# Werkzeuggestützte Konsistenz zwischen Anforderungen und Spezifikation

Dr. Michael Jastram, M.Sc. Lukas Ladenberger

Universität Düsseldorf
Formal Mind GmbH

Anforderungen beschreiben, was ein System tun soll, während die Spezifikation das "wie" (die Umsetzung) beschreibt. Anforderungen und Spezifikation werden im folgenden *Systembeschreibung* genannt. Die Systembeschreibung konsistent zu halten ist eine große Herausforderung, mit der sich dieser Vortrag beschäftigt. Es geht dabei primär um zwei Aspekte: (1) Strukturierung der Systembeschreibung und (2) Werkzeugunterstützung. In diesem Vortrag wird anhand eines Beispiels die Methode und deren Anwendung vorgestellt.

## I. Methode

Die erste Herausforderung ist es, die Systembeschreibung so zu strukturieren, dass deren Konsistenz überhaupt systematisch überprüft werden kann. Dies wird erreicht, indem die natürlichsprachigen Anforderungen (im folgenden *Artefakte* genannt) klassifiziert werden. Ein Artefakt kann u.A. eine Anforderung sein ($R$), eine Domänen-Eigenschaft ($W$), oder ein Spezifikations-Element ($S$). Artefakte brauchen ein Vokabular, und diese werden als *Phänomene* bezeichnet. Auch diese werden klassifiziert, und zwar bezüglich Sichtbarkeit und Kontrolle: Phänomene der Umgebung ($e$) können vom System wahrgenommen werden ($e_s$) oder nicht ($e_h$). Systemphenomene ($s$) sind ebenfalls unterteilt. Diese Methode ist in [JHL11, Jas12, HJL12] beschrieben und stellt eine Erweiterung des WRSPM-Referenzmodells [GJGZ00] dar.

Artefakte können über "realize"- und "justify"-Beziehungen miteinander verknüpft werden, was eine Nachverfolgbarkeit zwischen den Artefakten der Systembeschreibung ermöglicht.

Diese Klassifizierung leitet den Nutzer, durch die Klassifizierung die Systemgrenzen klar zu definieren. Weiterhin ermöglicht sie bereits die Prüfung einiger Konsistenzkriterien. Bspw. darf eine Anforderung ($R$) keine unsichtbaren Phänomene ($s_h$) nutzen, uns die Spezifikation hat keine Möglichkeit, auf nicht wahrgenommene Umgebungsphänomene ($e_h$) zu reagieren.

Die Traceability ermöglicht die Prüfung weiterer Konsistenzkriterien. Bspw. muss jede Anforderung in irgendeiner Art und Weise *realisiert* werden (also entsprechende realize-Beziehungen haben).

Dieses Vorgehen schränkt das Arbeiten mit traditionellen Anforderungen kaum ein, da lediglich die Klassifizierungen hinzugefügt werden müssen. Es ist aber auch möglich, noch einen Schritt weiter zu gehen: Teile des Systems können formal spezifiziert werden. Dies ermöglicht dann einen höheren Automatisierungsgrad bei der Erstellung der Systembeschreibung, sowie die Option, bestimmte Eigenschaften zu beweisen. In diesem Vortrag werden wir zeigen, wie Teile der Spezifikation mit dem Event-B-Formalismus modelliert werden und mit Anforderungen verlinkt werden können.

## II. Werkzeug

Die zweite Herausforderung ist es, den Ansatz durch entsprechende Werkzeugunterstützung skalierbar zu machen. In diesem Vortrag werden wir das Werkzeug ProR[1] mit einer entsprechenden Anpassung vorstellen. ProR ist Teil des Eclipse Requirements Modeling Frameworks, einer Open Source Werkzeugplattform, dessen Datenmodell auf dem Requirements Interchange Format (ReqIF) basiert [JG12]. Um eine Traceability zu formalen Modellen zu ermöglichen, wurde ein Integrations-Plug-in entwickelt, das ProR mit Rodin integriert. Rodin[2] ist ein quelloffenes, Eclipse-basiertes Modellierungswerkzeug, dass den Event-B-Formalismus unterstützt.

Die integrierte ProR-Rodin-Werkeugplatform ermöglicht es, textuelle Anforderungen zu erfassen und der hier beschriebenen Methodik entsprechend zu klassifizieren. Einige rudimentäre Konsistenzchecks werden in Echtzeit vorgenommen. Bspw. werden die erkannten Phänomene farblich hervorgehoben, und noch nicht definierte Phänomene entsprechend markiert.

Um anspruchsvollere Konsistenzeigenschaften zu überprüfen, wird die Systembeschreibung in ein Modell überführt, das dann mit dem integrierten ProB Model-Checker analysiert wird, um Inkonsistenzen zu finden.

## III. Ergebnis

Im Gegensatz zu manchen akademischen Ansätzen erfordert unsere Methode es nicht, einen neuen Entwicklungsprozess einzuführen. Statt dessen können diese Konzepte schrittweise in einen bestehenden Prozess eingearbeitet werdenwas dann unmittelbar einen Mehrwert bietet. Bei einer vollen Umsetzung ist eine konsistente Nachverfolgbarkeit in ein formales Modell gewährleistet, was sicherheitsrelevante Standards wie ISO 26262 oder IEC 61508 unterstützt.

Dieser Vortrag wird interaktiv gestaltet, indem schrittweise eine kleine Systembeschreibung entwickelt wird. Theoretischen Grundlagen werden beschrieben und deren Umsetzung im Werkzeug demonstriert.

## References

[GJGZ00] Carl A. Gunter, Michael Jackson, Elsa L. Gunter, and Pamela Zave, *A Reference Model for Requirements and Specifications*, IEEE Software **17** (2000), 37–43.

[HJL12] Stefan Hallerstede, Michael Jastram, and Lukas Ladenberger, *A Method and Tool for Tracing Requirements into Specifications*, Submitted to Science of Computer Programming, 2012.

[Jas12] Michael Jastram, *The ProR Approach: Traceability of Requirements and System Descriptions*, Inaugural-Dissertation, CreateSpace, 2012.

[JG12] Michael Jastram and Andreas Graf, *ReqIF – The New Requirements Standard and its Open Source Implementation Eclipse RMF*, Tech. report, Commercial Vehicle Technology Symposium, 2012.

[JHL11] Michael Jastram, Stefan Hallerstede, and Lukas Ladenberger, *Mixing Formal and Informal Model Elements for Tracing Requirements*, AVOCS 2011, 2011.

[1] http://eclipse.org/rmf/pror
[2] http://www.event-b.org/