Rudolf Hauber

(Rudolf.Hauber@HOOD-Group.com) betreut bei der HOOD Group als Senior Consultant das Thema Anforderungsmodellierung und ist dort für den Bereich Aerospace und Defense zuständig. Zuvor leitete er bei Kölsch & Altmann Software und Management Consulting GmbH den Bereich Technologietransfer. Als langjähriger IT-Berater arbeitete er in einer Vielzahl von Großprojekten, als Trainer schult er Modellierungsmethoden und Prozesse und veröffentlichte zahlreiche Artikel und Konferenzbeiträge. Sein Arbeitsschwerpunkt ist Modellierung und System Engineering. Er begann seine Karriere als Wissenschaftler im OO-Bereich an der Universität München

Susanne Mühlbauer

(Susanne.Muehlbauer@HOOD-Group.com) ist Senior Consultant bei der HOOD Group. Neben ihrer Beratungstätigkeit im Bereich Requirements Engineering & Management Methoden und Prozesse beschäftigt sie sich unter anderem mit Software-Entwicklungsprozessen und Use Case Modellierung. Weitere Schwerpunkte Ihrer Tätigkeit sind die Themen Configuration, Change und Version Management. Zusätzlich zu Beratungsprojekten im Software-Engineering-Umfeld hatte sie in der Vergangenheit die Projektleitung für Implementierungsprojekte im Bereich kundenspezifischer Softwareentwicklung und ERP-Systeme. Sie hat verschiedene Artikel und Studien zum Thema Collaboration Software und Anforderungsmanagement veröffentlicht.

Modellierung und Requirements Management – Ein starkes Team

In der Industrie ist der Einsatz von Modellierung fest etabliert, ebenso gilt das für Requirements Management. In der Praxis scheinen diese beiden Disziplinen jedoch oft isoliert voneinander zu arbeiten. Dabei kann eine starke Verzahnung von Requirements Management und Modellierung der entscheidende Erfolgsfaktor sein.
Ein methodisch integrierter Software-Entwicklungsprozess ermöglicht es, die jeweiligen Stärken von Modellierung und Requirements Management deutlich synergetischer zu nutzen.
Der Artikel zeigt verschiedene Lösungsansätze, wie auf Basis einer Geschäftsprozessmodellierung Anforderungen an ein Software-System mittels Modellierung abgeleitet werden können. Weiterhin zeigen wir Ihnen auf, wie eine methodisch korrekte und tool-übergreifende Traceability erreicht werden kann – und geben einen Einblick in mögliche Toolunterstützung.

Ausgangssituation

Modellierung und Requirements Management – heute immer noch Insellösungen

Vor allem in sicherheitskritischen Entwicklungen, für nicht-funktionale Anforderungen und in größeren Software-Projekten ist die Notwendigkeit von Requirements Management unbestritten. Requirements Management wird dabei vor allem in der Anfangsphase von Projekten verwendet, um die Anforderungen an ein System zu spezifizieren.

Der Einsatz von Modellierung ist in der Software-Entwicklung fest etabliert für die Software-Analyse und -Design. Modellierung wird in Unternehmen ferner auch für die Geschäftsprozessdokumentation verwendet. In der Praxis sehen wir leider, dass es keine Durchgängigkeit von der Geschäftsprozesserfassung über die Anforderungsspezifikation bis hin zur Software-Entwicklung gibt.

Welche Probleme ergeben sich daraus?

Ziel ist es, Entwicklungsprojekte erfolgreich durchzuführen, wobei wir „Erfolg als die Erfüllung der Anforderungen“ definieren. Anforderungen des Endanwenders lassen sich aber nur erfüllen, wenn diese auch allen Projektbeteiligten bekannt sind und für alle verständlich sind.

Wenn die Geschäftsprozessmodellierung von der Anforderungsspezifikation und der Software-Modellierung getrennt ist (oder gar nicht durchgeführt wird) und für die einzelnen Disziplinen auch noch verschiedene Sprachen (Notationen) verwendet werden, ist ein gemeinsames Verständnis für das zu entwickelnde System nur schwer zu erreichen. Insbesondere dort, wo man mit Schnittstellen zwischen vielen Projektbeteiligten umgehen muss, ist ein gemeinsames Verständnis jedoch essenziell: Fachabteilungen, Systemanalytiker, Software-Architekten und –Designer, Integratoren und Tester .

Ein weiterer Nachteil dieses Methodenbruches ist die fehlende Nachvollziehbarkeit zwischen Anforderungen, den Geschäftsprozessen, die durch die Lösung unterstützt werden sollen, und letztlich der Modellierung der Lösung; d. h. des IT-Systems. Die fehlende Durchgängigkeit macht es unmöglich sicherzustellen, dass die erstellte Lösung auch tatsächlich zum Erfolg, also der Erfüllung der Anforderungen führt.

Leider zeigen sich diese Auswirkungen üblicherweise erst in der Integration und beim Systemtest oder sogar erst bei der Einführung des IT-Systems.

Ein methodisch integrierter Software-Entwicklungsprozess ermöglicht es jedoch, die jeweiligen Stärken von Modellierung und Requirements Management deutlich synergetischer zu nutzen. Ein Vorteil dieser Integration ist die Nachvollziehbarkeit von Anforderungen aus den Geschäftsprozessmodellen und damit das verbesserte

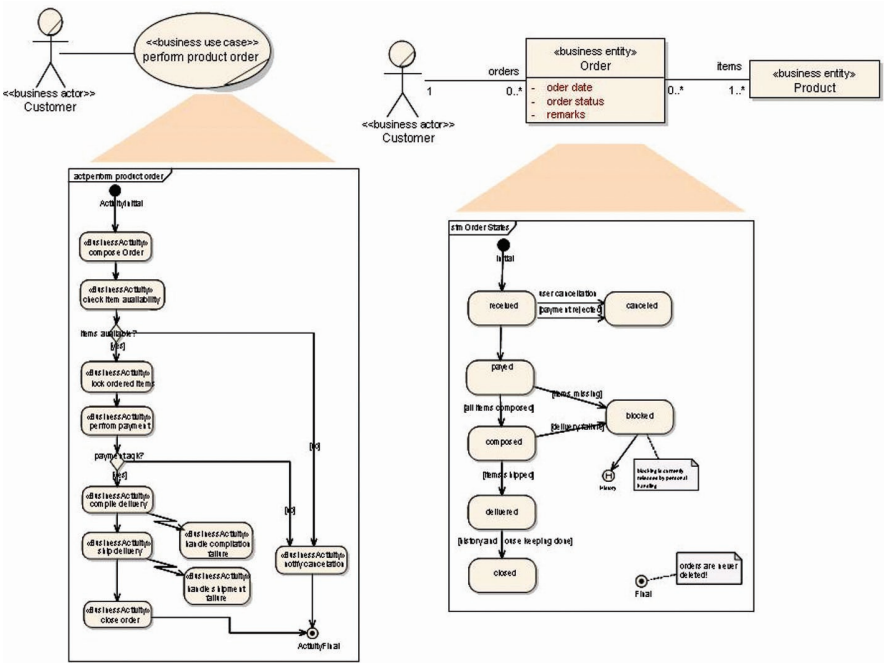

Abbildung 1: Geschäftsprozessmodellierung mit UML

Verständnis zwischen Fachabteilungen und Entwicklung.

Ein weiterer, wesentlicher Nutzen besteht in der Ableitbarkeit der Architekturmodelle aus den Anforderungen und damit einer verbesserten Grundlage für die Erweiterbarkeit, Skalierbarkeit und Wartbarkeit. Bleiben Fragen wie „Wo kommen Requirements her, wie sehen die zugrunde liegenden Geschäftsprozesse/ operationellen Abläufe aus? Wie werden diese durch die IT-Systeme unterstützt? Welche Auswirkungen auf das Design haben die Requirements?“ unbeantwortet, führt das zu Problemen beim Verständnis der Problemlage des Kunden und deshalb auch zu Problemen bei der Entwicklung, der Integration und der Abnahme des Systems. Die Umsetzung einer möglichst guten Prozessintegration, d. h. wie Modelle in textuelle Anforderungen zu übersetzen oder Modelle in Anforderungsdokumente einzubinden sind, kann der Schlüssel zum Erfolg sein.

Wie kann eine methodisch saubere und tool-gestützte Integration aussehen? Im Folgenden zeigen wir verschiedene Lösungsansätze, wie

- das Modell als Kommunikationsmittel für die Schaffung eines gemeinsamen Verständnisses genutzt werden kann
- mittels Modellierung auf Basis einer Geschäftsprozessmodellierung Anforderungen an ein Software-System abgeleitet werden können
- eine saubere und toolübergreifende Nachvollziehbarkeit zwischen Anforderungen, Modellen und Lösung etabliert werden kann.

Modelle als Kommunikationsmittel

Bei der Modellierung von Geschäftsprozessen geht es darum, das Wissen über die Abläufe in einem Unternehmen zu erfassen und adäquat zu dokumentieren. Diese Modelle sind eine exzellente Basis für Prozessverbesserungsmaßnahmen und stellen ein hervorragendes Kommunikationsmittel zwischen Fachabteilungen und IT-Abteilungen/ Dienstleistern dar. Sie ermöglichen es den Beteiligten, die für eine Software-Lösung relevanten Teilprozesse zu ermitteln und auf deren Basis die weitere Spezifikation der zu erstellenden Software voranzutreiben.

Die Notation der Geschäftsprozesse ist in der Regel sowohl für die Business-Seite als auch für Techniker leicht erlernbar und verständlich. Anders verhält es sich bei Modellen und Diagrammtypen, die in der Software-Modellierung verwendet werden. Gerade aus Fachbereichs-Sicht besteht hier in der Regel weniger die Bereitschaft, sich mit formalen Methoden wie UML detailliert auseinanderzusetzen. Letztlich landet man dann bei den klassischen Fachkonzepten, die versuchen, alle Sachverhalte in natürlicher Sprache auszudrücken.

Allerdings liegt gerade hierin ein enormes Verbesserungspotenzial, wenn einzelne Diagramme aus den Softwaremodellen in Lastenheften referenziert werden können, die auch von der Fachabteilung verstanden werden (siehe hierzu **Abbildung 4**). Man spart sich dadurch, denselben Sachverhalt mehrfach in unterschiedlichen Notationen beschreiben zu müssen, was letztlich nur dazu führt, dass Redundanzen entstehen, Inkonsistenzen auftreten und der Pflegeaufwand sich erhöht.

Abbildung 1 zeigt den Ausschnitt eines Geschäftsprozessmodells basierend auf dem UML Business Modelling Profile. Geschäftsprozesse werden als Business Use Cases dargestellt, deren Workflow durch Business Activities näher spezifiziert wird (**linke Seite der Abbildung 1**). Informationen/Geschäftsobjekte werden durch Business Entities dargestellt. Status eines Geschäftsobjekts können durch State Charts dargestellt werden (**rechte Seite der Abbildung 1**).

Ableitung von Anforderungen

Aus Geschäftsprozessmodellen können sehr gut Anforderungen an Software-Systeme,

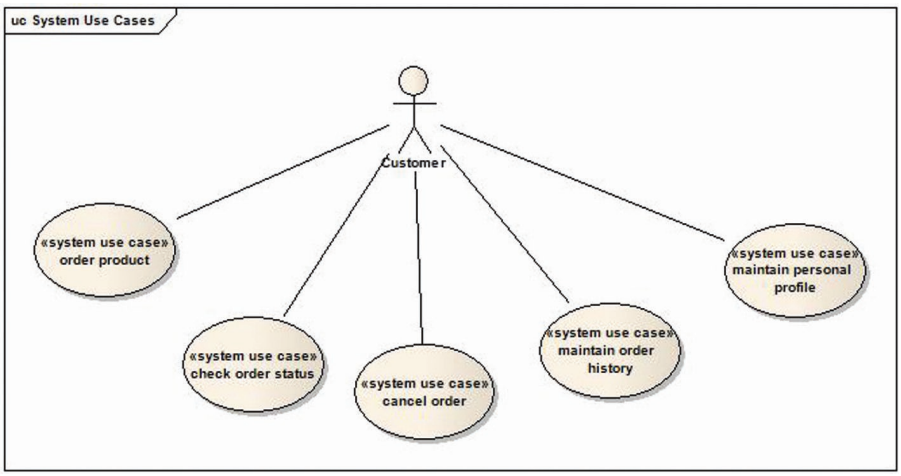

Abbildung 2: System Use Cases

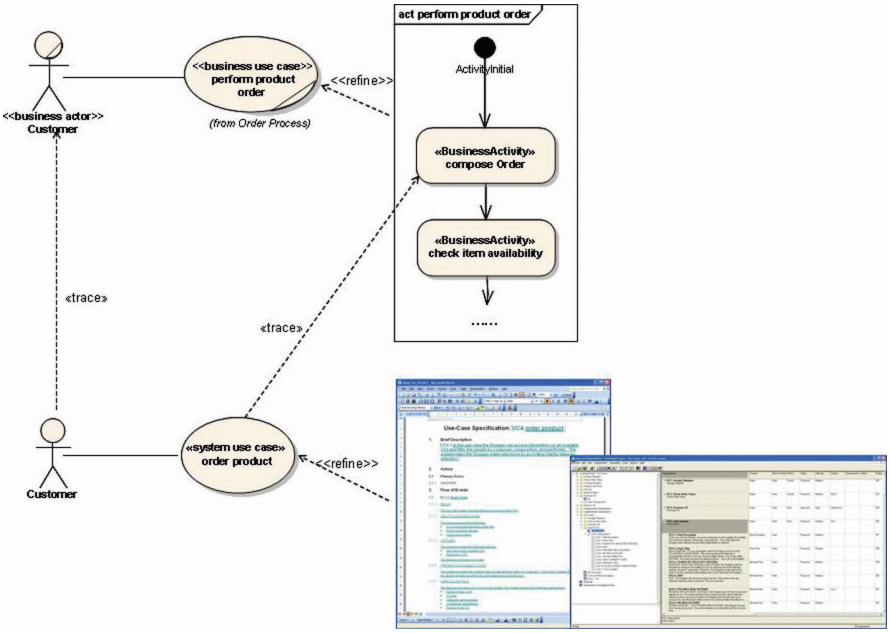

Abbildung 3: Use Cases und Requirements

die die Geschäftsprozesse automatisieren, abgeleitet werden. Betrachten wir beispielsweise **Abbildung 1**. Der Business Use Case *perform product order* aus **Abbildung 1** kann für einen Online-Bestellservice zu einer Reihe von System Use Cases für den Kunden führen (siehe **Abbildung 2**).

Dabei können einzelne Business Activities (wie beispielsweise *compose Order* aus **Abbildung 1**) zu einem eigenen System Use Case führen (hier *order product*). Die üblicherweise textuelle Beschreibung des System Use Cases definiert dann im Detail den Ablauf. Diese Use Case Beschreibung stellt somit die Anforderungsspezifikation dar und sollte daher in einem Requirements-Management (RM) Tool verwaltet werden (**Abbildung 3**).

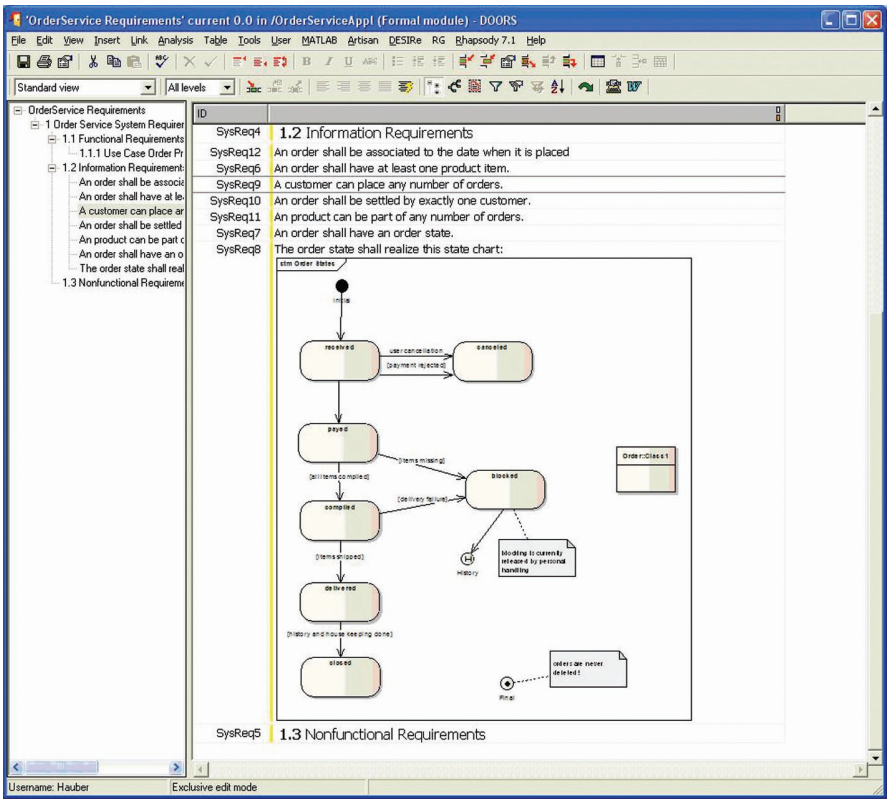

Abbildung 4: Abgeleitete Anforderungen

Werden Business Activities nicht im Rahmen des eigenen Geschäftskontexts ausgeführt, sondern an einen externen Dienstleister delegiert, kann dies im Modell durch tagged values wie *{executed by = supplier}* dargestellt werden. Die Business Activity *ship delivery* wird beispielsweise durch einen Zustelldienst realisiert. In diesem Fall sind dann nur die Prozessschnittstellen in dem Software-System umzusetzen (z. B. der Versandauftrag und der Status der Order).

Neben den rein geschäftsprozessorientierten Aspekten sind Geschäftsinformationen und Regeln ein weiterer wichtiger Punkt. Auch hier können aus den Geschäftsprozessmodellen direkt Anforderungen abgeleitet werden.

Generell kann das Zusammenspiel zwischen Modellen und Anforderungen konkret erfolgen über das

1. Einbinden von Modellen/ Diagrammen in Anforderungsdokumente oder
2. Ableiten von Anforderungen aus Modellen

Aus dem Business Entity Diagram in **Abbildung 1** kann nun das Modell als solches in die Anforderungsspezifikation für das Software-System als Anforderung eingebettet werden oder das Modell in mehrere natürlich-sprachliche Anforderungen transformiert werden. Die Anforderung SysReq8 in **Abbildung 4** zeigt den ersten Ansatz mit der Einbindung des State-Chart-Diagramms aus **Abbildung 1**. Die anderen Anforderungen aus **Abbildung 4** sind aus dem Business Entity Modell der **Abbildung 1** abgeleitete natürlich-sprachliche Anforderungen gemäß des zweiten Ansatzes.

Dabei ist darauf hinzuweisen, dass Ansatz 1 fehlerträchtige Redundanzen der Transformation in natürlich-sprachliche Anforderungen vermeidet, jedoch das Qualitätsmerkmale Atomarität von Anforderungen nicht erfüllt, da das Diagramm mehrere Anforderungen enthält.

Welchen Ansatz man verfolgt, sollte im Vorfeld der Spezifikation anhand der gegebenen Rahmenbedingungen und Teampräferenzen festgelegt werden.

Eine weitere Quelle für die Ableitung von Anforderungen - insbesondere für nichtfunktionale Anforderungen (z.B. Mengengerüste für Daten, Performance, Verfügbarkeit, Sicherheit, etc.) - sind die Business Goals. Hierüber lassen sich die Auswirkungen zentraler Anforderungen (sog. „Architekturtreiber“) auf die Systemarchitektur sichtbar machen (**siehe Abbildung 5**).

Traceability

Wie man bereits in Abbildung 5 sieht, ist es wichtig die Traceability zwischen Diagrammen und Anforderungen, Modellen und Anforderungsspezifikationen herzustellen.

An dieser Stelle sei auf die Idee von Architecture Frameworks wie [DoDAF] oder [TOGAF] hingewiesen (siehe Abbildung 6). Diese ermöglichen es, komplexe Architekturen durch die Erfassung verschiedener miteinander verlinkter Sichten darzustellen und darüber verständlich zu machen.

Innerhalb dieser Frameworks sind die Geschäftsziele die Basis jeglicher Aktivität. Aus diesen leitet sich der operationelle Kontext - die Geschäftsprozesse, Organisationsstruktur, Standorte, Rollen und Ressourcen ab (TOGAF Business Architecture bzw. DoDAF Operational Views). Die Geschäftsregeln und Informationen werden umgesetzt durch Systemfunktionen, Daten und Software-Systeme, die auf Systemplattformen an verschiedenen Standorten laufen. Dies wird in den DODAF Systems Views bzw. in der TOGAF Application Architecture und Data Architecture abgebildet. Verwendete Standards, Konventionen und Technologien, von denen das zu entwickelnde IT-System betroffen ist, finden durch die Technical Views bzw. Technology Architecture Eingang. Die verschiedenen Sichten und deren Elemente stehen über Links miteinander in Verbindung, so dass sie durchgängig nachverfolgbar sind.

Abbildung 7 zeigt die Nachverfolgbarkeit der operationellen Sichten auf die System-Sichten:

- wie (how) eine Aktivität umgesetzt wird (durch eine Funktion),
- wo (where) diese Funktion abläuft (in welchem System Node),
- wer (who) die Aktivität durchführt (Rolle auf Business-Seite, System auf System-Seite) und
- welche Informationen oder Daten (what) bearbeitet werden sollen

Ein einfaches Beispiel für die durchgängige Traceability des Informations- auf das Datenmodell ist in Abbildung 8 zu sehen.

Abbildung 9 zeigt die weiterführende durchgängige Traceability von Geschäftsprozessmodellen über Software-Anforderungen zu Software-Modellen und in den Code. Dem stehen im Sinne des V-Modells der linke Nachweis-Ast mit den verschiedenen Teststufen gegenüber, bei denen ebenfalls eine Nachvollziehbarkeit auf die An-

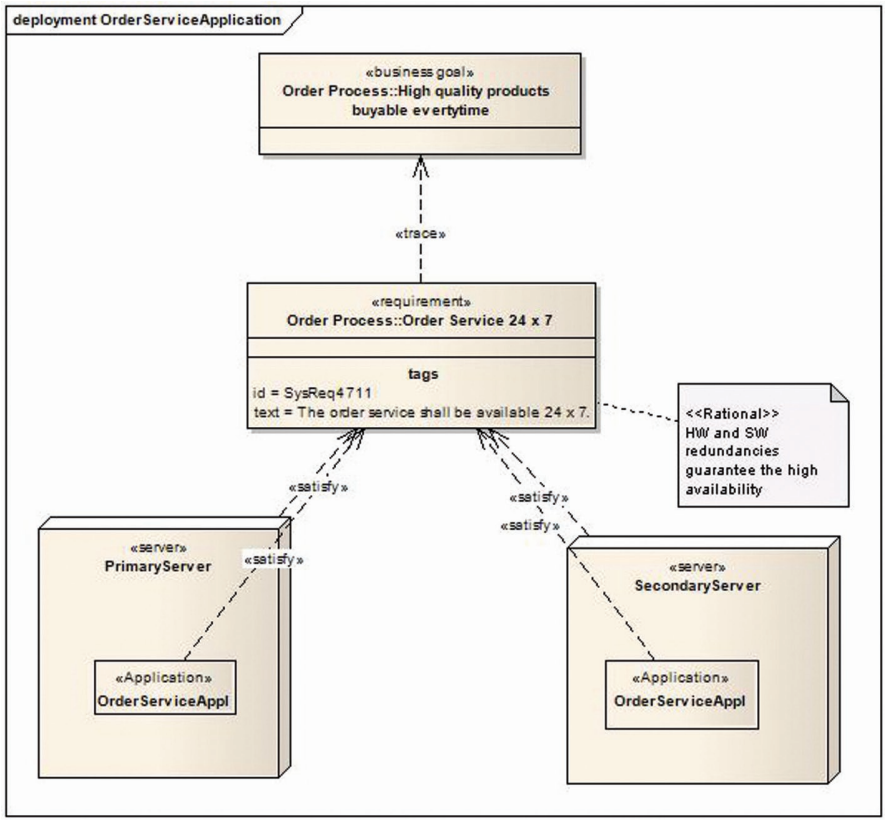

Abbildung 5: Business Goals und Requirements

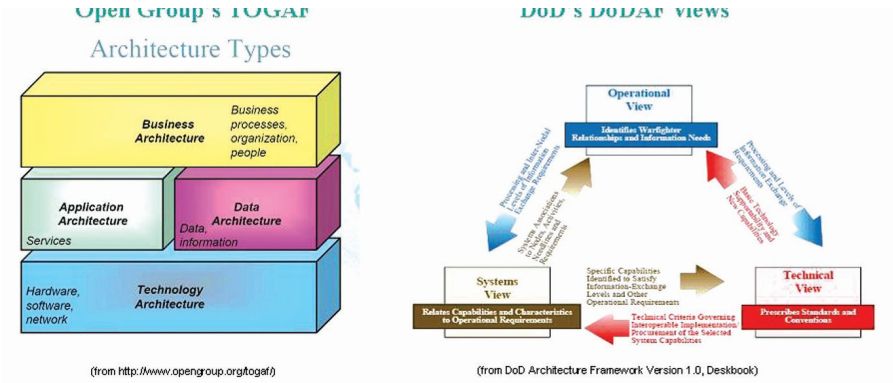

Abbildung 6: TOGAF und DoDAF Framework

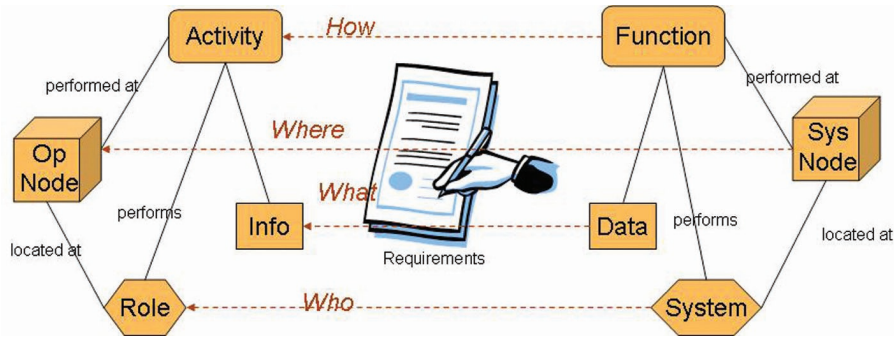

Abbildung 7: Traceability Business Elemente zu Software-Modellelementen

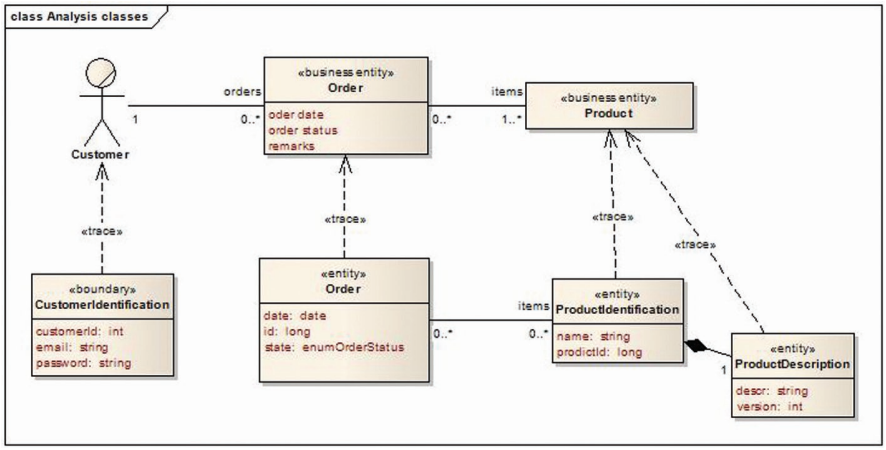

Abbildung 8: Traceability Datenmodell

forderungen und Systemabläufe vorhanden sein muss.

In einer integrierten Lösung sollte es möglich sein, zwischen den verschiedenen Darstellungen und Detaillierungsgraden zu navigieren. Dafür ist es notwendig, die Zusammenhänge der Elemente sichtbar zu machen und die Traceability herzustellen. Abbildung 9 zeigt dabei den Einsatz verschiedener SysML-Dependencies für die Darstellung unterschiedlicher Zusammenhänge zwischen Anforderungen und Modellelementen: <<deriveRqt>>, <<trace>>, <<refine>>, <<satify>>, <<verify>> (siehe [SysML]).

Nutzen der Traceability

Welche Vorteile erzielt man dadurch?

- Vollständigkeit (nichts wird vergessen)
  Traces unterstützen Sie dabei sicherzustellen, dass Sie alle Anforderungen (sei es nun aus der Anforderungsspezifikation oder aus der Geschäftsprozessanalyse) in der Lösung berücksichtigt haben.
- Notwendigkeit (nichts wird unaufgefordert realisiert)
  Traces unterstützen Sie dabei sicherzustellen, dass Sie nur die Dinge realisieren, die auch wirklich gefordert sind. Damit vermeiden Sie unnötige Realisierungsaufwände, aber vor allem auch daraus entstehende Test-, Pflege- und Wartungsaufwände!
- Auswirkungen (was ist von einer Änderung betroffen)
  Traces unterstützen Sie dabei herauszufinden, welche anderen Anforderungen oder welche Lösungsteile von einer geplanten Änderung tangiert werden und welche Auswirkungen diese Änderung auf das gesamte IT-System haben wird.
- Validierung von Anforderungen (passt die geplante Realisierung zur Anforderung)
  Traces unterstützen Sie dabei zu prüfen, ob das geplante IT-System auch wirklich die Anforderungen der Stakeholder erfüllt.

Traceability ist in vielen Fällen ein Erfolgskriterium und wird von einigen Entwicklungsstandards (insbesondere für sicherheitskritische Systeme) gefordert. In der Praxis zeigt sich, dass das nachträgliche Erstellen von Traceability viel Aufwand erzeugt oder schlichtweg nicht mehr möglich ist. Das heißt, Sie müssen dafür sorgen, dass Sie die Links unmittelbar erzeugen können. Deshalb sollten die von Ihnen eingesetzten Werkzeuge in der Lage sein, Sie beim Erstellen von Links zu unterstützen. Darüber hinaus müssen die Werkzeuge ein Navigieren durch die Elemente ermöglichen und auch Auswirkungsanalysen erlauben.

Je besser der Schritt des Erstellens von Links in die Umgebung des Erstellens von Entwicklungsartefakten integriert ist, desto einfacher gelingt es, sofort und mit reduziertem Aufwand Traceability aufzubauen und zu pflegen. Der Bruch zwischen Requirements Management und Modellierungstools kann auch heute schon überwunden werden.

Toolunterstützung

Die Unterstützung, die RM-Werkzeuge und Modellierungstools bei der Verlinkung von Anforderungen und Diagrammen/Diagrammelementen bieten, ist sehr unterschiedlich ausgeprägt.

Grundsätzlich unterscheiden wir im folgenden 3 Konzepte:

- manuelle Konzepte
- unidirektionale Toolinteroperabilitäts-Konzepte
- bidirektionale Toolinteroperabilitäts-Konzepte

Manuelle Konzepte beruhen darauf, dass die Verlinkung zwischen Anforderung und Modell über eindeutige Identifier hergestellt wird. Dabei können Tags oder Labels im Modell vergeben werden und diese werden dann in den RM-Werkzeugen über eine ID referenziert oder umgekehrt (siehe Abbildung 10).

Wie bei allen manuellen Vorgängen hat man hier mit einem hohen Abgleichaufwand und hoher Fehleranfälligkeit zu rech-

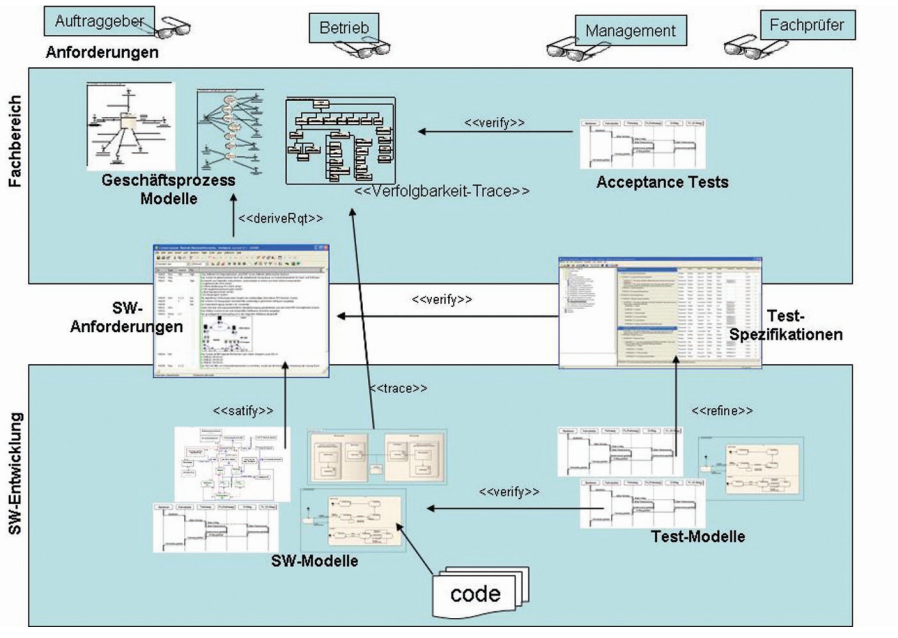

Abbildung 9: Durchgängige Traceability

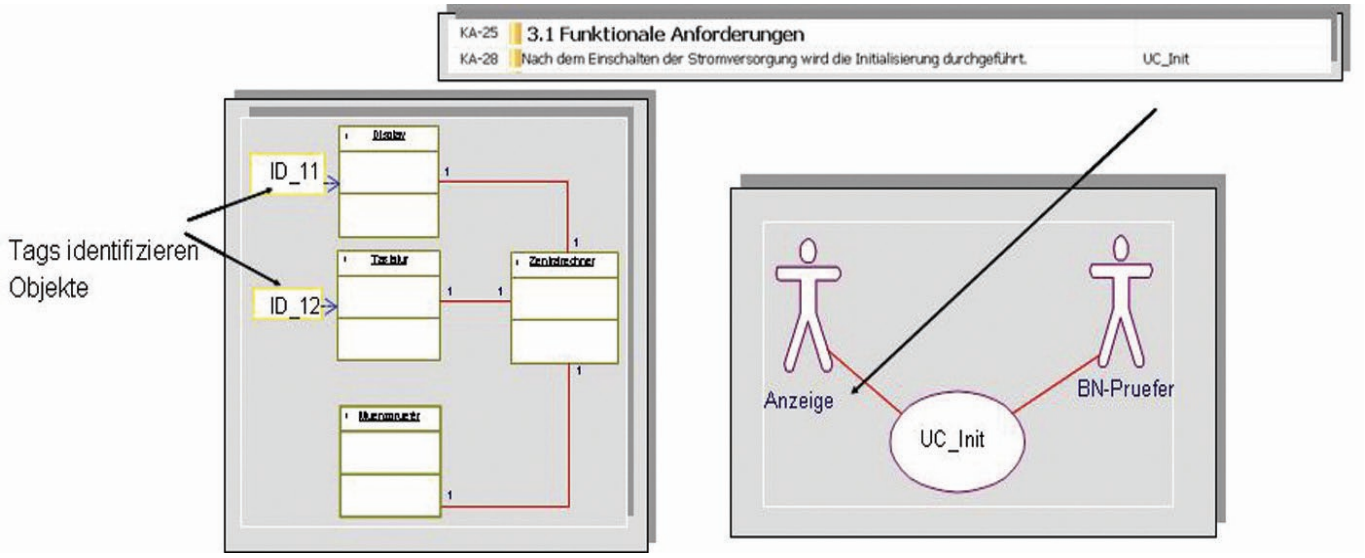

Abbildung 10: Manuelle Toolinteroperabilität

nen. Bei komplexen und umfangreichen Projekten ist ein manuelles Erstellen der Verlinkung nicht zu empfehlen.

Unidirektionale Toolinteroperabilität ermöglicht es, in einem RM-Tool Modellelemente zu verwalten oder Anforderungen in einem Modellierungstool abzubilden.

Im ersten Fall werden Modellelement-ID und -Name (und weitere Informationen) automatisiert ins RM-Tool kopiert bzw. abgeglichen. Dadurch können die Modellelemente im RM-Tool verlinkt werden und die starken Traceability-Analysefähigkeiten des RM-Tools voll genutzt werden.

Im zweiten Fall werden mittels des SysML-Modellelements <<requirement>> Anforderungen im Modellierungstool repräsentiert. Dadurch können die Auswirkungen von wichtigen Anforderungen auf die Systemarchitektur sehr gut visualisiert werden (siehe Abbildung 5).

Bidirektionale Toolinteroperabilität ermöglicht es, die Vorteile der beiden Toolinteroperabilitäts-Konzepte gleichzeitig zu nutzen. Im Idealfall sind die Tools nur Zugriffs- und Auswertungsfunktionen auf eine (virtuelle) gemeinsame Projektdatenbank.

Wichtig für die Zukunft ist, dass nicht nur proprietäre Schnittstellenformate zur Anwendung kommen, sondern diese standardisiert werden.

Zusammenfassung

Wir haben gezeigt, wie durch die Integration von traditionellem Requirements Management und Modellierung erheblicher Nutzen für die Projektarbeit gestiftet werden kann. Modelle dienen als exzellentes Kommunikationsmittel für die Schaffung eines gemeinsamen Verständnisses. Ferner können auf Basis einer Geschäftsprozessmodellierung Anforderungen an Software-Systeme klar verständlich abgeleitet werden. Letztlich kann damit eine saubere Nachvollziehbarkeit zwischen Geschäftsprozessen, Anforderungen, Software-Modellen bis in den Code und die Tests etabliert werden. Tooltechnische Lösungen dazu sind bereits vorhanden, auch wenn eine Standardisierung der proprietären Lösungen höchst wünschenswert wäre. ■

www.HOOD-Group.com

Referenzen

[DoDAF] http://www.architectureframework.com/dodaf/
[ToGAF] http://www.opengroup.org/togaf/
[SysML] http://www.sysml.org