Eine Referenzstrukturierung zur modellbasierten Kontextanalyse im Requirements Engineering softwareintensiver eingebetteter Systeme

Thorsten Weyer
Software Systems Engineering
Universität Duisburg-Essen
Schützenbahn 70, 45117 Essen
thorsten.weyer@sse.uni-due.de

Klaus Pohl
Software Systems Engineering
Universität Duisburg-Essen
Schützenbahn 70, 45117 Essen
klaus.pohl@sse.uni-due.de

Abstract: Dem Requirements Engineering (RE) kommt im Entwicklungsprozess die Aufgabe zu, die Anforderungen an das zu entwickelnde System zu spezifizieren. Der Ursprung von Anforderungen liegt in der Umgebung des zu entwickelnden Systems – dem Systemkontext. Um sicherzustellen, dass das entwickelte System den Erwartungen, Bedingungen und Reglementierungen seiner Umgebung genügt, muss der Systemkontext in den Anforderungen vollständig und fehlerfrei berücksichtigt werden. Dies setzt voraus, dass der Kontext des zu entwickelnden Systems in seiner Gesamtheit richtig erfasst wird. Die Erfassung des Systemkontexts im RE ist Gegenstand der Kontextanalyse. Das RE in der Entwicklung softwareintensiver eingebetteter Systeme sieht sich immer häufiger der Situation gegenüber, den zunehmend komplexer werdenden Kontext solcher Systeme unter bisweilen rigidesten Budget- und Zeitvorgaben vollständig und korrekt erfassen zu müssen, um die Anforderungen an das zu entwickelnde System spezifizieren zu können. In diesem Beitrag wird eine Referenzstrukturierung für den Systemkontext softwareintensiver eingebetteter Systeme vorgeschlagen, die ein Rahmenwerk zur strukturierten Erfassung und Dokumentation des Systemkontexts definiert und eine Komplexitätsreduktion der Kontextbetrachtung in der modellbasierten Kontextanalyse verspricht.

1 Einleitung

Softwareintensive eingebettete Systeme sind Systeme, bei denen die Software in eine technische und physikalische Umgebung eingebettet ist und wesentliche funktionale oder qualitative Eigenschaften des Systems softwareseitig realisiert werden. Im Entwicklungsprozess eines solchen Systems kommt dem RE die Aufgabe zu, die Anforderungen an das zu entwickelnde System zu spezifizieren.

1.1 Systemkontext und Kontextanalyse im RE

Anforderungen an ein zu entwickelndes Systems haben ihren Ursprung im Systemkontext (vgl. [JP93][ZJ97]). Der Systemkontext eines softwareintensiven eingebetteten Sys-

tems umfasst alle Aspekte in der Umgebung des Systems, die mit dem zu entwickelnden System in Beziehung stehen, etwa Personen und Gruppen, die ein Interesse an dem zu entwickelnden System haben oder technische und physikalische Prozesse, die mit dem zu entwickelnden System im Betrieb in einem Wirkzusammenhang stehen (vgl. [Gu95]). Die spezifizierten Anforderungen legen funktionale und qualitative Merkmale fest, die das System seiner Umgebung gegenüber aufweisen muss, um den Vorgaben, Bedingungen und Einschränkungen des Systemkontexts zu genügen.

In vielen Fällen ist ein Systemversagen, dessen Ursache vermeintlich in fehlerhaften oder unvollständigen Anforderungen liegt[1] tatsächlich auf eine unzureichende Analyse des Systemkontexts[2] zurückzuführen (vgl. z.B. [We75][LK95]). So wird in [HF00] von der Analyse von Fehlerfällen eines Avionic-Systems berichtet, bei der die Autoren feststellten, dass annähernd für die Hälfte aller aufgetretenen Fehlerfälle eine unvollständige oder fehlerhafte Erfassung des Systemkontexts ursächlich war. In der Literatur sind zahlreiche weitere Beispiele dafür zu finden, dass eine unzureichende Berücksichtigung des Systemkontexts in den Anforderungen eine häufige Ursache für Systemversagen im Betrieb ist, das mitunter katastrophale Konsequenz hat (vgl. etwa [Ja95][oV95][Le97]).

Die Erfassung des Systemkontexts im RE ist Aufgabe der Kontextanalyse. Im Rahmen der Kontextanalyse werden *Systemgrenzen* und *Kontextgrenzen* ermittelt und der Systemkontext untersucht, um *Informationen über den Kontext* des zu entwickelnden Systems zu gewinnen (vgl. z.B. [MP84]). Die Systemgrenzen separieren das zu entwickelnde System von dessen Umgebung und legen den Gegenstand der Entwicklung fest. Die Kontextgrenzen trennen den relevanten Teil der Umgebung des Systems (den Systemkontext) vom irrelevanten Teil. *Kontextinformationen* beziehen sich entweder darauf, welche *Kontextaspekte* im Systemkontext vertreten sind (z.B. bestimmte Gruppen, Systeme und physikalische Prozesse) oder auf relevante Eigenschaften und Beziehungen dieser Kontextaspekte (z.B. Intentionen einer Gruppe). Auf Grundlage der erarbeiteten Kontextinformationen werden im RE die Anforderungen an das zu entwickelnde System möglichst vollständig, präzise, widerspruchsfrei, in zweckmäßiger Detaillierung sowie mit allen relevanten Personen und Personengruppen abgestimmt spezifiziert. Anforderungen mit diesen Eigenschaften werden im Weiteren als *Systemanforderungen* bezeichnet. Systemanforderungen dienen als Grundlage für Architekturentwurf, Implementierung und Test und können z.B. in einem Lastenheft eine Referenz und Vertragsgrundlage für die Fremdvergabe der Entwicklung eines (Teil-)Systems darstellen.

1.2 Problemstellung

Um das „richtige“[3] System zu konstruieren, muss der Kontext des zu entwickelnden Systems im RE adäquat berücksichtigt werden. Voraussetzung dafür ist, dass der Sys-

[1] Kontext ist korrekt erfasst; Anforderungen sind allerdings fehlerhaft oder unvollständig spezifiziert
[2] Kontext ist fehlerhaft erfasst; Anforderungen sind im Sinne dieses Kontext „korrekt“ spezifiziert
[3] Im Sinne von [Bo84] „Am I building the right system?” im Kontrast zu „Am I building the system right?”

temkontext im Rahmen der Kontextanalyse vollständig und korrekt verstanden wird. Softwareintensive eingebettete Systeme müssen komplexe Wirkzusammenhänge und Bedingungen in der technischen und physikalischen Umgebung abbilden und dabei häufig mit anderen Systemen interagieren, um eine übergeordnete Aufgabe zu erfüllen („System-of-Systems"). Hinzu kommt, dass unterschiedlichste Personen und Gruppen ein Interesse an dem zu entwickelnden System haben, welches abgestimmt in den spezifizierten Anforderungen berücksichtigt werden muss. Der Kontext softwareintensiver eingebetteter Systeme ist in den meisten Fällen sehr komplex und zählt oftmals zu den am wenigsten verstandenen Aspekten solcher Systeme (vgl. [Za82][Br06][CA07]).

Ansätze, die die Kontextanalyse im RE softwareintensiver eingebetteter Systeme unterstützen (z.B. strukturierte Systemanalyseansätze für eingebettete Systeme [HP88] oder adaptierte Use-Case-Ansätze [PIÖ05]), berücksichtigen die tatsächliche Reichweite des Kontexts eines zu entwickelnden softwareintensiven eingebetteten Systems und die möglichen Beziehungen im Systemkontext nur teilweise. Dies birgt die Gefahr, dass wichtige Kontextaspekte oder Beziehungen zwischen Kontextaspekten nicht oder nur unzureichend in den Anforderungen abgebildet werden. So betrachten einschlägige Ansätze z.B. keine Wirkzusammenhänge zwischen Systemen in der Umgebung, die Auswirkungen auf das zu entwickelnde System haben können.

Um die Kontextanalyse im RE softwareintensiver eingebetteter Systeme weitreichender zu unterstützen, werden Ansätze benötigt, die die spezifischen Eigenschaften des Kontexts softwareintensiver eingebetteter Systeme berücksichtigen und die zunehmende Komplexität des Systemkontexts solcher Systeme beherrschbar machen.

1.3 Zielsetzung

In diesem Beitrag wird eine Referenzstrukturierung für den Systemkontext in der modellbasierten Kontextanalyse im RE vorgestellt, die darauf abzielt, den Kontext eines zu entwickelnden softwareintensiven eingebetteten Systems möglichst vollständig und strukturiert zu erfassen und dabei die Komplexität der Kontext*betrachtung* zu reduzieren. Ein zentrales Merkmal dieses Ansatzes ist die *Modellbasierung*, die sich in zwei Ausprägungen manifestiert: Zum einen ist die Struktur des zu analysierenden Systemkontexts und die Informationsstruktur der zu erarbeitenden Kontextinformationen durch Informationsmodelle definiert. Zum anderen werden die erarbeiteten Kontextinformationen in Form konzeptueller Modelle dokumentiert. Die vorgeschlagene Referenzstrukturierung basiert auf drei komplementären Konzepten:

1. *Drei Kontextsichten* zur grobgranularen Strukturierung des Kontexts zu entwickelnder softwareintensiver eingebetteter Systeme.
2. *Drei Informationsmodelle zur Strukturierung der Kontextsichten*, die für jede der drei Kontextsichten jeweils eine feingranulare Referenzstruktur definieren, durch die Informationen über den Systemkontext in der jeweiligen Kontextsicht strukturiert analysiert werden können.

3. *Ein Artefaktmodell* (auch: Produktmodell) für Kontextinformationen, das für die einzelnen Kontextsichten, auf Basis des jeweiligen Informationsmodells einer Kontextsicht, Typen von Artefakten zur Erarbeitung und Dokumentation detaillierter Kontextinformationen festlegt sowie geeignete Repräsentationsformen für einzelne Artefakttypen empfiehlt.

Dieses Papier konzentriert sich auf die beiden erstgenannten Konzepte der Referenzstrukturierung (d.h. Kontextsichten und zugehörige Informationsmodelle). Im Anschluss an die Einleitung wird in Abschnitt 2 der in diesem Beitrag vorgestellte Ansatz von anderen Arbeiten auf dem Gebiet der Kontextanalyse und Kontextmodellierung abgegrenzt. In Abschnitt 3 werden die drei Kontextsichten dieses Ansatzes erläutert. Abschnitt 4 stellt das Informationsmodell einer ausgewählten Kontextsicht (der operationellen Umgebung) im Detail vor. Abschnitt 5 zeigt einen Ausschnitt aus einem Anwendungsbeispiel und berichtet von Erkenntnissen einer ersten industriellen Erprobung des Ansatzes. Abschnitt 6 fasst die Inhalte dieses Papiers zusammen und gibt einen Ausblick auf künftige Forschungsaktivitäten.

2 Stand der Forschung: Kontextanalyse und Kontextmodellierung

Auf Grundlage der strukturierten Systemanalyseansätze (z.B. [RS77][De78]) der 70er Jahre wurden verschiedene spezialisierte Ansätze zur Systemanalyse eingebetteter Systeme vorgeschlagen (z.B. [WM85][HP88]). Diese Ansätze berücksichtigen den Systemkontext nur teilweise, da in den entsprechenden Kontextdiagrammen nur die unmittelbaren Beziehungen zwischen dem System und Aspekten im Kontext modelliert werden. Bezogen auf den Systemkontext betrachten Use-Case-Diagramme [Ja92] ausschließlich die Kommunikationsbeziehungen zwischen dem System und Akteuren im Systemkontext. Durch komplementäre Use-Case-Spezifikationen (z.B. [Ja92][Co01]) können diese Beziehungen zwar konkretisiert werden, allerdings beschränkt sich die Detaillierung auf die direkten Abhängigkeiten zwischen Akteuren und System. Erweiterungen von Use Cases für eingebettete Systeme (z.B. [DPB03][PIÖ05]) vergrößern die Analysereichweite von Use Cases jeweils nur um einzelne spezifische Aspekte des Systemkontexts eingebetteter Systeme (z.B. unmittelbar zur kontrollierende physikalische Prozesse in der Umgebung).

Zielbasierte Ansätze im RE (z.B. [La91][Yu93]) unterstützen die systematische Analyse und Dokumentation des Systemkontexts im Hinblick auf Kontextinformationen, die Intentionen von Personen oder Gruppen dokumentieren. Mittels szenariobasierten Ansätzen (z.B. [Po95]) können Interaktionen zwischen dem Kontext und dem zu entwickelnden System, aber auch Interaktionen im Systemkontext, analysiert und dokumentiert werden. Andere Arbeiten betrachten grobgranulare Heuristiken zur Kontextstrukturierung, wie z.B. Viewpoint-Ansätze [Mu79][JP93] oder „Problem Frames“ [Ja95][Ja01], die eine schrittweise Zerlegung des Systemkontexts in kleinere Problembereiche unterstützen. Einige Beiträge fokussieren auf die detaillierte Analyse und Modellierung spezifischer Teilaspekte des Systemkontexts. Hierzu gehören z.B. das „Vier-Variabelen-Modell“ [PM95], welches eine formale Analyse und Dokumentation der

Beziehungen zwischen physikalischer und technischer Umgebung und dem betrachteten System ermöglicht oder Arbeiten zum „Requirements Monitoring“ und zu kontextadaptiven Systemen (z.B. [FF95][SFS06]), die Teilaspekte der Umgebung detailliert modellieren, um relevante Kontextveränderungen identifizieren und erforderliche Systemadaptionen definieren zu können. In [Br07] wird ein Artefaktmodell für das RE softwareintensiver eingebetteter Systeme vorgeschlagen, das neben Informationen zu Anforderungen und Entwurf auch Kontextinformationen berücksichtigt (z.B. Unternehmensziele). Der Fokus dieses Ansatzes liegt allerdings weniger auf einer dezidierten Definition des im RE zu berücksichtigenden Systemkontexts sondern auf Systemanforderungen und Architekturentwurf und deren Beziehungen.

Jede der angeführten Arbeiten liefert wertvolle Beiträge zur Erarbeitung eines umfassenden Kontextverständnisses im RE softwareintensiver eingebetteter Systeme. Allerdings fokussieren diese Arbeiten entweder auf grobgranulare Kontextstrukturierungen oder auf feingranulare Betrachtungen spezifischer Ausschnitte des Systemkontexts. Um die Spezifikation der Anforderungen softwareintensive eingebetteter Systeme umfassend zu unterstützten, werden Ansätze zur Kontextanalyse im RE benötigt, die den Systemkontext in seiner Gesamtheit berücksichtigen und dabei *sowohl grobgranulare als auch feingranulare Strukturierungen* des Systemkontexts *in einem integrierten Modell* zusammenführen. Ein solcher Ansatz ist Gegenstand dieses Papiers.

3 Grobgranulare Strukturierung durch drei Kontextsichten

Zur grobgranularen Strukturierung des Systemkontexts unterscheidet der vorgestellte Ansatz die drei Kontextsichten „Interessenvertreter“, „operationelle Umgebung“ und „Systemdomäne“. Die Kontextsichten sind dabei in Bezug auf die in den einzelnen Sichten enthaltenen Kontextaspekte nicht disjunkt, d.h. es existieren Kontextaspekte (z.B. Gruppe oder Systeme), die in mehr als einer Kontextsicht vertreten sind. Solche „gemeinsamen“ Kontextaspekte sind Grundlage für die Integration der drei Kontextsichten.

3.1 Kontextsicht „Interessenvertreter“

Die Kontextsicht „Interessenvertreter“ umfasst Individuen und Personengruppen, die ein Interesse an dem zu entwickelnden System haben. *Interessenvertreter* sind entweder in den Entwicklungsprozess involviert oder haben ein Interesse an dem zu entwickelnden System aufgrund dessen, dass deren Handlungen direkt oder indirekt von dem zu entwickelnden System beeinflusst werden bzw. die Handlungen der Interessenvertreter direkt oder indirekt das spätere System beeinflussen (vgl. [Po97]). Beispiele von Interessenvertretern eines zu entwickelnden softwareintensiven eingebetteten Systems sind spätere Nutzer des (Gesamt-)Systems, der Gesetzgeber, Auftragnehmer, Auftraggeber, die Unternehmensleitung, Techniker, Wartungspersonal oder Standardisierungsinstitutionen. Neben Informationen darüber, welche Personen und Personengruppen ein Interesse an dem zu entwickelnden System haben und welche Beziehungen zwischen Interessenver-

tretern bestehen, dokumentieren Kontextinformationen den Gegenstand des Interesses genauer. Das Interesse eines Individuum oder einer Gruppe an einem zu entwickelnden softwareintensiven eingebetteten System wird durch *Intentionen*, *Wünsche*, *Bedingungen* und *Einschränkungen* artikuliert und durch Kontextinformationen dokumentiert (z.B. in Form von *Zielen*, *Szenarien* oder *Rahmenbedingungen* der Interessenvertreter). Beispiele für Kontextinformationen dieser Kontextsicht sind strategische Vorgaben der Unternehmensleitung, Intentionen späterer Nutzer des Systems oder normative Vorgaben des Gesetzgebers. Die Kontextsicht „Interessenvertreter" kann auch Informationen über die Interessen von Individuen und Gruppen hinsichtlich eines anderen Systems (z.B. eines abzulösenden Altsystems) umfassen, wenn das Interesse an diesem System unter Umständen Konsequenzen für das zu entwickelnde System hat bzw. wenn dieses Interesse möglicherweise von dem zu entwickelnden System beeinträchtigt wird.

3.2 Kontextsicht „operationelle Umgebung"

Die Kontextsicht „operationelle Umgebung" umfasst andere Systeme (adjazente Systeme), physikalische und technische Prozesse und zugehörige Ereignisse sowie Individuen oder Gruppen *in der Betriebsumgebung* des Systems. *Adjazente Systeme* stehen mit dem zu entwickelnden System in einer direkten oder indirekten Nutzungsbeziehung und stellen Anforderungen an das zu entwickelnde System, da sie dieses System im Betrieb nutzen oder von diesem genutzt werden sollen, um etwa technische Prozesse in der Umgebung zu steuern (vgl. [RR99]). Auch *Vorgaben hinsichtlich der technischen Kopplung* zwischen dem zu entwickelnden System und adjazenten Systemen können Ursprung von Anforderungen sein (z.B. wenn bestimmte Bus-Standards bzw. spezifische Schnittstellen- und Bussignale vorgegeben sind). Softwareintensive eingebettete Systeme müssen für gewöhnlich *physikalische Größen* und deren Ausprägungen in der Umgebung messen und *physikalische oder technische Prozesse* in der Umgebung des Systems steuern (vgl. [PM95]). Beispielsweise könnte ein Lichtsensor in einem Fahrzeug die physikalische Größe „Umgebungshelligkeit" messen und beim Unterschreiten eines definierten Helligkeitswertes (z.B. 1500 LUX) die Außenbeleuchtung des Fahrzeuges aktivieren. *Ereignisse* innerhalb der Kontextsicht „operationelle Umgebung" wirken auf das zu entwickelnde System bzw. rufen eine Reaktion dieses Systems hervor. Die Kontextsicht „operationelle Umgebung" schließt auch *Personen(-gruppen)* ein, die mit dem zu entwickelnden System direkt oder indirekt interagieren, indem sie z.B. das System im Betrieb nutzen, konfigurieren oder warten (→ Integration mit der Kontextsicht „Interessenvertreter). *Annahmen*, *Bedingungen* oder *Einschränkungen* der Betriebsumgebung des Systems sind ebenfalls Aspekte dieser Kontextsicht (z.B. gesetzliche Reglementierungen in Bezug auf die Ausleuchtung der Fahrbahn durch die Scheinwerfer eines Fahrzeuges). In Abschnitt 4 wird ein Informationsmodell vorgestellt, das eine generische Strukturierung für die Kontextsicht „operationelle Umgebung" definiert.

3.3 Kontextsicht „Systemdomäne“

Der Ursprung von Anforderungen an ein softwareintensives eingebettetes System kann nicht nur in den Kontextaspekten der spezifischen Umgebung des zu entwickelnden Systems liegen, sondern auch in allgemeinen Eigenschaften, Bedingungen und Einschränkungen, die *für alle Systeme eines Systemtypus* gelten (vgl. [LK95]). Die Kontextsicht „Systemdomäne“ umfasst Kontextaspekte und zugehörige Kontextinformationen sowie Systemmerkmale (Funktionen und Qualitäten), die für alle Systeme eines Systemtypus gelten (z.B. jedes Steuergerät für Wischermotoren besitzt die Funktion ‚Tippwischen’). Beispiele für Systemtypen softwareintensiver eingebetteter Systeme sind (aktive) Sicherheitssysteme in Fahrzeugen, Transportsteuerungssysteme im Schienennetz, Kollisionswarnsysteme in Flugzeugen, radiologische Systeme in der Medizintechnik oder Steuerungssysteme von Produktionsanlagen im Maschinen- und Anlagenbau. Ein softwareintensives eingebettetes System kann dabei mehrfach „typisiert“ sein. Kontextinformationen in der Kontextsicht „Systemdomäne“ können für gewöhnlich mindestens einer der beiden Kontextsichten „Interessenvertreter“ oder „operationelle Umgebung“ zugeordnet werden, da Aspekte, die für alle Systeme der betrachteten Systemtypen gelten, auch auf das entwickelnde System bzw. auf dessen Systemkontext bezogen werden können. Die explizite Berücksichtigung der Kontextsicht „Systemdomäne“ ermöglicht es allerdings, selektiv Kontextinformationen und Systemmerkmale zu betrachten, die für alle Systeme eines Systemtyps gelten, wodurch der Kontext eines Systems über dessen gesamten Lebenszyklus hinweg differenziert beobachtet und Veränderungen im Systemkontext systematisch identifiziert werden können. Außerdem können die Informationen im Systemkontext „Systemdomäne“ z.B. an Auftragnehmer kommuniziert und in anderen Entwicklungsprojekten wiederverwendet werden. Neben generischen Kontextinformationen und gemeinsamen Eigenschaften von Systemen eines bestimmten Systemtyps können in der Kontextsicht „Systemdomäne“ auch Variabilitäten der Systemdomäne in Bezug auf den Systemkontext oder bezüglich spezifischer Eigenschaften dieses Systemtypus explizit gemacht werden.

3.4 Integration der Kontextsichten

Abbildung 1 zeigt die zuvor erläuterten Sichten auf den Kontext eines zu entwickelnden softwareintensiven eingebetteten Systems. Wie in der Abbildung illustriert, sind die Kontextinformationen der drei Kontextsichten nicht disjunkt, d.h. es existieren Kontextaspekte (z.B. Systeme im Kontext), die in mehr als einer Kontextsicht vertreten sind und die dadurch auch eine Integration von Kontextsichten ermöglichen.

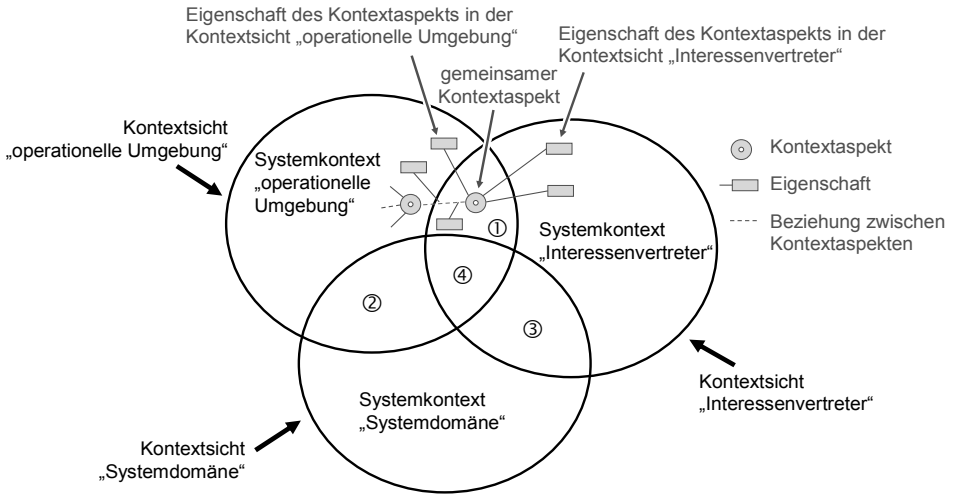

Abbildung 1. Integration der drei Kontextsichten

In der Schnittmenge der beiden Kontextsichten „Interessenvertreter“ und „operationelle Umgebung“ (① in Abbildung 1) befinden sich Interessenvertreter, die das zu entwickelnde System im Betrieb nutzen, konfigurieren, warten oder anderweitig mit dem System im Betrieb in Wechselwirkung stehen, z.B. der Interessenvertreter „Fahrzeugführer“, der mit einem zu entwickelnden Fahrassistenzsystems im Betrieb interagiert. Diese Schnittmenge umfasst darüber hinaus adjazente Systeme der technischen Umgebung, die Gegenstand relevanter Interessen von Individuen oder Gruppen der Kontextsicht „Interessenvertreter“ in Bezug auf das zu entwickelnde System sind. Die Integration der Kontextsichten „operationelle Umgebung“ und „Systemdomäne“ (② in Abbildung 1) ist durch physikalische Größen in der Umgebung und allgemeine Kontextinformationen gegeben, die für alle Systeme des Systemtypus gelten, z.B. Niederschlagsklassen bei der Entwicklung eines Steuergerätes für Wischermotoren. Zu den integrativen Elementen dieser beiden Kontextsichten gehören auch adjazente Systeme, die allgemein mit Systemen des betrachteten Systemtyps im Betrieb in einer Nutzungsbeziehung stehen, z.B. ein Wartungssystem hinsichtlich eines zu entwickelndes Motorsteuerungsgeräts. Die Schnittmenge der beiden Kontextsichten „Interessenvertreter“ und „Systemdomäne“ (③ in Abbildung 1) umfasst Aussagen über Interessenvertreter von Systemen des betrachteten Systemtyps, wie z.B. generelle Aussagen über Organisationen, die relevante Reglementierungen für die Entwicklung von Systemen dieses Typus definieren. Im Schnittbereich der drei Kontextsichten „Interessenvertreter“, „operationelle Umgebung“ und „Systemdomäne“ (④ in Abbildung 1) sind Aussagen über die Nutzung von Systemen durch Individuen oder Gruppen einzuordnen, die für alle Systeme des betrachteten Systemtypus gelten. Hierzu gehören etwa allgemeine Aussagen über Nutzungsgruppen von Systemen des betrachteten Typus.

Kontextaspekte werden durch deren Eigenschaften und durch die Beziehungen zu anderen Kontextaspekten genauer bestimmt. Bei Kontextaspekten, die in mehreren Kontextsichten enthalten sind, sind in den verschiedenen Sichten unterschiedliche Eigenschaften bzw. Beziehungen dieses Kontextaspektes von Bedeutung. Im oberen Teil von

Abbildung 1 wird dieser Sachverhalt exemplarisch angedeutet. Zum Beispiel kann bei der Entwicklung eines Fahrassistenzsystems der Kontextaspekt „Fahrzeugführer“ in der Kontextsicht „Interessenvertreter“ die Intentionen des Fahrzeugführers als Eigenschaft umfassen (etwa das Ziel „Vermeidung von Auffahrunfällen“). In der Kontextsicht „operationelle Umgebung“ ist der Kontextaspekt „Fahrzeugführer“ im Hinblick auf die Nutzung des Systems im Betrieb relevant. So zählen z.B. erwartete Interaktionen zwischen dem Fahrzeugführer und dem zu entwickelnden System oder relevante Wechselwirkungen mit einem adjazenten System zu den spezifischen Eigenschaften und Beziehungen des Kontextaspekts „Fahrzeugführer“ in der Kontextsicht „operationelle Umgebung“.

4 Informationsmodell der Kontextsicht „operationelle Umgebung“

Komplementär zur grobgranularen Strukturierung des Systemkontexts werden die drei Kontextsichten jeweils durch ein Informationsmodell detailliert, das eine generische Referenzstruktur für die jeweilige Kontextsicht definiert. Diese Informationsmodelle sind das zweite komplementäre Konzept der vorgestellten Referenzstrukturierung und ermöglichen es, den Systemkontext in der jeweiligen Sicht selektiv zu analysieren und zu dokumentieren. Im Folgenden wird das Informationsmodell einer ausgewählten Kontextsicht (der operationellen Umgebung) vorgestellt. Abbildung 2 zeigt das Informationsmodell der Kontextsicht „operationelle Umgebung“, das in Form eines Klassenmodells der Unified Modeling Language (UML) [OMG07] modelliert ist.

In der Kontextsicht „operationelle Umgebung“ wird die technische und physikalische Systemabgrenzung vorgenommen. Dabei gehören die Sensoren und Aktuatoren des Systems entweder zum Gegenstand der Entwicklung oder sind (teilweise) bereits vorgegeben und damit Aspekte der operationellen Umgebung des zu entwickelnden Systems. Die beiden unterschiedlichen Formen der Systemabgrenzung gehen mit der in [AM95] vorgeschlagenen Unterscheidung zwischen den Systemklassen „purposeful system“ und „reactive system“ einher. Die Art der Systemabgrenzung hat dabei weitreichende Auswirkungen auf das Gestaltungsspektrum und den Entwicklungsprozess des Systems, z.B. in Bezug auf die Reichweite, in welcher Innovationen möglich sind. Das in Abbildung 2 gezeigte Informationsmodell definiert eine generische Referenzstruktur für die Systemklasse „purposeful system“. Modellelemente zur Erweiterung des Informationsmodells für die Systemklasse „reactive system“ (d.h. Sensoren, Aktuatoren sowie deren Kopplung und Beziehungen) sind in Abbildung 2 gestrichelt dargestellt.

Innerhalb der Kontextsicht „operationelle Umgebung“ können drei Teilbereiche des Systemkontexts unterschieden werden: *technische Umgebung* (❶ in Abbildung 2), *physikalische Umgebung* (❷ in Abbildung 2) sowie *operationelle Vorgaben und Rollen* (❸ und ❹ in Abbildung 2).

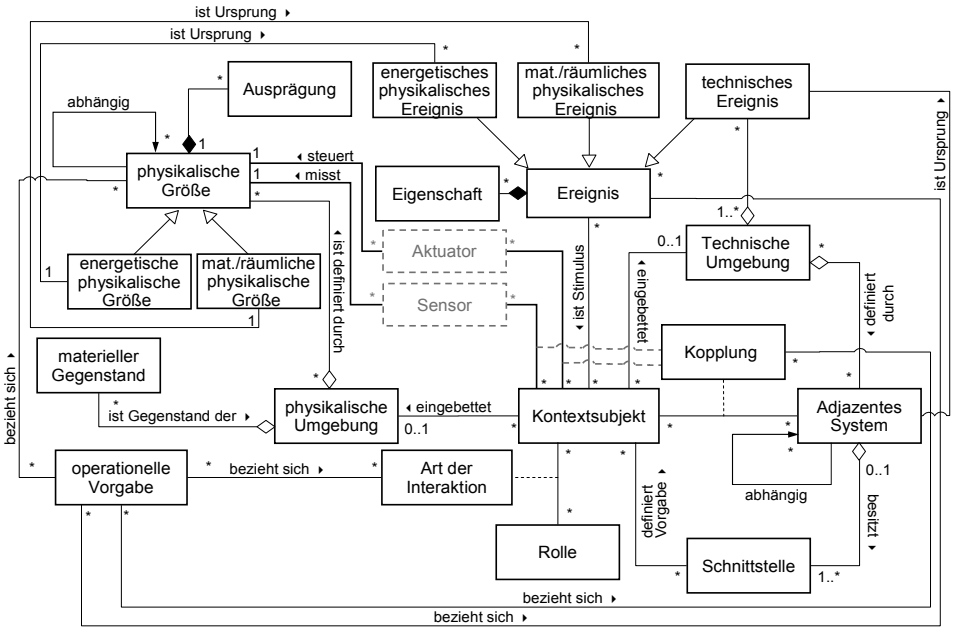

Abbildung 2. Informationsmodell für die Kontextsicht „operationelle Umgebung“

4.1 Technische Umgebung

Das zu entwickelnde System ist in eine technische Umgebung eingebettet (❶ in Abbildung 2). Die technische Umgebung umfasst externe Systeme, die mit dem zu entwickelnden System in einer direkten oder indirekten Nutzungsbeziehung stehen (*adjazente Systeme*). Adjazente Systeme stellen Anforderungen, da sie das zu entwickelnde System im Betrieb nutzen oder von diesem genutzt werden sollen, um z.B. physikalische Prozesse in deren Umgebung zu kontrollieren (vgl. [RR99]). Auch Vorgaben bezüglich der technischen *Kopplung* zwischen dem zu entwickelnden System und einem *adjazenten System* können der Ursprung spezifische Anforderungen sein (z.B. wenn zur Kopplung ein bestimmter Bus-Standard bzw. spezifische Bus- und Schnittstellensignale vorgegeben sind). Adjazente Systeme besitzen *Schnittstellen*, die sowohl Operationen enthalten können, die von dem zu entwickelnden System gefordert werden („Requires“-Schnittstelle) als auch Operationen, die das adjazente System dem zu entwickelnden System zur Verfügung stellen soll („Provides“-Schnittstelle). Adjazente Systeme können von anderen adjazenten Systemen im Systemkontext abhängig sein, dadurch dass diese etwa in einem Wirkzusammenhang stehen, der für das zu entwickelnde System von Bedeutung ist. Ein softwareintensives eingebettetes System soll auf definierte Ereignisse in der Umgebung reagieren, indem es auf die Ausprägung physikalische oder technische

Prozesse in der Umgebung einwirkt. *Technische Ereignisse* in der operationellen Umgebung werden von adjazenten Systemen ausgelöst.[4]

4.2 Physikalische Umgebung

Neben der Einbettung in eine technische Umgebung sind softwareintensive eingebettete Systeme in der Regel auch in eine *physikalische Umgebung* eingebettet, die sich durch *physikalische Größen* und deren *Ausprägungen* in der Umgebung (physikalische Prozesse) bestimmt (❷ in Abbildung 2). Physikalische Größen können in *materielle/räumliche physikalische Größen* (z.B. Abstand zum nächsten Hindernis) oder *energetische physikalische Größen* (z.B. Umgebungshelligkeit) unterschieden werden. Auch Abhängigkeiten zwischen physikalischen Größen können im Systemkontext des zu entwickelnde Systems von Bedeutung sein, etwa wenn ein physikalischer Prozess einen anderen physikalischen Prozess im Systemkontext beeinflusst und diese Abhängigkeit ggf. Auswirkungen auf das zu entwickelnde System hat. Physikalische Größen und zugehörige Prozesse sind Ursprung materieller bzw. energetischer physikalischer Ereignisse in der Umgebung des Systems, die einen *Stimulus* für das zu entwickelnde System darstellen, wie z.B. das energetische physikalische Ereignis „Umgebungshelligkeit < 1500 LUX“. Zu den Aspekten der physikalischen Umgebung zählen auch *materielle Gegenstände*, die in Beziehung zu dem zu entwickelnden System stehen, wie z.B. ein vorausfahrendes Fahrzeug für ein Fahrassistenzsystem in Fahrzeugen.

4.3 Operationelle Vorgaben und Rollen

Operationelle Vorgaben (❸ in Abbildung 2) gehen aus der Kontextsicht „Interessenvertreter“ oder aus der Kontextsicht „Systemdomäne“ hervor, z.B. indem eine operationelle Vorgabe aus einer Rahmenbedingung abgeleitet wird, die von einer Person artikuliert wurde. Operationelle Vorgaben sind integrative Kontextaspekte und können sich auf physikalische Größen, auf Ereignisse, auf die Kopplung adjazenter Systeme sowie auf die Interaktion zwischen Individuen und zu entwickelnden System im Betrieb beziehen. Ein weiterer integrativer Aspekt der Kontextsicht „operationelle Umgebung“ sind spezifische *Rollen* (❹ in Abbildung 2), die mit dem zu entwickelnden System im Betrieb in Beziehung stehen, wie z.B. die späteren Nutzer des Systems. Die Kontextsicht „operationelle Umgebung“ umfasst zudem Rollen von Individuen oder Gruppen, die das System konfigurieren oder warten, bzw. Individuen oder Gruppen, die das System zwar nicht unmittelbar nutzen, allerdings Eingaben für das System erarbeiten bzw. indirekt Dienste des Systems im Betrieb in Anspruch nehmen.

[4] Technische Ereignisse können auch von dem zu entwickelnden System durch einen internen Zustandsübergang initiiert werden. Zu dieser Klasse von Ereignissen gehören auch Zeitereignisse, die etwas durch das verstreichen einer definierten Zeitspanne initiiert werden. Solche Ereignisse werden in dieser Kontextsicht allerdings nicht berücksichtigt.

5 Anwendungsbeispiel: Steuergerät „dynamische Scheibentönung“

Im Rahmen einer ersten industriellen Erprobung wurde die Referenzstrukturierung zur Erfassung und Dokumentation des Systemkontexts eines zu entwickelnden prototypischen Systems im *DaimlerChrysler Software Technology Lab* eingesetzt. Bei dem zu entwickelnden System handelte es sich um das Steuergerät eines prototypischen Systems zur dynamischen Scheibentönung in Fahrzeugen (DST), das die Fahrzeuginsassen vor Blendbeeinträchtigung bei hoher Lichtintensität schützen soll, indem, abhängig von der Umgebungshelligkeit des Fahrzeuges und dem Lichteinfallwinkel, der Tönungsgrad der Fahrzeugscheiben elektronisch reguliert wird.

5.1 Beispiel: Kontextdiagramm „operationelle Umgebung“ des Steuergerät DST

Ein zentraler Artefakttyp im Artefaktmodell des vorgestellten Ansatzes (vgl. Abschnitt 1.3) sind Kontextdiagramme der einzelnen Kontextsichten, die überblicksartig den Systemkontext einer Kontextsicht dokumentieren. Das Kontextdiagramm einer Kontextsicht ist dabei eine Instanz des Informationsmodells der entsprechenden Kontextsicht. Das Kontextdiagramm einer Kontextsicht zeigt die in der jeweiligen Kontextsicht relevanten Kontextaspekte und deren Beziehungen und dient als Grundlage für die Erarbeitung und Dokumentation detaillierter Kontextinformationen, die durch spezifische im Artefaktmodell definierte Artefakte dokumentiert werden, etwa vorgegebene Schnittstellenmerkmale adjazenter Systeme (z.B. Operationen und Protokolle).

Abbildung 3 zeigt einen Ausschnitt des Kontextdiagramms „operationelle Umgebung“ des zu entwickelnden Steuergeräts DST, das auf dem in Abschnitt 4 vorgestellten Informationsmodell der Kontextsicht „operationelle Umgebung“ basiert und im Zuge der Kontextanalyse im RE für das zu entwickelnde Steuergerät erarbeitet wurde. Das Kontextdiagramm ist durch ein Objektdiagramm der UML [OMG07] modelliert, das aufgrund der Beschränkungen dieses Beitrags und zur besseren Verständlichkeit geringfügig vereinfacht wurde (siehe Anmerkungen zu „Vereinfachte Darstellung“ in Abbildung 3).

Gegenstand der prototypischen Entwicklung des Steuergerätes DST war das reaktive System, d.h. die zu verwendenden Sensoren und Aktuatoren des Systems waren vorgegeben und damit Aspekte im Systemkontext. Im Mittelpunkt des Kontextdiagramms steht das zu entwickelnde Steuergerät DST – repräsentiert durch das Objekt „*Stg. dynamische Scheibentönung (DST)*“.

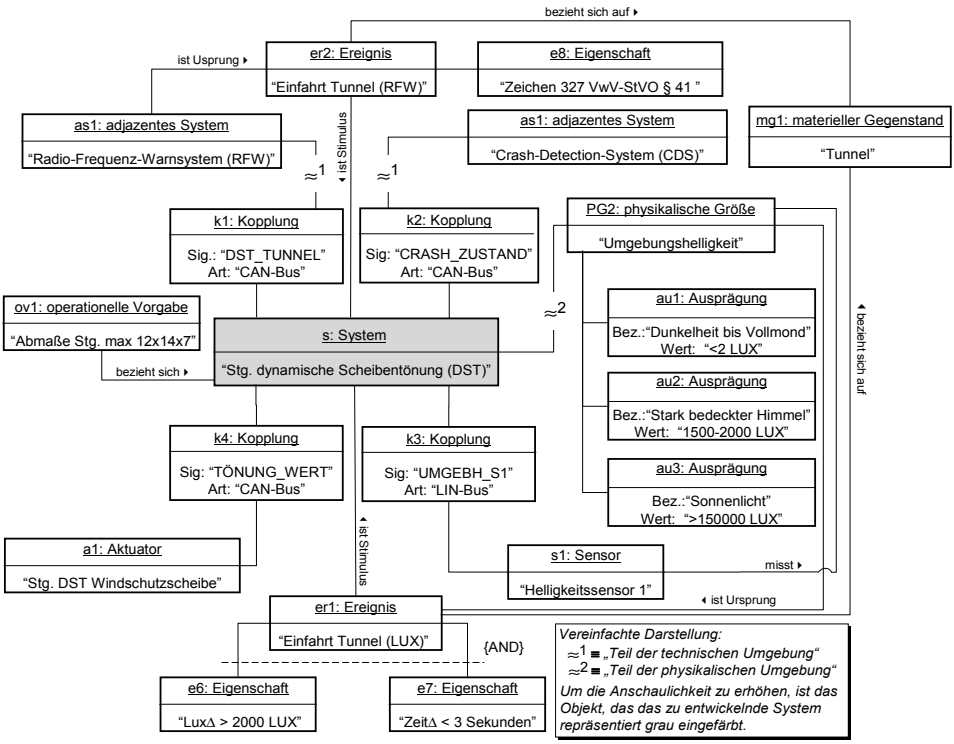

Abbildung 3. Kontextdiagramm der Kontextsicht „operationelle Umgebung" (Ausschnitt)

5.1.1 Technische Umgebung: Steuergerät DST

Für jedes adjazente System und für jeden Sensor und Aktuator waren die Ausprägung der Kopplung mit dem zu entwickelnden System sowie wichtige Bus-Signale bereits vorgegeben. So sollte das zu entwickelnde Steuergerät z.B. mit einem prototypischen *Radio-Frequenz-Warnsystem (RFW)* interagieren, dass über den CAN-Bus *(k1: Kopplung, Art: „CAN-Bus")* mit dem Steuergerät DST gekoppelt werden sollte. Vorgegeben war zudem, dass das *RFW* das Bus-Signal „*DST_TUNNEL*" ausgibt, um dem Steuergerät DST die Einfahrt in einen Tunnel zu signalisieren. *Einfahrt Tunnel (RFW)* ist ein Ereignis in der technischen Umgebung des zu entwickelnden Steuergeräts, welches vom RFW über das Verkehrszeichen „*Zeichen 327"* nach *„VwV-StVO § 41"* identifiziert wird. Auch das *Crash-Detection-System (CDS)* des Fahrzeuges sollte mit dem DST gekoppelt werden, um im Falle eines Unfalls alle Fahrzeugscheiben sofort zu enttönen. Ein Unfall wird vom CDS über das Bus-Signal *„CRASH_ZUSTAND"* signalisiert. Der Ausschnitt des Kontextdiagramms dokumentiert weitere Vorgaben für das zu entwickelnde Steuergerät DST, wie z.B. die vorgegebene Kopplung mit einem Steuergerät zur Tönung der Windschutzscheibe *(stg. DST Windschutzscheibe),* das einen Aktuator des Steuergeräts DST darstellt sowie ein vorgegebener Helligkeitssensor (*Helligkeitssensor 1*), der über den *LIN-Bus* mit dem Steuergerät DST gekoppelt werden soll und die aktuelle

Ausprägung der physikalischen Größe *Umgebungshelligkeit* über das Bus-Signal „*UMGEBH_S1*“ signalisiert.

5.1.2 Physikalische Umgebung und operationelle Vorgaben: Steuergerät DST

Im gezeigten Ausschnitt des Kontextdiagramms der Kontextsicht „operationelle Umgebung“ wird die Beziehung der physikalischen Größe *„Umgebungshelligkeit“* zu dem zu entwickelnden Steuergerät DST dokumentiert. Die Umgebungshelligkeit hat drei für das DST relevante Ausprägungen mit zugehörigen Lichtintensitätswerten, u.a. *„Dunkelheit bis Vollmond < 2 LUX“* und *„Sonnenlicht > 150000 LUX“*. Die Einfahrt in einen Tunnel kann, alternativ zur Identifikation durch das RFW, auch durch einen Abfall der Lichtintensität identifiziert werden und zwar durch die Bedingung, dass in weniger als 3 Sekunden die Lichtintensität um mehr als 2000 LUX fällt (*„LuxΔ > 2000 LUX“* und *„ZeitΔ < 3 Sekunden“*). Die beiden Ereignisse *„Einfahrt Tunnel (RFW)“* und *„Einfahrt Tunnel (LUX)“* beziehen sich auf den materiellen Gegenstand *„Tunnel“* im operationellen Kontext. Das Kontextdiagramm dokumentiert überdies eine operationelle Vorgabe, die die Abmaße des zu entwickelnden Steuergeräts DST reglementiert *(Abmaße Stg. max 12×14×7)*. Ursprung dieser operationellen Vorgabe ist ein Entwicklungsingenieur (→ Kontextsicht „Interessenvertreter“), der diese Abmaße aufgrund des zur Verfügung stehenden Einbauraums im Fahrzeug als Rahmenbedingung nannte.

5.2 Erkenntnisse aus der industriellen Erprobung

Im Rahmen einer ersten industriellen Erprobung hat sich der Ansatz hinsichtlich der strukturierten Erfassung und Dokumentation des Systemkontexts eines softwareintensiven eingebetteten Systems bewährt. Auch wenn die im Ansatz vorgesehene explizite Dokumentation des Systemkontexts durch konzeptuelle Modelle vielfach als ein zusätzlicher Aufwand empfunden wurde, half die Modellbasierung dabei, die Komplexität des Systemkontexts mit vertretbarem Aufwand beherrschbar zu machen. Allerdings setzte dies sowohl bei den Modellierern als auch bei den „Lesern“ der Modelle Kenntnisse hinsichtlich der verwendeten Modellierungssprachen voraus. Im Zuge der Erprobung haben sich eine Reihe weitere Vorteile des vorgestellten Ansatzes gezeigt. So ermöglichte die Referenzstrukturierung eine selektive Erfassung des Systemkontexts. Zudem erlaubte der Ansatz die Spezifikation der Systemanforderungen auf Grundlage einer dokumentierten Informationsbasis (den Kontextinformationen), wodurch auch die Nachvollziehbarkeit der Systemanforderungen zu deren Ursprung unterstützt wurde. Durch die strukturierte explizite Dokumentation der Kontextinformationen konnten die Systemanforderungen gegen die Kontextinformationen validiert werden. Dies förderte zusätzlich die Abstimmung der Systemanforderungen, da bei Unstimmigkeiten die assoziierten Kontextinformationen in die Abstimmung einfließen konnten. Die Erfahrungen lassen außerdem vermuten, dass der Ansatz das Management der Evolution des Systemkontexts über den gesamten Lebenszyklus des Systems hinweg unterstützt, da der Kontext selektiv auf Veränderungen hin untersucht und bei identifizierten Veränderungen ggf. Anpassungen der assoziierten Systemanforderungen initiiert werden können.

6 Zusammenfassung und Ausblick

In diesem Beitrag wurde eine Referenzstrukturierung des Systemkontexts softwareintensiver eingebetteter Systeme für die modellbasierte Kontextanalyse im RE vorgestellt. Das vorliegende Papier konzentrierte sich dabei auf zwei zentrale Konzepte der Referenzstrukturierung: Die drei Kontextsichten zur grobgranularen Strukturierung des Systemkontexts und die Informationsmodelle zur Detaillierung der einzelnen Kontextsichten. Zusätzlich umfasst die Referenzstrukturierung ein drittes Konzept in Form eines Artefaktmodells für Kontextinformationen. Dieses Artefaktmodell definiert, basierend auf den Informationsmodellen der drei Kontextsichten, Artefakttypen zur Erarbeitung und Dokumentation detaillierter Kontextinformationen und schlägt geeignete Repräsentationsformen für Kontextinformationen spezifischer Artefakttypen vor.

Die vorgestellte Referenzstrukturierung definiert ein Rahmenwerk zur strukturierten Erfassung und Dokumentation des Systemkontexts und bildet das Fundament einer methodischen Anleitung zur modellbasierten Kontextanalyse im RE softwareintensiver eingebetteter Systeme. Der Ansatz ermöglicht zudem eine selektive Erfassung des Systemkontexts, wenn etwa die zur Verfügung stehenden Ressourcen keine vollständige Kontextanalyse zulassen. Ebensolche Aspekte werden u.a. in den noch zu definierenden *Tailoring*-Richtlinien des Ansatzes zu berücksichtigen sein. In den nächsten Schritten ist eine weitergehende Validierung in Form einer industriellen Fallstudie vorgesehen, um die Eignung der vorgeschlagenen Konzepte im industriellen Umfeld intensiver zu überprüfen. Gehaltvolle Aussagen über Prozess- und Qualitätsverbesserungen werden erst in Kombination mit der methodischen Anleitung gewonnen werden können. Nichtsdestotrotz sind die Autoren der Überzeugung, dass die vorgeschlagene Referenzstrukturierung ein wichtiger Schritt in Richtung einer größtenteils noch zu entwickelnden Kontexttheorie für das modellbasierte RE softwareintensiver eingebetteter Systeme ist.

Danksagungen

Diese Arbeit wurde teilweise durch das BMBF gefördert: Projekt „REMsES“ (Förderkennzeichen: 01-IS-F06-D). Die Autoren danken Nadine Bramsiepe und Kim Lauenroth für die Unterstützung bei der Erstellung dieses Beitrags.

Literaturverzeichnis

[AM95] Alfred, C.; Mellor, S.: Observations on the role of patterns in object-oriented software development. Object Magazine, 5(2), 60-65.

[Bo84] Boehm, B.: Verifying and Validating Software Requirements and Design Specifications. IEEE Software, 1(1), IEEE Computer Society, 1984, 75-88.

[Br06] Broy, M.: The 'Grand Challenge' in Informatics – Engineering Software-Intensive Systems. IEEE Computer, 39(10), 72-80

[Br07] Broy, M.; Geisberger, E.; Kazmeier, J.; Rudorfer, A.; Beetz, K.: Ein Requirement-Engineering-Referenzmodell. Informatik-Spektrum 30(3), Springer, 127-142.

[CA07] Cheng, B. H.; Atlee J. M: Research Directions in Requirements Engineering. In: Proc. Int.'l Conf. on Software Eng., FOSE 07, IEEE Computer Society, 2007, 285-303.
[Co01] Cockburn, A.: Writing Effective Use Cases. Addison-Wesley, 2001.
[De78] DeMarco, T.: Structured Analysis and System Specification. Yourdon Press, New York, 1978.
[DPB03] Denger, C.; Paech, B.; Benz, S.: Guidelines – Creating Use Cases for Embedded Systems. QUASAR-Projekt, IESE-Report No. 078.03/E, Fraunhofer IESE, 2003.
[FF95] Fickas, S.; Feather, M.: Requirements Monitoring in Dynamic Environments. In: Proc. Int.'l Symp. on Requirements Engineering, IEEE Computer Society, 1995, 140-147.
[Gu95] Guha, R.: Contexts: a formalization and some applications. Stanford Univ., 1995.
[HF00] Hooks, I. A.; Farry, K. A.: Customer-Centred Requirements, Amacom, 2000.
[HP88] Hatley, D. J.; Pirbhai, I. A.: Strategies for Real Time System Specification. Dorset House, 1988.
[Ja01] Jackson, M.: Problem frames – Analysing and structuring software development problems. ACM Press, 2001.
[Ja92] Jacobson, I.; Christerson, M.; Jonsson, P.; Oevergaard, G.: Object Oriented Software Engineering – A Use Case Driven Approach. Addison Wesley, 1992.
[Ja95] Jackson, M.: The World and the Machine. In: Proc. Int.'l Conf. on Software Eng., ACM Press, 1995, 283-292.
[JP93] Jarke, M.; Pohl, K.: Establishing Visions in Context – Towards a Model of Requirements Processes, In: Proc. 14th Int.'l Conf. on Inf. Systems, 1993, 23-34.
[La91] Van Lamsweerde, A.; Dardenne, A.; Delcourt, B.; Dubisy, F.: The KAOS Project – Knowledge Acquisition in Automated Specification of Software. In: Proc. AAAI Spring Symposium Series, Stanford Univ., 1991, 69-82.
[Le97] Le Lann, G.: An Analysis Ariane 5 Flight 501 Failure – A System Engineering Perspec-tive. In: Proc. 10th IEEE Int'l ECBS Conf., IEEE Press, 1997, 339-346.
[LK95] Loucopoulos, P.; Karakostas, V.: System Requirements Engineering. McGraw Hill, 1995.
[MP84] McMenamin, S. M., Palmer, J. F. S: Essential Systems Analysis. Prentice Hall, Yourdon Press 1984.
[Mu79] Mullery, G.: CORE – A Method for Controlled Requirements Specification. In: Proc. 4th Int.'l Conf. on Software Eng., München 1979.
[OMG07] Object Management Group: Unified Modelling Language – Superstructure. Version 2.1.1., OMG, 2007. Abrufbar unter: http://www.omg.org.
[oV95] o.V.: The New York subway crash. In: New York Times vom 16. Juni 1995. Abrufbar unter: http//catless.ncl.ac.uk/Risks/17.17.html (abgerufen am: 23-08-2007).
[PIÖ05] Pettersson, F.; Ivarsson, M.; Öhman, P.: Automotive Use Case Standard for Embedded Systems. ACM SIGSOFT Software Engineering Notes 30(4), 2005, 1-6.
[PM95] Parnas, D. L.; Madey, J.: Functional documents for computer systems. Science of Computer Programming 25 (1995), Elsevier, 41-61.
[Po95] Potts, C.: Using Schematic Scenarios to Understand User Needs. In: Proc. ACM Symposium on Designing Interactive Systems (DIS 95), ACM Press, 1995, 247-266.
[Po97] Pouloudi, A.: Stakeholder Analysis as a Front-End to Knowledge Elicitation. AI & Society 11 (1), 122-137.
[RR99] Robertson, S.; Robertson, J.: Mastering the Requirements Process. Addison Wesley, 1999.
[RS77] Ross, D. T.; Schoman, K. E.: Structured Analysis for Requirements Definition. IEEE Trans. on Software Engineering, Vol. 3, Nr. 1, 1977, 6-15.
[SFS06] Sutcliffe, A. G.; Fickas, S.; Solhberg, M. M.: PC-Requirements Engineering – a method for personal and contextual Requirements Engineering with some experience. Requirements Engineering 11(3), 157-173.
[We75] Weinberg, G. M.: An introduction to general systems thinking, Dorset House, 2001.
[WM85] Ward, P.; Mellor, S.: Structured Development of Real-Time Systems – Introduction and Tools. Vol. 1. Prentice Hall, Upper Saddle River, 1985.
[Yu93] Yu, E.: An Organisational Modelling Framework for Multiperspective Information System Design. In: Requirements Engineering 1993, Univ. of Toronto, 1993, 66-86.
[Za82] Zave, P.: An operational approach to requirements specification for embedded systems. IEEE Trans. on Software Engineering 8(3), 250-269.
[ZJ97] Zave, P.; Jackson, M.: Four Dark Corners of Requirements Engineering. ACM Trans. on Software Engineering and Methodology 6(1), 1-30.