Analyse der Eignung domänenspezifischer Methoden der Anforderungsverfolgung für Produkt-Service-Systeme

Thomas Wolfenstetter[1], Sebastian Floerecke[2], Markus Böhm[1], und Helmut Krcmar[1]

[1] Technische Universität München, Lehrstuhl für Wirtschaftsinformatik, Garching, Deutschland
{thomas.wolfenstetter, markus.boehm, krcmar}@in.tum.de
[2] Universität Passau, Lehrstuhl für Wirtschaftsinformatik II, Passau, Deutschland
sebastian.floerecke@uni-passau.de

Abstract. Die Anforderungsverfolgung bringt bei der Entwicklung von Produkt-Service-Systemen (PSS) zahlreiche Herausforderungen mit sich. Gründe hierfür sind komplexe Schnittstellen zwischen den Domänen Produkt-, Software- und Dienstleistungsentwicklung, unterschiedliche Lebenszyklen und gegenseitige Beeinflussung einzelner Komponenten, ein hoher Grad an technischer Integration sowie eine kundenindividuelle Leistungserstellung. Verstärkt wird diese Komplexität dadurch, dass sich Anforderungen an das PSS entlang des gesamten Lebenszyklus ändern können. Während in der Literatur domänenspezifische Anforderungsverfolgungsmethoden zu finden sind, existiert bislang kein PSS-spezifischer Ansatz. Ziel dieses Beitrags ist daher, zu untersuchen, inwieweit sich diese Methoden für PSS eignen. Die Grundlage dafür bilden zehn, aus den Eigenschaften von PSS abgeleitete Kriterien. Die Analyse zeigt, dass keine der Methoden alle Kriterien vollständig erfüllt. Dennoch bieten einige bei der Verfolgung der Anforderungsherkunft, der Beziehung zwischen Anforderungen, der Anforderungsumsetzung sowie dem Versionsmanagement vielversprechende Ansätze. Diese Bewertung dient als Ausgangspunkt für eine gezielte Kombination und Erweiterung der Methoden, um eine adäquate Anforderungsverfolgung bei PSS zu ermöglichen.

Keywords: Produkt-Service-System, Anforderungsverfolgung, Anforderungsmanagement, Analyse, Literaturstudie

1 Ausgangssituation und Problemstellung

Um dem gestiegenen Wettbewerbsdruck entgegenzuwirken und sich von der Konkurrenz zu differenzieren, bieten Unternehmen unterschiedlichster Branchen vermehrt Komplettlösungen für individuelle Kundenprobleme an [1]. Ein Beispiel hierfür sind Carsharing-Angebote zur Erfüllung eines Mobilitätsbedürfnisses. Bei derartigen Lösungen handelt es sich um integrierte Leistungsbündel, bestehend aus Hardware-, Software- und Dienstleistungskomponenten, die als Produkt-Service-Systeme (PSS) bezeichnet werden [2]. Der Wertschöpfungsprozess im Kontext von PSS ist dabei auf eine dauerhafte und intensive Geschäftsbeziehung ausgerichtet, in welcher der Kunde nicht mehr länger nur als Wertschöpfungsempfänger, sondern als Wertschöpfungs-

12th International Conference on Wirtschaftsinformatik,
March 4-6 2015, Osnabrück, Germany

Wolfenstetter, T.; Floerecke, S.; Böhm, M.; Krcmar, H. (2015): Analyse der Eignung domänenspezifischer Methoden der Anforderungsverfolgung für Produkt-Service-Systeme, in: Thomas. O.; Teuteberg, F. (Hrsg.): Proceedings der 12. Internationalen Tagung Wirtschaftsinformatik (WI 2015), Osnabrück, S. 210-224

partner auftritt [3]. Hierbei trägt der Anbieter häufig die Verantwortung für den gesamten Lebenszyklus eines PSS [2, 4]. Die Entwicklung eines PSS erfordert eine integrative Zusammenarbeit von Produkt-, Software- und Dienstleistungsentwicklung. Jede dieser Disziplinen entwickelt einzelne Komponenten, die sich gegenseitig beeinflussen, nahtlos zusammenwirken sollen aber verschiedenen Lebenszyklen unterworfen sind [5, 6]. Oberstes Ziel eines PSS ist die möglichst optimale Erfüllung eines individuellen Kundenbedürfnisses. Daneben müssen Anbieter, zumindest im Business-to-Business-Bereich, die Geschäftsprozesse ihrer Kunden verstehen, um das PSS in deren Wertschöpfung integrieren zu können [3, 7]. Aus Anbietersicht ist es daher entscheidend, sämtliche Anforderungen an die Lösung genau zu erheben, zu spezifizieren und deren Abhängigkeiten zu kennen [8].

Entlang des gesamten PSS-Lebenszyklus kommt es jedoch fortlaufend zu Änderungen [9, 10], da dieser zyklischen Einflüssen unterworfen ist [11]. Beispiele für Zyklen sind die Änderung von Kundenwünschen und Gesetzen oder die Verfügbarkeit neuer Technologien [8]. Im Carsharing-Bereich etwa besteht eine zentrale Anforderung an den Dienstleistungsprozess darin, dass der Kunde vor Übernahme des Fahrzeugs dieses auf Vorschäden prüft und nicht erfasste Mängel meldet. Durch Einführung einer neuen Sensortechnologie könnten dagegen Schäden bereits bei Entstehung erfasst und weitergeleitet werden. Dies würde die ursprüngliche Anforderung überflüssig machen und die Nutzung vereinfachen, brächte aber Änderungen am Fahrzeug und der Software mit sich, welche wiederum weitere Änderungen anstoßen könnten. Zudem könnte diese Anpassung gegen gesetzliche Datenschutzregeln verstoßen und somit rechtliche Folgen haben. Im Rahmen des Anforderungsmanagements müssen PSS-Anbieter daher mögliche Änderungen antizipieren, die Auswirkungen auf weitere Anforderungen und Komponenten erkennen und Maßnahmen zum Umgang mit Änderungen anstoßen. Voraussetzung dafür ist, den Lebenszyklus einer Anforderung zu verfolgen und Abhängigkeiten zwischen Anforderungen und PSS-Komponenten zu identifizieren, diese abzubilden und zu pflegen [5]. Dieser Bereich des Anforderungsmanagements, dem besonders bei der PSS-Entwicklung eine große Bedeutung zukommt, wird als Anforderungsverfolgung bezeichnet [12, 13].

Obwohl in der Literatur verschiedene, domänenspezifische Anforderungsverfolgungsmethoden zu finden sind, existiert bislang kein domänenübergreifender Ansatz, der den Herausforderungen von PSS genügt [5, 14]. Dieser Beitrag untersucht daher, inwieweit sich diese domänenspezifischen Methoden der Anforderungsverfolgung für die PSS-Entwicklung eignen. Der Analyse liegen zehn Kriterien zugrunde, die aus den speziellen Eigenschaften von PSS und den Aufgaben der Anforderungsverfolgung bei PSS hergeleitet wurden.

2 Grundlagen der Anforderungsverfolgung bei PSS

Die Verfolgung von Anforderungen stellt eine Querschnittsfunktion des Anforderungsmanagements dar, die während des kompletten Lebenszyklus eines Produkts, einer Dienstleistung oder eines PSS zu gewährleisten ist [15, 16]. Der Lebenszyklus einer Anforderung wird dabei von ihrem Ursprung über alle Phasen der Entwicklung

verfolgt und die Abhängigkeiten zwischen verschiedenen Artefakten, wie etwa Anforderungen, PSS-Komponenten oder Testfälle, dokumentiert und gepflegt [12, 13]. Wie das Carsharing-Beispiel zeigt, können etwa Dienstleistungen das Design der Software oder der Hardware beeinflussen. Daher muss die Anforderungsverfolgung bei PSS domänenübergreifend durchgeführt werden [8]. Zur Verfolgung von Anforderungen werden semantische Abhängigkeiten, sogenannte „Trace Links", dokumentiert. Trace Links geben an, wie Artefakte zueinander in Beziehung stehen [17]. Durch Navigieren entlang der Trace Links lässt sich unter anderem nachvollziehen, warum eine Anforderung spezifiziert wurde, durch welche PSS-Komponenten eine Anforderung erfüllt wird oder welche Testfälle ihrer Absicherung dienen.

Die Anforderungsverfolgung lässt sich in drei Dimensionen unterteilen: Pre-Traceability erfasst Trace Links von der Anforderungsquelle, beispielsweise Kunde oder Gesetzgeber [18], bis zu ihrer Spezifikation [12]. Post-Traceability dagegen umfasst Trace Links einer Anforderung von ihrer Spezifikation bis zu ihrer Umsetzung, etwa als Softwarecode oder physische Komponente [12]. Drittens können Beziehungen zwischen Anforderungen auf derselben oder unterschiedlichen Abstraktionsebenen verfolgt werden. Diese Dimension wird als Inter-Traceability bezeichnet und beinhaltet Überlappungen, Konkretisierungen oder auch Konflikte zwischen Anforderungen [18, 19]. Für PSS ist Inter-Traceability von besonderer Bedeutung, da die Anforderungsspezifikation ein komplexes, domänenübergreifendes und multihierarchisches Netzwerk aus interdependenten Anforderungen darstellt [10]. Abbildung 1 fasst die Dimensionen der Anforderungsverfolgung zusammen:

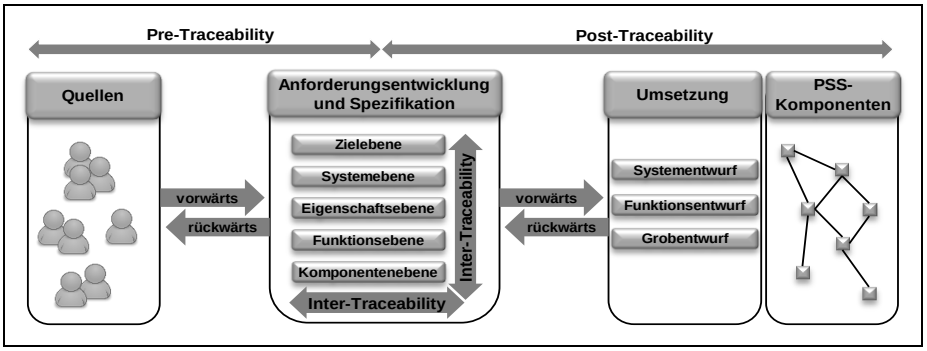

Abbildung 1. Dimensionen der Anforderungsverfolgung bei PSS (In Anlehnung an [12])

Durch eine systematische Anforderungsverfolgung lässt sich gezielt nachprüfen, ob und wie Anforderungen tatsächlich umgesetzt wurden. Des Weiteren können Komponenten oder Funktionen identifiziert werden, die keine Anforderung erfüllen und daher möglicherweise überflüssig sind [20, 21]. Gleichzeitig wird das nachträgliche Einschleichen von Anforderungen in die Spezifikation verhindert, da für jede Anforderung die Quelle sowie der Grund deren Aufnahme nachvollziehbar ist [16, 22]. Bei einer bevorstehenden Änderung lässt sich analysieren, welche Artefakte ebenfalls betroffen sind und gegebenenfalls angepasst werden müssen [19, 20]. Außerdem fördert die Anforderungsverfolgung die Wiederverwendung von Artefakten, da festge-

stellt werden kann, welches Artefakt verwendet werden könnte und welche Anpassungen für die Verwendung in einem anderen Kontext nötig sind [17, 21]. Ferner wird die Überwachung des Projektfortschritts, die Ressourcenallokation sowie das Controlling unterstützt [16, 23]. Zudem kann geprüft werden, welche Testfälle eine Anforderung verifizieren und ob jede Anforderung durch einen Test abgesichert wird [19, 22].

3 Forschungsdesign

Um verschiedene Anforderungsverfolgungsmethoden zu identifizieren, wurde eine Literaturstudie nach den Richtlinien von Webster und Watson [24] durchgeführt. Dabei wurde zunächst in den Publikationsdatenbanken Google Scholar, IEEE Xplore, ACM Digital Library, Springer Link und Emerald Insight nach relevanten Schlüsselwörtern wie „Requirements Traceability“, „Traceability“, „Tracing“ oder „Anforderungsverfolgung“ gesucht. Zudem wurden diese Begriffe in Verbindung mit Suchbegriffen wie „Product Development“, „Service“, „Product Service System“ oder „Hybrides Produkt“ geprüft. Anschließend bewerteten die Autoren getrennt voneinander die Relevanz der Veröffentlichungen, die in den Datenbanken jeweils unter den 500 besten Treffern lagen und mindestens einmal zitiert wurden. Im Fokus standen Publikationen, in denen Anforderungsverfolgungsmethoden beschrieben wurden. Grundlage der Publikationsauswahl für die detaillierte Prüfung war der Durchschnitt der Relevanzbewertung von Titel und Abstract seitens der Autoren. Danach wurden die Literaturverzeichnisse der identifizierten Beiträge systematisch geprüft. Gleichzeitig wurde kontrolliert, welche Quellen wiederum die bisher betrachteten Beiträge zitieren. Dieses Vorgehen wurde iterativ wiederholt, bis keine weiteren Methoden ausfindig gemacht werden konnten. Insgesamt wurden nach der Streichung von Duplikaten sowie derjenigen Publikationen, in den die Anforderungsverfolgung nur eine untergeordnete Rolle spielt, 15 Methoden aus 118 Veröffentlichungen identifiziert[1].

Die identifizierten Anforderungsverfolgungsmethoden wurden hinsichtlich ihrer Eignung für PSS anhand von zehn Kriterien analysiert. Hergeleitet wurden die Kriterien aus den Eigenschaften von PSS und den Aufgabenbereichen der Anforderungsverfolgung im Kontext von PSS. Die Methodenbewertung erfolgte anhand einer dreistufigen Skala, unabhängig voneinander durch die Autoren im Hinblick darauf, ob und in welchem Umfang die Kriterien in der Beschreibung der Methoden und deren Diskussion thematisiert wurden. Dabei ergab die Bewertung in 84,7 % der Fälle eine exakte Übereinstimmung (Krippendorffs Alpha: 0,709). Bei unterschiedlichen Bewertungen wurden gemeinsam Argumente abgewogen und eine Konsensentscheidung herbeigeführt. Voll erfüllt bedeutet in diesem Beitrag, dass höchstens nur leichte Erweiterungen für die Verwendung im PSS-Kontext nötig sind. Bei einer teilweisen Erfüllung wird zwar ein Kriterium in den Publikationen erwähnt, allerdings sind weitreichende Änderungen erforderlich. Wenn ein Kriterium überhaupt nicht thematisiert wurde, erfolgte eine Einstufung als nicht erfüllt.

[1] Ein vollständiger Überblick kann unter folgender Adresse abgerufen werden: tinyurl.com/traceability-publikationen

4 Methoden der Anforderungsverfolgung

Im Rahmen der Literaturstudie konnten 15 Methoden der Anforderungsverfolgung identifiziert werden. Viele Methoden stammen dabei aus dem Umfeld der Softwareentwicklung. Ein möglicher Grund dafür ist, dass in diesem Bereich das Anforderungsmanagement innerhalb der Forschung bislang stärker als bei den Ingenieurswissenschaften adressiert wurde [5]. Obwohl explizit danach gesucht wurde, konnte keine spezifische Methode für Dienstleitungen gefunden werden. Dies kann damit begründet werden, dass die Dienstleistungsentwicklung ein relativ junges Forschungsfeld darstellt [25] und dort oft Ansätze aus anderen Domänen adaptiert werden.

Insgesamt handelt es sich bei diesen Methoden um heterogene Ansätze, die sich auf unterschiedliche Aspekte der Anforderungsverfolgung konzentrieren. So legen einige Methoden den Fokus darauf, wie Trace Links identifiziert (z. B. [26]), dokumentiert oder gepflegt werden können (z. B. [27]), während sich andere damit beschäftigen, wie diese Informationen visualisiert (z. B. [28]) oder wie Auswirkungen von Änderungen prognostiziert werden können. Konzeptuelle Ansätze hingegen beschreiben, welche Artefakte bei der Anforderungsverfolgung berücksichtigt werden müssen, welche Arten von Beziehungen existieren und welche Informationen über jedes Artefakt benötigt werden (z. B. [13]). Daneben gibt es Prozessmodelle, die definieren, welche Aktivitäten bei der Anforderungsverfolgung durchzuführen sind. Die in diesem Beitrag als Methode verstandenen Ansätze beschreiben hingegen, wie diese Aktivitäten durchgeführt beziehungsweise unterstützt werden können. Die den einzelnen Methoden zugrunde liegenden Ideen werden nachfolgend kurz vorgestellt:

(1) Information Retrieval basiert auf einem Ähnlichkeitsvergleich zweier Artefakte zur automatischen Identifikation möglicher Trace Links. Ein Artefakt fungiert als Anfrage und ein anderes als Dokument, das hinsichtlich der Anfrage durchsucht wird. So agiert beispielsweise der Softwarecode als Anfrage und die Anforderungsspezifikation als Dokument. Artefakte mit hohen Ähnlichkeitswerten gelten dabei als Kandidaten für Trace Links. Um Unterschiede zwischen verschiedenen Artefaktversionen zu erkennen, kann dieser Ansatz mehrfach angewandt werden [26].

(2) Bei *Event-based-Traceability*, einem Architekturkonzept für Softwaretools, werden Trace Links als Publisher-Subscriber-Beziehungen abgebildet. Diese Methode setzt existierende Trace Links voraus. Sobald eine Änderung eintritt, werden davon betroffene Stellen benachrichtigt, um mögliche Auswirkungen zu analysieren [27].

(3) Der Grundgedanke von *Rule-based-Traceability* ist die Verwendung von XML-basierten Regeln zur automatischen Generierung von Trace Links. Dazu müssen neben den Regeln selbst, das Anwendungsfalldokument und das Analyseobjektmodell im XML-Format vorliegen [29].

(4) Während andere Methoden keine Unterscheidung bei den Anforderungen bezüglich ihrer Bedeutung für die Anforderungsverfolgung vornehmen, setzt das *Value-based-Traceability* genau an dieser Stelle an. Ziel ist es, besonders wichtige Anforderungen detaillierter zu verfolgen als solche mit niedriger Priorität. Dadurch ergeben sich Kostensenkungspotentiale bei der Anforderungsverfolgung [30].

(5) Der Ausgangspunkt von *Feature-Model-based-Traceability* ist, dass der Unterschied im Abstraktionsgrad und der Formalität zwischen Anforderungen und Features

geringer als zwischen einer Anforderung und einem Lösungsartefakt ist. Ein Feature beschreibt eine Produkteigenschaft aus Sicht des Kunden und stellt das Bindeglied zwischen Anforderungen und Lösungsartefakten dar. Die Trace Links verlaufen somit von den Anforderungen zu den Features bis hin zu den Lösungselementen [31].

(6) Das Ziel des *Feature-oriented-Traceability* ist das Identifizieren von Trace Links basierend auf priorisierten Anforderungen hinsichtlich Kosten und Aufwand. Daran lässt sich erkennen, dass dieses Verfahren eine Erweiterung des Value-based-Traceability darstellt [32].

(7) Scenario-based-Traceability gleicht statische Informationen aus Entwicklungsmodellen mit dynamischen Informationen aus dem aktuell implementierten Softwareprodukt ab. Dazu wird das Verhalten der Software mit Testszenarios, die während der Entwicklung definiert wurden, beobachtet. Da die meisten Beobachtungen direkt mit Szenarien korrespondieren, können auf diese Weise Trace Links zwischen Szenarien und dem System identifiziert werden [33].

(8) Hypertext-based-Traceability verwendet ein Hypertextmodell, welches das komplexe Verbinden und Versionieren von Nachvollziehbarkeitsbeziehungen ermöglicht. Zur Generierung der Trace Links wird das Information Retrieval eingesetzt. Es nutzt XML als Datenformat zur Repräsentation der Modelle und der generierten Trace Links. Diese müssen daher in einer XML-Darstellung vorliegen [34].

(9) Goal-centric-Traceability zielt darauf ab, nicht-funktionale Anforderungen zu verfolgen. Nicht-funktionale Anforderungen, wie Zuverlässigkeit, Sicherheit oder Wartbarkeit, gelten als besonders anspruchsvoll zu verfolgen, da sie Auswirkungen auf das System als Ganzes haben und zwischen ihnen zahlreiche Abhängigkeiten und Zielkonflikte bestehen [35].

(10) Der Ausgangspunkt von *Pre-Requirements Specification Traceability* ist die Annahme, dass Pre-Traceability im Vergleich zu Post-Traceability deutlich schwieriger ist, da Trace Links zwischen dem Problemraum – den Bedürfnissen – und dem Lösungsraum – der Anforderungsspezifikation – etabliert werden müssen. Da die Distanz zwischen beiden Räumen groß ist, wird ein Übergangsraum eingefügt, um diese Komplexitätslücke zu reduzieren [36].

(11) Einen weiteren Ansatz, der speziell zur Unterstützung des Pre-Traceability entwickelt wurde, stellen *Contribution Structures* dar. Dabei werden Trace Links zwischen Anforderungen und Stakeholdern generiert. Beispielsweise wird die Anforderungsquelle oder die Verantwortlichkeit für die Umsetzung einer Anforderung dokumentiert [37].

(12) Bei *Traceability-Matrizen*, wie Design-Structure- oder Domain-Mapping-Matrizen, müssen die Trace Links manuell erzeugt werden. Sie werden häufig in der Praxis verwendet, um sowohl Beziehungen zwischen Anforderungen untereinander als auch zwischen Anforderungen und anderen Entwicklungsartefakten, wie Testfällen, Codemodulen und dem Design, zu etablieren und visualisieren [28].

(13) Das *Reference Model for Requirements Traceability* definiert bestimmte Typen von Artefakten und Trace Links. Dabei wird zwischen verschiedenen Detaillierungsstufen der Anforderungsverfolgung unterschieden [13].

(14) Beim *Quality Function Deployment* handelt es sich um einen strukturierten Ansatz zur Übersetzung von Kundenanforderungen in Designziele sowie zur Analyse

der Abhängigkeiten zwischen Komponenten. Damit lässt sich darstellen, welche Kundenanforderung durch welche Produktkomponente umgesetzt wird [38].

(15) Die *Fehlermöglichkeits- und Einflussanalyse* wird eingesetzt, um eine möglichst fehlerfreie Gestaltung von Produkten und Prozessen unter Einhaltung aller Kunden- und Qualitätsanforderungen zu erzielen. Sie basiert auf dem Prinzip der vorausschauenden Fehlervermeidung. Wie auch beim Quality Function Deployment lässt sich abbilden, welche Komponente welche Anforderung erfüllt [39].

5 Bewertungskriterien

Der Methodenbewertung liegen folgende zehn Kriterien zugrunde, die anhand der Literaturstudie aus den Eigenschaften von PSS und den Aufgabenbereichen der Anforderungsverfolgung im Kontext von PSS hergeleitet wurden:

(1) Verfolgung der Anforderungsherkunft: Ist dieses Kriterium erfüllt, so kann unter anderem nachvollzogen werden, warum eine Anforderung spezifiziert wurde und wer welche Verantwortung trägt [18, 40]. Allerdings wird häufig darauf hingewiesen, dass Pre-Traceability in der Praxis die größte Herausforderung darstellt und deshalb auch nur selten angewendet wird [12, 36]. Im Kontext von PSS ist Pre-Traceability besonders wichtig und zugleich schwierig, da meist viele, heterogene Interessensgruppen mit unterschiedlichen Anforderungen berücksichtigt werden müssen [5].

(2) Verfolgung der Beziehung zwischen Anforderungen: Bei der Detaillierung von Anforderungen an ein PSS werden abstrakte Geschäftsziele iterativ heruntergebrochen bis konkrete, domänenspezifische Anforderungen an jede PSS-Komponente spezifiziert werden können. Die strukturierte Abbildung der Beziehungen zwischen Anforderungen an Sach- und Dienstleistungskomponenten eines PSS dient somit der Koordination der verschiedenen Domänen [10]. Bei der Anforderungsverfolgung für PSS muss daher berücksichtigt werden, dass mehrere Abstraktionsebenen, von Geschäftszielen bis hin zu detaillierten Komponentenanforderungen, existieren [14]. Anforderungen können hierbei in Konflikt zueinander stehen, sich gegenseitig einschränken oder erweitern [19, 41]. Durch die Erfassung dieser Beziehungen kann somit die Anforderungsbasis strukturiert werden [14, 18].

(3) Verfolgung der Anforderungsumsetzung: Dieses Kriterium umfasst Trace Links einer Anforderung von ihrer Spezifikation bis zur Umsetzung durch eine Lösungskomponente des PSS [12]. Zu jedem Zeitpunkt der Entwicklung muss erkennbar sein, zu welchem Grad Anforderungen umgesetzt oder umsetzbar sind [17, 42]. Außerdem sollen Komponenten oder Funktionen identifiziert werden können, die keine Anforderung erfüllen und daher möglicherweise überflüssig sind [13, 22].

(4) Versionsmanagement: Da Artefakte bei der Entwicklung von PSS einer ständigen Veränderung unterworfen sind, muss eine Methode zur Anforderungsverfolgung in der Lage sein, verschiedene Versionen eines Artefakts abzubilden und miteinander zu verknüpfen [10, 17]. Nur anhand dieser Evolutionskette kann nachvollzogen werden, welche Änderungen aus welchem Grund und zu welchem Zeitpunkt vorgenommen wurden. Unterschiedliche Versionen müssen dabei dokumentiert und gegebenenfalls auch wiederhergestellt werden können [18].

(5) Varianten- und Konfigurationsmanagement: PSS zielen meist auf die Lösung eines spezifischen Kundenproblems ab und müssen daher individuell gestaltet werden. Bei einer unsystematischen Individualisierung und Variantenbildung ist es für den PSS-Anbieter jedoch schwierig, profitabel zu wirtschaften [43, 44]. Um gewisse Komponenten wiederverwenden zu können und damit Kosten zu sparen, wird das Konzept der Modularisierung eingesetzt. Dadurch können individuelle Kundenanforderungen oft durch die Kombination bestehender Module abgedeckt werden [45]. Die Anforderungsverfolgung sollte dabei aufzeigen, wie verschiedene Komponenten kombiniert werden müssen, um sämtliche Anforderungen zu erfüllen [14]. Hierdurch ergeben sich zusätzliche Herausforderungen, da sich die Menge der Informationen und die Komplexität, Änderungen zu managen, erhöht [46].

(6) Integration des Wertschöpfungsnetzwerks: Nachdem PSS oftmals aus vielen Komponenten bestehen, ist es für Anbieter zumeist nicht möglich, sämtliche Bestandteile selbst zu entwickeln oder anzubieten [47]. Auch wenn sie in der Regel alleine die Geschäftsbeziehung mit ihren Kunden unterhalten, sind PSS-Anbieter von der Qualität ihrer Modullieferanten abhängig [48]. Daher müssen Anbieter die Lieferanten und Partner aus ihrem Wertschöpfungsnetzwerk in die Entwicklung einbinden. Anforderungsverfolgung bei Entwicklung von PSS sollte daher ebenfalls unternehmensübergreifend durchführbar sein. So muss Partnern der aktuelle Stand, der für sie relevanten Teilmenge, der Anforderungsbasis zugänglich gemacht werden können.

(7) Simultane Entwicklung und unterschiedliche Sichten: Entwicklungsaufgaben werden häufig so aufgeteilt, dass Teams gleichzeitig und weitgehend autonom an bestimmten Teilaufgaben arbeiten können [49]. Dies sollte durch die Anforderungsverfolgung unterstützt werden. Daneben muss es möglich sein, verschiedene Sichten auf das Anforderungsmodell und das gesamte PSS-Modell zu definieren. Dabei sollten die verschiedenen Rollen, wie Projektmanager oder Entwickler, gezielt mit Informationen versorgt werden können, die sie auch tatsächlich benötigen [50].

(8) Robustheit gegenüber Unsicherheiten: Artefakte, wie Designmodelle oder Anforderungen, zeichnen sich oftmals durch ungenaue oder sogar gänzlich fehlende Daten aus. Diese Unsicherheit stammt aus dem Umfeld des Entwicklungsprojekts, der Entwicklungsarbeit selbst oder der Projektdefinition [51]. Unsicherheit tritt zum Beispiel dann auf, wenn die Dauer eines Dienstleistungsprozesses nur ungenau bestimmt werden kann. Bei der Anforderungsverfolgung im Kontext von PSS sollten deshalb derartige Unsicherheiten und Informationslücken adressiert werden können [14].

(9) Berücksichtigung des gesamten PSS-Lebenszyklus: PSS-Anbieter sind meist für den kompletten Lebenszyklus verantwortlich [2, 4]. Auch die Anforderungsverfolgung sollte daher den gesamten Lebenszyklus erfassen. Dabei muss ein breites Spektrum von Lösungsartefakten berücksichtigt werden. So muss etwa abgebildet werden, wie Hardwarekomponenten hergestellt und instand gehalten, wie Dienstleistungen erbracht oder wie Softwaremodule aktualisiert werden. Oft können Erfahrungen aus späten Lebenszyklusphasen Anforderungen an die nächste Generation eines PSS aufzeigen. Dazu muss aber der Anforderungslebenszyklus nachvollziehbar sein [52].

(10) Reduzierung des Aufwands: Da die Erfassung aller relevanter Informationen mit einem hohen Aufwand verbunden ist und dabei nicht jede Anforderung gleich wichtig ist, wäre eine vollständige, aber unsystematische Anforderungsverfolgung in

vielen Fällen unwirtschaftlich [16, 53]. Dies gilt insbesondere für PSS, die oftmals durch eine hohe Anzahl und einer hohen Verflechtung von Anforderungen gekennzeichnet sind [14]. Daher sollte es möglich sein, kritische Anforderungen zu identifizieren und genauer zu verfolgen als unkritische.

6 Bewertung der Anforderungsverfolgungsmethoden

Die Bewertung der 15 Methoden zeigt, dass keine der untersuchten Methoden alle Kriterien erfüllt. Dennoch werden bereits, wie Tabelle 1 verdeutlicht, alle der für PSS relevanten Kriterien von den Methoden, zumindest teilweise, abgedeckt.

Tabelle 1. Zusammenfassung der Methodenbewertung

Kriterium / Methode	(1) Verfolgung der Anforderungsherkunft	(2) Verfolgung der Beziehung zwischen Anforderungen	(3) Verfolgung der Anforderungsumsetzung	(4) Versionsmanagement	(5) Varianten- und Konfigurationsmanagement	(6) Integration des Wertschöpfungsnetzwerks	(7) Simultane Entwicklung und unterschiedliche Sichten	(8) Robustheit gegenüber Unsicherheiten	(9) Berücksichtigung des gesamten PSS-Lebenszyklus	(10) Reduzierung des Aufwands	Anzahl positiver Bewertungen
Information Retrieval	○	●	●	◐	◐	○	○	○	○	○	3
Event-based-Traceability	○	●	●	●	○	◐	●	○	○	○	4,5
Rule-based-Traceability	○	●	◐	●	◐	◐	◐	○	○	○	4
Value-based-Traceability	○	○	●	○	○	○	○	○	○	●	2
Feature-Model-based-Traceability	○	○	●	●	●	○	◐	◐	◐	○	4,5
Feature-oriented-Traceability	○	○	●	○	●	○	○	○	◐	●	3,5
Scenario-based-Traceability	◐	○	◐	○	○	○	○	○	○	○	1
Hypertext-based-Traceability	○	●	●	●	○	○	○	○	◐	○	3,5
Goal-centric-Traceability	○	◐	◐	●	○	○	○	○	◐	◐	3
Pre-Requirements Specification Traceability	●	◐	○	○	○	○	○	○	○	○	1,5
Contribution Structures	●	◐	○	○	○	◐	○	○	○	○	2
Traceability-Matrizen	○	●	●	○	◐	○	○	○	○	○	2,5
Reference Models for Requirements Traceability	◐	●	●	●	○	○	◐	○	◐	◐	5
Quality Function Deployment	○	◐	◐	○	◐	○	○	◐	○	○	2
Fehlermöglichkeits- und Einflussanalyse	○	○	◐	○	◐	○	○	◐	○	○	1,5
Anzahl positiver Bewertungen	**3**	**8**	**10,5**	**6,5**	**4,5**	**1,5**	**2,5**	**1,5**	**2,5**	**3**	
Erfüllungsgrad	● *voll erfüllt*			◐ *teilweise erfüllt*			○ *nicht erfüllt*				

Das erste Kriterium, *(1) Verfolgung der Anforderungsherkunft*, wird nur von wenigen Techniken abgedeckt. Allerdings können die Ansätze der positiv bewerteten Methoden auf PSS übertragen werden, da Beziehungen zwischen der Anforderungsquelle und der Spezifikation als weitestgehend domänenneutral angesehen werden können.

Die *(2) Verfolgung der Beziehung zwischen Anforderungen* berücksichtigen zehn der 15 Methoden, zumindest aus ihrer domänenspezifischen Perspektive heraus. Für die Strukturierung von Anforderungen an PSS wurde durch ein Artefaktmodell [10] bereits ein erster, domänenübergreifender Ansatz vorgeschlagen. Dieser unterstützt durch die Aufteilung von Anforderungen in mehreren Abstraktionsebenen die interdisziplinäre Erhebung und die Konkretisierung der Anforderungen. Durch eine Erweiterung dieses Modells um verschiedene Arten von Trace Links kann es für die Verfolgung der Beziehung zwischen Anforderungen eingesetzt werden.

Die *(3) Verfolgung der Anforderungsumsetzung* steht bei den meisten der Methoden im Mittelpunkt. Jedoch werden auch hier nur domänenspezifische Artefakte betrachtet. Bei der Entwicklung von PSS müssen allerdings Beziehungen zwischen heterogenen Artefakten in allen beteiligten Domänen abgebildet werden können. Beispielsweise muss analysierbar sein, wie sich eine Änderung im Konstruktionsentwurf für ein physisches Produkt auf die Steuerungssoftware auswirkt. Insgesamt bilden die Verfahren eine geeignete Basis, die für PSS allerdings erweitert werden müsste.

Das *(4) Versionsmanagement* nimmt in etwa bei der Hälfte der Methoden, jedoch ausschließlich im Softwarebereich, einen hohen Stellenwert ein. Für die Anforderungsverfolgung im Kontext von PSS ist das Managen von Änderungen und somit der Umgang mit verschiedenen Versionen der Artefakte in Anbetracht der zyklischen Wechselwirkungen von enormer Wichtigkeit. Für die Anforderungsverfolgung bei der PSS-Entwicklung gilt es deshalb zu prüfen, wie die Versionsverwaltung in einer domänenübergreifenden Umgebung aussehen sollte.

Der Bereich *(5) Varianten- und Konfigurationsmanagement* stand bisher innerhalb der Forschung zur Anforderungsverfolgung weniger im Fokus. Wie aus der Analyse hervorgeht, unterstützen zwar einige Methoden das Kriterium, aber bezogen auf PSS, nur im weiteren Sinne. Dabei ist die Modularisierung die Voraussetzung, um als PSS-Anbieter profitabel arbeiten zu können [44]. Es ist davon auszugehen, dass das Anbieten unterschiedlicher Varianten, basierend auf einer modularen Architektur, die Komplexität bei der Anforderungsverfolgung erheblich steigert [46].

Dass die Entwicklung und die Erbringung von PSS oft innerhalb eines *(6) Wertschöpfungsnetzwerks* erfolgt und dies daher ebenfalls von der Anforderungsmethode unterstützt werden sollte, geht aus den untersuchten Publikationen nur unzureichend hervor. Denkbar wäre, Event-based-Traceability unternehmensübergreifend einzusetzen. Bei dem eng damit verwandten Kriterium der *(7) simultanen Entwicklung* zeigt sich wiederum, dass nur diese Methode einen zufriedenstellenden Ansatzpunkt liefert. Diese Forschungslücke der simultanen Entwicklung ist bereits bekannt und gilt als einer der größten Schwachstellen der existierenden Methoden [41].

Bezüglich der *(8) Robustheit gegenüber Unsicherheiten* wird ersichtlich, dass lediglich drei Methoden in gewisser Weise mit Informationslücken umgehen können. Ein PSS-spezifischer Ansatz sollte im Idealfall in der Lage sein, Informationslücken auszugleichen und Unsicherheiten bei der Analyse von Änderungsauswirkungen zu

berücksichtigen. Davon scheint die Anforderungsverfolgung aber weit entfernt zu sein. Bevor allerdings dieses Spezialkriterium angegangen wird, sollten zunächst Lösungen für grundlegendere Probleme erarbeitet werden.

Die untersuchten Methoden beschränken sich meist auf spezielle Phasen der Entwicklung oder sind nur in den Lebenszyklus einer domänenspezifischen Komponente abzubilden. Da jedoch bei PSS unterschiedliche Lebenszyklen von Komponenten in einer integrierten, domänenübergreifenden Lösung aufeinandertreffen, deckt keine der betrachteten Methoden den *(9) kompletten PSS-Lebenszyklus* vollständig ab. Die Methoden müssen für einen Einsatz im PSS-Kontext in dieser Richtung erweitert werden, da die Verantwortung für den gesamten Lebenszyklus zu den elementarsten Anforderungen der PSS-Entwicklung zählt [2]. Grund für diese Lücke ist, dass der Verantwortungsbereich innerhalb des Produktlebenszyklus bei einem klassischen Produkthersteller zumeist deutlich früher als bei einem PSS-Anbieter endet [48].

Das letzte betrachtete Kriterium, *(10) Reduzierung des Aufwands*, erfüllen nur zwei der untersuchten Ansätze vollständig. Sie versuchen einen Kompromiss dahingehend zu finden, ausschließlich die kritischen Anforderungen einzubeziehen und wichtige Anforderungen detaillierter zu verfolgen, um die anfallenden Kosten zu senken. Die geringe Anzahl an unterstützenden Methoden ist erstaunlich, da nach [23, 54] zahlreiche Praktiker den mit der Anforderungsverfolgung verbundenen Aufwand beklagen.

7 Diskussion

Insgesamt gesehen ist keine der Methoden uneingeschränkt für die PSS-Entwicklung geeignet. Gleichzeitig kann aber auch kein Verfahren als besonders nachteilig, bezogen auf PSS, ausgewiesen werden. Schließlich unterstützen sie alle auf ihre Art einen Teilaspekt der Anforderungsverfolgung. Insbesondere weisen einige der Methoden bei den Kriterien (1) Verfolgung der Anforderungsherkunft, (2) Verfolgung der Beziehung zwischen Anforderungen, (3) Verfolgung der Anforderungsumsetzung und (4) Versionsmanagement bereits vielversprechende Ansätze auf. Allerdings muss an dieser Stelle betont werden, dass die betrachteten Methoden nicht direkt miteinander vergleichbar sind, da jede einen etwas anderen Fokus aufweist. Es zeigt sich aber, dass jedes Kriterium durch mindestens eine Methode, zumindest teilweise, erfüllt wird. Somit ergibt sich ein großes Potential für eine gezielte Kombination und Erweiterung der Methoden in Bezug auf die Herausforderungen der PSS-Entwicklung.

Für die Forschung ergeben sich neben der aufwendigen Neuerstellung eines PSS-spezifischen Ansatzes zwei Möglichkeiten: Entweder wird versucht, bestehende Verfahren tiefgreifend zu erweitern oder sie in geeigneter Weise zu kombinieren. Dies wirft die Frage auf, wie bestimmte Verfahren kombiniert werden können, damit sie die Anforderungen von ihrer Quelle bis zu ihrer Umsetzung verfolgen und gleichzeitig Informationen aus späteren Lebenszyklusphasen einbinden, um diese bei der Weiterentwicklung des PSS nutzen zu können. In diesem Zusammenhang ist es sinnvoll, die Anforderungsverfolgung als Prozess mit den Aufgaben (1) Identifikation und Dokumentation der relevanten Informationen und Trace Links, (2) die fortlaufende Aktualisierung dieser und (3) die Nutzung der Informationen und Erkenntnisse dar-

aus, beispielsweise im Änderungsmanagement, zu betrachten. Bezogen auf diese Phasen sollten die untersuchten Methoden in geeigneter Weise kombiniert und erweitert werden. So wäre es denkbar, mit Information Retrieval mögliche Trace Links zu identifizieren und Event-based-Traceability einzusetzen, um diese zu pflegen und Entwickler über sie betreffende Änderungen zu informieren. Inwieweit diese Strategie allerdings umsetzbar ist, sollte in einer weiteren Forschungsarbeit genauer untersucht werden, da eine unstrukturierte, nicht-integrierte, gleichzeitige Verwendung mehrerer Verfahren zu Überlappungen und redundanter Arbeit führen kann.

8 Zusammenfassung und Ausblick

In diesem Beitrag wurde der Frage nachgegangen, inwieweit sich die existierenden domänenspezifischen Anforderungsverfolgungsmethoden für den Einsatz bei PSS eignen. Dieser Analyse lagen zehn Kriterien zugrunde, anhand derer die Anforderungsverfolgungsmethoden im Kontext von PSS bewertet wurden. Bei dieser Bewertung wurde deutlich, dass keine Anforderungsverfolgungsmethode alle der betrachteten Kriterien erfüllt. Dennoch zeigen einige bei den Kriterien (1) Verfolgung der Anforderungsherkunft, (2) Verfolgung der Beziehung zwischen Anforderungen, (3) Verfolgung der Anforderungsumsetzung und (4) Versionsmanagement vielversprechende Ansätze. Insgesamt erscheint es sinnvoll und auch möglich, die Methoden so zu kombinieren und zu erweitern, dass alle geforderten Kriterien hinreichend erfüllt werden.

Um Entwickler von PSS in der Praxis bei der Anforderungsverfolgung zu unterstützen, bedarf es dreier Bausteine: Erstens ist ein Datenmodell erforderlich, das spezifiziert, welche Artefakte bei der Anforderungsverfolgung berücksichtigt werden müssen und welche Arten von Abhängigkeiten, Trace Links, zwischen diesen bestehen. Ein solches Datenmodell könnte beispielsweise als Ontologie realisiert werden, welche die semantischen Beziehungen zwischen verschiedenen Artefakten definiert. Dies sollte jedoch so gestaltet sein, dass es, beispielsweise hinsichtlich des Detaillierungsgrads, für unternehmens- beziehungsweise projektspezifische Zwecke angepasst werden kann. Daneben wird ein Prozessmodell benötigt, das darlegt, welche Aktivitäten in welchen Phasen des Entwicklungsprozesses in Unternehmen stattfinden müssen, um die Trace Links und weitere Informationen, die im Kontext der Anforderungsverfolgung wichtig sind, zu dokumentieren, zu pflegen und zu nutzen. Abgerundet werden sollte dies durch eine Art Methodenbaukasten, der in Abhängigkeit verschiedener Faktoren, wie dem Projektkontext oder der Art des PSS, aufzeigt, welche Methoden bei der Anforderungsverfolgung angewandt werden sollten.

Danksagung

Diese Veröffentlichung entstand im Rahmen des Sonderforschungsbereichs 768 „Zyklenmanagement von Innovationsprozessen – Verzahnte Entwicklung von Leistungsbündeln auf Basis technischer Produkte“. Das Forschungsvorhaben wird gefördert durch die Deutsche Forschungsgemeinschaft (DFG).

Literatur

1. Leimeister, J.M., Glauner, C.: Hybride Produkte – Einordnung und Herausforderungen für die Wirtschaftsinformatik. Wirtschaftsinformatik 50, 248-251 (2008)
2. 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. Journal of Engineering Manufacture 221, 1543-1552 (2007)
3. Tuli, K.R., Kohli, A.K., Bharadwaj, S.G.: Rethinking Customer Solutions: From Product Bundles to Relational Processes. Journal of Marketing 71, 1-17 (2007)
4. Herzfeldt, A., Schermann, M., Krcmar, H.: A Product Service System Lifecycle Model for the IT Service Industry. Multikonferenz der Wirtschaftsinformatik, pp. 53-64, Braunschweig (2012)
5. Berkovich, M., Leimeister, J.M., Krcmar, H.: Requirements Engineering für Product Service Systems – Eine State-of-the-Art-Analyse. Wirtschaftsinformatik 53, 357-370 (2011)
6. Krcmar, H.: Informationsmanagement. Springer Verlag, Heidelberg, Berlin (2010)
7. Böhmann, T., Krcmar, H.: Hybride Produkte: Merkmale und Herausforderungen. In: Bruhn, M., Stauss, B. (eds.) Wertschöpfungsprozesse bei Dienstleistungen: Forum Dienstleistungsmanagement, pp. 239-255. Gabler Verlag, Wiesbaden (2007)
8. Berkovich, M., Mauro, C., Leimeister, J.M., Weyde, F., Krcmar, H.: Towards Cycle-Oriented Requirements Engineering. Multikonferenz der Wirtschaftsinformatik, pp. 963-972, Zürich (2011)
9. Herzfeldt, A., Schermann, M., Krcmar, H.: Towards a Set of Requirements for a Holistic IT Solution Engineering Approach. Australasian Conference on Information Systems, pp. 1-10, Brisbane (2010)
10. Berkovich, M., Esch, S., Mauro, C., Leimeister, J.M., Krcmar, H.: Towards an Artifact Model for Requirements to IT-enabled Product Service Systems. Multikonferenz der Wirtschaftsinformatik, pp. 241-251, Zürich (2011)
11. Langer, S., Lindemann, U.: Managing Cycles in Development Processes – Analysis and Classification of external Context Factors. 17th International Conference on Engineering Design, pp. 539-550, Kalifornien (2009)
12. Gotel, O., Finkelstein, A.: An Analysis of the Requirements Traceability Problem. In: Society, I.C. (ed.) First International Conference on Requirements Engineering, pp. 94-101, Colorado Springs (1994)
13. Ramesh, B., Jarke, M.: Toward Reference Models for Requirements Traceability. IEEE Transactions on Software Engineering 27, 58-93 (2001)
14. Wolfenstetter, T., Goswami, S., Krcmar, H.: Herausforderungen auf dem Weg zu einer zyklengerechten Requirements Traceability für Produkt-Service Systeme. Zyklenmanagement Aktuell 4, 7-9 (2013)
15. Cheng, B.H.C., Atlee, J.M.: Research Directions in Requirements Engineering. Future of Software Engineering, pp. 285-303, Minneapolis (2007)
16. Ebert, C.: Systematisches Requirements Engineering: Anforderungen ermitteln, spezifizieren, analysieren und verwalten. dpunkt.verlag, Heidelberg (2012)
17. Spanoudakis, G., Zisman, A.: Software Traceability: A Roadmap. In: Chang, S.K. (ed.) Handbook of Software Engineering and Knowledge Engineering, pp. 395-428. World Scientific, New Jersey (2005)
18. Pinheiro, F.A.C.: Requirements Traceability. In: Prado Leite, J.C., Doorn, J.D. (eds.) Perspectives on Software Requirements, pp. 91-113. Springer US, Boston (2004)

19. Pohl, K.: Requirements Engineering: Grundlagen, Prinzipien, Techniken. dpunkt.verlag, Heidelberg (2008)
20. Watkins, R., Neal, M.: Why and how of Requirements Tracing. IEEE Software 11, 104-106 (1994)
21. Kotonya, G., Sommerville, I.: Requirements Engineering: Processes and Techniques. John Wiley & Sons, Inc, West Sussex (1998)
22. Kirova, V., Kirby, N., Kothari, D., Childress, G.: Effective Requirements Traceability: Models, Tools, and Practices. Bell Labs Technical Journal 12, 143-157 (2008)
23. Torkar, R., Gorschek, T., Feldt, R., Svahnberg, M., Raja, U.A., Kamran, K.: Requirements Traceability: A systematic Review and Industry Case Study. International Journal of Software Engineering and Knowledge Engineering 22, 385-433 (2012)
24. Webster, J., Watson, R.T.: Analyzing the Past to prepare for the Future: Writing a Literature Review. MIS Quarterly 26, 13-23 (2002)
25. Leimeister, J.M.: Dienstleistungsengineering und -management. Springer Verlag, Berlin, Heidelberg (2012)
26. Antoniol, G., Canfora, G., Casazza, G., Lucia, A., Merlo, E.: Recovering Traceability Links between Code and Documentation. IEEE Transactions on Software Engineering 28, 970-983 (2002)
27. Cleland-Huang, J., Chang, C.K., Sethi, G., Javvaji, K., Hu, H., Xia, J.: Automating speculative Queries through Event-based Requirements Traceability. IEEE Joint International Conference on Requirements Engineering, pp. 289-296, Essen (2002)
28. Cleland-Huang, J.: Toward improved Traceability of non-functional Requirements. 3rd International Workshop on Traceability in emerging Forms of Software Engineering, pp. 14-19, Long Beach (2005)
29. Zisman, A., Spanoudakis, G., Perez-Minana, E., Krause, P.: Towards a Traceability Approach for Product Families Requirements. 3rd ICSE Workshop on Software Product Lines: Economics, Architectures, and Implications, pp. 19-25, Orlando (2002)
30. Heindl, M., Biffl, S.: A Case Study on Value-based Requirements Tracing. 10th European Software Engineering Conference held jointly with 13th ACM SIGSOFT International Symposium on Foundations of Software Engineering, pp. 60-69, Lissabon (2005)
31. Riebisch, M.: Supporting Evolutionary Development by Feature Models and Traceability Links. 11th IEEE International Conference and Workshop on the Engineering of Computer-Based Systems, pp. 370-377, Brno (2004)
32. Ahn, S., Chong, K.: A Feature-oriented Requirements Tracing Method: A Study of Cost-Benefit Analysis. International Conference on Hybrid Information Technology, pp. 611-616, Washington DC (2006)
33. Egyed, A.: A Scenario-Driven Approach to Traceability. 23rd International Conference on Software Engineering, pp. 123-132, Toronto (2001)
34. Maletic, J.I., Munson, E.V., Marcus, A., Nguyen, T.N.: Using a Hypertext Model for Traceability Link Conformance Analysis. 3rd International Workshop on Traceability in Emerging Forms of Software Engineering, pp. 47-54, Monteray Bay (2003)
35. Cleland-Huang, J., Settimi, R., BenKhadra, O., Berezhanskaya, E., Christina, S.: Goal-centric Traceability for Managing non-functional Requirements. 27th International Conference on Software Engineering, pp. 362-371, St. Louis (2005)
36. Ravichandar, R., Arthur, J.D., Pérez-Quiñones, M.: Pre-Requirement Specification Traceability: Bridging the Complexity Gap through Capabilities. International Symposium on Grand Challenges in Traceability, Lexington (2007)
37. Gotel, O., Finkelstein, A.: Contribution Structures [Requirements Artifacts]. Second IEEE International Symposium on Requirements Engineering, pp. 100-107, York (1995)

38. Akao, Y.: Quality Function Deployment: Integrating Customer Requirements into Product Design. Productivity Press, New York (1990)
39. Teng, S.H.G., Ho, S.Y.M.: Failure Mode and Effects Analysis: An integrated Approach for Product Design and Process Control. International Journal of Quality & Reliability Management 13, 8-26 (1996)
40. Maeder, P., Riebisch, M., Philippow, I.: Traceability for Managing Evolutionary Change. 15th International Conference on Software Engineering and Data Engineering, pp. 1-8, Los Angeles (2006)
41. Winkler, S., Pilgrim, J.: A Survey of Traceability in Requirements Engineering and Model-driven Development. Software and Systems Modeling (SoSyM) 9, 529-565 (2010)
42. Sahraoui, A.: Requirements Traceability Issues: Generic Model, Methodology and Formal Basis. International Journal of Information Technology & Decision Making 4, 59-80 (2005)
43. Galbraith, J.R.: Organizing to Deliver Solutions. Organizational Dynamics 31, 194-207 (2002)
44. Sawhney, M.: Going Beyond the Product: Defining, Designing and Delivering Customer Solutions. In: Lusch, R.F., Vargo, S.L. (eds.) Toward a Service-Dominant Logic of Marketing: Dialog, Debate, and Directions, pp. 65-80. M. E. Sharpe, Armonk (2006)
45. Böhmann, T., Langer, P., Schermann, M.: Systematische Überführung von kundenspezifischen IT-Lösungen in integrierte Produkt-Dienstleistungsbausteine mit der SCORE-Methode. Wirtschaftsinformatik 50, 196-207 (2008)
46. Mohan, K., Ramesh, B.: Change Management Patterns in Software Product Lines. Communications of the ACM 49, 68-72 (2006)
47. Gebauer, H., Paiola, M., Saccani, N.: Characterizing Service Networks for Moving from Products to Solutions. Industrial Marketing Management 42, 31-46 (2012)
48. Reichwald, R., Mayer, D., Bonnemeier, S.: Risikomanagement bei hybrider Wertschöpfung. In: Schäfer, K., Burghof, H.-P., Johanning, L., Wagner, H.F., Rodt, S. (eds.) Risikomanagement und kaptialmarktorientierte Finanzierung, pp. 209-228. M. Knappe, Frankfurt am Main (2009)
49. Sharafi, A., Wolfenstetter, T., Wolf, P., Krcmar, H.: Comparing Product Development Models to identify Process Coverage and current Gaps: A Literature Review. IEEE International Conference on Industrial Engineering and Engineering Management, pp. 1732-1736, Macao (2010)
50. Berkovich, M., Leimeister, J.M., Krcmar, H.: Ein Bezugsrahmen für Requirements Engineering hybrider Produkte. Multikonferenz der Wirtschaftsinformatik, pp. 1-12, Göttingen (2010)
51. Ebert, C., De Man, J.: Requirements Uncertainty: Influencing Factors and concrete Improvements. 27th International Conference on Software Engineering, pp. 553-560, St. Louis (2005)
52. von Knethen, A., Paech, B., Kiedaisch, F., Houdek, F.: Systematic Requirements Recycling through Abstraction and Traceability. IEEE Joint International Conference on Requirements Engineering, pp. 273-281, Essen (2002)
53. Jarke, M.: Requirements Tracing. Communications of the ACM 41, 32-36 (1998)
54. Egyed, A., Biffl, S., Heindl, M., Grünbacher, P.: A value-based Approach for Understanding Cost-Benefit Trade-offs during automated Software Traceability. 3rd International Workshop on Traceability in emerging Forms of Software Engineering, pp. 2-7, Long Beach (2005)