Herausforderungen in der Laufzeitdarstellung von Anforderungen für eingebettete Systeme

Fabian Kneer und Erik Kamsties
Fachhochschule Dortmund
{erik.kamsties, fabian.kneer}@fh-dortmund.de

Motivation. Es gibt eine Klasse von Softwaresystemen, die sich dadurch auszeichnet, dass die Umgebung des Systems zur Entwicklungszeit nicht vollständig bekannt ist oder die Umgebung unterliegt häufigen Änderungen. Die Entwicklung dieser Systeme ist daher mit traditionellen Methoden des Requirements Engineering nicht adäquat beherrschbar. Diese Klasse von Systemen wird als dynamisch anpassbare Systeme (DAS) bezeichnet. In aktuellen Arbeiten wurde daher nach Wegen gesucht, um Anforderungen zur Laufzeit verfügbar zu machen (engl. *requirements at runtime*), damit ein System Entscheidungen zu seiner Anpassung selbst treffen kann. Umsetzungen dieses Ansatzes wurden unter anderem im Bereich Service-orientierter Systeme vorgeschlagen [4].

Forschungsfragestellung. In diesem Beitrag werden zwei Herausforderungen adressiert. Die erste Herausforderung liegt in der Kommunikation der erfolgten Anpassungen. Zum einen kann ein Anforderungsanalyst aus den Anpassungen wertvolle Erkenntnisse über die Benutzung eines Systems für dessen Weiterentwicklung gewinnen, zum anderen sollte der Benutzer über die Anpassung des Systems informiert werden. Das Problem liegt darin, dass ein Laufzeit-Anforderungsmodell formal sein muss, um automatisiert ausgewertet werden zu können, jedoch Änderungen an den Anforderungsanalyst und Benutzer möglichst natürlichsprachlich kommuniziert werden sollten.

Die zweite Herausforderung ergibt sich im Kontext von eingebetteten Systemen. Diese Systeme sind als Massenprodukte ressourcenarm (u.a. hinsichtlich Speicher, Rechenleistung) ausgelegt, gleichzeitig werden höhere Anforderungen an die Vorhersagbarkeit des zeitlichen Verhaltens gestellt. Etablierte Ansätze lassen sich folglich nicht direkt übertragen.

Dieser Beitrag baut auf einem Research Poster auf [3], in dem das zugrunde liegende Konzept der Nachvollziehbarkeit zwischen Entwicklungs- und Laufzeitanforderungen vorgestellt wurde. Der vorliegende Beitrag diskutiert die Forschungsergebnisse hinsichtlich der Realisierung dieses Konzepts.

Lösungsansatz. Es wird in diesem Beitrag ein Ansatz vorgestellt, der eine Nachvollziehbarkeit zwischen traditionellen, textbasierten Anforderungen und Laufzeitdarstellungen von Anforderungen durch entsprechende Links vorsieht, siehe Abbildung 1. Mit Hilfe dieser Links können Änderungen an den Laufzeitanforderungen für die erwähnten Interessengruppen zurückübersetzt werden.

Der vorgestellte Ansatz besteht aus mehreren Komponenten (siehe Abbildung 1). Der Anforderungsmonitor überwacht das System, d.h. die eigentliche Anwendung, er setzt sich wiederum aus mehreren Modulen zusammen:

- einer Laufzeitdarstellung der Anforderungen,
- dem eigentlichen Monitor zur Überwachung von Annahmen über die Umgebung, sowie
- einer Auswirkungsanalyse.

Das Konzept des Anforderungsmonitors wird im folgenden kurz vorgestellt.

Laufzeitdarstellung von Anforderungen. Im Ansatz wird gegenwärtig für die Laufzeitanforderungen ein erweitertes i*-Zielmodell basierend auf openOME [1] verwendet. Zur Erstellung dieses Modells wird ein einfacher textbasierter Editor verwendet, der aus einem Ecore-Metamodell (Teil des Eclipse Modeling Framework EMF) des i*-Metamodells generiert wird.

Monitor. Auf dem i*-Laufzeitmodell werden Regeln definiert, die von einem Monitor überwacht werden. Bei der Realisierung dieser Komponente wurde auf ein etabliertes Framework mit geringer Ressourcennutzung zurückgegriffen [2]. Es gibt zwei Formen von Regeln, die eine ist mit *Softgoals* assoziiert und die andere mit sogenannten *Contribution Links*. Mit Hilfe des Monitors werden Abweichungen der Umgebung vom erwarteten Verhalten erkannt und in Bezug zu den Laufzeitanforderungen (i*-Modell) gesetzt.

Auswirkungsanalyse. In der Auswirkungsanalyse wird mit Hilfe des Laufzeit-Anforderungsmodells die notwendige Anpassung einer Systemkonfiguration berechnet, falls eine Regel verletzt wurde. Es wurde ein entsprechendes Verfahren entwickelt, das berechnet, ob die Zielerfüllung noch gegeben ist und ggf. die Parameter angepasst werden müssen.

Wir fokussieren im hier vorgestellten Ansatz zunächst auf die Anpassung von Parametern, die gerade im Bereich von eingebetteten Systemen eine große Bedeutung besitzen, da über sie ein Großteil der

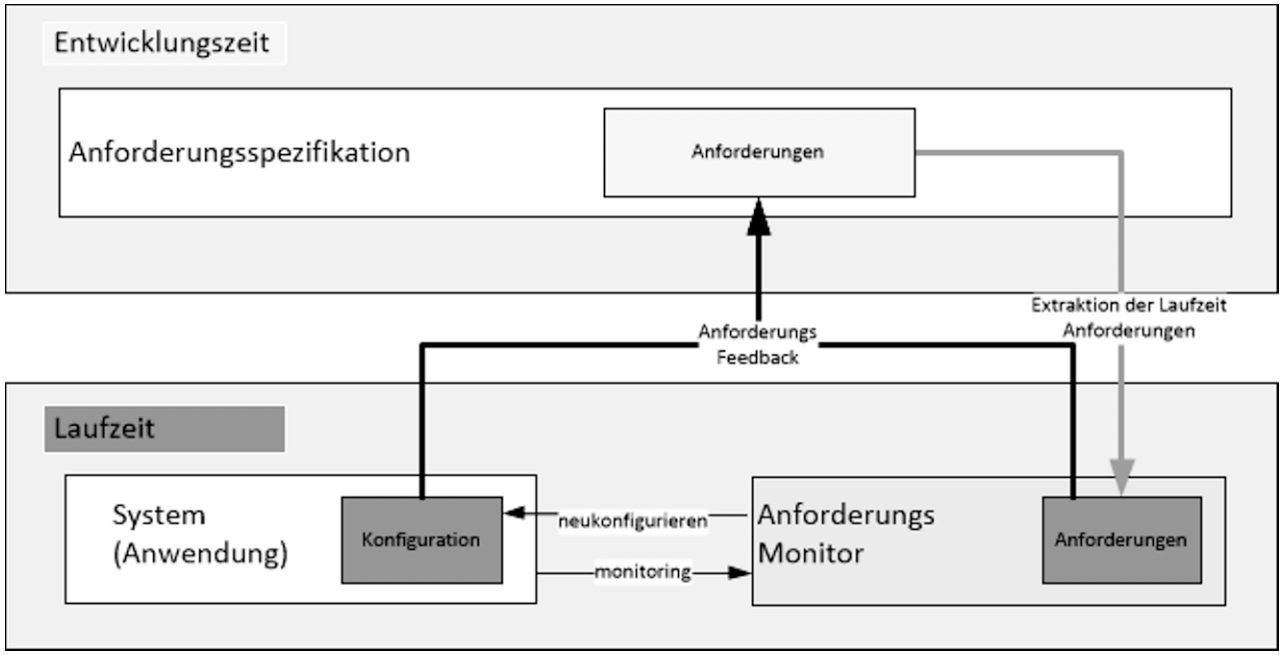

Abbildung 1: Bridging the gap between development time and runtime artifacts

umgebungsspezifischen Anpassungen erfolgt ("Codierung", "Kalibrierung").

Der vorgestellte Ansatz wurde in einer prototypischen Form in JAVA realisiert und in einer Fallstudie erprobt. JAVA ist auf 32-bit Embedded Plattformen häufig verfügbar (z.B. dem populären Raspberry Pi). Gegenwärtig wird der Ansatz auf C portiert, damit können kleinere Plattformen adressiert werden. Es wurde im Vortrag auf dem Fachgruppentreffen über den aktuellen Stand der Arbeiten berichtet.

Literatur

[1] openome
home page. https://se.cs.toronto.edu/trac/ome/. Accessed: 2013-10-12.

[2] roolie
project home page. http://roolie.sourceforge.net/. Accessed: 2013-10-12.

[3] Fabian Kneer and Erik Kamsties. Representation of requirements changes to stakeholders in self-adaptive systems. In Richard Berntsson Svensson and Daniel M. Berry etal, editors, *Workshop Proceedings of the International Working Conference on Requirements Engineering: Foundation for Software Quality (REFSQ2013)*, pages 253–254. University of Duisburg-Essen, ICB-Research Report No. 56, July 2013. Extended Abstract.

[4] P. Sawyer, N. Bencomo, J. Whittle, E. Letier, and A. Finkelstein. Requirements-aware systems: A research agenda for RE for self-adaptive systems. In *Requirements Engineering Conference (RE), 2010 18th IEEE International*, pages 95–103, 2010.