Ein Metamodell zur formalen Darstellung und Konfiguration komplexer Dienstleistungen

Michael Becker, Stephan Klingner
Abteilung für Betriebliche Informationssysteme
Johannisgasse 26
04103 Leipzig
{mbecker, klingner}@informatik.uni-leipzig.de

Kurzfassung: Die zunehmende Industrialisierung von Dienstleistungen verlangt von Anbietern neue, effektive Ansätze, um sich Herausforderungen wie steigender Konkurrenz, Komplexität und vermehrtem Preisdruck stellen zu können. Verschiedene Ansätze wie beispielsweise Mass Customisation können dabei aus anderen Wirtschaftssektoren auf den Dienstleistungssektor übertragen werden. Dieser Beitrag stellt dazu ein Metamodell vor, welches die Konzepte der Standardisierung und Komponentisierung umsetzt sowie einen formalen Rahmen zur Modellierung und Konfiguration von Dienstleistungen bietet.

1 Einleitung

Dienstleistungsunternehmen stehen heutzutage vielfach vor der Herausforderung, in hochkompetitiven und zunehmend internationalen Märkten zu bestehen. Auch industrielle Dienstleistungen sind dabei in steigendem Maße kundenindividuell und zu wettbewerbsfähigen Preisen zu erbringen [Pap04]. Um der Dichotomie aus Preisdruck und steigender Komplexität aufgrund von Individualisierung adäquat begegnen zu können, liegt die Adaption von Ansätzen aus dem industriellen Sektor sowie der Softwartechnik nahe, welche in der Vergangenheit ähnlichen Problemen gegenüberstanden [McI68, Sun94]. Ein entsprechendes Konzept wird mit dem Begriff *Mass Customisation* beschrieben [Har95, PI99]. Die grundlegende Intention des Mass Customisation ist die Einteilung komplexer, monolithischer Strukturen in einfachere, funktional abgegrenzte sowie standardisierte Komponenten. Durch die sogenannte Komponentisierung (bzw. Modularisierung) können trotz kundenindividuell konfigurierbarer Dienstleistungen die Vorteile von Standardisierung wie Skaleneffekte oder vereinfachte Wartung nutzbar gemacht werden [HPT05, BK11].

Die Anwendbarkeit von Komponentisierung bzw. Mass Customisation im Bereich von Dienstleistungen ist in der Literatur bereits umfänglich diskutiert worden [Sun94, HPT05, BK10]. Zur Umsetzung dieser grundlegenden methodischen Konzepte in nutzbare Werkzeuge ist ein formaler Rahmen zu erstellen, welcher die Basis für eine zu erstellende Software bildet. Nachfolgend wird dementsprechend ein Metamodell zur Dienstleistungsbeschreibung vorgestellt, welches die Ansätze der Komponentisierung und Standardisierung berücksichtigt. Der anvisierte Funktionsumfang des Metamodells geht dabei über die bloße formale Beschreibung von Dienstleistungen hinaus. Das Ziel ist die Unterstützung der Erstellung kundenindividueller Konfigurationen. Dazu sind wirtschaftliche Kennzahlen und verschiedene Abhängigkeitsbeziehungen darzustellen. Auf

Tabelle 1: Konzepte zur Modellierung von Dienstleistungen und ihre Umsetzung im Metamodell

Konzept	Bestandteile
Modularisierte Beschreibung von Funktionalitäten	Komponente
Komposition und Dekomposition von Komponenten	Hierarchische Abhängigkeiten
	Kardinalitäten
Allgemeine Beziehungen zwischen Modellelementen	Logische Abhängigkeiten
Prozesssicht auf Komponenten	Zeitliche Abhängigkeiten
Produktivitätsbetrachtung	Key Performance Indicators
Externe Einflussfaktoren	Variablen
Nichtfunktionale Eigenschaften	Attribute
Funktionale Eigenschaften	Prozessmodelle

Basis dieser Beschreibung lassen sich dann kundenindividuelle Angebote hinsichtlich Validität und Produktivität bewerten.

Das vorgestellte Metamodell fasst dabei als Ergebnis Anforderungen und Anregungen zusammen, welche auf verschiedenen Wegen in Zusammenarbeit mit Vertretern der Praxis identifiziert wurden. Grundlegender Bedarf an Methoden oder Tools zur Strukturierung von Dienstleistungsportfolios wurde dabei im Rahmen einer qualitativen Befragung von Dienstleistungsunternehmen festgestellt [BM12]. Ebenso wurden in zahlreichen Workshops mit Projektpartnern funktionale Anforderungen und Weiterentwicklungen diskutiert [KBBD11]. Die Praxisrelevanz des Themas zeigte sich auch in verschiedenen Arbeitskreistreffen, welche zum Thema “Produktivität von Dienstleistungssystemen” durchgeführt wurden.

Im Rahmen dieses Beitrags wird das formale Metamodell vorgestellt. Im Fokus dieser Arbeit steht dabei die ganzheitliche Formalisierung der Konzepte zur Modellierung komplexer Dienstleistungen. Dazu werden in Abschnitt 2 zunächst Konzepte zur Modellierung von Dienstleistungen eingeführt und beschrieben. Aufbauend auf diesen allgemeinen Konzepten wurde das konkrete Metamodell entwickelt; die Formalisierung des Metamodells ist Bestandteil von Abschnitt 3. Ziel der Nutzung des Modells ist die kundenindividuelle Konfiguration komplexer Dienstleistungen. Dies wird beispielhaft in Abschnitt 4 gezeigt. Zur praktischen Anwendbarkeit ist es notwendig, geeignete Werkzeuge bereitzustellen, die auf dem formal definierten Modell aufbauen. Der Prototyp eines Werkzeugs wird in Abschnitt 5 gezeigt. Um einen Überblick über die Einsatzbarkeit des Modells zu geben, wird in Abschnitt 6 gezeigt, in welchen Phasen des Service Engineering das Modell Unterstützung leisten kann. Abschnitt 7 schließlich fasst den Beitrag zusammen und zeigt Ansätze für weitere Forschungen im Bereich Dienstleistungsmodellierung.

2 Konzepte zur Modellierung von Dienstleistungen

In diesem Abschnitt werden die Konzepte des Metamodells vorgestellt, die zur Definition und Modellierung von Dienstleistungen genutzt werden können. Mit Hilfe dieser Übersicht kann von den Spezifika des Metamodells abstrahiert und ein Vergleich mit anderen Ansätzen zur Modellierung von Dienstleistungen durchgeführt werden. Die einzelnen Konzepte sind in Tabelle 1 zusammengefasst dargestellt und werden in den folgenden Abschnitten genauer erläutert.

2.1 Modularisierte Beschreibung von Funktionalitäten

Durch die zunehmende Wichtigkeit von Dienstleistungen und aufgrund gestiegener Kundenanforderungen ist es heutzutage nicht mehr mit vertretbarem Aufwand möglich, komplexe Dienstleistungen als einen monolithischen Block zu betrachten. Aus diesem Grund rücken bei der Entwicklung von Dienstleistung zunehmend Konzepte zur Modularisierung in den Vordergrund. Durch modulare Beschreibungen lassen sich komplexe Produkte aus kleineren Einzelteilen zusammensetzen, die unabhängig voneinander entwickelt werden können [BC97].

Das Metamodell setzt das Konzept der Modularisierung durch die Nutzung von *Dienstleistungskomponenten* um. Eine Komponente stellt eine wohldefinierte, abgrenzbare Funktionalität dar, die durch logisch zusammengehörige Aktivitäten erbracht wird [BF09]. Zur Erbringung der Funktionalität einer Komponente ist es notwendig, Ressourcen zu verbrauchen und zu verändern. Das Metamodell fokussiert die Beschreibung einzelner Komponenten und die Beschreibung der Interaktion zwischen Komponenten. Um die Interaktion zu ermöglichen, wird die Funktionalität einer Komponente über präzise definierte Schnittstellen bereitgestellt.

Die Nutzung von Komponenten ermöglicht, einen Ausgleich zwischen dem steigenden Bedarf nach Individualisierung und der Notwendigkeit der Standardisierung von Dienstleistungen herzustellen. Dadurch können die Ideen des Mass Customisation umgesetzt werden. Die Individualisierung ist notwendig zur Erstellung kundenindividueller Angebote und ermöglicht damit unter anderem eine Abgrenzung von Konkurrenten. Demgegenüber steht die Standardisierung zur Steigerung der Produktivität von Dienstleistungen [HPT05]. Im Gegensatz zur schwierig umzusetzenden Standardisierung eines gesamten Dienstleistungsangebot ist es möglich, einzelne Komponenten getrennt voneinander zu betrachten und diese zu standardisieren. Durch die Kombination verschiedener, standardisierter Komponenten lassen sich kundenindividuelle Angebote erstellen.

2.2 Komposition und Dekomposition von Komponenten

Um die Generierung kundenindividueller Angebote basierend auf standardisierten Komponenten zu ermöglichen, sind Regeln notwendig, anhand derer sich Komponenten zusammensetzen lassen. Die Zusammensetzung der Komponenten wird auch *Komposition* genannt. Die Funktionalität einer zusammengesetzten Komponente entspricht der Summe der Funktionalitäten der einzelnen Komponenten [BK11]. Umgekehrt ist es auch möglich, eine Komponente in ihre Teilbestandteile zu zerlegen und diese getrennt voneinander zu betrachten (*Dekomposition*).

Im Metamodell ist ein *Portfolio* die strukturierte Aufstellung aller Komponenten, d.h. im Portfolio ist definiert, wie sich einzelne Komponenten zusammensetzen und welche Beziehungen sie untereinander haben. Die hierarchischen Beziehungen von Komponenten werden mittels des *Konfigurationsgraphen* dargestellt. Abbildung 1 zeigt einen beispielhaften Ausschnitt hierarchischer Beziehungen zwischen Komponenten einer Wartungsdienstleistung. Dabei besteht die Gesamtleistung *Wartung* aus den einzelnen Komponenten *Schulung*, *Reinigung* und *Sichtprüfung*.

Zu beachten ist, dass in Abbildung 1 nur obligatorische Beziehungen vorhanden sind. In der Realität tritt allerdings oftmals der Fall auf, dass es verschiedene alternative Möglichkeiten gibt, eine Gesamtleistung auszuführen. Beispielsweise kann die Schulung bei der Wartung fakultativ

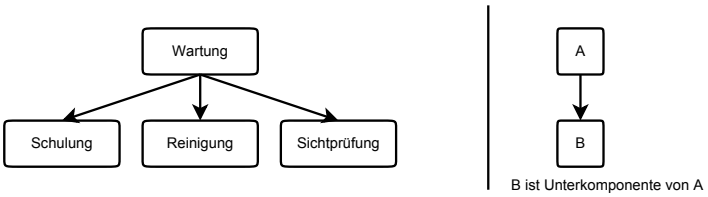

Abbildung 1: Hierarchische Beziehungen zwischen Komponenten einer Wartungsdienstleistung

sein und muss nicht zwangsweise erworben werden. Das Metamodell erlaubt daher mit Hilfe von *Kardinalitäten* den Typ der Zusammensetzung einer Komponente genauer zu spezifizieren (siehe Abschnitt 3.1). Dies ist in Analogie zur Feature-Modellierung im Software Engineering zu sehen [CHE04]. Neben den in diesem Abschnitt vorgestellten hierarchischen Beziehungen lassen sich zwischen Komponenten auch nichthierarchische Beziehungen identifizieren. Diese werden in Abschnitt 2.5 genauer vorgestellt.

2.3 Produktivitätsbetrachtung

Sollen Komponenten genauer beschrieben werden, ist dies mittels *Produktivitätskennzahlen* (Key Performance Indicators, KPIs) möglich. KPIs ermöglichen es, Dienstleistungen dahingehend zu untersuchen, inwiefern sie vorgegebene Geschäftsziele erfüllen. Ein großes Problem bei der Definition von KPIs zu Dienstleistungen ist die hohe Komplexität der Dienstleistungen. Dadurch ist oftmals nicht genau klar, welche relevanten Einflüsse auf die Produktivität von Dienstleistungen beachtet werden müssen [GO04]. Komponenten hingegen bilden nur eine gewisse Teilleistung mit klar umrissener Funktionalität ab; dies vereinfacht die Identifikation geeigneter Kennzahlen zur Analyse der Funktionalität.

Im Metamodell werden dementsprechend Kennzahlen zu Komponenten zugewiesen. Durch die Komposition von Komponenten werden Kennzahlen aggregiert und lassen damit von der Betrachtung der Produktivität einer einzelnen Komponente auf die Produktivität einer Gesamtleistung schließen. Durch die Heterogenität von Dienstleistungen existieren eine Reihe verschiedener Kennzahlen, mit denen die Produktivität gemessen werden kann. Um diese Vielzahl von Kennzahlen zu beherrschen, empfiehlt sich die Nutzung einer Bibliothek [FLKB11].

2.4 Externe Einflussfaktoren

Zur kundenindividuellen Konfiguration des Portfolios eines Dienstleisters ist es notwendig, externe Einflussfaktoren in die Betrachtung miteinzubeziehen. Diese Einflussfaktoren können mitunter vielfältige Auswirkungen auf die Erbringung einzelner Komponenten haben. So können gewisse Gegebenheiten beim Kunden dazu führen, dass einzelne Komponenten im Zuge der Konfiguration nicht mehr ausgewählt werden können, da sie nicht die benötigte Funktionalität implementieren.

Ein Beispiel eines externen Einflussfaktors für einen Call-Center-Anbieter ist die erwartete Anzahl eingehender Anrufe. Je nach Anzahl müssen verschiedene zusätzliche Komponenten gewählt werden, bzw. lässt sich auch feststellen, dass eine Dienstleistung nur mit hohem Risiko erbracht werden kann, weil nicht genügend Kapazitäten vorhanden sind.

Das Metamodell ermöglicht die Darstellung externer Einflussfaktoren durch die Verwendung von *Variablen*. Die Werte der Variablen dienen während der Modellierung als Platzhalter und werden erst zur Konfiguration gesetzt, da sie kundenabhängig sind und vom Anbieter einer Dienstleistung nicht vorhergesehen werden können. Mit Hilfe von Variablen lässt sich die Produktivitätsbetrachtung flexibilisieren, da keine statischen Werte verwendet werden müssen und Komponenten daher auch in verschiedenen Kontexten verwendet werden können.

2.5 Allgemeine Beziehungen zwischen Modellelementen

Neben hierarchischen Beziehungen zwischen Komponenten (siehe Abschnitt 2.2) müssen zur vollständigen Beschreibung von Dienstleistungen auch allgemeine Beziehungen zwischen konstitutiven Elementen des Metamodells beschrieben werden. Diese beschreiben Abhängigkeiten zwischen Elementen, die sich auf die Möglichkeiten der Konfiguration auswirken - sie können entweder mögliche Konfigurationsentscheidungen einschränken oder empfehlenden Charakter haben. In der Literatur wird unter anderem genannt, dass die Produktivität einer Anlage nicht alleine von ihrem Wartungszustand abhängt sondern auch vom Qualifikationsstand der bedienenden Mitarbeiter [UMB+08]. Daneben ist es auch möglich, dass eine Kombination verschiedener Elemente zu Änderungen an Eigenschaften (z.B. dem Wert) einer Komponente führt [Mon02].

Im Metamodell werden allgemeine Beziehungen mittels *logischer Abhängigkeiten* dargestellt. Diese Abhängigkeiten können zwischen allen konstituierenden Elementen des Metamodells auftreten, d.h. zwischen Komponenten, Produktivitätskennzahlen, externen Einflussfaktoren sowie nichtfunktionalen Eigenschaften. Dabei lassen sich verschiedene Typen von Abhängigkeiten definieren. *Vorschlagende Abhängigkeiten* beschreiben lose Zusammenhänge zwischen Elementen. Sie schränken die Konfigurationsmöglichkeiten nicht ein, da ihre Anwendung optional ist. *Einschränkende Abhängigkeiten* beschreiben Beziehungen, welche die Menge der möglichen Konfigurationen verringern. *Modifizierende Abhängigkeiten* beschreiben Änderungen von Eigenschaften der Modellelemente.

2.6 Prozesssicht auf Komponenten

Im Gegensatz zu Sachleistungen sind Dienstleistungen ausführbare Prozesse. Daher muss das Metamodell prozessuale Bestandteile enthalten, die beschreiben, in welcher Reihenfolge Komponenten ausgeführt werden können und ob es zeitliche Einschränkungen gibt. Im Beispiel der Wartungsdienstleistung ist eine Einschränkung der zeitlichen Anordnung von Komponenten, dass bei der Reparatur vor Ort zunächst ein Mitarbeiter des Kundendienstes anfahren muss bevor die eigentliche Reparatur beginnen kann. Die Beachtung zeitlicher Einschränkungen zur Modellierung von Dienstleistungen ist in der Literatur schon seit geraumer Zeit bekannt [Sho84].

Die Prozesssicht wird im Metamodell durch die Definition *zeitlicher Abhängigkeiten* zwischen Komponenten dargestellt. Existieren in einem modellierten Portfolio keine explizit definierten zeitlichen Abhängigkeiten lassen sich die konfigurierten Komponenten quasi-parallel ausführen. Dessen ungeachtet existieren in der Regel implizite zeitliche Abhängigkeiten, wenn beispielsweise zwei Komponenten von einer Person ausgeführt werden. Mit Hilfe der Prozesssicht ist es möglich, aufeinander aufbauende Komponenten zu beschreiben. Unter Beachtung der zeitlichen Abhängigkeiten ist es möglich, konfigurierte Dienstleistungsmodelle in eine Prozessdarstellung zu transformieren und damit innerhalb von Systemen zur Unterstützung der Prozessausführung weiterzuverwenden [BK12b].

2.7 Nichtfunktionale Eigenschaften

Die Definition nichtfunktionaler Eigenschaften ist notwendig, um die Möglichkeiten und Grenzen von Dienstleistungskomponenten zu modellieren. Verschiedene Komponenten, die ähnliche Funktionen bereitstellen, unterscheiden sich oftmals durch ihre Leistungsfähigkeit voneinander. Obwohl dies keine Einschränkung der Funktionalität einer Komponente ist, ist die Darstellung nichtfunktionaler Eigenschaften notwendig, um die passende Komponente für ein bestimmtes Einsatzgebiet zu ermitteln. Eine große Anzahl nichtfunktionaler Eigenschaften sind in [O'S08] dargelegt und umfassen u.a. zeitliche und örtliche Verfügbarkeit, verfügbare Bezahlmethoden sowie Rechte und Pflichten von Vertragspartnern.

Zur Beschreibung nichtfunktionaler Eigenschaften werden im Metamodell *Attribute* verwendet. Attribute sind Komponenten zugeordnet und beschreiben deren individuelle Leistungsfähigkeit. Die Attribute ermöglichen es, verschiedene Komponenten mit ähnlicher Funktionalität voneinander abzugrenzen.

2.8 Funktionale Eigenschaften

Die Definition der eigentlichen Funktionalität einer Komponente kann auf mehreren Wegen erfolgen. Zunächst ist es möglich, die *funktionalen Eigenschaften* durch die Beschreibung von Zustandsänderungen, die sich durch die Ausführung einer Komponente ergeben, zu definieren. Daneben kann dies durch verschiedene Klassifikationsansätze oder durch die Beschreibung der Änderungen an beteiligten Ressourcen erfolgen [BF09]. Im Rahmen des Metamodells erfolgt keine Einschränkung der Darstellungsmöglichkeiten der Funktionalität. In [BK12b] wurde mit der Überführung von Komponenten des Metamodells in eine Prozessdarstellung eine Möglichkeit der Beschreibung der Funktionalität näher untersucht.

3 Formales Metamodell

Nachdem im letzten Abschnitt die Bestandteile des Metamodells anhand von Konzepten zur Modellierung von Dienstleistungen beschrieben wurden, werden diese an dieser Stelle formal defi-

niert. Mit Hilfe dieser formalen Definition ist es möglich, eindeutige Beschreibungen von Portfolios zu entwickeln und aufbauend auf dieser Beschreibung kundenindividuelle Konfigurationen zu erstellen. Die Formalisierung ist weiterhin notwendige Voraussetzung einer softwaretechnischen Umsetzung. Durch die Integration des Konzepts der Modularisierung ist es nicht notwendigerweise erforderlich, Portfolios in allen Details zu spezifizieren. Stattdessen können zunächst grobe Portfoliobeschreibungen erstellt und im Laufe der Verwendung des Metamodells weiter detailliert werden.

Ein Portfolio ist ein 11-Tupel $P = (C, K, E, card, L, T, kpi, kpiV, att, attV, var)$:

- C ist eine endliche, nicht-leere Menge von *Komponenten*,
- K ist eine endliche, nicht-leere Menge von *Konnektoren*,
- $E \subseteq (C \times K) \cup (K \times C) \cup (K \times K)$ ist eine Menge von Kanten, die einen azyklischen *Konfigurationsgraphen* bilden und *hierarchische Abhängigkeiten* zwischen Komponenten darstellen,
- $card : K \rightarrow \mathcal{P}(\mathbb{N} \times \mathbb{N})$ ist eine Menge von *Kardinalitäten*, um Konnektoren mit weiterer Semantik anzureichern,
- L ist eine endliche Menge *logischer Abhängigkeiten*,
- T ist eine endliche Menge *zeitlicher Abhängigkeiten*,
- kpi ist eine endliche Menge von *Produktivitätskennzahlen*, $kpiV$ deren mögliche Werte,
- att ist eine endliche Menge von *Attributen*, $attV$ deren mögliche Werte,
- var ist eine endliche Menge von *Variablen*.

3.1 Hierarchische Beziehungen zwischen Komponenten

Der Konfigurationsgraph besteht aus zwei Knotentypen: Komponenten (C) und Konnektoren (K). In Abbildung 3 (Seite 13) ist ein Portfolio dargestellt, das 14 Komponenten und 6 Konnektoren enthält. Anhand der Definition des Portfolios ist ersichtlich, dass Komponenten nur mittels Konnektoren miteinander verbunden werden dürfen. Dies ist notwendig, um Eigenschaften der hierarchischen Verbindung zwischen Komponenten genauer zu definieren.

Die Semantik eines Konnektors wird durch die ihm zugewiesenen Kardinalitäten bestimmt. Im Allgemeinen lassen sich Kardinalitäten als Tupel natürlicher Zahlen (min, max) darstellen. Diese legen fest, wie viele nachfolgende Knoten bei der Konfiguration mindestens und höchstens aktiviert[1] sein dürfen. Um die Erfüllbarkeit von Konnektoren mit Kardinalitäten zu ermöglichen, müssen diese einigen Regeln gehorchen, die in Tabelle 2 aufgezeigt sind.

[1] Ein Knoten ist aktiviert, wenn mindestens einer seiner nachfolgenden Knoten aktiviert ist. Das Konzept der Aktivierung wird im Zuge der Beschreibung der Konfiguration (Abschnitt 4) genauer erläutert und formalisiert.

Tabelle 2: Regeln für Kardinalitäten

Name	Formalisierung
MINMAX	$\forall k \in K, \forall (m,n) \in card(k) : m \leq n$
POSTNODES	$\forall k \in K, \forall (m,n) \in card(k) : n \leq \lvert postnodes(k) \rvert$
OVERLAPS	$\forall k \in K, \forall (m,n) \in card(k) :$
	$\neg \exists (m^\star, n^\star) \in card(k) : m \leq m^\star \leq n \vee m \leq n^\star \leq n$

Tabelle 3: Vordefinierte Konnektortypen

Name	Formalisierung
K_{ALL}	$\forall k \in K_{ALL} : card(k) = \{(\lvert postnodes(k) \rvert, \lvert postnodes(k) \rvert)\}$
K_{ONE}	$\forall k \in K_{ONE} : card(k) = \{(1,1)\}$
K_{ANY}	$\forall k \in K_{ANY} : card(k) = \{(0, \lvert postnodes(k) \rvert)\}$
K^n_{MIN}	$\forall k \in K^n_{MIN} : card(k) = \{(n, \lvert postnodes(k) \rvert)\}$
K^n_{MAX}	$\forall k \in K^n_{MAX} : card(k) = \{(0,n)\})$

Um die Regeln aufzustellen ist die Definition der Abbildung $postnodes$ notwendig. Diese ermittelt für ein Element des Konfigurationsgraphen die Menge der nachfolgenden Knoten:

$$postnodes(v) = \{v^\star | (v, v^\star) \in E\}$$

Die Regel MINMAX legt fest, dass in allen Kardinalitäten die Mindestzahl erforderlicher nachfolgender Knoten höchstens so groß sein darf wie deren Maximalzahl. Um festzulegen, dass eine Kardinalität nicht mehr Knoten erfordern darf, als es nachfolgende Knoten gibt, wird die Regel POSTNODES verwendet. Schließlich wird mit der Regel OVERLAPS festgelegt, dass es keine Überlappungen zwischen den Kardinalitäten eines Konnektors geben darf.

Um den Umgang mit dem Modell zu vereinfachen, existieren vordefinierte Konnektortypen, die auf verschiedenen Kardinalitäten basieren (siehe Tabelle 3). Mit Hilfe dieser Konnektoren lassen sich bereits eine Reihe unterschiedlicher Gegebenheit spezifizieren. Mit Hilfe des K_{ALL}-Konnektors lässt sich festlegen, dass bei der Konfiguration alle nachfolgenden Knoten aktiviert sein müssen. Der K_{ONE}-Konnektor beschreibt eine Situation, in der genau ein nachfolgender Knoten aktiviert sein muss. K_{ANY}-Konnektoren definieren keine weiteren Restriktionen auf die Anzahl aktivierter nachfolgender Knoten. Schließlich lässt sich mit den Konnektoren K^n_{MIN} und K^n_{MAX} festlegen, dass mindestens n nachfolgende Knoten aktiviert sein müssen bzw. dass höchstens n nachfolgende Knoten aktiviert sein dürfen. Um die Erfüllbarkeit der beiden letzten Konnektorentypen zu gewährleisten, dürfen die entsprechenden Minimal- und Maximalwerte die Anzahl nachfolgender Knoten nicht überschreiten.

3.2 Produktivitätskennzahlen

Die Repräsentation von Kennzahlen erfolgt im Metamodell durch die Mengen kpi und $kpiV$, die Namen und Werte von Kennzahlen enthalten. Da sich der Wert einer Kennzahl auch aus anderen Kennzahlen berechnen lässt, enthält die Menge $kpiV$ neben statischen Werten auch Formeln zur

Berechnung von Kennzahlen. Die Zuweisung von Kennzahlen zu einzelnen Komponenten erfolgt mittels der Abbildung $KPIValue$:

$$KPIValue : C \times kpi \rightarrow kpiV$$

Durch Aggregation von Kennzahlen verschiedener Komponenten wird eine Aussage über die Produktivität einer Gesamtleistung ermöglicht. Dementsprechend können KPIs einer Komponente auf verschiedene Arten berechnet werden. Im einfachsten Fall wird der KPI ein einzelner Wert zugewiesen, z.B. $time = 5$. Eine Unterkomponente mit einer KPI gibt diese KPI an ihre übergeordnete Komponente weiter (*Descendent Propagation*). Werden Kennzahlen aus Unterkomponenten in einer Oberkomponente verändert bzw. zur Berechnung neuer KPIs genutzt, spricht man von *Descendent Calculation*. Damit kann zum Beispiel ausgedrückt werden, dass der Zeitaufwand für eine zusammengesetzte Leistung die Summe der Zeitaufwände der einzelnen Teilleistungen ist. Dies ist möglich mit der Formel $time = SUM(\$.time)$.

Um nicht-hierarchische Auswirkungen auf KPIs zu definieren, können modifizierende logische Abhängigkeiten (siehe Abschnitt 3.4) verwendet werden. In diesem Fall spricht man von *Intra-Tree Calculation*. Dies ist dann von Relevanz, wenn die Auswahl einer Komponente Auswirkungen auf KPIs einer anderen Komponente hat. So können z.B. durch die Auswahl einer komplexen Dienstleistungskomponente, die viel Unterstützung durch den Anbieter benötigt, die Kosten für den Support steigen. Die möglichen Formen der KPI-Zuweisung und Aggregation sind in Abbildung 2 dargestellt.

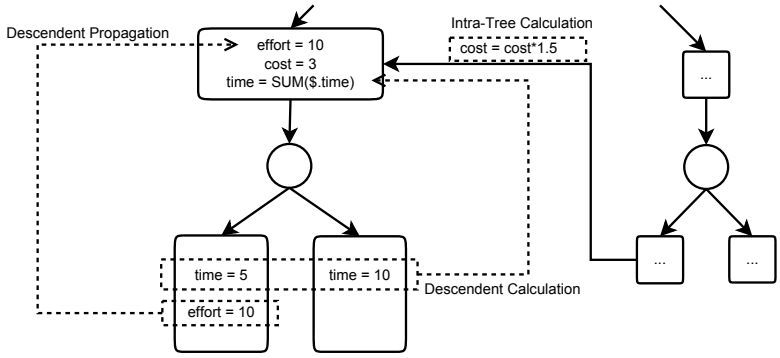

Abbildung 2: Zuweisung und Aggregation von KPIs im Konfigurationsgraphen

3.3 Variablen

Variablen zur Darstellung externer Einflussfaktoren werden im Metamodell hinsichtlich ihres Datentyps nicht eingeschränkt. Dadurch ist es möglich, flexible Abhängigkeiten zu definieren. Die Menge var enthält alle Variablen. Die Werte von Variablen ($varV$) werden erst zur Konfiguration mit Hilfe der Abbildung $VarValue$ festgelegt:

$$VarValue : var \times varV$$

Tabelle 4: Vordefinierte logische Abhängigkeiten

	Vorschlagende Abhängigkeiten
Alternative	Alternativen sind nur für Komponenten definiert. Zwei Komponenten sind Alternativen voneinander, wenn sie die gleiche Funktionalität auf unterschiedliche Art und Weise erfüllen, z.B. *In-House-Entwicklung* und *Auslagerung an externen Anbieter*.
	Einschränkende Abhängigkeiten
Voraussetzung	Voraussetzungen schränken die Möglichkeiten der Konfiguration ein, indem sie Abhängigkeiten definieren, die erfüllt sein müssen, wenn Elemente in einer Konfiguration aktiv sind. So ist denkbar, dass die Dienstleistung *Effizienzvergleich* mit anderen Anlagen nur erbracht werden kann, wenn die installierte Basis größer als 100 ist.
Ausschluss	Ein Ausschluss definiert, dass verschiedene Elemente nicht gleichzeitig in einer gültigen Konfiguration aktiv sein dürfen. In einem Call Center mit verschiedenen Angeboten kann eine Komponente nicht aktivierbar sein, wenn die Anzahl erwarteter eingehender Anrufe einen Schwellwert übersteigt.
	Modifizierende Abhängigkeiten
Wertänderung	Eine Wertänderung führt dazu, dass in einer Konfiguration, in der gewisse Elemente aktiv sind, Werte von Kennzahlen geändert werden. Dies kann verwendet werden, um Rabatte oder ähnliches darzustellen, z.B. kann bei der Konfiguration der Wartungsdienstleistung aus Abbildung 3 die Auswahl der beiden Komponenten Sichtprüfung und Fernwartung dazu führen, dass es einen Rabatt von 10% auf die Gesamtkosten der geplanten Wartung gibt.

3.4 Logische Abhängigkeiten

Um eine größtmögliche Flexibilität bei der Definition logischer Abhängigkeiten zwischen Modellelementen zu gewährleisten, lassen diese sich mit Hilfe formaler Logik definieren. Da die Anwendung formaler Logik aufgrund ihrer Komplexität für Nicht-Fachleute ungeeignet ist, gibt es eine Reihe vordefinierter Abhängigkeiten, die der Literatur entnommen sind und häufig in Dienstleistungen auftreten. Eine Auswahl dieser Abhängigkeiten ist in Tabelle 4 dargestellt. Weitere Abhängigkeiten sowie deren Formalisierungen finden sich in [BK12a].

Damit logische Abhängigkeiten bei der Konfiguration eingesetzt werden können, muss ihre Semantik formal definiert werden. Während der Konfiguration wird zunächst überprüft, ob es vorschlagende Abhängigkeiten gibt, die angewendet werden können. Wird eine der möglichen vorschlagenden Abhängigkeiten angewendet, ändert sich die Konfiguration, weil z.B. zusätzliche Komponenten aktiviert oder vormals aktivierte Komponenten deaktiviert werden. Da die Anwendung der vorschlagenden Abhängigkeiten optional ist, müssen diese manuell geprüft werden. Nach dieser Prüfung werden wertändernde Abhängigkeiten angewendet, wodurch es möglich ist, dass sich die Konfiguration erneut ändert. Um die Gültigkeit der Gesamtkonfiguration zu prüfen, wird anschließend die Einhaltung einschränkender Abhängigkeiten geprüft. Falls eine einschränkende Abhängigkeit nicht erfüllt ist, muss die Konfiguration angepasst werden.

Die Möglichkeit zur Definition logischer Abhängigkeiten ermöglicht es, vielfältige Beziehungen

Tabelle 5: Vordefinierte zeitliche Abhängigkeiten

Bezeichnung	Beschreibung
Vorgänger	Eine Abhängigkeit $before(A) = B$ legt fest, dass in allen Konfigurationen, die sowohl Komponente A als auch Komponente B enthalten, mindestens einmal die Komponente B ausgeführt werden muss, bevor Komponente A ausgeführt werden kann.
Direkter Vorgänger	Die Abhängigkeit $iBefore(A) = B$ legt fest, dass direkt bevor Komponente A ausgeführt wird, Komponente B ausgeführt werden muss. Das heißt, zwischen der Ausführung beider Komponenten darf keine andere Komponente ausgeführt werden.

zwischen Elementen einer Dienstleistung darzustellen. Auch wenn die Verwendung vordefinierter Beziehungen für Modellierer nur geringen Aufwand bedeutet, sollte dies nicht im Übermaß eingesetzt werden. Die Beziehungen verringern sowohl die Lesbarkeit als auch die Verständlichkeit von Dienstleistungsmodellen und erhöhen die Komplexität bei der Konfiguration; in [TBK09] wurde dieses Phänomen für Featuremodelle gezeigt. Da Komponenten als in sich abgeschlossene Einheiten angesehen werden, sollte das Verhältnis zwischen Kohäsion einer Komponente und Kopplung mit anderen Komponenten nicht außer Acht gelassen werden [AB11].

3.5 Zeitliche Abhängigkeiten

In Analogie zu nichthierarchischen logischen Abhängigkeiten, die mit Hilfe der Aussagen- und Prädikatenlogik definiert werden, lassen sich die zeitlichen Beziehungen mit Hilfe der Linearen Temporalen Logik (LTL) darstellen. Dieser Ansatz wird unter anderem auch in [AP06] zur verständlichen Modellierung von Prozessen verfolgt. Allerdings besitzt die LTL im Vergleich zur Prädikatenlogik eine höhere Komplexität, weshalb auch hier der Ansatz verfolgt wird, oftmals genutzte Beziehungen vorzudefinieren. Eine Auswahl dieser Abhängigkeiten findet sich in Tabelle 5; ihre Formalisierung ist in [BK12b] beschrieben.

Analog zu logischen Abhängigkeiten wird die formale Semantik der zeitlichen Abhängigkeiten nicht direkt bei der Modellierung ausgewertet. Stattdessen erfolgt dies erst im Zuge der Transformation eines konfigurierten Dienstleistungsmodells in ein Prozessmodell, welches dann an eine Ausführungsumgebung weitergegeben werden kann. Wird in eine prozedurale Prozessdarstellung transformiert, müssen dabei die zeitlichen Abhängigkeiten direkt bei der Generierung des Prozessmodells beachtet werden. Wird hingegen eine deklarative Sprache zur Prozessbeschreibung (z.B. [AP06]) verwendet, ist es erforderlich, die entsprechenden LTL-Konstrukte anzugeben.

3.6 Attribute

Attribute zur Darstellung nichtfunktionaler Eigenschaften werden im Metamodell durch die Mengen att und $attV$ dargestellt. Diese enthalten die eigentlichen Attribute sowie deren Werte. Im

Gegensatz zu Kennzahlen, deren Wertmenge auf reelle Zahlen beschränkt ist, besitzen Attribute einen beliebigen Datentyp. Die Zuweisung von Attributen zu einzelnen Komponenten erfolgt mittels der Abbildung $AttValue$:

$$AttValue : C \times att \rightarrow attV$$

Da der Wertebereich von Attributen unbeschränkt ist, lassen sich eine Reihe nichtfunktionaler Eigenschaften definieren, so wie es auch in [O'S08] gefordert ist. Im Portfolio eines Call-Center-Dienstleisters ist eine wichtige nichtfunktionale Eigenschaft die maximale Anzahl eingehender Anrufe pro Tag (Kapazität). Um beispielsweise zu beschreiben, dass die Komponente C_1 eine Kapazität von 5000 Anrufen pro Tag hat, wird folgende Zuweisung verwendet:

$$AttValue(C_1, capacity) = 5000$$

Durch die Kombination von Attributen und logischen Abhängigkeiten lassen sich Anforderungen an die Funktionalität einer konfigurierten Dienstleistung definieren. Im Call-Center-Beispiel dürfen nur Komponenten aktiviert werden, deren Kapazität größer als die erwartete Anzahl eingehender Anrufe ist. Im Folgenden repräsentiert die Variable $incomingCalls$ die erwartete Anzahl an Anrufen und das Attribut $capacity$ der Komponente $CallCenter$ stellt die Kapazität des Call Centers dar.[2]

$$requires((incomingCalls, >, X)) = (capacity, CallCenter, > X)$$

3.7 Graphische Darstellung

Um die Verständlichkeit und Nutzbarkeit des Metamodells zu erhöhen, wurde eine graphische Notation entwickelt, mit der die formalen Bestandteile dargestellt werden können. Dadurch lassen sich Modelle auch ohne Kenntnisse formaler Logik erstellen. Ein beispielhaftes Portfolio eines Wartungsdienstleisters ist in Abbildung 3 skizziert. Dabei sind Komponenten als abgerundete Rechtecke und Konnektoren in Kreisform dargestellt.

Das Portfolio enthält die Wurzelkomponente *Unternehmensportfolio*, welche die verschiedenen anderen Komponenten kapselt. Mit Hilfe des vorgestellten Metamodells ist es möglich, das gesamte Portfolio in einem Konfigurationsgraphen darzustellen. Dies ermöglicht es, Beziehungen zwischen Dienstleistungen zu modellieren, die sonst in keiner direkten Relation zueinander stehen. Unter anderem müssen Teilleistungen nur einmal modelliert werden und lassen sich dann im gesamten Modell verwenden. Direkte Unterkomponenten des Unternehmensportfolios sind die Komponenten *Wartung*, *Neuinstallation* und *Inbetriebnahme*; diese repräsentieren die jeweiligen Dienstleistungen. Die drei Dienstleistungen sind mit der Portfolio-Komponente mittels eines K^1_{MIN}-Konnektors verbunden, d.h. mindestens einer der nachfolgenden Knoten muss für eine gültige Konfiguration aktiviert sein. Aus Gründen der Übersichtlichkeit wird an dieser Stelle nur

[2] Im Beispiel werden Vergleichsoperatoren für Variablen und für Attribute verwendet. Die genaue Semantik dieser Operatoren ist in [BK12a] beschrieben. An dieser Stelle soll hinreichend sein, X als ein Element der natürlichen Zahlen zu definieren.

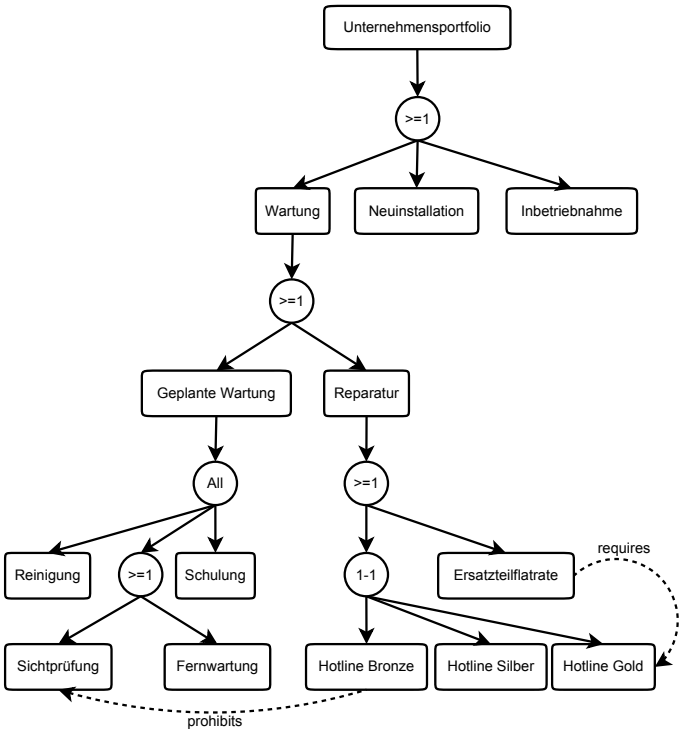

Abbildung 3: Portfolio eines Wartungsdienstleisters

die Komponente Wartung weiter detailliert. Die beiden Komponenten Neuinstallation und Inbetriebnahme sind hier nur angedeutet. Diese Möglichkeit der Komposition und Dekomposition wurde oben bereits angesprochen und hilft hier, die Übersichtlichkeit des Modells zu erhöhen.

Die Komponente Wartung lässt sich weiter in die Bestandteile *Geplante Wartung* und *Reparatur* detaillieren. Anhand des verbindenden Konnektors ist ersichtlich, dass mindestens eine dieser Komponenten aktiviert sein muss. Wählen Kunden die geplante Wartung wird in einem periodischen Abstand eine Wartung durchgeführt. Diese Wartung besteht darin, dass eine Anlage gereinigt wird (Komponente *Reinigung*) und dass Mitarbeiter im Umgang mit der Anlage angeführt werden (*Schulung*). Die Prüfung der Anlage erfolgt dann entweder per Sichtprüfung, per Fernwartung oder per Kombination dieser beiden Möglichkeiten. Durch den K_{ALL}-Konnektor müssen alle nachfolgenden Knoten der geplanten Wartung aktiviert sein, d.h. sowohl die Reinigung als auch die Schulung sind bei diesem Angebot obligatorisch vorhanden. Für die Prüfung muss mindestens eine der Möglichkeiten ausgewählt werden.

Ist die Komponente Reparatur ausgewählt, können Kunden entscheiden, ob sie einen Vertrag über eine *Ersatzteilflatrate* abschließen. Dabei werden bei notwendigen Reparaturmaßnahmen automatisch Ersatzteile erworben, ohne dass diese einzeln abgerechnet werden. Stattdessen wird ein jährlicher Pauschalbetrag gezahlt. Daneben gibt es eine Hotline des Anbieters, die entweder in der Form *Hotline Bronze*, *Hotline Silber* oder *Hotline Gold* gewählt werden kann. Die ver-

schiedenen Hotline-Varianten unterscheiden sich hinsichtlich der Verfügbarkeit der Mitarbeiter sowie der Kosten eines Anrufes. Durch den K_{ONE}-Konnektor, mit dem die Hotline-Angebote verbunden sind, kann nur genau eine der verschiedenen Möglichkeiten ausgewählt werden, um eine gültige Konfiguration zu erzeugen.

Neben den hierarchischen Abhängigkeiten zwischen Komponenten gibt es weiterhin logische Abhängigkeiten. Wählen Kunden während der Konfiguration die Ersatzteilflatrate aus, müssen sie auch die Hotline Gold auswählen. Dadurch ist eine kontinuierliche Verfügbarkeit zur Bereitstellung der Ersatzteile garantiert. Eine weitere Abhängigkeit besteht darin, dass bei der Auswahl der Komponente Hotline Bronze die Sichtprüfung nicht gewählt werden darf. Hotline Bronze deckt im Beispiel nur einen geringen Teil der Wartung ab, zu dem die Sichtprüfung nicht gehört.

Zur Vereinfachung sind im Beispiel nur logische Abhängigkeiten zwischen Komponenten dargestellt. Prinzipiell sind weitere Abhängigkeiten möglich, aber nicht modelliert. Als Beispiel lässt sich definieren, dass die Ersatzteilflatrate nur besonderen Kunden angeboten wird, die mehr als zehn Anlagen des Betreibers im Einsatz haben. Dies lässt sich als Abhängigkeit an eine externe Variable darstellen.

4 Konfiguration

Die Erstellung von Modellen von Dienstleistungsportfolios, welche zu dem Metamodell konform sind, ist die Basis für die Durchführung kundenindividueller Konfigurationen. Ziel der modellbasierten Konfiguration ist dabei die Bewertung einer auf Kundenanforderungen basierenden Dienstleistung hinsichtlich Validität sowie Produktivität.

4.1 Aufstellen einer Konfiguration

Zur Konfiguration ist es notwendig, dass Knoten im Konfigurationsgraphen *aktiviert* werden. Die Aktivierung eines Knotens kann auf mehreren Wegen erfolgen. Zunächst werden Komponenten aktiviert, indem sie manuell selektiert, d.h. während der Konfiguration ausgewählt werden.

$$\forall c \in C : c \in selected \leftrightarrow c \in activated$$

Ein Knoten wird automatisch aktiviert, wenn mindestens einer seiner nachfolgenden direkten Kindknoten aktiviert ist.

$$\forall n \in C \cup K : n \in activated \leftrightarrow \exists n^{\star} \in postnodes(n) : n^{\star} \in activated$$

Nachfolgende Konnektoren einer Komponente werden aktiviert, wenn die übergeordnete Komponente aktiviert ist.

$$\forall c \in C \forall k \in postnodes(c) : c \in activated \rightarrow k \in activated$$

Komponenten sind die einzigen Knotentypen, die manuell aktiviert werden können, indem sie während der Konfiguration gewählt werden. Durch Auswahl einer Komponente werden die mit

dieser Komponente verbundenen Konnektoren entsprechend der oben genannten Regel automatisch aktiviert. Eine Konfiguration enthält dann alle Komponenten, die aktiviert sind; sei es, weil sie manuelle ausgewählt wurden oder weil sie durch nachfolgende Knoten aktiviert sind.

$$\forall c \in C : c \in Configuration \leftrightarrow c \in activated$$

Zur Bewertung der Produktivität werden die den Komponenten zugeordneten Produktivitätskennzahlen sowie Variable herangezogen. Variable sind im Zuge der Konfiguration mit Hilfe der Abbildung $varValue$ mit einem Zahlenwert zu versehen, während sich die Werte von Produktivitätskennzahlen aus den ausgewählten Komponenten bestimmen.

Aufgrund der generischen Konzeption von Produktivitätskennzahlen im Metamodell ist auch die Nutzung von KPIs über die reine Produktivitätsbetrachtung hinaus möglich. Beispielsweise können Kennzahlen zur Bewertung des wirtschaftlichen Risikos einzelner Konfigurationen verwendet werden.

4.2 Validierung einer Konfiguration

Die Komposition von Komponenten in kundenindividuellen Konfigurationen erfolgt nicht nur anhand von Kundenanforderungen und Produktivitätsgesichtspunkten. Die im Modell definierten hierarchischen sowie logischen Abhängigkeiten sind ebenso zu berücksichtigen. Dabei wirken die verschiedenen Regeltypen auf unterschiedliche Art und Weise, was ebenso in die Validierung der Konfiguration einzubeziehen ist. Die chronologische Vorgehensweise wäre demnach die Präsentation vorschlagender Regeln, die Anwendung modifizierender Regeln sowie die Validierung einschränkender Regeln. In Verbindung mit den hierarchischen Abhängigkeiten lässt sich somit die Validität kundenindividueller Konfigurationen des Modells bestimmen.

Eine Konfiguration ist nur dann valide, wenn folgende Bedingungen erfüllt sind:

- Die Konnektoren der Konfiguration sind valide, d.h. es wurde eine entsprechende Menge an nachfolgenden Knoten gewählt, so dass eine der Kardinalitäten erfüllt ist.
- Die einschränkenden logischen Abhängigkeiten sind erfüllt, d.h. die Konfiguration darf keine Komponenten enthalten, die dazu führen, dass logische Abhängigkeiten verletzt werden.

4.3 Konfiguration am Beispiel

Zur kundenindividuellen Konfiguration einer Dienstleistung lassen sich zwei unterschiedliche Ansätze verfolgen. Zunächst ist es möglich, die Konfiguration *Top-Down* vorzunehmen. Dazu wird beim Wartungsbeispiel in Abbildung 3 zunächst die Komponente *Unternehmensportfolio* ausgewählt. Basierend auf der Definition der Aktivierung wird der nachfolgende Konnektor ausgewählt. Dieser legt fest, dass mindestens ein nachfolgender Knoten aktiviert sein muss, d.h. mindestens eine der Komponenten *Wartung*, *Neuinstallation* oder *Inbetriebnahme* muss gewählt

werden, damit der Konnektor valide ist. Bei der Auswahl der Wartung wird analog mit den nachfolgenden Knoten dieser Komponente verfahren. Für eine gültige Konfiguration müssen dementsprechend die *Geplante Wartung*, die *Reparatur* oder beide ausgewählt werden. Eine gültige Konfiguration entsteht, wenn alle Konnektoren valide sind und damit die Komponenten bis zu den Blättern hin ausgewählt wurden. Im Rahmen der Top-Down-Konfiguration werden also allgemeine, grobgranulare Komponenten immer weiter verfeinert, um eine Konfiguration zu erzeugen.

Bei der *Bottom-Up*-Konfiguration werden zunächst spezifische Komponenten ausgewählt und dann zu einer Gesamtleistung zusammengefasst. Dadurch wird z.B. zunächst die Komponente *Hotline Silber* gewählt. Durch den K_{ONE}-Konnektor ist es im Rahmen einer validen Konfiguration nicht möglich, eine weitere Hotline-Variante auszuwählen. Durch die Auswahl einer der Hotline-Komponenten werden automatisch alle übergeordneten Knoten aktiviert, d.h. die Komponenten *Reparatur*, *Wartung* und *Unternehmensportfolio* sowie die verbindenden Konnektoren sind ebenfalls aktiviert.

Der direkt der Komponente *Reparatur* nachfolgend Konnektor erlaubt auch die Auswahl der Komponente *Ersatzteilflatrate*. Wird diese ausgewählt, wird die logische Abhängigkeit ausgewertet. Diese besagt, dass bei der Auswahl der *Ersatzteilflatrate* die *Hotline Gold* ebenfalls aktiviert sein muss; dies wird automatisch durchgeführt. Da der Konnektor der Hotline-Varianten nur einen einzigen aktivierten nachfolgenden Knoten erlaubt, ist die zuvor ausgewählte Komponente *Hotline Silber* zu deaktivieren. Dadurch ist die Konfiguration wieder valide.

5 Softwaretechnische Umsetzung

Zur Evaluation, Visualisierung und Weiterentwicklung des oben vorgestellten Metamodells wurden die Konzepte in einem Software-Werkzeug umgesetzt. Da das Metamodell lediglich den formalen Rahmen der Modellierung definiert, ist zur praktischen Nutzung der Ideen eine Software notwendig. Das entsprechend erstellte, prototypische Tool "Service Modeller" setzt einen Großteil der vorgestellten Konzepte um und erlaubt einen ersten Eindruck, wie die Modellierung und Konfiguration von Dienstleistungen aussehen kann. Abbildung 4 zeigt die Umsetzung des Portfolios aus Abbildung 3 im Service Modeller.

Die Vorgehensweise zur Erstellung kundenindividueller Dienstleistungen mit dem Service Modeller gliedert sich in zwei wesentliche Schritte, welche durch Sichten (Views) in der Software umgesetzt wurden. Die Editor-View dient dabei der Erstellung des Dienstleistungsmodells, d.h. Komponenten und Konnektoren werden in einem Baum angeordnet. Des Weiteren werden Eigenschaften von Komponenten wie KPIs, Beschreibungen sowie Attribute und die Kardinalitäten von Konnektoren definiert. Eine Fehlerprüfung gibt dabei im Falle nicht erfüllbarer Kardinalitäten (beispielsweise Kardinalität K^4_{MIN}, der Konnektor hat aber nur drei Kindkomponenten) entsprechende Warnungen aus.

Ist ein Modell vollständig im Editor erstellt worden, so kann dieses in der Konfigurator-View konfiguriert werden. Dabei sind gewünschte Komponenten zu wählen und Variablen Werte zuzuordnen. Zur Vereinfachung des Konfigurationsprozesses können durch die Vergabe von Attributen Komponenten gefiltert werden. So werden Komponenten, welche bestimmte Eigenschaften nicht erfüllen, ausgeblendet. Dies vereinfacht insbesondere bei komplexen Modellen die Konfigurati-

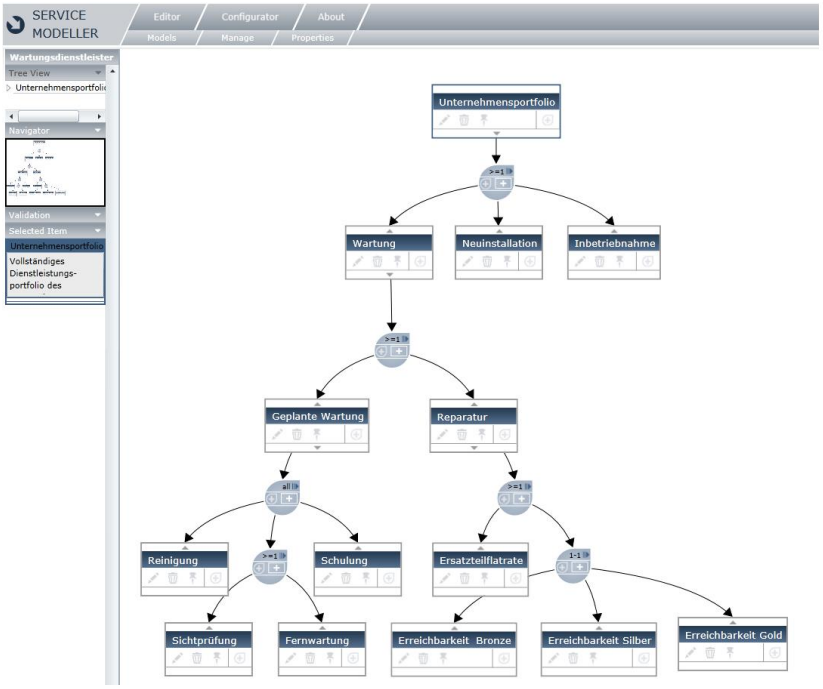

Abbildung 4: Service Modeller

on. Entsprechend der gewählten Komponenten werden KPIs berechnet, wodurch sich auch die Wirtschaftlichkeit von Konfigurationen bewerten lässt.

Da der Service Modeller als Prototyp angelegt wurde, sind noch nicht alle Konzepte des Metamodells umgesetzt. Die Software unterliegt ebenso wie das Metamodell einer permanenten Weiterentwicklung, welche sich aus Forschungsergebnissen und neuen Anforderungen und Vorschlägen der Praxispartner speist. Aktuell lassen sich mit dem Service Modeller Komponenten (mit KPIs und Attributen) sowie durch Konnektoren deren hierarchische und durch logische Abhängigkeiten deren nicht-hierarchische Beziehungen untereinander modellieren. Darauf aufbauend können verschiedene kundenindividuelle Konfigurationen erstellt und hinsichtlich ihrer Produktivität miteinander verglichen werden.

6 Phasenbasierte Unterstützung der Dienstleistungsentwicklung

Durch Nutzung des auf dem Metamodell basierenden Service Modellers werden verschiedene Phasen der Dienstleistungsentwicklung unterstützt. Tabelle 6 zeigt die Phasen der Dienstleistungsentwicklung nach [Mey08] sowie deren Unterstützung durch das Modell. Dabei lässt sich zwischen expliziter und impliziter Unterstützung unterscheiden. Im ersten Fall führt die Nutzung des Metamodells direkt zu einer Erhöhung der Effizienz und Effektivität der durchzuführenden

Tabelle 6: Phasen der Dienstleistungsentwicklung nach [Mey08] und Metamodell-Unterstützung

Phase	Unterstützung	Beschreibung
Ideenfindung	Implizit	Inspiration, Wiederverwendung
Definition	Explizit	Spezifikation von Komponenten
Anforderungsanalyse	Implizit	Nichtfunktionale Eigenschaften
Konzeption	Explizit	Spezifikation, Wiederverwendung
Realisierung	Explizit	Konfiguration von Dienstleistungen
Test	Implizit	Abhängigkeiten, KPIs
Markteinführung		
Betrieb	Keine	
Rückzug		

Aktivitäten der jeweiligen Phase. Durch die implizite Unterstützung hingegen können Aktivitäten verschiedener Phasen vereinfacht werden.

Bei der *Ideenfindung* werden Anregungen von Kunden, Wettbewerbern und aus dem eigenen Unternehmen gesammelt und zu konkreten Ideen weiterentwickelt. Das Metamodell kann hier durch eine strukturierte Darstellung existierender Dienstleistungskomponenten unterstützend beitragen. Diese lassen sich einerseits als Inspirationsquelle zur Entwicklung neuer Dienstleistungen nutzen, andererseits können neue Dienstleistungen mit Hilfe existierender Komponenten umgesetzt werden. Die Bewertung von Ideen sowie Methoden der Kreativitätssteigerung sind allerdings nicht Bestandteil des Metamodells.

Soll eine Idee als Dienstleistung umgesetzt werden, schließt sich die *Definitionsphase* an. Dabei werden konkrete Konzepte zur Umsetzung einer Idee in Dienstleistungsform entwickelt und getestet. Weiterhin werden grundlegende Schritte zur Erbringung einer Dienstleistung definiert. Durch die mit dem Metamodell fokussierte Modularisierung von Dienstleistungen lassen sich schon während der Definition einzelne, grobgranulare Komponenten beschreiben. Diese Komponenten können dem existierenden Portfolio bereits hinzugefügt werden. Durch die Nutzung von Kennzahlen lassen sich bereits erste Rückschlüsse auf die Produktivität verschiedener Varianten einer Dienstleistung ziehen, wodurch Varianten, die sich nicht mit vertretbarem Aufwand realisieren lassen, bereits frühzeitig erkannt und ausgeschlossen werden können.

Im Rahmen der *Anforderungsanalyse* werden die Erwartungen der Kunden mit den Zielsetzungen, Kernelementen und Rahmenbedingungen einer geplanten Dienstleistung abgeglichen. Darauf aufbauend wird eine genauere Spezifikation der Dienstleistung erstellt. Das Metamodell unterstützt die Analyse der Anforderungen, indem Kunden bereits frühzeitig erste Entwürfe einer geplanten Dienstleistung präsentiert werden. Daneben lassen sich durch nichtfunktionale Eigenschaften einzelner Komponenten Rahmenbedingungen der Dienstleistungserbringung extrahieren.

In der *Konzeptionsphase* wird eine Reihe von Aktivitäten durchgeführt, die Auswirkungen auf die Qualität der Dienstleistung haben. Die bisherigen Ideen werden verfeinert und die einzelnen Dienstleistungskomponenten genauer spezifiziert. Dabei werden die Potential-, Prozess- und Ergebnisdimensionen der Dienstleistung gestaltet. Durch die Unterstützung der Spezifikation und die Möglichkeit der Wiederverwendung von Komponenten gibt das Metamodell hierbei direkte

Unterstützung. Einerseits können Dienstleistungen detailliert in Teilleistungen zerlegt und damit strukturiert werden. Andererseits erlaubt die Nutzung von Kennzahlen die Durchführung einer Wirtschaftlichkeitsanalyse. Durch die Möglichkeit der Definition von Abhängigkeiten (seien es zeitliche oder logische) zwischen Komponenten wird weiterhin eine sehr feingranulare Spezifikation ermöglicht.

In die Konzeptionsphase fällt auch die Aufstellung von Teams, welche die Dienstleistung erbringen. Hier kann der Service-Modeller insofern helfen, als dass durch die Komponentisierung visualisiert wird, welche Kompetenzen zur Erbringung einzelner Teilleistungen notwendig sind und so die Zuordnung geeigneter Mitarbeiter zu den jeweiligen Dienstleistungskomponenten vereinfacht wird.

Nachdem Dienstleistungen konzipiert wurden, werden sie in der *Realisierungsphase* implementiert. Dabei wird die Dienstleistung für die Erbringung vorbereitet und mit entsprechenden Attributen versehen. Basierend auf der Ausgestaltung einzelner Komponenten der Dienstleistung wird die Dokumentation erstellt. Durch die Möglichkeit der Konfiguration wird die Realisierung explizit unterstützt. Zum einen kann mit wenig Aufwand eine an die Bedürfnisse bestimmter Kunden angepasste Dienstleistung erstellt werden. Zum anderen lässt sich mit Hilfe des Service Modellers auch der Aufwand zur Erstellung der Dokumentation verringern; teilweise kann diese anhand der vormodellierten Komponenten automatisch generiert werden. Durch die Transformation einer konfigurierten Dienstleistung in entsprechende Prozesse und die Anbindung eines Workflowmanagementsystems lässt sich eine weitere Automatisierung erreichen, indem fertig konfigurierte Dienstleistungen übergeben werden.

Um zu ermitteln, ob Dienstleistungen tatsächlich in der geforderten Qualität und vor allem fehlerfrei erbracht werden können, ist es notwendig diese zu *testen*. Hierbei ergibt sich allerdings eine Schwierigkeit, da Dienstleistungen per se immateriell sind und im gleichen Moment erbracht und konsumiert werden (Uno-actu-Prinzip [CG07]). Dadurch wird eine Qualitätsbewertung erschwert und Qualitätsmängel werden erst zu spät (bei der Erbringung) aufgedeckt. Der Service Modeller kann das Testen von Dienstleistungen durch die Modellierung der Komponenten zumindest teilweise ermöglichen. Die Abhängigkeiten zwischen einzelnen Elementen einer Dienstleistung lassen sich formalisieren, wodurch sich Mitarbeiter, die mit der Erbringung beauftragt sind, an diesem Schema orientieren können. Bei der Konfiguration helfen zudem die Kardinalitäten an den Konnektoren dabei, keine ungültigen Instanzen zu bilden.

Bisher ist noch keine vollständige Evaluierung vorgenommen worden, ob der Einsatz des Metamodells tatsächlich zu einem effizienteren Service Engineering führt. Trotzdem lässt sich anhand der vorliegenden Beschreibung der Zielsetzung annehmen, dass vor allem die frühen Service-Engineering-Phasen von der Nutzung des Metamodells und dem Einsatz des Service Modellers profitieren. Im Gegensatz dazu bieten beide Ergebnisse allerdings keine direkte Unterstützung für die Markteinführung, den Betrieb und den Rückzug von Dienstleistungen an. Die Erbringung von Dienstleistungen erfordert teilweise völlig andere Konzepte als die Dienstleistungsentwicklung, weshalb sie nicht im Fokus steht. Trotz allem können z.B. konfigurierte Dienstleistungen an ein Workflowmanagementsystem übergeben werden, wodurch der Betrieb der Dienstleistung vereinfacht wird.

7 Zusammenfassung

In diesem Beitrag wurde ein Metamodell vorgestellt, das verschiedene aus der Literatur bekannte Konzepte zur Modellierung von Dienstleistungen integriert. Das Metamodell hat zum Ziel, kundenindividuelle Konfigurationen komplexer Dienstleistungen zu erstellen. Aus diesem Grund ist eine Formalisierung der Bestandteile des Metamodells notwendig. Dies erfolgt mit Mitteln der formalen Logik. Zur praktischen Einsetzbarkeit wurde mit dem Service Modeller ein Werkzeug vorgestellt, welches auf dem Metamodell basiert. Zuletzt wurde skizziert, in welchen Phasen der Dienstleistungsentwicklung die Nutzung des Metamodells und des Werkzeugs vielversprechend ist.

Trotz der bereits erfolgten praktischen Evaluierung des Metamodells sind noch einige relevante Punkte offen, die in zukünftigen Weiterentwicklungen integriert werden müssen. Eine Anforderung aus der Praxis, die insbesondere Unternehmen betrifft, deren Dienstleistungen einer saisonal schwankenden Nachfrage unterliegen, ist die Integration und Beachtung dieser Schwankungen im Metamodell. Im oben angesprochenen Call-Center-Beispiel kann die Anzahl eingehender Anrufe unter anderem zum Weihnachts- und Ostergeschäft besonders hoch sein, während sie in den Sommermonaten zurückgeht. Diese Schwankungen können z.B. beachtet werden, indem Variablen statt eines festen Wertes ein Intervall zugewiesen wird. Dadurch lassen sich Auswirkungen auf die Produktivität bereits während der Konfiguration ermitteln.

Neben saisonal schwankender Nachfrage sind außerdem weitere Restriktionen hinsichtlich der Erbringung von Dienstleistungen relevant. Eine Wartungsdienstleistung für Windkraftanlagen kann z.B. nur bei mäßigem Wind durchgeführt. Diese dynamischen Einflüsse lassen sich momentan nicht direkt darstellen, sondern müssen statisch definiert werden. Aus diesem Grund ist es notwendig, Vorbedingungen zu definieren, die erfüllt sein müssen, damit Komponenten ausgeführt werden können. Dafür kann auf die bereits vorhandenen Möglichkeiten zur Definition allgemeiner Beziehungen zwischen Elementen zurückgegriffen werden. Weitere Formalisierungsansätze sind auf ihre Eignung zur Beschreibung der Bedingungen hin zu untersuchen.

Neben der in diesem Beitrag gezeigten Nutzung des Metamodells und des Service Modellers zur Modellierung von Dienstleistungen, können diese auch bei der Entwicklung sogenannter Product Service Systems (PSS) bzw. hybriden Produkten eingesetzt werden. Einer der Grundgedanken hinter der Entwicklung von PSS ist die Bereitstellung einer bestimmten Funktionalität durch Dienstleistungen statt durch Produkte [Mon02]. Dadurch sind auch Anbieter klassischer Produkte zunehmend mit der Erbringung von Dienstleistungen konfrontiert. Bei der Modellierung von PSS sind neben der reinen Dienstleistungsperspektive auch die Produktperspektive sowie Interaktionen zwischen beiden Bestandteilen zu beachten. In [BK12a] wurde eine erste Übertragung der Konzepte des Metamodells auf die PSS-Domäne gezeigt.

8 Danksagung

Diese Arbeit wurde durch das Bundesministerium für Bildung und Forschung (BMBF) im Rahmen des Forschungsprojekts KoProServ unter dem Kennzeichen 01FL09004 gefördert (Projektträger DLR). Weitere Informationen zum Projekt finden sich unter http://koproserv.uni-leipzig.de.

Literatur

[AB11] APEL, Sven ; BEYER, Dirk: Feature cohesion in software product lines: an exploratory study. In: *Proceedings of the 33rd International Conference on Software Engineering*. New York, NY, USA : ACM, 2011 (ICSE '11). – ISBN 978–1–4503–0445–0, S. 421–430

[AP06] AALST, W. van d. ; PESIC, M.: DecSerFlow: Towards a Truly Declarative Service Flow Language. In: BRAVETTI, Mario (Hrsg.) ; NÚÑEZ, Manuel (Hrsg.) ; ZAVATTARO, Gianluigi (Hrsg.): *Web Services and Formal Methods* Bd. 4184. Springer Berlin / Heidelberg, 2006. – ISBN 978–3–540–38862–3, S. 1–23

[BC97] BALDWIN, C. Y. ; CLARK, K. B.: Managing in an age of modularity. In: *Harvard Business Review* 75 (1997), Nr. 5, S. 84–93. – ISSN 0017–8012

[BF09] BÖTTCHER, Martin ; FÄHNRICH, Klaus-Peter: Service Systems Modeling. In: ALT, R. (Hrsg.) ; FÄHNRICH, K.-P. (Hrsg.) ; FRANCZYK, B (Hrsg.): *Proceedings First International Symposium on Services Science*. Leipzig, Germany : Logos, March 2009

[BK10] BÖTTCHER, Martin ; KLINGNER, Stephan: Der Komponentenbegriff der Dienstleistungsdomäne. In: FÄHNRICH, Klaus-Peter (Hrsg.) ; FRANCZYK, B. (Hrsg.): *Informatik 2010 – GI Jahrestagung* Bd. 1, Lecture Notes in Informatics (LNI), 2010. – ISBN 978–3–88579–269–7, S. 59–66

[BK11] BÖTTCHER, Martin ; KLINGNER, Stephan: Providing a Method for Composing Modular B2B-Services. In: *Journal of Business and Industrial Marketing* 26 (2011), Nr. 5, S. 320–331. – ISSN 10.1108/08858621111144389

[BK12a] BECKER, Michael ; KLINGNER, Stephan: Formalisierung von Regeln zur Darstellung von Abhängigkeiten zwischen Elementen von Product-Service-Systems / Abteilung für Betriebliche Informationssysteme, Universität Leipzig. Leipzig, Germany, February 2012. – Forschungsbericht

[BK12b] BECKER, Michael ; KLINGNER, Stephan: *Towards customer-individual configuration of business process models*. September 2012. – to appear at BPMDS'12

[BM12] BÖTTCHER, Martin (Hrsg.) ; MEIREN, Thomas (Hrsg.): *Anforderungen an die Produktivität und Komponentisierung von Dienstleistungen: Ergebnisse einer Studie unter technischen Dienstleistern*. Stuttgart : Fraunhofer Verlag, 2012. – ISBN 978–3–8396–0242–3

[CG07] CORSTEN, Hans ; GÖSSINGER, Ralf: *Dienstleistungsmanagement*. Oldenbourg, 2007

[CHE04] CZARNECKI, Krzysztof ; HELSEN, Simon ; EISENECKER, Ulrich: Staged Configuration Using Feature Models. In: NORD, Robert (Hrsg.): *Software Product Lines* Bd. 3154. Boston, MA, USA : Springer Berlin / Heidelberg, 2004. – ISBN 978–3–540–22918–6, S. 162–164

[FLKB11] FREITAG, Mike ; LAMBERTH, Sabrina ; KLINGNER, Stephan ; BÖTTCHER, Martin: Method of collecting and categorising performance indicators to measure the productivity of modular services using an IT tool. In: GANZ, Walter (Hrsg.) ; KICHERER, Florian (Hrsg.) ; SCHLETZ, Alexander (Hrsg.): *RESER 2011 Productivity of Services NextGen - Beyond Output / Input. Conference Proceedings*. Hamburg, Germany, September 2011

[GO04] GRÖNROOS, Christian ; OJASALO, Katri: Service productivity: Towards a conceptualization of the transformation of inputs into economic results in services. In: *Journal of Business Research* 57 (2004), Nr. 4, S. 414 – 423. `http://dx.doi.org/10.1016/S0148-2963(02)00275-8`. – DOI 10.1016/S0148–2963(02)00275–8. – ISSN 0148–2963. – European Research in service marketing

[Har95] HART, Christopher W.: Mass customization: conceptual underpinnings, opportunities and limits. In: *International Journal of Service Industry Management* 6 (1995), Nr. 2, S. 36–45. – ISSN 0956–4233

[HPT05] HEISKALA, Mikko ; PALOHEIMO, Kaija-Stiina ; TIIHONEN, Juha: Mass Customisation of Services: Benefits and Challenges of Configurable Services. In: *Frontiers of E-Business Research*, 2005

[KBBD11] KLINGNER, Stephan ; BÖTTCHER, Martin ; BECKER, Michael ; DÖHLER, Arndt: Managing complex service portfolios. In: GANZ, Walter (Hrsg.) ; KICHERER, Florian (Hrsg.) ; SCHLETZ, Alexander (Hrsg.): *RESER 2011 Productivity of Services NextGen - Beyond Output/Input. Conference Proceedings*, 2011. – ISBN 978–3–8396–0298–0

[McI68] MCILROY, M. D.: Mass-Produced Software Components. In: BUXTON, J. M. (Hrsg.) ; NAUR, Peter (Hrsg.) ; RANDELL, Brian (Hrsg.): *Software Engineering Concepts and Techniques (1968 NATO Conference of Software Engineering)*, NATO Science Committee, 1968, S. 88–98

[Mey08] MEYER, Kyrill: Software-Service-Co-Design – eine Methodik für die Entwicklung komponentenorientierter IT-basierter Dienstleistungen. In: GATERMANN, Inken (Hrsg.) ; FLECK, Myriam (Hrsg.): *Technologie und Dienstleistung: Innovationen in Forschung, Wissenschaft und Unternehmen. Beiträge der 7. Dienstleistungstagung des BMBF*, 2008

[Mon02] MONT, O. K.: Clarifying the concept of product-service system. In: *Journal of Cleaner Production* 10 (2002), Nr. 3, S. 237 – 245. – ISSN 0959–6526

[O'S08] O'SULLIVAN, Justin J.: *Towards a precise understanding of service properties*, Queensland University of Technology, Faculty of Information Technology, Diss., 2008

[Pap04] PAPATHANASSIOU, Eleutherios A.: Mass customisation: management approaches and internet opportunities in the financial sector in the UK. In: *International Journal of Information Management* 24 (2004), Nr. 5, S. 387 – 399. – ISSN 0268–4012

[PI99] PINE II, B. J.: *Mass customization: The new frontier in business competition*. Harvard Business School Press, 1999. – ISBN 0–87584–372–7

[Sho84] SHOSTACK, G. L.: Designing Services That Deliver. In: *Harvard* 62 (1984), S. 133–139

[Sun94] SUNDBO, Jon: Modulization of service production and a thesis of convergence between service and manufacturing organizations. In: *Scandinavian Journal of Management* 10 (1994), Nr. 3, S. 245–266. `http://dx.doi.org/\url{10.1016/0956-5221(94)90002-7}.` – DOI `10.1016/0956--5221(94)90002--7`

[TBK09] THUM, Thomas ; BATORY, Don ; KASTNER, Christian: Reasoning about edits to feature models. In: *Proceedings of the 31st International Conference on Software Engineering*. Washington, DC, USA : IEEE Computer Society, 2009 (ICSE '09). – ISBN 978–1–4244–3453–4, S. 254–264

[UMB+08] UHLMANN, E. ; MEIER, H. ; BOCHNIG, H. ; GEISERT, C. ; SADEK, K. ; STELZER, C.: Customer-driven development of product-service-systems. In: PHAM, D. T. (Hrsg.) ; ELDUKHRI, E. E. (Hrsg.) ; SOROKA, A. J. (Hrsg.): *Innovative production machines and systems : Fourth I*PROMS Virtual International Conference, 1st - 14th July, 2008*, Whittles Publ. [u.a.], 2008. – ISBN 978–1–904445–81–4