Konsistenzprüfung von Architekturbeschreibungen mit Anforderungen mittels linguistischer Analyse

Kai Niklas[1,2], Stefan Gärtner[2], und Kurt Schneider[2]

[1]Talanx Systeme AG, Hannover, Germany
kai.niklas@talanx.com
[2]Leibniz Universität, Software Engineering Group, Hannover, Germany
{kai.niklas,stefan.gaertner,kurt.schneider}@inf.uni-hannover.de

Zusammenfassung

Zur Qualitätssicherung von Architekturbeschreibungen, z.B. Komponenten- oder Schnittstellenbeschreibungen, ist eine Konsistenzprüfung gegen die gegebenen Anforderungen sinnvoll. Gerade wenn neue oder geänderte Anforderungen die Änderung bestehender Architekturbeschreibungen zur Folge haben, sollte die Konsistenz zu den Anforderungen bewahrt bleiben. Wir präsentieren eine Methode, basierend auf linguistischer Analyse, um eine solche Konsistenzprüfung zu unterstützen. Wir demonstrieren unsere Methode anhand eines industriellen Fallbeispiels.

1 Inkonsistenz zwischen Anforderungen und Architekturbeschreibungen durch Evolution

Die begriffliche Konsistenz zwischen Anforderungen und Architekturbeschreibungen hilft Entwicklern und Architekten komplexe Systeme besser zu verstehen. Der Grund ist die vereinfachte Zuordnung von Begriffen aus den Anforderungen zu den dazu passenden Ausschnitten der Architekturbeschreibung. Dadurch kann der Wartungsaufwand reduziert und die Produktivität gesteigert werden [SZ01]. Inkonsistenzen können entstehen, wenn Anforderungen hinzugefügt oder geändert werden. Wird beispielsweise ein bestehender Begriff in den Anforderungen ersetzt, so muss die Architekturbeschreibung entsprechend angepasst werden. Andernfalls geht die meist implizite Zuordnung zu den Anforderungen verloren. Aus den gleichen Gründen kann es aber auch, durch Änderung der Architekturbeschreibung, zu Inkonsistenzen mit bestehenden Anforderungen kommen.

Der Aufwand zur manuellen Prüfung von Konsistenz ist bei komplexen Softwaresystemen mit umfangreichen Anforderungen sehr hoch. Daher ist eine automatische Überprüfung und Korrekturunterstützung sinnvoll. Ein wesentliches Problem ist, dass Anforderungen in der Praxis überwiegend in natürlicher Sprache vorliegen und meist schwach strukturiert sind. Existierende Ansätze erfordern jedoch formale Notationen [CG01, Kim08] oder besondere Formen von Anforderungen, z.B. Szenarien [KZ02]. Eine Überführung von natürlichsprachlichen Anforderungen in formale Beschreibungen, durch die Entwickler, ist jedoch, gerade im Kontext von Evolution, zeitaufwändig und arbeitsintensiv. Das hat zur Folge, dass bestehende Ansätze zur Konsistenzprüfung nicht skalieren und für viele oder komplexe Artefakte ungeeignet sind [PSM+09].

2 Automatische Konsistenzprüfung durch linguistische Analyse

Mit Hilfe von linguistischen Methoden haben wir einen Ansatz zur automatischen Konsistenzprüfung zwischen natürlichsprachlichen Anforderungen und Architekturbeschreibungen entwickelt. Der Ansatz besteht im Wesentlichen aus zwei Schritten.

Schritt 1) *Extraktion von Begriffen aus Anforderungen und Architekturbeschreibungen*: Die wesentlichen Begriffe werden aus den jeweiligen Artefakten mit Hilfe von linguistischen Methoden extrahiert. Hierzu werden NLP-Tools für Part-of-Speech Tagging und Lemmatisierung benutzt [Sch94]. Die extrahierten Begriffe stehen in einer Beziehung zueinander, wenn sie innerhalb eines Artefakts gemeinsam genutzt werden. Bei natürlichsprachlichen Anforderungen wird hierzu die lexikalische Struktur des Textes mit einbezogen. Zum Beispiel stehen zwei Begriffe in Beziehung zueinander, wenn sie innerhalb einer Aufzählung gemeinsam verwendet werden. Bei Architekturbeschreibungen, z.B. in einem UML Diagramm, entstehen Beziehungen durch Hierarchie- und Geschwisterbeziehungen der Klassen und Attribute.

Domänenspezifische Begriffe, die untereinander in Beziehung stehen, bilden sogenannte Konzeptgraphen [Sow76]. Hierbei werden Begriffe als Knoten, und Beziehungen zwischen Begriffen als Kanten abgebildet.

Schritt 2) *Vergleich der extrahierten Begriffe*: Im Rahmen unserer Arbeit sind zwei Artefakte zueinander konsistent, wenn sie Begriffe in gleicher Art und Weise verwenden. Das bedeutet, dass verwendete Begriffe, und deren Beziehungen zueinander, aus den Anforderungen mit den Begriffen, und deren Beziehungen zueinander, aus der Architekturbeschreibung übereinstimmen müssen. Es wird nicht geprüft, ob alle Begriffe und Beziehungen aus den Anforderungen auch tatsächlich verwendet werden. Zur Prüfung der Konsistenz werden hierzu die aus den Anforderungen und der Architekturbeschreibung erstellten Konzeptgraphen automatisch miteinander verglichen [Bun00]. Passen die Begriffe in den Knoten, unter Einbeziehung der Struktur des Graphen, semantisch nicht zueinander oder sind die Knoten unterschiedlich miteinander verbunden, so liegt eine Inkonsistenz vor. Das heißt, dass die verwendeten Begriffswelten voneinander abweichen. Aus dem Ergebnis des Vergleichs können somit Rückschlüsse auf die Konsistenz der jeweiligen Artefakte gezogen und Empfehlungen zur Korrektur abgeleitet werden.

Der Ansatz kann zur Entwicklungszeit zwischen der Anforderungs- und Entwurfsphase angewendet werden. Er eignet sich insbesondere für die Wartung langlebiger Systeme, da geänderte Anforderungen kontinuierlich gegen die Architekturbeschreibungen geprüft werden können.

Durch die Verwendung linguistischer Methoden kann unserer Ansatz auf schwach strukturierte Anforderungen in natürlicher Sprache, ohne manuelle Vorverarbeitung, automatisiert angewendet werden. Vorteile sind somit die Ausnutzung bestehender Anforderungen im industriellen Umfeld und die Unterstützung verschiedener Projektgrößen. Die Methode wird nur durch die Qualität der Artefakte und Leistungsfähigkeit verwendeter linguistischer Methoden beschränkt. Zum Beispiel wird die sinnvolle Anwendung des Ansatzes erschwert, wenn innerhalb der Anforderungen Begriffe nicht einheitlich verwendet werden.

3 Industrielles Fallbeispiel

Das folgende Beispiel demonstriert den Einsatz der vorgeschlagenen Methode anhand eines industriellen Fallbeispiels. Im Kontext einer serviceorientierten Architektur (SOA) soll die Konsistenz einer Service-Schnittstellenbeschreibung gegen die zugrundeliegenden Anforderungen geprüft werden. Services werden zur Integration von Anwendungen und (Teil-) Systemen genutzt. Sie werden mit Hilfe einer unternehmensinternen, domänenspezifischen Sprache beschrieben und sind zentrale, langlebige Architekturbeschreibungen. Aus den formalen Beschreibungen werden später Code und Dokumentation zur Umsetzung von Services generiert. Die natürlichsprachlichen Anforderungen liegen als schwach strukturiertes Fachkonzept vor, welches zwischen Fachbereich und IT abgestimmt wurde. Es beinhaltet keine spezifische Servicebeschreibung, sondern beschreibt nur Funktionalitäten aus fachlicher Sicht.

In Abbildung 1 sind Ausschnitte der extrahierten Konzeptgraphen dargestellt: G_a (Anforderungen) und G_b (Schnittstellenbeschreibung). Zu erkennen ist die Übereinstimmung der markierten Knoten und Kanten in G_a und G_b. Der verwendete Begriff „Produktkategorie“ aus G_b ist jedoch nicht in G_a vorhanden und wird somit auch nicht als Begriff in den Anforderungen verwendet. Dies stellt eine Inkonsistenz zu G_a dar. In G_a finden wir jedoch andere Begriffe an entsprechender Stelle im Graphen, z.B. „Zuordnung“ oder „Spezifizierung“. Zur Herstellung der Konsistenz zwischen Anforderung und Schnittstellenbeschreibung können dem Entwickler diese Alternativen vorgeschlagen werden. Dieser kann dann einen zum Kontext passenden Begriff für die Schnittstellenbeschreibung auswählen.

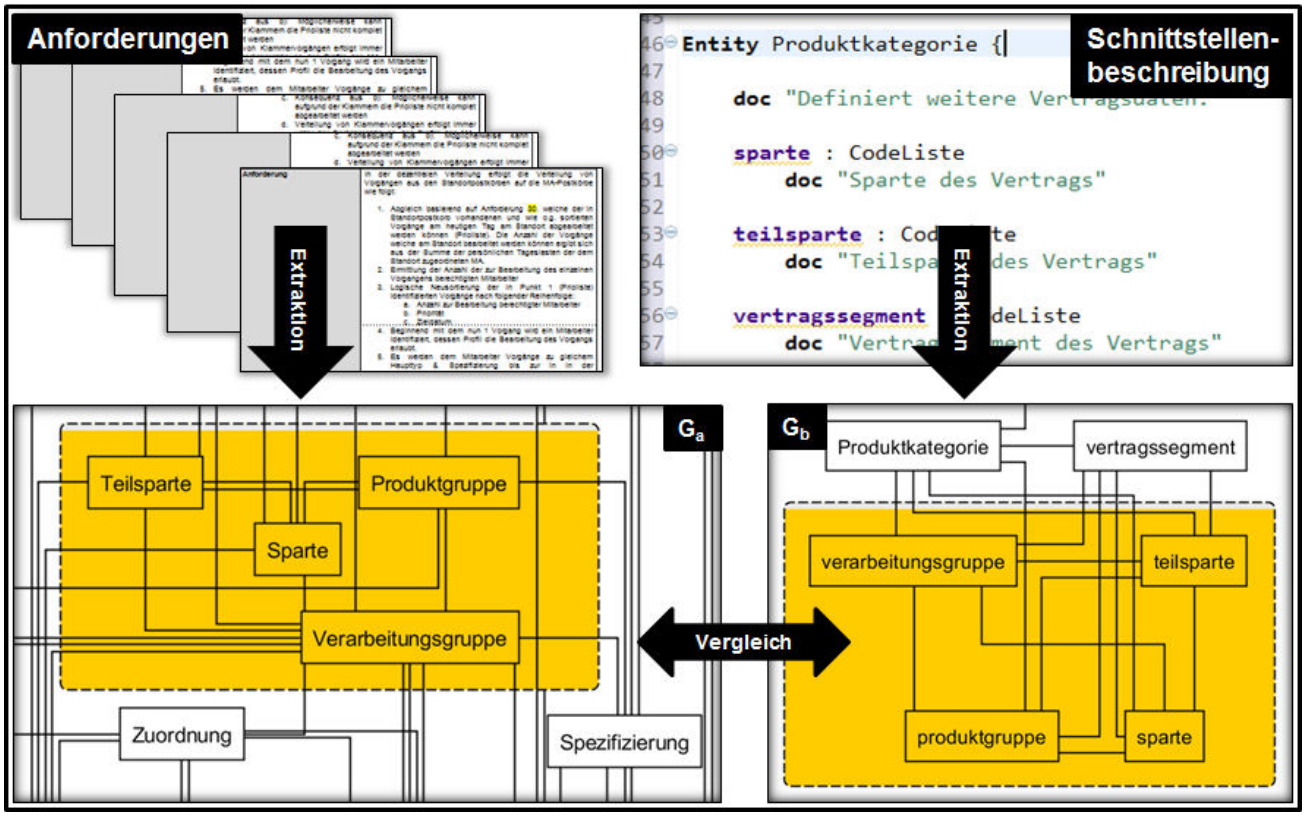

Abbildung 1: Ausschnitte aus Konzeptgraphen von Anforderungen und Schnittstellenbeschreibung

Literatur

[Bun00] Horst Bunke. Graph Matching: Theoretical Foundations, Algorithms, and Applications. In *International Conference on Vision Interface*, Seiten 82–84, Mai 2000.

[CG01] Marsha Chechik und John Gannon. Automatic analysis of consistency between requirements and designs. *IEEE Transactions on Software Engineering*, 27(7):651–672, 2001.

[Kim08] Do Hyung Kim. Method and Implementation for Consistency Verification of DEVS Model against User Requirement. *Proceedings of the 10th International Conference on Advanced Communication Technology*, Seiten 400–404, 2008.

[KZ02] A. Kozlenkov und A. Zisman. Are their design specifications consistent with our requirements? *Proceedings of the IEEE Joint International Conference on Requirements Engineering*, Seiten 145–154, 2002.

[PSM+09] Hendrik Post, Carsten Sinz, Florian Merz, Thomas Gorges und Thomas Kropf. Linking Functional Requirements and Software Verification. *Proceedings of the 17th IEEE International Requirements Engineering Conference*, Seiten 295–302, 2009.

[Sch94] Helmut Schmid. Probabilistic part-of-speech tagging using decision trees. In *Proceedings of the international conference on new methods in language processing*, Jgg. 12, Seiten 44–49. Manchester, UK, 1994.

[Sow76] John F. Sowa. Conceptual graphs for a data base interface. *IBM Journal of Research and Development*, 20(4):336–357, 1976.

[SZ01] George Spanoudakis und Andrea Zisman. Inconsistency management in software engineering: Survey and open research issues. *Handbook of software engineering and knowledge engineering*, 1:329–380, 2001.