Formalisierung von Regeln zur Darstellung von Abhängigkeiten zwischen Elementen von Product-Service-Systems

Michael Becker[1] and Stephan Klingner[1]

Abteilung Betriebliche Informationssysteme, Universität Leipzig
[mbecker|klingner]@informatik.uni-leipzig.de

Zusammenfassung In diesem Bericht werden beispielhaft Regeln, die bei der Modellierung von Product-Service-Systems (PSSs) Abhängigkeiten zwischen einzelnen Elementen beschreiben, dokumentiert und mittels prädikatenlogischer Ausdrücke formalisiert. Die Regeln sind sowohl aus der wissenschaftlichen Literatur zu PSSs abgeleitet als auch im Rahmen von Workshops mit Praxispartnern entwickelt worden. Zur praktischen Anwendbarkeit wurde ein Prolog-Listing entwickelt, welches ein Beispielportfolio sowie die Regeln enthält. Mit Hilfe dieser Anwendung kann die Gültigkeit der Regeln geprüft werden.

1 Einleitung

Die in den vergangenen Jahren stark gestiegene wirtschaftliche Bedeutung von Dienstleistungen [8] begünstigt auch eine zunehmende Konvergenz von Produkten und Dienstleistungen. Der darauf basierende Ansatz, Produkte und Dienstleistungen in sogenannten Product-Service-Systems (PSSs) zu verbinden [13] und gebündelt anzubieten, hat daher in der Vergangenheit verstärkt an Relevanz gewonnen [12,11,14]. Treiber dafür sind beispielsweise eine Verschiebung des Fokusses vom Erwerb einzelner Produkte bzw. Dienstleistungen hin zum Erwerb von Lösungen bzw. Funktionalitäten [1,9] und die Nutzung von Dienstleistungen zur Abgrenzung des Angebotsportfolios gegenüber Wettbewerbern [11].

Obgleich die Idee der PSSs bereits ihren Platz in der wissenschaftlichen Diskussion gefunden hat, ist ihr Durchdringungsgrad in der Industrie bislang begrenzt [1]. Dies liegt an einer Vielzahl von Herausforderungen, welche sich aufgrund einer stärkeren holistischen Betrachtungsweise von Sach- 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 Zusammenarbeit zwischen existierenden Abteilungen zu intensivieren ist [12]. Aufgrund der Interdependenzen ergibt sich weiterhin eine gesteigerte Komplexität, da so die Gestaltung von PSSs deutlich aufwändiger ist als die abgegrenzte Betrachtung von Sach- und Dienstleistungen [9,12]. So führen beispielsweise diese Interdependenzen zu erweiterten Anforderungen an das Portfoliomanagement, da ggf. Änderungen im Portfolio der Sachleistungen Modifikationen im Dienstleistungsportfolio nach sich ziehen und umgekehrt. Ebenso ist der Anspruch der Kunden nach individuellen Angeboten auch im Bereich PSSs immanent, wodurch sich die Problematik der kundenindividuellen Angebotskonfiguration auf die beiden Felder der Sach- und Dienstleistungen erweitert.

Das Ziel, trotz der hohen Komplexität PSSs handhabbar zu gestalten, erfordert geeignete softwaretechnische Unterstützung [7]. Besonderes Augenmerk ist dabei neben prozessualen Aspekten der Beschreibung von Abhängigkeiten zwischen den Produkt- und Dienstleistungsbereichen zu widmen [2]. Der Fokus unserer 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.

Umfangreiche Interdependenzen zwischen und Sach- und Dienstleistungen sowie der Wunsch nach einer möglichst homogenen Modellierungslandschaft [17] sprechen dafür, Produkte, Dienstleistungen und deren Interdependenzen in einem einzigen holistischen Modell abzubilden. Zur

Modellierung von Dienstleistungen aus Komponenten wurde in der Vergangenheit ein Modell entwickelt, welches die kundenindividuelle Konfiguration von Dienstleistungen fokussiert [5]. Dieses Modell dient im Rahmen dieses Beitrags als Ausgangspunkt zur Beschreibung von Interdependenzen zwischen Bestandteilen in PSSs.

Im folgenden Abschnitt werden zunächst Mengen und Relationen vorgestellt, die notwendig sind, um die Interdependenzen darzustellen. Diese basieren auf der in [3] vorgestellten Formalisierung des Modells von Dienstleistungskomponenten. Anschließend werden die Regeln in Abschnitt 3 eingeführt. Dazu werden zunächst allgemeine Eigenschaften der Regeln definiert, woraufhin eine Vorstellung der Regeln unterteilt in die Klassen *Vorschlagend*, *Einschränkend*, *Modifizierend* und *Sonstige* erfolgt. Jeder Regel werden Eigenschaften zugeordnet und es wird definiert, wie sie im Zusammenspiel mit anderen Regeln interagiert.

Zu beachten ist hierbei, dass die vorgestellten Regeln nicht alle möglichen PSS-Szenarien abdecken können sondern nur beispielhaft zeigen, welche Interdependenzen existieren. Die Regeln wurden sowohl der Literatur zu PSSs sowie zur Produkt- und Dienstleistungsmodellierung entnommen als auch in Praxisworkshops entwickelt. Insbesondere in spezifischen Domänen lassem sich weitere Interdependenzen identifizieren. Durch die formale Darstellung des Modells lassen sich neue Regeln allerdings leicht einfügen und ihre Beziehungen zu existierenden Regeln spezifizieren.

2 Definition notwendiger Mengen und Relationen

Vor der eigentlichen Definition der Regeln ist es notwendig, Mengen und Relationen zu spezifizieren, aus denen Regeln aufgebaut sind. Dadurch ist es möglich, nicht nur Regeln aufbauend aus Produkt- bzw. Dienstleistungskomponenten zu entwerfen, sondern auch die Betrachtung anderer Elemente eines PSSs mit einzubeziehen.

2.1 Allgemeine Mengen und Relationen

Die folgend vorgestellten Definitionen zur Beschreibung von Abhängigkeiten zwischen Elementen in PSSs nutzen die in [3] vorgestellten Formalisierungen zur Beschreibung von Dienstleistungen. Zum besseren Verständnis werden daher zunächst diese Formalisierungen skizziert.

Ein wesentlicher Bestandteil der Beschreibung komplexer Dienstleistungen sind Interdependenzen zwischen Dienstleistungskomponenten. Diese Interdependenzen werden durch hierarchische und nichthierarchische Beziehungen zwischen den Komponenten dargestellt. Im Rahmen von PSSs existieren neben Dienstleistungs- auch Produktkomponenten. Alle Komponenten beider Arten sind in der Menge *Component* zusammengefasst. Zur Darstellung hierarchischer Beziehungen werden die Komponenten in einem sogenannten *Konfigurationsgraphen* miteinander verbunden. Der Konfigurationsgraph bzw. die Komponentenhierarchie ist an dieser Stelle nicht weiter von Bedeutung, daher sei auf die Darstellung in [3] verwiesen.

Jede Komponente kann mit verschiedenen Produktivitätskennzahlen (KPIs) versehen werden. Durch die Kombination verschiedener KPIs lassen sich damit Aussagen zur Gesamtproduktivität einer Konfiguration treffen. Alle verwendeten KPIs eines modellierten Systems sind in der Menge *KPIName* enthalten. Der Wert einer bestimmten KPI einer Komponente ist definiert durch die Abbildung *KPI*.

$$KPI : Component \times KPIName \rightarrow KPIValue \tag{1}$$

Die Menge *KPIValue* der Werte von KPIs ist dabei im Regelfall $\mathbb{R}$, kann aber je nach Anwendungsfall auch domänenspezifisch angepasst sein.

Neben KPIs sind Komponenten weiterhin durch nichtfunktionale Eigenschaften gekennzeichnet. Dies kann bei Produktkomponenten z.B. deren Gewicht oder ihre Größe sein, bei Dienstleistungskomponenten hingegen die Kapazität (für ein Call-Center); Eigenschaften der verschiedenen Komponenten finden sich u.a. in [17]. Die nichtfunktionalen Eigenschaften eines Systems sind in

der Menge *AttributeName* definiert. Der Wert eines bestimmten Attributs einer Komponente ist definiert durch die Abbildung *Attribute*.

$$Attribute : Component \times AttributeName \rightarrow AttributeValue \tag{2}$$

Die Menge *AttributeValue* enthält beliebige Werte, z.B. lassen sich gesprochene Sprachen eines Call Centers als Textwert darstellen. Zu beachten ist hierbei allerdings, dass der Vergleich von Attributwerten (siehe Abschnitt 2.3) nur über reellen Zahlen definiert ist. Dies kann im Zuge künftiger Anpassungen noch erweitert werden.

Um externe Einflüsse auf die Implementierung eines PSSs zu beschreiben, werden Variablen verwendet. Variablen repräsentieren kundenspezifische Gegebenheiten und damit die Umgebung, in der Dienstleistungen erbracht und Produkte installiert werden. Dies können z.B. die erwartete Anzahl eingehender Anrufe bei einem Call Center oder der Dachneigungswinkel bei der Installation einer Photovoltaik-Anlage sein. Die Variablen sind in der Menge *VariableName* definiert. Der Wert einer Variablen ist definiert durch die Abbildung *Variable*.

$$Variable : VariableName \rightarrow VariableValue \tag{3}$$

Zu beachten ist, dass die Werte von Variablen erst während der Konfiguration eines PSSs gesetzt werden. Dadurch müssen während der Modellierung keine Annahmen bezüglich des genauen Wertes getroffen werden. Die Menge *VariableValue* enthält, wie auch die Menge der Attributwerte, beliebige Elemente, wobei der Vergleich über Variablenwerten bisher nur für reelle Zahlen definiert ist.

2.2 KPI-Modifikator

Mit Hilfe des KPI-Modifikators wird die KPI einer Komponente mit einem bestimmten Wert multipliziert. Dadurch lassen sich Regeln ableiten, die z.B. bei der Auswahl bestimmter Komponenten einen Rabatt auf den Preis anderer Komponenten ermöglichen. Der KPI-Modifikator ist definiert als

$$KPIModifier = (Component, KPIName, \mathbb{R}). \tag{4}$$

Die Anwendung bewirkt die Multiplikation der KPI $KPIName$ mit dem angegebenen Faktor

$$\begin{aligned} &\forall c \in Component, k \in KPIName, m \in \mathbb{R} : \\ &(c, k, m) \in KPIModifier \rightarrow KPI(c, k) = KPI(c, k) \cdot m \end{aligned} \tag{5}$$

Das oben angerissene Beispiel zur Definition eines Rabatts lässt sich demnach mit einem KPI-Modifikator realisieren, der den Preis einer Komponente verringert. Im folgenden Beispiel legt der Modifikator *discountA* fest, dass sich der Preis der Komponente A (dargestellt durch die KPI *price*) auf 90 Prozent des eigentlichen Wertes verringert.

$$discountA = (A, price, 0.9) \tag{6}$$

2.3 Vergleichsoperatoren

Mittels Vergleichsoperatoren ist es möglich, ein numerisches Notationselement (Variable, Attribut, KPI) mit einem bestimmten Wert zu vergleichen. Dabei lassen sich die bekannten arithmetischen Vergleichsoperatoren *kleiner als*, *größer als* und *gleich* verwenden.

$$Comparator = \{<, >, =\} \tag{7}$$

Vergleichsoperatoren für Variablen Mit Hilfe von Vergleichsoperatoren für Variablen lässt sich ermitteln, ob eine externe Variable kleiner als, größer als oder gleich zu einem gegebenen Vergleichswert ist.

$$VariableComparator = (VariableName, Comparator, \mathbb{R}) \tag{8}$$

Vergleichsoperatoren für Attribute Mit Hilfe von Vergleichsoperatoren für Attribute lässt sich ermitteln, ob ein Attribut einer bestimmten Komponente kleiner als, größer als oder gleich zu einem gegebenen Vergleichswert ist.

$$AttributeComparator = (AttributeName, Component, Comparator, \mathbb{R}) \tag{9}$$

Vergleichsoperatoren für KPIs Mit Hilfe von Vergleichsoperatoren für KPIs lässt sich ermitteln, ob eine KPI einer bestimmten Komponente kleiner als, größer als oder gleich zu einem gegebenen Vergleichswert ist.

$$KPIComparator = (KPIName, Component, Comparator, \mathbb{R}) \tag{10}$$

3 Regeln

In diesem Abschnitt werden die einzelnen Regeln genauer vorgestellt und formalisiert. Dazu wird zunächst der für alle Regeltypen gültige Vorbereich definiert, d.h. die Vorbedingungen, die erfüllt sein müssen, damit eine Regel zum Tragen kommt. Weiterhin werden mögliche Eigenschaften der Regeln spezifiziert. Im Anschluss folgt die Vorstellung der Regeln unterteilt in die drei Bereiche *Vorschlagend*, *Einschränkend* und *Modifizierend*. Für die einzelnen Regeln wird definiert, welche Eigenschaften sie besitzen und wie sie mit anderen Regeln interagieren.

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

$$R : VB \rightarrow NB \tag{11}$$

3.1 Definition des Vorbereichs

Im einfachsten Fall besteht der Vorbereich einer Regel aus einer Komponente (d.h. in der Konfiguration wurde die entsprechende Komponente gewählt), einem Variablenvergleich, einem Attributvergleich oder einem Vergleich über eine KPI.

$$VB = \{Component, VariableComparator, AttributeComparator, KPIComparator\} \tag{12}$$

Um nicht nur atomare Aussagen im Vorbereich zu verwenden, ist es notwendig, diesen erweitern zu können. Dies wird durch die induktive Erweiterung aussagenlogischer Verknüpfungen zwischen atomaren Formeln ermöglicht.

$$\begin{gathered} a \in VB \wedge b \in VB \rightarrow (a \wedge b) \in VB \\ a \in VB \wedge b \in VB \rightarrow (a \vee b) \in VB \\ a \in VB \rightarrow \neg a \in VB \end{gathered} \tag{13}$$

Mit Hilfe der Erweiterung des Vorbereichs ist es nun möglich auch Kombinationen atomarer Formeln zuzulassen. Dies ermöglicht z.B. die Aussage, dass eine Regel mit dem Vorbereich v zum Tragen kommt, wenn die Komponente A gewählt wurde und der Wert der Variablen x zwischen 5 und 7 beträgt.

$$v = A \wedge (x, >, 4) \wedge (x, <, 8) \tag{14}$$

3.2 Eigenschaften von Regeln

Die Eigenschaften der Regeln orientieren sich an den bekannten Eigenschaften von Relationen. Indem Regeln mit Eigenschaften versehen werden ist es möglich, ihre Semantik sowie Grenzen der Anwendbarkeit genauer zu definieren.

Eigenschaft	Formalisierung	Beschreibung
Symmetrie	$\forall v_1, v_2 \in VB :$ $(v_1, v_2) \in R \rightarrow (v_2, v_1) \in R$	Wenn eine Regel den Vorbereich v_1 besitzt und den Nachbereich v_2, dann gibt es auch eine Regel, die v_2 als Vorbereich und v_1 als Nachbereich hat.
Transitivität	$\forall v_1, v_2, v_3 \in VB :$ $(v_1, v_2) \in R \wedge (v_2, v_3) \in R \rightarrow (v_1, v_3) \in R$	Wenn es eine Regel gibt, die v_1 als Vorbereich und v_2 als Nachbereich hat und eine Regel, die v_2 als Vorbereich und v_3 als Nachbereich hat, so gibt es auch eine Regel, die v_1 als Vorbereich und v_3 als Nachbereich hat.
Irreflexivität	$\forall v \in VB :$ $(v, v) \notin R$	Es ist nicht möglich, eine Regel dieses Typs zu definieren, die den gleichen Vor- wie Nachbereich hat.
Asymmetrie	$\forall v_1, v_2 \in VB :$ $(v_1, v_2) \in R \rightarrow (v_2, v_1) \notin R$	Wenn eine Regel den Vorbereich v_1 und den Nachbereich v_2 hat, dann darf keine Regel existieren, die v_2 als Vorbereich und v_1 als Nachbereich hat.

Tabelle 1. Eigenschaften von Regeln

3.3 Vorschlagende Regeln

Alternative Nach [1] ist eine Motivation für den Denkansatz hinter der Implementierung von PSSs, die Betonung 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. Dementsprechend sind der Vorbereich VB_{sur}, der Nachbereich NB_{sur} sowie das Prädikat *surrogates* wie folgt definiert:

$$\begin{aligned} VB_{sur} &= Component \\ NB_{sur} &= Component \\ surrogates &: VB_{sur} \rightarrow NB_{sur} \end{aligned} \tag{15}$$

Eigenschaften Die Regel zur Beschreibung von Alternativen ist transitiv, d.h. wenn definiert wurde, dass Komponente B eine Alternative der Komponente A und Komponente C eine Alternative der Komponente B ist, dann ist C auch eine Alternative von A. Sie ist weiterhin symmetrisch, wodurch sich aus der Definition, dass B eine Alternative von A ist ergibt, dass A auch eine Alternative von B ist. Schließlich ist die Regel irreflexiv, da keine Komponente eine Alternative von sich selbst sein kann.

Existieren mehrere Alternativen mit dem gleichen Vorbereich, lassen sich diese zusammenfassen, wobei der Nachbereich aus der Disjunktion der vorhandenen Nachbereiche gebildet wird.

$$\begin{aligned} &\forall a \in VB_{sur}, \forall b, c \in NB_{sur} : \\ &(a, b) \in surrogates \wedge (a, c) \in surrogates \rightarrow (a, b \vee c) \in surrogates \end{aligned} \tag{16}$$

Beziehungen zu anderen Regeln Eine Komponente, die eine Alternative zu einer anderen Komponente ist, darf keine exklusive Alternative (siehe Abschnitt 3.4) dieser Komponente sein.

$$\begin{aligned} &\forall a \in VB_{sur} \cap VB_{xsu}, \forall b \in NB_{sur} \cap NB_{xsu} : \\ &(a, b) \in surrogates \rightarrow (a, b) \notin xSurrogates \end{aligned} \tag{17}$$

Beispiel Die Komponenten A und B sind eine Alternative von einander.

$$surrogates(A) = B \tag{18}$$

Empfehlung Dieser Abhängigkeitsbeziehung liegt die Annahme zugrunde, dass Anbieter ihren Kunden empfehlen, bei der Wahl eines Produkts eine bestimmte Dienstleistung ebenfalls auszuwählen. So legt Uhlmann bspw. dar, dass die Produktivität einer Anlage nicht nur von deren materiellen Eigenschaften abhängt, sondern auch vom Qualifikationsstand der Mitarbeiter und dem Wartungszustand der Maschine [16]. 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 die Erwartungen von Kunden übertreffen [6]. Zur Darstellung einer Empfehlung wird das Prädikat *recommends* verwendet. Der Vorbereich VB_{rec}, Nachbereich NB_{rec} sowie das Prädikat *recommends* sind wie folgt definiert:

$$\begin{aligned} VB_{rec} &= VB \\ NB_{rec} &= VB \\ recommends &: VB_{rec} \rightarrow NB_{rec} \end{aligned} \tag{19}$$

Eigenschaften Die Empfehlung ist irreflexiv, d.h. eine Komponente (oder ein beliebiges anderes Element des Vorbereichs) kann sich nicht selbst empfehlen.

Existieren mehrere Empfehlungen mit dem gleichen Vorbereich, lassen sich diese zusammenfassen, wobei der Nachbereich aus der Disjunktion der vorhandenen Nachbereiche gebildet wird.

$$\begin{aligned} &\forall a \in VB_{rec}, \forall b, c \in NB_{rec} : \\ &(a, b) \in recommends \wedge (a, c) \in recommends \rightarrow (a, b \vee c) \in recommends \end{aligned} \tag{20}$$

Beziehungen zu anderen Regeln Elemente, die sich gegenseitig empfehlen, dürfen nicht in der Nicht-Empfehlungs-Relation (siehe nachfolgende Regel) miteinander stehen.

$$\begin{aligned} &\forall a \in VB_{rec} \cap VB_{\neg rec}, b \in NB_{rec} \cap NB_{\neg rec} : \\ &(a, b) \in recommends \rightarrow (a, b) \notin recommendsNot \end{aligned} \tag{21}$$

Elemente, die sich gegenseitig empfehlen, dürfen sich nicht ausschließen (siehe Abschnitt 3.4).

$$\begin{aligned} &\forall a \in VB_{rec} \cap VB_{pro}, b \in NB_{rec} \cap NB_{pro} : \\ &(a, b) \in recommends \rightarrow (a, b) \notin prohibits \end{aligned} \tag{22}$$

Elemente, die sich gegenseitig empfehlen, dürfen keine exklusive Alternative voneinander sein.

$$\begin{aligned} &\forall a \in VB_{rec} \cap VB_{xsu}, b \in NB_{rec} \cap NB_{xsu} : \\ &(a, b) \in recommends \rightarrow (a, b) \notin xSurrogates \end{aligned} \tag{23}$$

Beispiel Die Kombination der Komponente A und der Erwartung, dass in einem Call Center mehr als 5000 Anrufe pro Tag eingehen (dargestellt durch die Variable *incomingCalls*) empfiehlt einerseits die Auswahl der Komponente B und andererseits, dass die KPI *price* der Komponente C kleiner als 500 sein soll.

$$\begin{aligned} incomingMin &= (incomingCalls, >, 5000) \\ priceC &= (price, C, <, 500) \\ recommends(A \wedge incomingMin) &= (B \wedge priceC) \end{aligned} \tag{24}$$

Nicht-Empfehlung Hiermit wird eine abgeschwächte Form der Ausschlussbeziehung (siehe Abschnitt 3.4) beschrieben. Die Auswahl verschiedener Elemente kann in ihrer Kombination dazu führen, dass die Produktvititität des PSSs sinkt oder die Kosten des Systems steigen, ohne dass sich für die Kunden ein direkter Mehrwert ergibt. Um diese Auswirkungen zu visualisieren, kann

es angebracht sein, die jeweiligen Abhängigkeiten mittels des Prädikats *recommendsNot* explizit darzustellen. Vor- und Nachbereich dieses Prädikats können dabei beliebig gewählt werden.

$$\begin{aligned} VB_{\neg rec} &= VB \\ NB_{\neg rec} &= NB \\ recommendsNot &: VB_{\neg rec} \rightarrow NB_{\neg rec} \end{aligned} \tag{25}$$

Eigenschaften Wie auch die Empfehlung ist die Nicht-Empfehlung irreflexiv, d.h. ein Element des Vorbereichs kann sich nicht selber nicht empfehlen.

Existieren mehrere Nicht-Empfehlungen mit dem gleichen Vorbereich, lassen sich diese zusammenfassen, wobei der Nachbereich aus der Disjunktion der vorhandenen Nachbereiche gebildet wird.

$$\begin{aligned} &\forall a \in VB_{\neg rec}, \forall b, c \in NB_{\neg rec} : \\ &(a, b) \in recommendsNot \wedge (a, c) \in recommendsNot \rightarrow (a, b \vee c) \in recommendsNot \end{aligned} \tag{26}$$

Beziehungen zu anderen Regeln Die Nicht-Empfehlung lässt sich auch durch die Empfehlung darstellen, indem deren Nachbereich negiert wird.

$$\begin{aligned} &\forall a \in VB_{\neg rec}, \forall b \in NB_{\neg rec} : \\ &(a, b) \in recommendsNot \rightarrow (a, \neg b) \in recommends \end{aligned} \tag{27}$$

Elemente, die sich gegenseitig nicht empfehlen, dürfen sich nicht gegenseitig empfehlen.

$$\begin{aligned} &\forall a \in VB_{\neg rec} \cap VB_{rec}, b \in NB_{\neg rec} \cap NB_{rec} : \\ &(a, b) \in recommendsNot \rightarrow (a, b) \notin recommends \end{aligned} \tag{28}$$

Elemente, die sich gegenseitig nicht empfehlen, dürfen keine Voraussetzung voneinander sein.

$$\begin{aligned} &\forall a \in VB_{\neg rec} \cap VB_{req}, b \in NB_{\neg rec} \cap NB_{req} : \\ &(a, b) \in recommendsNot \rightarrow (a, b) \notin requires \end{aligned} \tag{29}$$

Beispiel Die Kombination der Komponente A und der Erwartung, dass in einem Call Center mehr als 5000 Anrufe pro Tag eingehen (dargestellt durch die Variable *incomingCalls*) empfiehlt nicht die Auswahl der Komponente B und empfiehlt weiterhin nicht, dass die KPI *price* der Komponente C kleiner als 500 sein soll.

$$\begin{aligned} incomingMin &= (incomingCalls, >, 5000) \\ priceC &= (price, C, <, 500) \\ recommendsNot(A \wedge incomingMin) &= (B \wedge priceC) \end{aligned} \tag{30}$$

3.4 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 [4,15]. Sun definiert eine Schnittstelle, über die Dienstleistungen angeboten werden als Service Carrier [15]. 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 skizzieren Böhmann und Krcmar,

indem Dienstleistungen und Produkte in den Wertschöpfungsprozess der Kunden integriert werden müssen [4]. Eine Voraussetzung wird durch das Prädikat *requires* dargestellt.

$$\begin{aligned} VB_{req} &= VB \\ NB_{req} &= NB \\ requires &: VB_{rec} \rightarrow NB_{req} \end{aligned} \tag{31}$$

Eigenschaften Voraussetzungen sind irreflexiv, d.h. ein Element kann sich nicht selber voraussetzen.

Existieren mehrere Voraussetzungen mit dem gleichen Vorbereich, lassen sich diese zusammenfassen, wobei der Nachbereich aus der Konjunktion der vorhandenen Nachbereiche gebildet wird.

$$\begin{aligned} &\forall a \in VB_{req}, \forall b, c \in NB_{req} : \\ &(a,b) \in requires \wedge (a,c) \in requires \rightarrow (a, b \wedge c) \in requires \end{aligned} \tag{32}$$

Beziehungen zu anderen Regeln Ein PSS-Element, welches ein anderes Element als Voraussetzung besitzt, darf dieses nicht gleichzeitig nicht empfehlen.

$$\begin{aligned} &\forall a \in VB_{req} \cap VB_{\neg rec}, \forall b \in NB_{req} \cap NB_{\neg rec} : \\ &(a,b) \in requires \rightarrow (a,b) \notin recommendsNot \end{aligned} \tag{33}$$

Ein PSS-Element, welches ein anderes Element als Voraussetzung besitzt, kann dieses nicht gleichzeitig ausschließen.

$$\begin{aligned} &\forall a \in VB_{req} \cap VB_{pro}, \forall b \in NB_{req} \cap NB_{pro} : \\ &(a,b) \in requires \rightarrow (a,b) \notin prohibits \end{aligned} \tag{34}$$

Elemente, die sich als Voraussetzung besitzen, dürfen keine exklusive Alternative voneinander sein.

$$\begin{aligned} &\forall a \in VB_{req} \cap VB_{xsu}, \forall b \in NB_{req} \cap NB_{xsu} : \\ &(a,b) \in requires \rightarrow (a,b) \notin xSurrogates \end{aligned} \tag{35}$$

Beispiel Die Kombination der Komponente A und der Erwartung, dass in einem Call Center mehr als 5000 Anrufe pro Tag eingehen (dargestellt durch die Variable *incomingCalls*) hat als Voraussetzung, dass die Komponente B ausgewählt wurde sowie die KPI *price* der Komponente C kleiner als 500 ist.

$$\begin{aligned} incomingMin &= (incomingCalls, >, 5000) \\ priceC &= (price, C, <, 500) \\ requires(A \wedge incomingMin) &= (B \wedge priceC) \end{aligned} \tag{36}$$

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. Dienstleistungshierarchien ist es möglich, durch die Verwendung von Exklusiv-Oder-Konnektoren gegenseitige Ausschlüsse von Komponenten zu definieren. Im Gegensatz dazu müssen nichthierarchische Ausschlüsse zwischen Komponenten bzw. zwischen unterschiedlichen Elementtypen explizit gekennzeichnet werden. Dies erfolgt mittels des Prädikats *prohibits*.

$$\begin{aligned} VB_{pro} &= VB \\ NB_{pro} &= NB \\ prohibits &: VB_{pro} \rightarrow NB_{pro} \end{aligned} \tag{37}$$

Eigenschaften Die Regel Ausschluss ist irreflexiv, d.h. ein Element eines PSSs kann sich nicht selber ausschließen. Der Ausschluss ist symmetrisch, d.h. wenn Element A Element B ausschließt, folgt daraus, dass B auch A ausschließt.

Existieren mehrere Ausschlüsse mit dem gleichen Vorbereich, lassen sich diese zusammenfassen, wobei der Nachbereich aus der Disjunktion der vorhandenen Nachbereiche gebildet wird.

$$\begin{aligned} &\forall a \in VB_{pro}, \forall b, c \in NB_{pro} : \\ &(a, b) \in prohibits \wedge (a, c) \in prohibits \rightarrow (a, b \vee c) \in prohibits \end{aligned} \tag{38}$$

Beziehungen zu anderen Regeln Elemente, die sich gegenseitig ausschließen, dürfen sich nicht empfehlen.

$$\begin{aligned} &\forall a \in VB_{pro} \cap VB_{rec}, \forall b \in NB_{pro} \cap NB_{rec} : \\ &(a, b) \in prohibits \rightarrow (a, b) \notin recommends \end{aligned} \tag{39}$$

Elemente, die sich gegenseitig ausschließen, dürfen keine Voraussetzung voneinander sein.

$$\begin{aligned} &\forall a \in VB_{pro} \cap VB_{req}, \forall b \in NB_{pro} \cap NB_{req} : \\ &(a, b) \in prohibits \rightarrow (a, b) \notin requires \end{aligned} \tag{40}$$

Beispiel Die Kombination der Komponente A und der Erwartung, dass in einem Call Center mehr als 5000 Anrufe pro Tag eingehen (dargestellt durch die Variable *incomingCalls*) schließt die Auswahl der Komponente B aus und verbietet weiterhin, dass die KPI *price* der Komponente C kleiner als 500 ist.

$$\begin{aligned} incomingMin &= (incomingCalls, >, 5000) \\ priceC &= (price, C, <, 500) \\ prohibits(A \wedge incomingMin) &= (B \wedge priceC) \end{aligned} \tag{41}$$

Exklusive Alternative Mit der exklusiven Alternative werden Elemente beschrieben, die Alternativen voneinander sind, aber nicht gemeinsam in einer Konfiguration auftreten dürfen. Dies können bspw. Produkt- und Dienstleistungskomponenten sein, die die gleiche Funktionalität anbieten aber nicht miteinander kompatibel sind. Die exklusive Alternative wird durch das Prädikat *xSurrogates* dargestellt. Eine exklusive 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.

$$\begin{aligned} VB_{xsu} &= Component \\ NB_{xsu} &= Component \\ xSurrogates &: VB_{xsu} \rightarrow NB_{xsu} \end{aligned} \tag{42}$$

Eigenschaften Die Regel exklusive Alternative ist irreflexiv, d.h. ein Element eines PSSs kann keine Alternative zu sich sein und sich nicht gleichzeitig selber ausschließen. Die exklusive Alternative ist symmetrisch, d.h. wenn Element A eine exklusive Alternative von Element B ist, dann ist auch Element B eine exklusive Alternative von Element A.

Existieren mehrere exklusive Alternativen mit dem gleichen Vorbereich, lassen sich diese zusammenfassen, wobei der Nachbereich aus der Disjunktion der vorhandenen Nachbereiche gebildet wird.

$$\begin{aligned} &\forall a \in VB_{xsu}, \forall b, c \in NB_{xsu} : \\ &(a, b) \in xSurrogatse \wedge (a, c) \in xSurrogates \rightarrow (a, b \vee c) \in xSurrogates \end{aligned} \tag{43}$$

Beziehungen zu anderen Regeln Die exklusive Alternative lässt sich durch die Kombination der Alternative mit dem Ausschluss darstellen, indem beide kombiniert werden

$$\begin{aligned} &\forall a \in VB_{xsu}, \forall b \in NB_{xsu} : \\ &(a,b) \in xSurrogates \rightarrow (a,b) \in surrogates \wedge (a,b) \in prohibits \end{aligned} \tag{44}$$

Elemente, die eine exklusive Alternative voneinander sind, dürfen sich nicht empfehlen.

$$\begin{aligned} &\forall a \in VB_{xsu} \cap VB_{rec}, \forall b \in NB_{xsu} \cap NB_{rec} : \\ &(a,b) \in xSurrogates \rightarrow (a,b) \notin recommends \end{aligned} \tag{45}$$

Elemente, die eine exklusive Alternative voneinander sind, dürfen keine Voraussetzung voneinander sein.

$$\begin{aligned} &\forall a \in VB_{xsu} \cap VB_{req}, \forall b \in NB_{xsu} \cap NB_{req} : \\ &(a,b) \in xSurrogates \rightarrow (a,b) \notin requires \end{aligned} \tag{46}$$

Beispiel Die Komponenten A und B sind eine exklusive Alternative von einander.

$$xSurrogates(A) = B \tag{47}$$

Wie oben beschrieben lässt sich diese Beziehung auch durch eine Kombination der Regeln Alternative und Ausschluss beschreiben. Demnach ist das Beispiel auch durch folgende zwei Regeln darstellbar.

$$\begin{aligned} surrogates(A) &= B \\ prohibits(A) &= B \end{aligned} \tag{48}$$

3.5 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 [12]. 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.

$$\begin{aligned} VB_{chv} &= VB \\ NB_{chv} &= KPIModifier \\ changesValue &: VB_{chv} \rightarrow NB_{chv} \end{aligned} \tag{49}$$

Eigenschaften Existieren mehrere Wertänderungen mit dem gleichen Vorbereich, lassen sich diese zusammenfassen, wobei der Nachbereich aus der Konjunktion der vorhandenen Nachbereiche gebildet wird.

$$\begin{aligned} &\forall a \in VB_{chv}, \forall b, c \in NB_{chv} : \\ &(a,b) \in changesValue \wedge (a,c) \in changesValue \rightarrow (a, b \wedge c) \in changesValue \end{aligned} \tag{50}$$

Beziehungen zu anderen Regeln Ein Element, das den Wert eines anderen Elements ändert, darf dieses Element nicht gleichzeitig ausschließen. Der Nachbereich der Wertänderung legt fest, dass nur KPIs von Komponenten geändert werden dürfen; daher darf die Vorbedingung einer Wertänderungsregel diese Komponente nicht ausschließen.

$$\begin{aligned}&\forall a \in VB_{chv} \cap VB_{pro}, \forall b \in NB_{chv}, \forall c \in Components, \forall k \in KPINames, \forall r \in \mathbb{R} : \\ &(a,b) \in changesValue \wedge a = (c,k,r) \rightarrow (a,c) \notin prohibits\end{aligned} \tag{51}$$

Ein Element, das den Wert eines anderen Elements ändert, darf keine exklusive Alternative dieses Elements sein. Hier gilt wie oben bereits die Bedingung, dass die entsprechende Komponente des Nachbereichs keine exklusive Alternative des Vorbereichs sein darf.

$$\begin{aligned}&\forall a \in VB_{chv} \cap VB_{xsu}, \forall b \in NB_{chv}, \forall c \in Components, \forall k \in KPINames, \forall r \in \mathbb{R} : \\ &(a,b) \in changesValue \wedge a = (c,k,r) \rightarrow (a,c) \notin xSurrogates\end{aligned} \tag{52}$$

Beispiel Die Auswahl der Komponenten A und B führt zu einem Rabatt von 10 Prozent bei der Komponente C. Der Rabatt wird dargestellt, indem die KPI *price* mit dem Faktor 0.9 multipliziert wird.

$$\begin{aligned}discountC &= (C, price, 0.9) \\ changesValue(A \wedge B) &= discountC\end{aligned} \tag{53}$$

3.6 Sonstige Regeln

Externe Abhängigkeit Bestimmte Dienstleistungen lassen sich nur durch Kollaboration mit anderen Unternehmen durchführen, z.B. die Warenrücknahme von Produkten über die Filialen konkurrierender Anbieter. Dies ist insbesondere in komplexen PSSs häufig anzutreffen [15]. Um die Konsistenz des Modells zu wahren und auch die Einhaltung eventueller vertraglicher Rahmenbedingungen zu gewährleisten, ist es sinnvoll, diese externen Abhängigkeiten in das PSS-Modell zu integrieren. Da die externen Leistungen außerhalb des Zuständigkeitsbereichs des jeweiligen Anbieters liegen, referenziert das Prädikat *requiresExternal* nur die generische Leistungsbeschreibung eines externen Anbieters, dargestellt durch die Menge *CapabilityDescription*.

$$\begin{aligned}VB_{ree} &= VB \\ NB_{ree} &= CapabilityDescription \\ requiresExternal &: VB_{ree} \rightarrow NB_{ree}\end{aligned} \tag{54}$$

Eigenschaften Existieren mehrere externe Abhängigkeiten mit dem gleichen Vorbereich, lassen sich diese zusammenfassen, wobei der Nachbereich aus der Konjunktion der vorhandenen Nachbereiche gebildet wird.

$$\begin{aligned}&\forall a \in VB_{ree}, \forall b, c \in NB_{ree} : \\ &(a,b) \in requiresExternal \wedge (a,c) \in requiresExternal \rightarrow (a, b \wedge c) \in requiresExternal\end{aligned} \tag{55}$$

Beziehungen zu anderen Regeln Da die externe Abhängigkeit im Nachbereich keine Elemente des modellierten PSS fokussiert, bestehen keine Beziehungen zu anderen Regeln.

Beispiel Werden in einem Call Center System werden mehr als 5000 Anrufe pro Tag (dargestellt durch die Variable *incomingCalls*) erwartet, ist es notwendig, von externen Anbietern Weiterleitungsfunktionalitäten (dargestellt durch die Leistungsbeschreibung *routing*) zu beziehen.

$$\begin{aligned}incomingMin &= (incomingCalls, >, 5000) \\ requiresExternal(incomingMin) &= routing\end{aligned} \tag{56}$$

4 Fazit

Das vorliegende Dokument beschreibt Regeln zur Definition von Abhängigkeiten zwischen Elementen eines PSSs. Durch die Formalisierung lassen sich die Regeln auch automatisch und somit schon während der Konfiguration von PSSs überprüfen. Dies wurde bisher exemplarisch am Beispiel von Dienstleistungen für Photovoltaik-Anlagen geprüft. Dabei liegt die Beschreibung des Systems in Form von Prolog-Regeln dar[1]. Damit ist es möglich, die Eigenschaften der Regeln sowie ihre Beziehungen untereinander zu prüfen.

Die nächsten Schritte zur Verbesserung des Modells bestehen darin, das existierende Werkzeug zur Modellierung von Dienstleistungen aus Komponenten [10] auch auf die Anwendung auf PSSs zu erweitern. Darauf aufbauend werden die existierenden Regeln mit Praxispartnern weiter validiert und auf Verbesserungen hin untersucht. Außerdem sollen durch weitere Workshops zusätzliche Regeln gefunden werden, die Relevanz in der Praxis besitzen. Die damit gewonnen Erkenntnisse werden mit Hilfe der bereitgestellten Methoden formalisiert.

Literatur

1. Baines, T.S., Lightfoot, H.W., Evans, S., Neely, A., Greenough, R., Peppard, J., Roy, R., Shehab, E., Braganza, A., Tiwari, A., Alcock, J.R., Angus, J.P., Bastl, M., Cousens, A., Irving, P., Johnson, M., Kingston, J., Lockett, H., Martinez, V., Michele, P., Tranfield, D., Walton, I.M., Wilson, H.: State-of-the-art in product-service systems. Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture 221(10), 1543–1552 (2007), http://pib.sagepub.com/content/221/10/1543.abstract
2. Becker, J., Beverungen, D., Knackstedt, R.: Reference models and modeling languages for product-service systems? status-quo and perspectives for further research. In: Hawaii International Conference on System Sciences. p. 105. IEEE Computer Society, Waikoloa, Big Island, Hawaii (2008)
3. Becker, M.: Formales Metamodell für Dienstleistungskomponenten. Tech. rep., Universität Leipzig (2011), to appear
4. Böhmann, T., Krcmar, H.: Hybride produkte: Merkmale und herausforderungen. In: Bruhn, M., Stauss, B. (eds.) Wertschöpfungsprozesse bei Dienstleistungen, pp. 239–255. Gabler, Wiesbaden, Germany (2007), http://www.springerlink.com/content/m485574h83501362/
5. Böttcher, M., Becker, M., Klingner, S.: Dienstleistungsmodularisierung zur kundenindividuellen Konfiguration. In: Hans-Ulrich Heiß, Peter Pepper, H.S.J.S. (ed.) Informatik 2011: Informatik schafft Communities. Lecture Notes in Informatics, Gesellschaft für Informatik, Bonner Köllen Verlag, Berlin, Germany (October 2011)
6. Chase, R.B., Hayes, R.H.: Beefing up operations in service firms. Sloan Management Review 33(1), 15–26 (1991), http://sloanreview.mit.edu/the-magazine/1991-fall/3312/beefing-up-operations-in-service-firms/
7. Dietze, V.: Festigung zwischenbetrieblicher Kollaboration durch den Einsatz von Group Decision Support Software - ein strategischer Planungsansatz. Grin Verlag (October 2008)
8. Hildebrand, W.C., Klostermann, T.: Dienstleistungsverkehr in industriellen wertschöpfungsprozessen. In: Bruhn, M., Stauss, B. (eds.) Wertschöpfungsprozesse bei Dienstleistungen, pp. 215–236. Gabler, Wiesbaden, Germany (2007), http://dx.doi.org/10.1007/978-3-8349-9285-7_10
9. Isaksson, O., Larsson, T.C., Rönnbäck, A.h.: Development of product-service systems: challenges and opportunities for the manufacturing firm. Journal of Engineering Design 20(4), 329–348 (April 2009), http://www.tandfonline.com/doi/abs/10.1080/09544820903152663
10. Klingner, S., Böttcher, M., Becker, M., Döhler, A.: Managing complex service portfolios. In: Ganz, W., Kicherer, F., Schletz, A. (eds.) RESER 2011 Productivity of Services NextGen - Beyond Output/Input. Conference Proceedings. (September 2011)
11. Knackstedt, R., Pöppelbuß, J., Winkelmann, A.: Integration von sach- und dienstleistungen – ausgewählte internetquellen zur hybriden wertschöpfung. WIRTSCHAFTSINFORMATIK 50, 235–247 (2008), http://dx.doi.org/10.1365/s11576-008-0050-0
12. Mont, O.K.: Clarifying the concept of product-service system. Journal of Cleaner Production 10(3), 237 – 245 (2002), http://www.sciencedirect.com/science/article/pii/S0959652601000397

[1] https://sourceforge.net/projects/kpstools/

13. Morelli, N.: Designing product/service systems: A methodological exploration. Design Issues 18(3), pp. 3–17 (2002), http://www.jstor.org/stable/1512062
14. Stille, F.: Produktbegleitende dienstleistungen gewinnen weiter an bedeutung. Wochenbericht 70(21), 335–342 (2003), http://EconPapers.repec.org/RePEc:diw:diwwob:70-210-20
15. Sun, H.: Product service relationship: defining, modelling and evaluating. International Journal of Internet Manufacturing and Services 2(2), 128–141 (February 2010), http://inderscience.metapress.com/content/2804x154t0107314/
16. Uhlmann, E., Meier, H., Bochnig, H., Geisert, C., Sadek, K., Stelzer, C.: Customer-driven development of product-service-systems. In: Pham, D.T., Eldukhri, E.E., Soroka, A.J. (eds.) Innovative production machines and systems : Fourth I*PROMS Virtual International Conference, 1st - 14th July, 2008. Whittles Publ. [u.a.] (2008), http://conference.iproms.org/customer_driven_development_of_product_service_systems
17. Weber, C., Steinbach, M., Botta, C., Deubel, T.: Modelling of product-service systems (pss) based on the pdd approach. In: D., M. (ed.) Proceedings of the 8th International Design Conference DESIGN 2004. pp. 547–554. Dubrovnik, Croatia (May 2004), http://scidok.sulb.uni-saarland.de/volltexte/2008/1619/