeichert und ausgewertet werden, um auf deren Basis einen Effizienzvergleich mit anderen Anlagen durchführen zu können. Zusätzlich zur Installation und Wartung von PV-Anlagen beim Kunden wird außerdem der Betrieb einer PV-Anlage, die nicht beim Kunden montiert ist, als Dienstleistung angeboten. Ein dementsprechendes Dienstleistungsportfolio zeigt Abbildung 2.

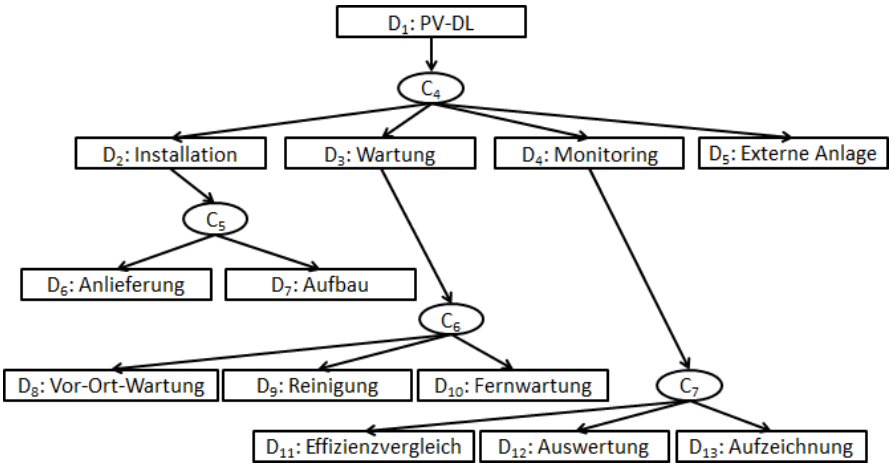

Abbildung 2: Dienstleistungsportfolio des PSSs „PV-Anlage“

Wie auch das Produktportfolio *PP* lässt sich das Dienstleistungsportfolio *DP* formal darstellen.

$$DP = (C_D, G_D)$$
$$C_D = \{D_1, D_2, D_3, D_4, D_5, D_6, D_7, D_8, D_9, D_{10}, D_{11}, D_{12}, D_{13}\}$$
$$G_D = (V_D, E_D)$$
$$V_D = C_D \cup \{C_4, C_5, C_6, C_7\}$$
$$E_D = \{(D_1, C_4),$$
$$(C_4, D_2), (C_4, D_3), (C_4, D_4), (C_4, D_5),$$
$$(D_2, C_5), (C_5, D_6), (C_5, D_7),$$
$$(D_3, C_6), (C_6, D_8), (C_6, D_9), (C_6, D_{10}),$$
$$(D_4, C_7), (C_7, D_{11}), (C_7, D_{12}), (C_7, D_{13})\}$$
$$card_D = \{\big(C_4, (1,4)\big), \big(C_5, (1,2)\big), \big(C_6, (1,3)\big), (C_7, (1,3))\}$$

Mit der unabhängigen Modellierung der Dienstleistungs- und Produktportfolios in separaten Bäumen sind Abhängigkeiten zwischen den Bäumen noch nicht darstellbar. Diese sind jedoch für die holistische Konfiguration von großer Wichtig-

keit. Aus diesem Grund werden im folgenden Kapitel die Abhängigkeitsbeziehungen zwischen PSS-Bestandteilen untersucht.

4 Abhängigkeiten in Product-Service-Systems

Eine der wichtigsten Anforderungen an Notationen zur Modellierung von PSS ist die Repräsentation der Abhängigkeiten zwischen den modellierten Komponenten. Mit dem bisher vorgestellten Modell lassen sich diese Abhängigkeiten formal darstellen. In diesem Abschnitt werden zunächst Erläuterungen zu Typen und zur Struktur von Abhängigkeitsregeln gemacht. Darauf aufbauend werden spezifische Abhängigkeitsbeziehungen, die in der PSS-Literatur genannt werden, vorgestellt. Im Gegensatz zu bisherigen Arbeiten werden die Abhängigkeiten in dieser Arbeit auch formal definiert, so dass sie während einer (teil)automatisierten Konfiguration überprüft werden können. Jede Abhängigkeitsbeziehung wird sowohl textuell durch ein Beispiel als auch im Rahmen des oben vorgestellten Metamodells formal beschrieben.

Abhängigkeitsbeziehungen in PSSs sind auf verschiedenen Ebenen möglich. Angelehnt an die Ausführungen von Böttcher beschreiben *Descendent Dependencies* (Typ I) hierarchische Beziehungen innerhalb eines Teilbaums (Böttcher & Klingner, 2011). Da sich Böttcher lediglich auf die Beschreibung von Beziehungen innerhalb eines Dienstleistungsmodells fokussiert, sollte der Begriff der *Cross-Tree Dependencies* zur Beschreibung nichthierarchischer Abhängigkeiten aufgrund des erweiterten Kontexts von PSSs ersetzt werden.

Bei nichthierarchischen Abhängigkeiten in PSSs lassen sich zwei Typen unterscheiden. So werden Beziehungen innerhalb der Bäume als *Intra-Tree Dependencies* (Typ II) bezeichnet, während Beziehungen zwischen Dienstleistungs- und Produktbaum mit dem Begriff *Inter-Tree Dependencies* (Typ III) beschrieben werden. Abbildung 3 gibt eine Übersicht aller möglichen Typen von Abhängigkeiten. Während somit Descendent Dependencies sowie Intra-Tree Dependencies jeweils Beziehungen zwischen Komponenten des gleichen Typs beschreiben, sind Inter-Tree Dependencies stets Abhängigkeiten zwischen Produkt- und Dienstleistungskomponenten. Um den Mehrwert für PSSs zu unterstreichen, werden die nachfolgend aufgeführten Abhängigkeiten vorzugsweise mit Beispielen von Beziehungen zwischen Produkt- und Dienstleistungskomponenten illustriert, obgleich sie sich ebenso zur Beschreibung von Abhängigkeitsbeziehungen vom Typ I und II eignen.

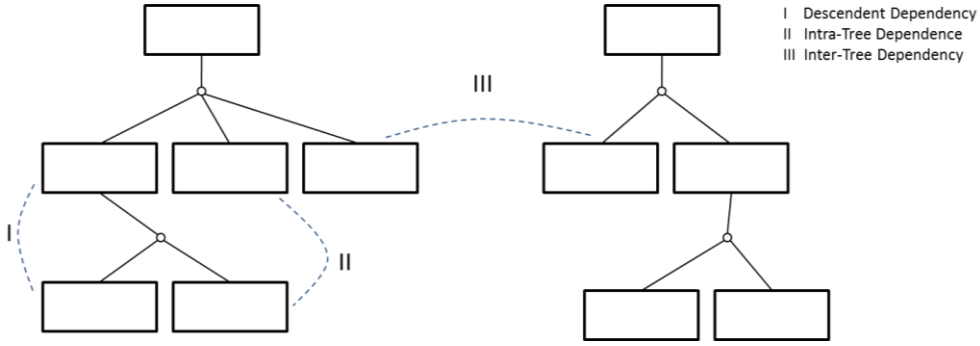

Abbildung 3: Typen von Abhängigkeitsbeziehungen eines PSSs

Eine zweite Dimension der Strukturierung kann auf Ebene der Regeln selbst durchgeführt werden, welche hinsichtlich verschiedener Regeltypen klassifiziert werden können. In der Literatur vorgeschlagene Herangehensweisen zur Definition von Abhängigkeitsregeln bei Konfigurationen beschränken sich oftmals auf den Aspekt der einschränkenden Anweisungen, sogenannter Constraints (Faltings & Weigel, 1994; Gelle & Weigel, 1996; Jinsong, et al., 2005). Bei einem weiter gefassten Regelverständnis, welches insbesondere im Kontext von PSSs notwendig wird, ergibt sich der Bedarf weiterer Regeln, wie beispielsweise Empfehlungen (Te'eni & Shufer, 2006) oder Alternativen (Baines, et al., 2007). Daraus lassen sich zusammenfassend drei Gruppen von Regeltypen identifizieren:

- *Vorschlagend*
  Vorschlagende Regeln beschreiben lose Zusammenhänge zwischen Komponenten. Diese Regeln schränken die Zahl gültiger Konfigurationsmöglichkeiten nicht weiter ein, da ihre Anwendung optional ist.
- *Einschränkend*
  Einschränkende Regeln beschreiben Beziehungen, welche die Menge der möglichen Konfigurationen verringern.
- *Modifizierend*
  Modifizierende Regeln beschreiben Änderungen von Eigenschaften von Komponenten.

Während der Konfiguration werden zuerst vorschlagende Regeln präsentiert. Dem folgt die Anwendung der modifizierenden Regeln, bevor abschließend die gesamte Konfiguration auf Einhaltung der einschränkenden Regeln geprüft wird. Die Formalisierung der Anwendung dieser Regeln findet sich in (Becker & Klingner, 2011). Zur praktischen Anwendbarkeit wurden die Regeln prototypisch als Prolog-Programm spezifiziert. Damit ist es möglich, Konfigurationen auf ihre Gültigkeit zu prüfen.

4.1 Struktur der Regeln

Bevor im folgenden Abschnitt einige Regeln beispielhaft gezeigt werden, soll an dieser Stelle die allgemeine Struktur der Regeln definiert werden. Ausgehend von dieser Struktur lassen sich auch weitere Regeln formulieren, die aus Platzgründen in dieser Arbeit nicht aufgelistet und in (Becker & Klingner, 2011) ausformuliert sind. Mit Hilfe der grundlegenden Struktur ist es weiterhin möglich, Regeln für eine spezifische Domäne zu erstellen.

Grundsätzlich ist eine Regel als Abbildung von einem Vorbereich *VB* auf einen Nachbereich *NB* definiert:

$$rule: VB \rightarrow NB$$

Der Vorbereich besteht dabei entweder aus einer Komponente, einem Vergleich über Variablen, über Attribute, über KPIs oder einer logischen Verknüpfung dieser Elemente. Bei den Vergleichsoperationen werden die entsprechenden Bestandteile mit einem bestimmten Wert verglichen. Sie lassen sich wie folgt definieren:

$VariableRule = (Variable, Comparator, \mathbb{R})$

$AttributeRule = (Attribute, Comparator, \mathbb{R})$

$KPIRule = \{(KPI, Comparator, \mathbb{R}), (KPI, Component, Comparator, \mathbb{R})\}$

Die Menge $Comparator = \{<, >, =\}$ beschreibt Vergleichsoperationen über der Menge der reellen Zahlen. Eine entsprechende Regel sagt demzufolge aus, dass eine Variable, ein Attribute oder eine KPI größer, kleiner oder gleich einer reellen Zahl sein soll. Als Spezialfall kann die Anwendung eines Vergleichs über KPIs außerdem auf eine spezifische Komponente eingeschränkt werden.

Anhand dieser Definitionen ist es nun möglich den Vorbereich der Regeln zu definieren:

$$VB = \{Component, VariableRule, AttributeRule, KPIRule\}$$

Um aussagenlogische Verknüpfungen zwischen den atomaren Elementen des Vorbereichs zu ermöglichen ist der Vorbereich induktiv zu erweitern:

$$a \in VB \wedge b \in VB \rightarrow (a \wedge b) \in VB$$
$$a \in VB \wedge b \in VB \rightarrow (a \vee b) \in VB$$
$$a \in VB \rightarrow \neg a \in VB$$

Die Erweiterung des Vorbereichs ist notwendig, damit Regeln nicht nur anhand atomarer Elemente aufgestellt werden können. In der Praxis führt oftmals die Kombination mehrere Elemente zu Auswirkungen auf andere Elemente. Mit Hilfe der aufgestellten Definitionen ist es nun möglich, verschiedene Elemente des Vorbereichs zu definieren. Beispielsweise stellt sich der Term „*Komponenten A und B sind gewählt und die Variable incomingCalls ist größer als 10.000 oder die KPI price der Komponente C ist größer als 200"* wie folgt dar:

$$\big(A \wedge B \wedge (incomingCalls, >, 10000)\big) \vee (price, C, >, 200)$$

Während der Vorbereich einer Regel für die meisten Regeln gleich definiert ist, trifft dies für den Nachbereich nicht zu. Die gültigen Elemente des Nachbereichs sind abhängig von der Semantik der jeweiligen Regel. Zur Illustration dessen wer-

den den oben aufgestellten Gruppen von Regeltypen nun verschiedene Regeln zugeordnet. Eine zusammenfassende Übersicht der Regeltypen, in dieser Arbeit untersuchte Regeln, sich daraus ergebenden Einschränkungen hinsichtlich des Nachbereiches sowie entsprechende Literaturquellen zeigt Tabelle 2. Nachfolgend werden die Regeln formalisiert und anhand des PV-Beispiels erläutert.

Tabelle 2: Regeltypen und Regeln zur Definition von Abhängigkeiten in PSSs

Regeltyp	Regel und Quelle	Nachbereich
Vorschlagend	**Alternative** (Baines, et al., 2007)	Komponente | Logische Verknüpfung[2]
	Empfehlung (Uhlmann, et al., 2008)	Komponente | Attributvergleich | Variablenvergleich | KPI-Vergleich | Logische Verknüpfung
Einschränkend	**Voraussetzung** (Sun, 2010; Böhmann & Krcmar, 2007)	Komponente | Attributvergleich | Variablenvergleich | KPI-Vergleich | Logische Verknüpfung
	Ausschluss (Faltings & Weigel, 1994; Gelle & Weigel, 1996; Jinsong, et al., 2005)	Komponente | Logische Verknüpfung
Modifizierend	**Wertänderung** (Mont, 2002)	KPI-Modifikator[3]

Vorschlagende Regeln

Vorschlagende Regeln umfassen Alternativen sowie Empfehlungen. Alternativen definieren im Ergebnis ihrer Leistung identische und daher miteinander substituierbare Komponenten, während durch Empfehlungen Elemente bestimmt werden, welche bei Auswahl die aktuelle Konfiguration sinnvoll ergänzen.

[2] Eine logische Verknüpfung zwischen Elementen des Nachbereichs ist analog zur oben definierten aussagenlogischen Verknüpfung von Elementen des Vorbereichs definiert.

[3] Modifikatoren über KPIs ändern deren Wert. Sie sind definiert als $(Component, KPI, Modifier)$. Bei der Anwendung wird die KPI der angegebenen Komponente mit dem Wert von *Modifier* multipliziert. Weitere Formalisierungen finden sich in (Becker & Klingner, 2011).

Alternative

Nach (Baines, et al., 2007) ist eine Motivation für die Implementierung von PSSs, den Fokus auf den Verkauf eines Nutzens und nicht eines bestimmten Produkts zu lenken. Dies basiert auf der Annahme, dass Kunden keine Präferenzen haben, ob ihre Bedürfnisse durch Dienstleistungen oder durch Produkte erfüllt werden.

Zur Kennzeichnung alternativer Angebote wird das Prädikat *surrogates* verwendet. Eine Alternative bildet lediglich von einer logischen Verknüpfung von Komponenten in eine logische Verknüpfung von Komponenten ab. Dies liegt darin begründet, dass mit dieser Regel nur alternative Produkt- bzw. Dienstleistungsbestandteile beschrieben werden, mit denen identische Kundenanforderungen erfüllbar sind.

Im PV-Beispiel erfüllt die Dienstleistung „Externe Anlage“ die gleichen Bedürfnisse wie das Produkt „PV-Anlage“ – Kunden werden mit Strom versorgt. Dies führt zu folgender Formel:

$$surrogates(PV - Anlage) = Externe\ Anlage$$

Empfehlung

Dieser Abhängigkeitsbeziehung liegt die Annahme zugrunde, dass Anbieter ihren Kunden empfehlen, bei der Wahl eines Produkts eine bestimmte Dienstleistung ebenfalls auszuwählen. Uhlmann legt dar, dass die Produktivität einer Anlage nicht nur von deren materiellen Eigenschaften abhängt, sondern auch vom Qualifikationsstand der Mitarbeiter und der generellen Wartung (Uhlmann, et al., 2008). Die Empfehlung einer bestimmten Komponente lässt sich noch weiter verallgemeinern, indem Komponenten mit bestimmten Eigenschaften empfohlen werden. Dadurch ist es möglich, von spezifischen Details zu abstrahieren. Schließlich können Umgebungsparameter die Auswahl einer bestimmten Komponente empfehlen. Mit Hilfe dieser Abhängigkeitsbeziehungen können Unternehmen innovative Portfolios erstellen und somit auch Kundenerwartungen übererfüllen (Chase & Hayes, 1991). Zur Darstellung einer Empfehlung wird das Prädikat *recommends* verwendet.

Im Rahmen der PV-Anlage wird bei der Auswahl des Gesamtprodukts „Anlage“ die Dienstleistung „Reinigung“ empfohlen, damit die PV-Module gleichbleibende Leistung erbringen. Die Dienstleistung „Vor-Ort-Wartung“ empfiehlt die Produktkomponente „RS485“, da sich mit diesem Anschluss Daten über die PV-Anlage leicht auslesen lassen. Zusätzlich wird empfohlen, dass Kunden die Anlage nur selber installierten sollten (also die Dienstleistung „Aufbau“ nicht gewählt wird), wenn die Dachneigung (dargestellt durch die externe Variable *roofPitch*) zwischen 20 und 50 Grad beträgt. Ist dies nicht der Fall sollte der Aufbau durch ein spezialisiertes Unternehmen erfolgen. Diese Regeln lassen sich wie folgt formalisieren.

$$recommends(PV - Anlage) = Reinigung$$
$$recommends(VorOrtWartung) = RS485$$
$$recommends\big((roofPitch, <, 20) \vee (roofPitch, >, 50)\big) = Aufbau$$

Einschränkende Regeln

Einschränkende Regeln reduzieren die gültigen Varianten des Konfigurationsgraphs. Dazu sind Regeln der Prädikate „Voraussetzungen“ sowie „Ausschluss“ zu zählen. Voraussetzungen definieren Elemente oder Eigenschaften, welche für Konfiguration anderer Elemente zwingend benötigt werden, wohingegen sich durch ausschließende Regeln Elemente identifizieren lassen, welche auf Basis der aktuellen Konfiguration nicht gewählt werden dürfen.

Voraussetzung

Diese Abhängigkeit ist in der PSS-Literatur weit verbreitet (Böhmann & Krcmar, 2007; Sun, 2010). In (Sun, 2010) wird eine Schnittstelle, über die Dienstleistungen angeboten werden, *Service Carrier* genannt. Dies kann entweder ein Produkt sein oder aber die Kombination von Produkten und anderen Dienstleistungen (z.B. enthält das PSS „medizinische Behandlung“ die Dienstleistung „Untersuchung“ und das Produkt „Medikamente“). Ein Beispiel für externe Variablen im Vorbereich ist in (Böhmann & Krcmar, 2007) skizziert, indem Dienstleistungen und Produkte in den Wertschöpfungsprozess der Kunden integriert werden müssen. Eine Voraussetzung wird durch das Prädikat *requires* dargestellt.

Im PV-Beispiel ist zur Durchführung der Dienstleistung „Fernwartung“ eine LAN-Schnittstelle am Wechselrichter notwendig. Die Dienstleistung „Effizienzvergleich“ erfordert eine installierte Basis (dargestellt durch die externe Variable *installedBase*) von mehr als 100, damit der Vergleich auch sinnvoll erbracht werden kann.

$$requires(Fernwartung) = LAN$$
$$requires(Effizienzvergleich) = (installedBase, >, 100)$$

Ausschluss

Diese Regel wird verwendet, um Elemente zu beschreiben, die in einer Konfiguration nicht gemeinsam auftreten dürfen. Dies betrifft im Allgemeinen alternative Angebote, die sich nicht miteinander kombinieren lassen. In reinen Produkt- bzw. Dienstleistungsangeboten ist es möglich, durch die Verwendung von Exklusiv-Oder-Konnektoren gegenseitige Ausschlüsse von Komponenten zu definieren. Im Gegensatz dazu müssen nichthierarchische Ausschlüsse zwischen unterschied-

lichen Elementtypen explizit gekennzeichnet werden. Dies erfolgt mittels des Prädikats *prohibits*.

Im PV-Beispiel verbietet die Dienstleistung „Externe Anlage“ die Auswahl des Produkts „PV-Anlage“. Demnach können Kunden nicht gleichzeitig eine Anlage vor Ort installieren und Beteiligungen an einer externen Anlage erwerben.

$$prohibits(Externe\ Anlage) = PV - Anlage$$

Modifizierende Regeln

Modifizierende Regeln führen zu Änderungen an Elementen. Bestimmte Konfigurationen können somit Wertänderungen an KPIs von Komponenten nach sich ziehen, welche mithilfe dieser Regeln quantifiziert werden.

Wertänderung

Für produzierende Unternehmen besteht der Vorteil von PSSs darin, dass sie den Wert existierender Produkte durch zusätzliche Dienstleistungen erhöhen können (Mont, 2002). Um diese Beziehung detaillierter darzustellen, lässt sich auf der Ebene der Komponenten definieren, welche Dienstleistungskomponente den Wert welcher Produktkomponenten ändert und umgekehrt. Die Wertänderung kann sich z.B. in der Qualität oder auch im Preis widerspiegeln. Aus diesem Grund wird das Prädikat *changesValue* verwendet, mit dem generische Wertänderungen mit Hilfe eines Modifikators angegeben werden können.

Im PSS der PV-Anlage erhöht der professionelle Aufbau der Anlage durch den Anbieter die Qualität der Unterkonstruktion im Vergleich zur Qualität, die sich durch die manuelle Installation durch den Kunden ergeben würde.

$$changesValue(Aufbau) = (Unterkonstruktion, quality, 1.2)$$

5 Transformation von Produktmodellen

Beschreibungsansätze für Produkte sowie der Bedarf des kundenindividuellen Konfigurierens sind bereits seit längerer Zeit in der Praxis etabliert (Hegge & Wortmann, 1991). Demzufolge kann von einer Vielzahl existierender Modelle ausgegangen werden.

Um die Möglichkeit zu bieten, eventuell bereits in Unternehmen vorhandene Modelle weiter nutzen zu können sowie gleichzeitig das im vorangegangenen Abschnitt vorgeschlagene Metamodell zur Beschreibung von PSSs auf Vollständigkeit zu evaluieren, werden im Folgenden zwei verbreitete Ansätze zur Beschreibung von Produkten hinsichtlich ihrer Transformierbarkeit in das o.g. Metamodell hin untersucht. Die unten vorgestellten Transformationen sind im Rahmen einer

OpenSource Java-Anwendung implementiert[4]. Durch die freie Verfügbarkeit ist gewährleistet, dass auch andere Formate integriert werden können.

Die Untersuchungen werden anhand der Beispiele Bill of material (BOM) sowie der Featuremodellierung vorgenommen. Zur Vergleichbarkeit der verschiedenen Modellierungsmethoden wird im Folgenden jeweils das bereits eingeführte Beispiel des Produktmodells einer (PV-Anlage) entsprechend eines BOMs sowie eines Feature-Modells modelliert.

5.1 Bill of Material

Das Erstellen von sogenannten Bills of Material zur logischen Repräsentation von Produktdaten und Informationen zur Produktion ist im Bereich der Sachgüterherstellung weit verbreitet (Jiao, et al., 2000; Zhu, et al., 2007). Allgemein werden in einer BOM Produktbestandteile sowie deren Verbindung untereinander beschrieben (Stonebraker, 1996; Guoli, et al., 2003). Jiao et al. führen dazu die Notwendigkeit der Berücksichtigung dreier Aspekte und damit die wesentlichen Bestandteile einer BOM auf (Jiao, et al., 2000):

- *Items* als Komponenten zur Strukturierung des Gesamtprodukts
- *Goes-into relationships* als Verbindung zwischen Eltern- und Kindkomponente. Diese Verknüpfung wird zumeist mit der Anzahl der Kindkomponenten, welche zur Erstellung der Elternkomponente notwendig sind, ergänzt.
- *Employment* als Einflussfaktor des Anwendungsgebiets. In Abhängigkeit des Einsatzbereiches bzw. der Position des Produktlebenszyklus‘ existieren verschiedene Varianten und damit Herangehensweisen der Erstellung von BOMs.

Im Rahmen dieser Arbeit wird vorrangig die allgemeine Form der BOM betrachtet. Für die Unterscheidungen der Varianten wie beispielsweise modular BOM (MBOM), engineering BOM (EBOM), Generic BOM (GBOM) oder variant BOM (VBOM) sei auf die Literatur verwiesen (Hegge & Wortmann, 1991; van Veen & Wortmann, 1992; Jiao, et al., 2000; Zhu, et al., 2007).

Abbildung 4 illustriert diese drei Aspekte an der Umsetzung des oben beschriebenen Beispiels einer Photovoltaik-Anlage in eine BOM. Wichtig sei an dieser Stelle der Hinweis, dass die abgebildete BOM lediglich eine mögliche Konfiguration darstellt (diejenige, bei der sowohl die Komponente „LAN“ als auch die Komponente „RS485“ ausgewählt wurde). Für weitere Konfigurationen wären ähnliche BOMs zu erstellen, welche eine Vielzahl von Redundanzen aufwiesen (Hegge & Wortmann, 1991).

[4] https://sourceforge.net/projects/pmtransformator/

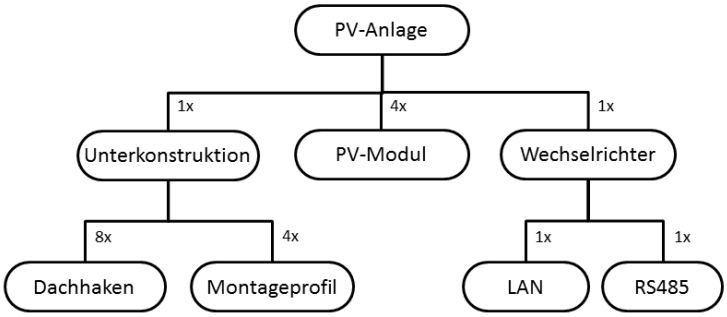

Abbildung 4: Photovoltaik-Anlage als BOM

Die Erstellung des BOMs erfolgte exemplarisch in der cloudbasierten Applikation Arena[5]. Dabei wurde das Beispiel der PV-Anlage modelliert. Über die Exportfunktion wurde eine XML-Datei erstellt, welche als Datengrundlage der Transformation diente.

Der Prozess der Transformation besteht im Wesentlichen aus der Extraktion der Items, welche in Komponenten überführt werden. Die Struktur von Eltern- und Kind-Items im BOM entspricht dabei der des transformierten Baumes, wobei zwischen Eltern- und Kindkomponenten des Metamodells noch Konjunktiv-Konnektoren eingefügt werden. Eine weitere Besonderheit des BOMs ist die Verwendung von Quantitäten, welche im Metamodell mittels Klonen repräsentiert werden.

Die Darstellung komplexer logischer Zusammenhänge bezüglich der Struktur ist mit BOMs nicht möglich, weshalb nur eine Teilmenge der Ausdrucksstärke des Metamodells genutzt wird, wie in Tabelle 3 dargestellt.

Tabelle 3: Transformation BOM – PSS

Konzept BOM	Konzept Metamodell
Item	Komponente
Item Bill-of-Material	Konjunktiv-Konnektoren
Quantity	Klone von Komponenten

5.2 Feature-Modellierung

Featuremodelle sind insbesondere im Software Engineering weit verbreitet, um sogenannte Produktfamilien zu beschreiben. Ein Feature ist eine für Endbenutzer sichtbare Eigenschaft eines Systems bzw. eine wahrnehmbare Charakteristik eines

[5] http://www.arenasolutions.com

Konzepts, die für bestimmte Stakeholder relevant ist (Czarnecki & Eisenecker, 2000). In einem Featuremodell sind die verschiedenen Features hierarchisch in einer Baumstruktur angeordnet. Dabei lassen sich folgende Kompositionstypen unterscheiden:

- *Obligatorische Features* müssen gewählt werden, wenn ihr jeweiliges Oberfeature gewählt wurde.
- Aus einer Menge *alternativer* Features darf nur genau eins gewählt werden, damit die Konfiguration gültig ist.
- *Optionale* Features können im System vorhanden sein, müssen es aber nicht.

Zusätzlich zu diesen hierarchischen Beziehungen lassen sich in Featuremodellen nichthierarchische Abhängigkeiten definieren. Batory stellt allgemein *simple inclusion* (Auswahl eines Features führt dazu, dass ein anderes Feature automatisch ausgewählt wird) und *simple exclusion* (Auswahl eines Features führt dazu, dass ein anderes Feature automatisch deaktiviert wird und nicht ausgewählt werden kann) vor (Batory, 2005). Diese grundlegenden Abhängigkeiten lassen sich durch die Anwendung aussagenlogischer Formeln auf die formale Repräsentation eines Featuremodells generalisieren. Weitergehende Erweiterungen für Featuremodelle, z.B. Kardinalitäten und Annotationen von Features, zeigen (Czarnecki, et al., 2004). Im Rahmen dieser Arbeit werden allerdings nur einfache Featuremodelle betrachtet. Abbildung 5 zeigt die Darstellung der PV-Anlage als Featuremodell, welches mit dem Eclipse Plug-In FeatureIDE erstellt wurde (Kastner, et al., 2009). Dabei sind Features, die mit einem ausgefüllten Kreis verbunden sind, obligatorisch und diejenigen mit nicht ausgefülltem Kreis optional.

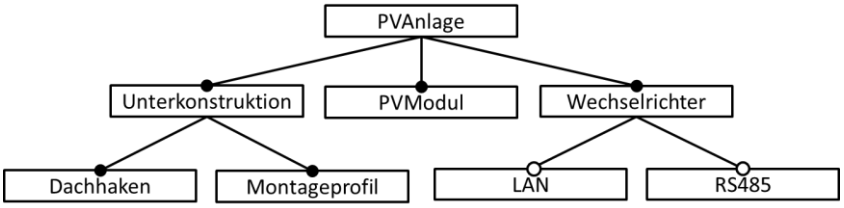

Abbildung 5: PV-Anlage als Featuremodell

Die Konzepte der Featuremodelle lassen sich wie in Tabelle 4 dargestellt in Elemente des PSS-Metamodells transformieren. Durch die Ausdrucksstärke von Featuremodellen ist die Transformation des Modells identisch zur direkten Spezifikation mittels des PSS-Metamodells. Lediglich die Kardinalitäten der Konnektoren lassen sich ohne Kardinalitäten in Featuremodellen nicht weiter darstellen. Daher wird je Feature nur genau eine Komponente erstellt, so dass nach der Transformation nur ein Dachhaken, ein Montageprofil und ein PV-Modul vorhanden sind. Die fehlenden Angaben zur Quantität lassen sich z.B. mittels Attributen nachtragen.

Tabelle 4: Transformation Featuremodell – PSS

Konzept FM	Konzept KPS
Konzept	Gesamtdienstleistung / Produkt
Feature	Komponente
Beziehungen	Konnektoren
Obligatorische Features	Konjunktiv-Konnektor
Optional Features	Disjunktiv-Konnektor
Alternative Features	Exklusiv-Oder-Konnektor

6 Related Work

In diesem Abschnitt werden vergleichbare Arbeiten vorgestellt, die sich auch im Themengebiet PSSs positionieren. Da die Thematik eher der Fachrichtung der Dienstleistungsforschung zuzuordnen ist, werden dazu zunächst Ansätze zur Modellierung von Dienstleistungen vorgestellt. Daran schließt sich eine überblicksartige Darstellung über Möglichkeiten zur Beschreibung von PSSs an. Für eine ausführliche Darstellung verschiedener Methoden der Produktmodellierung sei an dieser Stelle auf (Yang, et al., 2008) verwiesen.

Ein umfassendes Modell zur Beschreibung komplexer Dienstleistungen liefert die Unified Service Description Language (USDL) (Barros, et al., 2011). Die USDL fokussiert die Beschreibung des Zusammenspiels zwischen Dienstleistungskomponenten sowie einer ganzheitlichen Beschreibung mit Organisationen, Verantwortlichkeiten etc. Insbesondere die Preisbildung mit möglichen Rabattierungen findet hierbei explizit Beachtung. Im Gegensatz dazu ist die Preisbildung im hier vorgestellten Modell eher implizit durch die Definition und Berechnung von KPIs realisiert. Die USDL ist mittels eines UML-Modells formalisiert, bietet allerdings keine direkte Unterstützung der Konfiguration.

Im Zentrum der Betrachtungen von (Dong, et al., 2011) steht dagegen neben der Beschreibung der Abhängigkeiten zwischen Komponenten durch eine Ontologie auch die Konfiguration. Gültige Konfigurationen müssen dabei vordefinierten Constraints genügen. Zur praktischen Anwendbarkeit wurde außerdem ein Expertensystem entwickelt.

Neben der reinen Beschreibung der Struktur von Dienstleistungen adressieren (Brocke, et al., 2010) und (Emmrich, 2005) auch die notwendigen Anpassungen an Dienstleistungen über die einzelnen Phasen des Lebenszyklus‘. Während Emmrich produktorientierte Dienstleistungen untersucht und die Zustände und Zustandsänderungen von Produkten beschreibt, orientieren sich Brocke et al. am kontinuierlichen Anpassungsbedarf der Kunden bei IT-basierten Dienstleistungen.

Gegenüber Dienstleistungen sind Untersuchungen zu PSSs sowie deren Modellierung in der wissenschaftlichen Literatur erst in den letzten Jahren verstärkt in den Fokus gerückt. Insbesondere im deutschsprachigen Raum findet sich auch die

Bezeichnung hybride Leistungsbündel (Langer, et al., 2009; Becker, et al., 2009; Böhmann & Krcmar, 2007). Frühe Definition zu PSSs finden sich in (Goedkoop, et al., 1999) und (Mont, 2002). Dort wird dargelegt, dass zu den Elementen von PSSs nicht nur Produkte und Dienstleistungen sondern auch deren Beziehungen zueinander gehören. Basierend auf dieser Feststellung existieren vergleichbare Ansätze zur Modellierung von PSSs.

Ein Ansatz zur Beschreibung von PSSs mit dem Ziel der Konfiguration durch Anbieter (Vorabkonfiguration) sowie der kundenindividuellen Konfiguration wird in (Becker, et al., 2009) vorgestellt und mittels eines Entity-Relationship-Diagramms definiert. Die Besonderheit des Modells ist die explizite Fokussierung auf die Bewertung verschiedener Konfigurationen mittels Finanzplänen, die aufbauend auf einer Konfiguration erstellt werden. Im Gegensatz dazu lassen sich Konfigurationen mit dem in dieser Arbeit vorgestellten Modell basierend auf KPIs vergleichen, die bereits im Rahmen der Konfiguration berechnet werden.

Ausgehend von einem ursprünglich für die Modellierung von Produkten entwickelten Ansatz beschreiben (Weber, et al., 2004) das Property-Driven Design/Development-Modell. Dabei wird in Merkmale zur Beschreibung der Bestandteile und der Struktur von PSS-Elementen sowie in Eigenschaften zur Beschreibung der Bewertung eines PSSs durch Kunden unterschieden. Daneben wird ein Vorgehensmodell zur systematischen Entwicklung von PSSs vorgestellt. Auch (Uhlmann, et al., 2008) beschreiben ein solches Vorgehensmodell und fokussieren dabei die Berücksichtigung von Kundenanforderungen. Die Bestandteile beider Ansätze spiegeln sich bspw. durch die Definition von Variablen (als Kundenanforderungen) sowie durch Attribute und KPIs (Eigenschaften) in unserem Modell wider. Allgemeine Anforderungen und Besonderheiten bei der Entwicklung und Umsetzung von PSSs finden sich in (Morelli, 2002).

Weitere Ansätze, z.B. (Sun, 2010), untersuchen die Beziehungen zwischen Produkten und Dienstleistungen; dabei liegt der Fokus allerdings auf den Beziehungen zwischen Unternehmen, die PSSs anbieten und nutzen. Einen direkten Praxisbezug stellt (Walter, 2009) mit der Definition eines PSSs für die Domäne technischer Kundendienst dar.

7 Fazit

Die durchgeführten Arbeiten zeigen, wie eine holistische Modellierungsmethode für PSSs gestaltet werden kann, welche Produkt- und Dienstleistungskomponenten sowie Abhängigkeiten zwischen diesen berücksichtigt. Dazu wurde ein bislang im Dienstleistungskontext entwickeltes Metamodell um Möglichkeiten zur Darstellung von Produktmodellen erweitert. Im Beitrag wird in diesem Zusammenhang die Auffassung einer im Rahmen der Strukturierung prinzipiellen Ähnlichkeit von Produkt- und Dienstleistungskomponenten postuliert. Daneben wurde insbesondere die Sammlung und Strukturierung vielfältiger Abhängigkeitsregeln

fokussiert. Gemeinsam mit der komponentenbasierten Modellierung der Struktur bilden diese Regeln die Basis für die Konfiguration komplexer PSSs.

Die Ausgestaltung der Modellierungsmethode wurde dabei so vorgenommen, dass die Transformation existierender Modelle in die vorgeschlagene Modellierungsmethode für PSSs möglich ist. Dies wurde exemplarisch anhand des Beispiels einer PV-Anlage gezeigt, welche von einem BOM respektive Featuremodell in ein Modell konform zum Metamodell transformiert wurde.

Auch wenn gezeigt werden konnte, dass eine Transformation von Produktbäumen prinzipiell möglich ist, sind zukünftig Erweiterungen des Metamodells denkbar, welche die Spezifika von Produktmodellen besser unterstützen. Ein Beispiel wäre die Möglichkeit der expliziten Angabe von Quantitäten, anstatt diese über Klone zu konfigurieren. Aufbauend darauf wäre dann auch die Formulierung von Abhängigkeitsregeln wie „Wenn Produktkomponente A mehr als zehnmal gewählt wurde, ist Dienstleistungskomponente B zu wählen“ möglich. Jinsong bezeichnet diese Regeln als „Cardinality constraints“ (Jinsong, et al., 2005).

Das vorgeschlagene Metamodell zur Modellierung von PSSs dient als formale Basis für die Entwicklung von Software zur Konfiguration und Bewertung solcher Systeme. Für den Teilbereich der Dienstleistungskonfiguration existiert bereits ein auf dem Metamodell basierendes Tool, welches entsprechend von Anforderungen der Praxis entwickelt wurde (Klingner, et al., 2011). Die in diesem Beitrag vorgeschlagene Erweiterung auf PSSs ist hingegen noch softwaretechnisch umzusetzen.

Die Softwareunterstützung dient dazu, die hohe Komplexität des Modells, die sich durch dessen Formalisierung ergibt, besser handhabbar zu machen. Trotzdem ist der Aufwand zur Erstellung eines Modells recht hoch. Dies wird noch dadurch verstärkt, dass bisher zu dem vorgestellten Ansatz kein zugehöriges Vorgehensmodell existiert. Dies kann in der Praxis dazu führen, dass die einzelnen Elemente eines PSSs nicht in optimaler Granularität bestimmt werden, was im Endeffekt zu einem höheren Aufwand durch notwendige Anpassungen oder sogar fehlerhaften Implikationen führt. Im Zuge des Vorgehensmodells ist vor allem relevant, wie die Beziehungen zwischen Elementen definiert werden. Gegebenenfalls lassen sich hier auch existierende Verfahren zur Entwicklung von PSSs nutzen, z.B. (Thomas, et al., 2008; Abramovici & Schulte, 2005).

Die Anwendbarkeit in der Praxis kann gegebenenfalls erhöht werden, indem domänenspezifische Komponenten bzw. allgemeine PSS-Bestandteile vordefiniert werden, die sich dann in einem neuen PSS einsetzen lassen. Dies ermöglicht die einfache Integration von Best Practices und Referenzmodellen verschiedener Domänen.

Eine Anforderung aus der Praxis, die mit dem Modell nicht ohne weiteres dargestellt werden kann, ist die Änderung von PSSs über den Zeitraum des Einsatzes beim Kunden. Dies kann z.B. dann notwendig sein, wenn das PSS eines Webshops mit saisonal unterschiedlichen Anforderungen (z.B. Spitzenlast während der Weihnachtszeit) modelliert wird. Dies ist allerdings notwendig, um die Kosten und das Verhalten des Systems korrekt abschätzen zu können. Einige Ansätze da-

zu finden sich in (Brocke, et al., 2010) und (Emmrich, 2005). Sie lassen sich allerdings nicht ohne größere Änderungen am Modell an sich implementieren.

Daneben ist insbesondere bei komplexen PSSs die Interaktion zwischen verschiedenen, an der Umsetzung beteiligten Unternehmen, von Interesse. Dies wurde in (Sun, 2010) dargestellt. Das Modell erlaubt bisher nur die Darstellung von PSS-Elementen eines Unternehmens. Ansatzpunkte zur Integration verschiedener Unternehmen lassen sich über den Import von Produktmodellen erreichen, wobei allerdings keine Unterscheidung zwischen verschiedenen Besitzern bzw. Verantwortlichen einer Komponente gemacht wird.

Weitere möglicherweise notwendige Erweiterungen können ggf. identifiziert werden, wenn weitere Methoden zur Produktmodellierung auf ihre Transformierbarkeit in Modelle entsprechend des Metamodells überprüft werden. Dies können beispielsweise STEP-basierte Methoden (Yang, et al., 2008) oder in UML modellierte Konfigurationsgraphen (Felfernig, et al., 2001) sein.

Danksagung

Dieser Beitrag wurde ermöglicht durch die Förderung des Projekts „KoPro-Serv“ („Produktivitätssteigerung durch komponentenbasierte Dienstleistungen“) mit Mitteln des Bundesministeriums für Bildung und Forschung (BMBF). Das Projekt (Förderkennzeichen 01FL09004) wird betreut vom Projektträger im Deutschen Zentrum für Luft- und Raumfahrt (PT-DLR). Wir danken weiterhin den anonymen Gutachtern für ihre Kommentare und Anmerkungen, die zur Verbesserung des Beitrags beigetragen haben.

Literaturverzeichnis

Abramovici, M. & Schulte, S., 2005. Lifecycle Management von Produkt-Service-Systemen (PSS) für einen maximierten Kundennutzen. In: K. Grote, Hrsg. *Tagungsband : 3. Gemeinsames Kolloquium Konstruktionstechnik 2005 am 16. und 17.06.2005 im Herrenkrug Parkhotel Magdeburg.* Magdeburg: Shaker, pp. 1-11.

Baines, T. S. et al., 2007. State-of-the-art in product-service systems. *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture,* 221(10), pp. 1543-1552.

Barros, A., Kylau, U. & Oberle, D., 2011. *Unified Service Description Language 3.0 (USDL) Overview,* SAP: SAP Research.

Batory, D., 2005. Feature Models, Grammars, and Propositional Formulas. In: H. Obbink & K. Pohl, Hrsg. *Software Product Lines.* Rennes, France: Springer Berlin / Heidelberg, pp. 7-20.

Becker, J., Beverungen, D. & Knackstedt, R., 2008. *Reference Models and Modeling Languages for Product-Service Systems? Status-Quo and Perspectives for Further Research.* Waikoloa, Big Island, Hawaii, IEEE Computer Society, p. 105.

Becker, J., Beverungen, D., Knackstedt, R. & Müller, O., 2009. {Konzeption einer Modellierungssprache zur softwarewerkzeugunterstützten Modellierung, Konfiguration und Bewertung hybrider Leistungsbündel}. In: O. Thomas & M. Nüttgens, Hrsg. *Dienstleistungsmodellierung.* Berlin, Germany: Physica-Verlag HD, pp. 53-70.

Becker, M. & Klingner, S., 2011. *Formalisierung von Regeln zur Darstellung von Abhängigkeiten zwischen Elementen von Product-Service-Systems,* Leipzig: Universität Leipzig.

Becker, M., Klingner, S. & Böttcher, M., 2011. Configuring services regarding service environment and productivity indicators. In: M. Ganzha, L. Maciaszek & M. Paprzkycki, Hrsg. *Computer Science and Information Systems (FedCSIS), 2011 Federated Conference on.* Stettin: IEEE CS, pp. 505-512.

Böhmann, T. & Krcmar, H., 2007. Hybride Produkte: Merkmale und Herausforderungen. In: M. Bruhn & B. Stauss, Hrsg. *Wertschöpfungsprozesse bei Dienstleistungen.* Wiesbaden, Germany: Gabler, pp. 239-255.

Böttcher, M. & Fähnrich, K.-P., 2009. *Service Systems Modeling.* Leipzig, Logos.

Böttcher, M. & Klingner, S., 2011. Komponentisierung zur Steigerung der Dienstleistungsproduktivität. In: M. Bruhn, D. Georgi & K. Hadwich, Hrsg. *Forum Dienstleistungsmanagement 2011 - Dienstleistungsproduktivität.* Wiesbaden: Gabler, pp. 59-80.

Böttcher, M., Klingner, S. & Becker, M., 2011. Komponentenbasiertes Produktivitätscontrolling komplexer Dienstleistungsportfolios. *Controlling - Zeitschrift für erfolgsorientierte Unternehmenssteuerung*, 1 Oktober, 23(10), pp. 509-513.

Brocke, H., Uebernickel, F. & Brenner, W., 2010. {Zwischen Kundenindividualität und Standardisierung – Konzept und Referenz-Datenstruktur eines konfigurierbaren IT-Produktmodells}. In: O. Thomas & M. Nüttgens, Hrsg. *Dienstleistungsmodellierung 2010.* Klagenfurt, Österreich: Physica-Verlag HD, pp. 231-253.

Chase, R. B. & Hayes, R. H., 1991. Beefing Up Operations in Service Firms. *Sloan Management Review,* 33(1), pp. 15-26.

Czarnecki, K. & Eisenecker, U. W., 2000. *Generative programming: methods, tools, and applications.* 1 Hrsg. New York, NY, USA: ACM Press/Addison-Wesley Publishing Co..

Czarnecki, K., Helsen, S. & Eisenecker, U., 2004. Staged Configuration Using Feature Models. In: R. Nord, Hrsg. *Software Product Lines.* Boston, MA, USA: Springer Berlin / Heidelberg, pp. 162-164.

Dietze, V., 2008. *Festigung zwischenbetrieblicher Kollaboration durch den Einsatz von Group Decision Support Software - ein strategischer Planungsansatz.* 1. Hrsg. München: Grin Verlag.

Dong, M., Yang, D. & Su, L., 2011. Ontology-based service product configuration system modeling and development. *Expert Systems with Applications,* 38(9), pp. 11770-11786.

Emmrich, A., 2005. *Ein Beitrag zur systematischen Entwicklung produktorientierter Dienstleistungen,* Paderborn, Germany: Universität Paderborn.

Faltings, B. & Weigel, R., 1994. *Constraint-based knowledge representation for configuration systems,* Lausanne, Switzerland: Department d'Informatique, Laboratoire d'Intelligence Artificielle, Ecole Polytechnique Federale de Lausanne.

Felfernig, A., Friedrich, G. & Jannach, D., 2001. Conceptual modeling for configuration of mass-customizable products. *Artificial Intelligence in Engineering*, 15(2), pp. 165-176.

Gelle, E. & Weigel, R., 1996. *Interactive Configuration using Constraint Satisfaction Techniques.* London, UK, Menlo Park, AAAI Press.

Geum, Y., Kwak, R. & Park, Y., 2012. Modularizing services: A modified HoQ approach. *Computers & Industrial Engineering,* 62(2), pp. 579 - 590.

Goedkoop, M., Halen, C. v., Riele, T. H. & Rommens, P., 1999. *Product Service systems, Ecological and Economic Basics,* The Hague, The Netherlands: Pre.

Guoli, J., Daxin, G. & Tsui, F., 2003. Analysis and implementation of the BOM of a tree-type structure in MRPII. *Journal of Materials Processing Technology*, 139(1-3), pp. 535-538.

Hegge, H. & Wortmann, J., 1991. Generic bill-of-material: a new product model. *International Journal of Production Economics*, 23(1-3), pp. 117-128.

Hildebrandt, W.-C. & Klostermann, T., 2007. Dienstleistungsverkehr in industriellen Wertschöpfungsprozessen. In: M. Bruhn & B. Stauss, Hrsg. *Wertschöpfungsprozesse bei Dienstleistungen.* Wiesbaden: Gabler, pp. 215-236.

Isaksson, O., Larsson, T. C. & Rönnbäck, A. Ö., 2009. Development of product-service systems: challenges and opportunities for the manufacturing firm. *Journal of Engineering Design*, 1 April, 20(4), pp. 329-348.

Jiao, J., Tseng, M. M., Qinhai Ma & Yi Zou, 2000. Generic Bill-of-Materials-and-Operations for High-Variety Production Management. *Concurrent Engineering*, 8(4), pp. 297-321.

Jinsong, Z., Qifu, W., Li, W. & Yifang, Z., 2005. Configuration-oriented product modelling and knowledge management for made-to-order manufacturing enterprises. *The International Journal of Advanced Manufacturing Technology*, 25(1-2), pp. 41-52.

Kastner, C. et al., 2009. *FeatureIDE: A tool framework for feature-oriented software development.* Washington, DC, USA, IEEE Computer Society, pp. 611-614.

Klingner, S. et al., 2011. Managing complex service portfolios. In: W. Ganz, F. Kicherer & A. Schletz, Hrsg. *RESER 2011 Productivity of Services NextGen - Beyond Output/Input.* Hamburg, Germany: Fraunhofer, p. 150.

Knackstedt, R., Pöppelbuß, J. & Winkelmann, A., 2008. Integration von Sach- und Dienstleistungen – Ausgewählte Internetquellen zur hybriden Wertschöpfung. *WIRTSCHAFTSINFORMATIK*, 1 March, 50(3), pp. 235-247.

Langer, S. et al., 2009. {Entwicklungsprozesse hybrider Leistungsbündel – Evaluierung von Modellierungsmethoden unter Berücksichtigung zyklischer Einflussfaktoren}. In: O. Thomas & M. Nüttgens, Hrsg. *Dienstleistungsmodellierung.* Berlin, Germany: Physica-Verlag HD, pp. 71-87.

Mendonca, M., Wasowski, A. & Czarnecki, K., 2009. *SAT-based analysis of feature models is easy.* Pittsburgh, Carnegie Mellon University, pp. 231-240.

Mont, O. K., 2002. Clarifying the concept of product-service system. *Journal of Cleaner Production,* 10(3), pp. 237-245.

Morelli, N., 2002. Designing Product/Service Systems: A Methodological Exploration. *Design Issues,* 18(3), pp. pp. 3-17.

Stille, F., 2003. Produktbegleitende Dienstleistungen gewinnen weiter an Bedeutung. *Wochenbericht des DIW*, Band 21, pp. 336-342.

Stonebraker, P. W., 1996. Restructuring the bill of material for productivity: A strategic evaluation of product configuration. *International Journal of Production Economics*, 45(1-3), pp. 251-260.

Sundbo, J., 1994. Modulization of service production and a thesis of convergence between service and manufacturing organizations. *Scandinavian Journal of Management,* 10(3), pp. 245-266.

Sun, H., 2010. Product service relationship: defining, modelling and evaluating. *International Journal of Internet Manufacturing and Services,* 2(2), pp. 128-141.

Te'eni, M. & Shufer, I., 2006. *Component upgrading with dependency conflict resolution, knowledge based and rules.* USA, Patentnr. 7140013.

Thomas, O., Walter, P. & Loos, P., 2008. Product-Service Systems: Konstruktion und Anwendung einer Entwicklungsmethodik. *WIRTSCHAFTSINFORMATIK,* 50(3), pp. 208-219.

Uhlmann, E. et al., 2008. *Customer-driven development of product-service-systems.* Villa Erba, Cernobbio, Italy, Whittles Publ. [u.a.].

van Veen, E. & Wortmann, J. C., 1992. New developments in generative BOM processing systems. *Production Planning & Control*, 3(3), pp. 327-335.

Vazsonyi, A., 1954. The Use of Mathematics in Production and Inventory Control. I. *Management Science,* 1(1), pp. 70-85.

Walter, P., 2009. Modellierung technischer Kundendienstprozesse des Maschinen- und Anlagenbaus als Bestandteil hybrider Produkte. In: O. Thomas & M. Nüttgens, Hrsg. *Dienstleistungsmodellierung.* Berline, Germany: Physica-Verlag HD, pp. 129-145.

Weber, C., Steinbach, M., Botta, C. & Deubel, T., 2004. *Modelling of Product-Service Systems (PSS) based on the PDD Approach.* Dubrovnik, Croatia, , pp. 547-554.

Yang, W. Z., Xie, S. Q., Ai, Q. S. & Zhou, Z. D., 2008. Recent development on product modelling: a review. *International Journal of Production Research*, 46(21), pp. 6055-6085.

Zhu, S., Cheng, D., Xue, K. & Zhang, X., 2007. A Unified Bill of Material Based on STEP/XML. In: *Computer Supported Cooperative Work in Design III*. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 267-276.