TraCo - Entwicklung und Einführung eines Traceability Controllings zur verbesserten Qualitätssicherung im Requirements Management

Autoren: Horst Kargl, Alexandra Mazak

Wie Erfahrungen und Befragungen aus der Praxis belegen, fehlen in Unternehmen oft effiziente Abläufe, um Fehler in der Systementwicklung frühzeitig zu identifizieren. Weitere signifikante Ursachen für Fehlschläge bei der Softwareentwicklung von großen Programm Systemen sind oftmals die mangelnde „Management-Awareness“ und eine hohe Personalfluktuation (Steinpichler & Kargl, 2011). Meistens ergeben sich Probleme aus folgenden Punkten:

- Der Systemarchitekt muss das Modell sehr gut kennen, um jederzeit den Überblick über die im Modellierungsprozess getroffenen Designentscheidungen behalten zu können (zurzeit sehr intuitiv).
- Mitarbeiter scheiden vorzeitig aus dem Projekt aus und für neue Projektmitarbeiter fehlt oftmals die nötige Dokumentation, um sich rasch einen Überblick über den Systemarchitektur und die Relevanz der einzelnen Systemkomponenten im Hinblick auf die Implementierung verschaffen zu können.
- Oftmals wird schon zu Beginn der Analysephase eine Lösung „anmodelliert“, wodurch lösungsspezifische Festlegungen und Einschränkungen voreilig getroffen werden, die in der Anforderungsliste nirgendwo gefordert sind.
- Die Management- bzw. Projektleiter-Ebene wird nicht bewusst in den Prozess der Überleitung von Kundenanforderungen in Systemanforderungen involviert, wodurch es vermehrt zu Fehlinterpretationen hinsichtlich der Umsetzung kommt.

Ein für den gesamten Projektzyklus durchgehender Nachweis der Anforderungsabdeckung ist ein gängiger Standard und in Bereichen mit höchstem Sicherheitsbedarf (Medizin-, Flug-, und Raumfahrtechnik) zwingend vorgesehen (Pohl, 2008). Der Fokus im „Requirements Traceability“, das ein wesentliches Teilgebiet des Requirements Managements darstellt, liegt in der lückenlosen Nachvollziehbarkeit von Anforderungen. Requirements Traceability stellt heute einen Schlüsselfaktor im Software Projektmanagement dar und ist ein wichtiger Teil in der Qualitätssicherung im Software Engineering (Pohl & Rupp, 2011). Es wird eingesetzt um sicherzustellen, dass alle Anforderungen hinreichend in der Spezifikation berücksichtigt wurden (Validierung) und in die Umsetzung richtig einfließen (Verifikation). Es fungiert als Gradmesser der Systementwicklungsreife. Zielsetzung ist es, komplexe Systeme effizient und fehlerarm zu entwickeln.

Lücke in verfügbaren Werkzeugen aufgedeckt

Gängige Requirements-Management-Tools bieten die Möglichkeit zur Erfassung, Analyse, Spezifizierung und Validierung von Anforderungen und ermöglichen so eine standardisierte Form des Requirements Managements. Dabei findet vor allem die „Traceability Matrix“, erstmals eingeführt von (MacMillian & Vosburgh, 1986), Anwendung. Dieses Konstrukt ist eine Art „Completeness Indicator“, das die Verlinkung von Design-Artefakten mit den entsprechenden Anforderungen visualisiert.

Das Framework Scrum (Kniberg, 2007) bietet ein iteratives und inkrementelles Vorgehensmodell zur Risikokontrolle und nachhaltigen Entwicklung komplexer Systeme. Scrum basiert auf der Theorie der empirischen Prozesssteuerung. Man geht von der Annahme aus, dass Wissen aus Erfahrung gewonnen wird und dass Entscheidungen aufgrund von bekannten Fakten getroffen werden. Die Prozesssteuerung stützt sich auf drei Säulen: (1) Transparenz, (2) Überprüfung und (3) Anpassung.

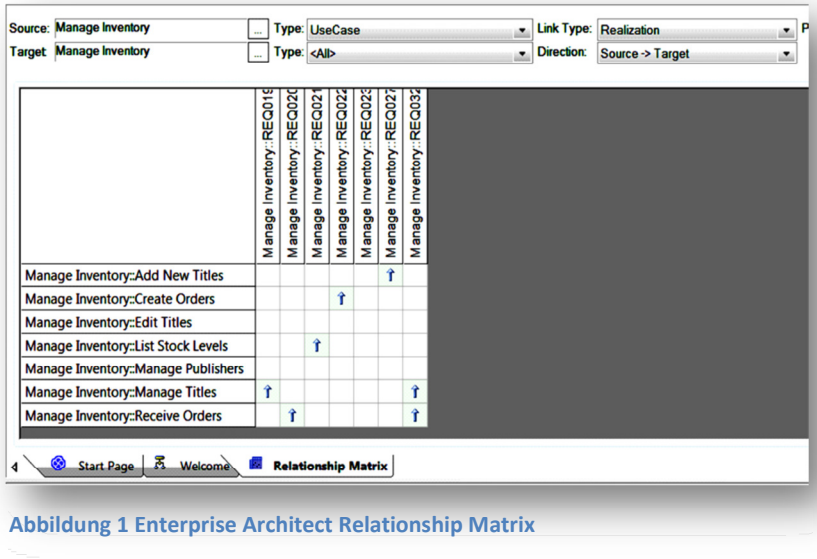

Abbildung 1 Enterprise Architect Relationship Matrix

Aktuell zeigt sich jedoch, dass eine abgesetzte Führung von Requirements-Systemen neben der Modellhaltung (z.B. durch parallel laufende nicht-integrierte Tools) ein aufwendiges, kostenintensives und vor allem fehleranfälliges Vorgehen ist (Steinpichler & Kargl, 2011). Die auf aktuelle Literatur basierende Recherche hat gezeigt, dass die Traceability Matrix (siehe Abbildung 1) ein geeignetes Werkzeug ist um die Abdeckung der Anforderungen nachzuvollziehen. Es kann daraus jedoch nicht die Relevanz der Verlinkung zwischen Designelementen und Anforderungen abgeleitet werden, um so beispielsweise die Signifikanz der Elemente im Hinblick auf die Implementierung zu evaluieren.

Um der Anforderung nach „Relevanzinformation“ von Verlinkungen nachzukommen, haben wir einen kognitionsbasierten heuristischen Ansatz entwickelt. Der Ansatz trägt den Namen Traceability Controlling (TraCo) und wurde im Modellierungswerkzeug Enterprise Architect (EA) von Sparx Systems integriert.

Entwicklung eines Traceability Controllings (TraCo)

In TraCo werden kognitive Techniken in den Bereich Requirements Management in der frühen Phase der Software und System Entwicklung integriert (siehe Abbildung 2). Wir gehen dabei der Frage nach, wie sich Nutzen und Qualität im Kontext der Systemmodellierung operationalisieren lassen. Der Fokus liegt dabei auf der Entwurfs- und Designphase in einem Software Entwicklungsprojekt und auf der Qualitätssicherung der, in der Phase der Lösungsfindung und Lösungsspezifikation, vom Modellierer getroffenen Designentscheidungen. TraCo ist ein phasenbezogenes Monitoring-Verfahren zur kontinuierlichen Überprüfung der Entwicklungsfacette. Es nimmt Bezug auf jene Kontextaspekte, die die Entwicklung des

Struktur	Anforderung	Realisiert durch
Kapitel A		
	FA 001	Komponente X
	FA 002	Komponente Y
	NFA 01	Komponente Z
Kapitel B		
	FA 003	Komponente Z
	NFA 02	Komponente X
	NFA 03	Komponente Y

Abbildung 2 Realisierungen zwischen Kundenanforderungen und Designkomponenten

Systems nachhaltig beeinflussen. So kann validiert werden, ob das Architekturmodell bzw. feingranular die Systembausteine/Design Artefakte im Modell in entsprechender Qualität vorliegen und ob diese den Ansprüchen des Kunden genügen. Die speziellen, qualitätsbezogenen Kontextkriterien basieren auf der abstrakten Qualitätsnorm ISO/IEC 25010 – System and software quality models. Generell können Kontexte aber auch anwenderspezifisch generiert werden.

In TraCo wird die Integrationssicht auf Architektur- und Designebene (Sicht des Modellierers) mit der Kundensicht auf Anforderungsebene gekoppelt berücksichtigt. Die Überprüfung der Umsetzung von Kundenanforderungen und Qualitätsmerkmalen im Architektur-/Designmodell ist für die Einschätzung der Produktqualität erforderlich und dient zudem als Entscheidungsgrundlage für das Projektmanagement. Grundsätzlich kann TraCo aber auch in jeder anderen Phase der Systementwicklung verwendet werden, wie beispielsweise in der Implementierungs- und Integrationsphase auf Komponentenebene. Hier liegt der Fokus auf der Umsetzung der Systembausteine in Code. Zentral ist dann die Entwicklersicht im Entwicklungskontext.

Kernidee in TraCo ist es, implizites Wissen (die Wahrnehmung/Intention der Modellierer zum Zeitpunkt des Systemdesigns) zu formalisieren, indem dieses situative und kontextbezogene Wissen explizit in Zahlen gegossen wird. Auf Grund der Zahlendarstellung wird eine Vergleichbarkeit zwischen den Modellierern und ihrer individuellen Sicht auf das Architekturmodell hergestellt. In TraCo wird „qualitätsbezogener Kontext“, ausgehend von der menschlichen Wahrnehmung, analysiert und als Meta-Information im Modell hinterlegt. Die dabei im jeweiligen Kontext gemessene Eigenschaft der Wahrnehmung ist die qualitative Relevanz der Realisierungen (siehe Abbildung 3). Das bedeutet, dass der Modellierer den Effekt quantifiziert, den eine Realisierung im

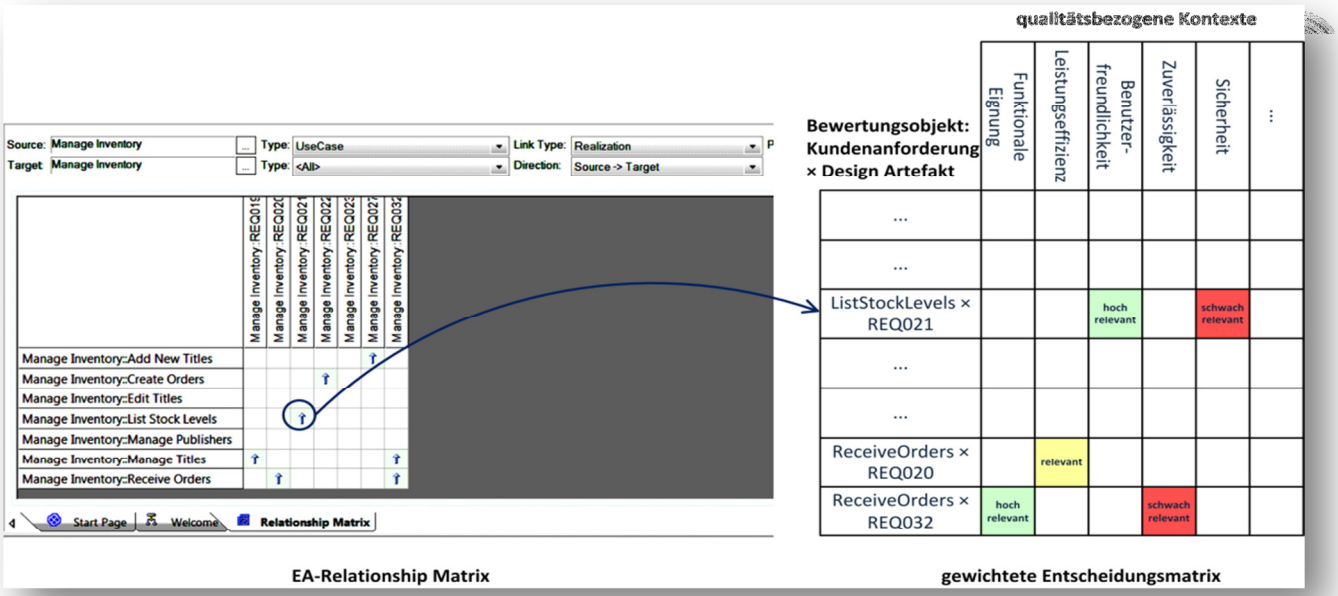

Abbildung 3 Überleitung EA-Relationship Matrix in gewichtete Entscheidungsmatrix

momentanen Kontext hat.

Relationship Matrix

Im Enterprise Architect werden die Beziehungen von Kundenanforderungen und Systembausteinen in der Enterprise Architect Relationship Matrix visualisiert. Je nach Modellierungssprache wird ein

spezieller Verknüpfungslink verwendet (UML::Realization, SysML::Satisfy). Durch die Verlinkung wird eine Trace-Kette aufgebaut, die eine Impact-Analyse erlaubt und bei Änderungen von Anforderungen oder Design Artefakten eine wertvolle Informationsquelle darstellt. Abbildung 1 zeigt einen Ausschnitt der Matrix. In den Kreuzpunkten wird eine vorhandene Kopplung in Form eines Symbols (Pfeil) dargestellt.

Die Matrix hilft dabei nachzuvollziehen, welche Kundenanforderungen durch welche Systembausteine umzusetzen sind. Der Modellierer kann so beispielsweise überprüfen, ob alle Kundenanforderungen in der Lösung berücksichtigt werden (Vollständigkeit) und ob die Anforderungen an das System gemeinsam erfüllbar sind (Konsistenz). Die Matrix hilft, die Menge an Anforderungen überschaubar zu halten und Kopplungen nachvollziehbar zu tracken. Allerdings wird durch diese Art der visualisierten Realisierungen lediglich dargestellt, dass eine Anforderung von einer anderen Anforderung abgeleitet ist bzw. durch einen Systembaustein umgesetzt werden muss. Es kann keine Information herausgelesen werden, inwieweit die Umsetzung schon vorangeschritten ist (Umsetzungssicht), noch wird die Information berücksichtigt warum eine Kundenanforderung mit einer Designkomponente verknüpft wurde. Das heißt, die Relevanz der Kopplung aus Integrationssicht — beispielsweise im Hinblick auf inhaltliche Qualitätsaspekte (Notwendigkeit, Auswirkung, Korrektheit, Adäquatheit) — bleibt unberücksichtigt. Das bedeutet, dass das Wissen, welches der Modellierer bei der Erstellung der Verlinkungen im Kopf hat (die Bedeutung der Semantik des Modells im Kontext), nicht aufgezeichnet werden kann und daher verloren geht. Dieses situative kontextbezogene Wissen ist in der Struktur des Modells implizit hinterlegt und daher nicht transparent für andere, im Projekt beteiligte, Akteure (z.B. Projektleiter, Entwickler, Kunde).

Impact

Der Einsatz von TraCo hilft auf der Ebene der Architektur und des Designs im Software Entwicklungsprozess dabei:

- Entscheidungen in Teams zu unterstützen;
- die gemeinsam tragbare Lösung zu finden und den dafür erforderlichen Zeitaufwand zu minimieren;
- die Entscheidungsfindung und das Ergebnis nachvollziehbar zu machen;
- eventuell Inkonsistenzen in der Entscheidungsfindung aufzudecken;
- subjektive „Bauch-Entscheidungen“ zu überprüfen und zu ergänzen;
- qualitative Gewichtungsentscheidungen herauszuarbeiten;
- Differenzen im Bereich der Systemauffassung/des Systemverständnisses im Team aufzuzeigen;
- eine End-Entscheidung und Bewertung der Modellstruktur strukturiert darzustellen.

Heuristischer Ansatz

Aufwändige Qualitätsanalysen finden angesichts eines oft hohen Projektdrucks kaum Akzeptanz. Gerade in Großprojekten ist es sehr schwierig, Softwarequalität neben Zeit und Budget gleichermaßen zu steuern, da kontinuierliche und umfassende Rückmeldungen über den aktuellen Status häufig fehlen (Großmann, 2010).

Im Projekt TraCo wird ein heuristischer Ansatz gewählt, um komplexe Entscheidungen vereinfacht darzustellen und dadurch transparent und nachvollziehbar zu machen. Dabei wird informelles kontextbezogenes Situationswissen – der Modellierungsfokus – formalisiert und explizit in Zahlen gegossen. Als Ergebnis erhält der Nutzer ein Grid, das Designentscheidungen um eine „Relevanz-Sicht“ erweitert wiederspiegelt. Die in TraCo umgesetzte analytische Qualitätssicherungsmaßnahme dient der Feststellung, ob die Komponenten des Architekturmodells (Systembausteine bzw. Designelemente) der Spezifikation (Verifikation) und dem Kundenwunsch (Validierung) entsprechen. Wird die gewichtete Entscheidungsmatrix (siehe Abbildung 6) von mehreren Akteuren befüllt, erhält man eine Landkarte der verschiedenen Bewertungen in unterschiedlichen Kontexten, die als Diskussionsgrundlage herangezogen werden kann, um das gemeinsame Verständnis aus den verschiedenen Sichten zu validieren (siehe Abbildung 4). Darüber hinaus erhalten die Nutzer wertvolle Kennzahlen, um beispielsweise bei Änderungen von Kundenanforderungen oder Systembausteinen die Auswirkungen auf die Architektur und vorausblickend auf die Implementierung und Integration abschätzen zu können. Es gilt dabei, die relevanten Bauteile

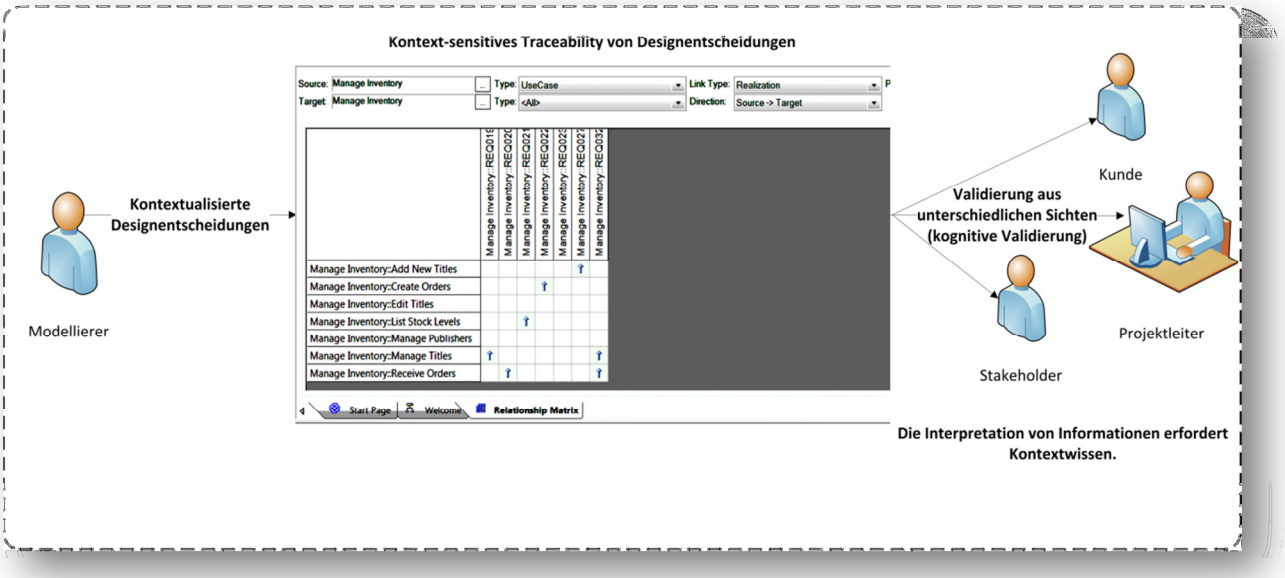

Abbildung 4 Kommunikationsgrundlage: Transparenz des Modellierungsfokus im Kontext

(innere Softwarequalität) mittels Kennzahlen zu steuern.

TraCo soll helfen, den Modellierern frühzeitig erkennen zu lassen, welche Anforderungen nicht oder nicht genügend oder aber auch zu detailliert in das Modell übernommen wurden (Erfüllung der Korrektheit). Dadurch können Realisierungen im Modell identifiziert werden, die keinen oder nur einen minimalen Nutzen stiften (Erfüllung der Notwendigkeit). Zudem können determinante Systembausteine („Business Driver“) erkannt und vorrangig bearbeitet werden. Letztere Identifizierung kann beispielsweise als Ausgangspunkt für den „Product Owner“ im „Product Backlog“ (Scrum) von Nutzen sein (Kniberg, 2007). Die Früherkennung von Designfehlern dient der Vermeidung von „Flickwerken“, die zu teuren Folgefehlern in der Phase der Implementierung Anlass geben können (Schatten, et al., 2010). Fehler können sich negativ auf die Entwicklungszeit und –kosten auswirken und bis zum Projektabbruch bzw. zu einer kompletten Neuerstellung des gesamten Produkts führen, falls diese Fehler Architektur, Entwurf oder Design betreffen (Schatten, et al., 2010).

Durch den TraCo Ansatz wird den direkt (Systemarchitekt, Requirements Engineer, Modellierer) und indirekt (Kunde, Projektleiter) am Software Entwicklungsprozess beteiligten Akteuren eine neue Kommunikationsgrundlage geboten. Dadurch soll die interne sowie externe Kommunikation zwischen den Stakeholdern im Sinne der Umsetzung einer effizienten „Requirements Negotiation“ wesentlich erleichtert und verbessert werden. Sowohl der gedankliche Prozess der Kontextkriterien (Auswahl und Erfassung der Gewichtungen) als auch deren Auswertung machen komplexe Abhängigkeiten sichtbar und führen zur Klärung von Prioritäten, gerade wenn das Ergebnis mit anderen diskutiert und hinterfragt wird. Dadurch soll das Risiko minimiert werden, dass Projekte aufgrund falsch verstandener Anforderungen scheitern.

Kognitiver Strukturierungsprozess

Das System in TraCo ist darauf ausgerichtet, dass sich der Modellierer im Sinne der (realen) Umsetzung des „Reflective Practitioner“ von Schön (Schön, 1984) selbst überwacht („Self-Monitoring“) und bei Bedarf auch selbst reguliert („Self-Regulation“). Dabei kommt es zu keinem Einsatz extern gesteuerter Tools oder dem Eingriff übergeordneter Personen, die diese Aufgabe übernehmen. Stattdessen wird jene wertvolle Ressource herangezogen, die mit dem nötigen Hintergrundwissen (Domänenwissen, Modellierungswissen) am Designprozess aktiv teilnimmt — der Modellierer selbst. Der Fokus des Modellierers ist dabei auf seine Tätigkeit im Modellierungsprozess — dem Treffen von Designentscheidungen — und den damit verbunden Objekten gerichtet.

Zu diesem Zweck wird in den Designprozess ein intern gesteuerter Kreislauf in Form eines iterativen 4-phasigen Prozesses integriert. Dieser Kreislauf dient dem Strukturierungsprozess der Wahrnehmung. Das bedeutet, dass der prozedurale Ablauf vorgibt, wie Entscheidungen strukturiert und analysiert werden. Dabei wird dem Modellierer eine kontinuierliche Evaluation seiner (Design)Entscheidungen ermöglicht. Die Kernfunktionen des sogenannten TEMoR-Kreislaufs (Transact — Evaluate — Monitor — Regulate, siehe Abbildung 5) stellen einen kontinuierlichen Verbesserungsprozess dar und entsprechen damit dem Grundprinzip des Qualitätsmanagements nach ISO 9001. Der Kreislauf wird von außen, durch eine Aufforderung des Systems motiviert, jedoch intern durch den Modellierer selbst aktiv kontrolliert und gesteuert. Der TeMoR-Kreislauf ist dem PDCA-Zyklus (Plan — Do — Check — Act) entlehnt, der hauptsächlich im Controlling Anwendung findet und seinen Ursprung im Managementsystem Kaizen hat.

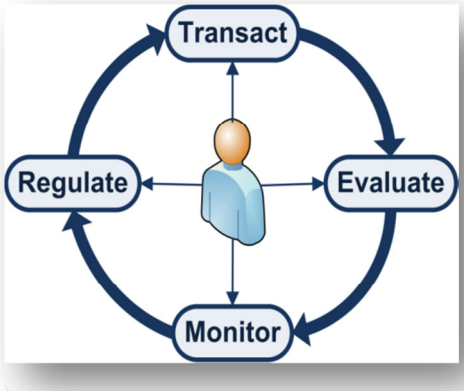

Abbildung 5 TEMoR-Kreislauf

Transact: durchzuführender Task (Aktivität: Treffen von Designentscheidungen = kognitiver Prozess bei der Überführung von Kundenanforderungen in Design Artefakte)

Evaluate: Bewertung der getroffenen Designentscheidungen unter Berücksichtigung qualitativer Kontextkriterien (Cognitive Walkthrough)

Monitor: überwachen des Modellstatus, d.h. überprüfen ob die Realisierungen den Erwartungen des Kunden entsprechen (z.B. hinsichtlich Notwendigkeit und Korrektheit)

Regulate: wenn nötig zeitgerecht und kostensparend gegensteuern

TraCo unterstützt den Anwender aktiv bei der prozessbegleitenden Auseinandersetzung mit der Qualität des Architektur-/Designmodells. Dabei wird der Modellierer kontinuierlich aufgefordert in Reflexion mit dem Modell und dessen Nutzen zu gehen. Dadurch kann rückblickend die Bewertung der Entscheidungsqualität (für künftige Entscheidungen) evaluiert werden. Das bedeutet, dass die Folgen und Auswirkungen einer Entscheidung frühzeitig erkannt und bei Bedarf abgeändert werden können. TraCo wird angewendet, um subjektive Bewertungsgrundsätze zu spezifizieren und validieren zu können.

In TraCo wird versucht Handlungen und deren Folgen nachvollziehbar zu machen, indem die kognitive Entscheidungsbildung im Modell explizit hinterlegt wird. Die Grundlage dafür bildet die jeweilige Entscheidungssituation in qualitätsbezogenen Kontexten. Implementiert wird das Verfahren in zwei Controlling Instrumente — der „Gewichteten Entscheidungsmatrix (GEM)“ und dem „Design Decision Traceability Grid (DDT-Grid)“.

Gewichtete Entscheidungsmatrix

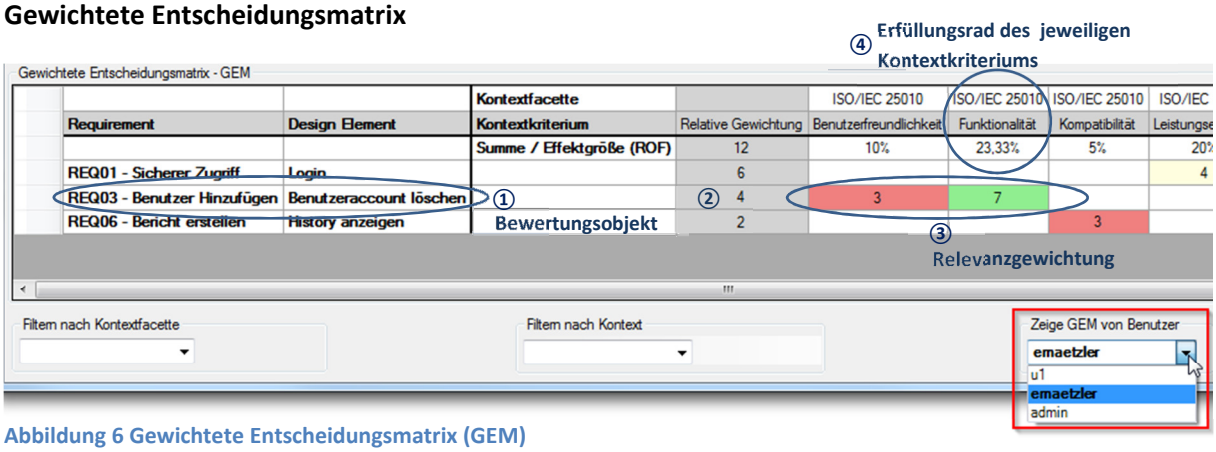

Abbildung 6 Gewichtete Entscheidungsmatrix (GEM)

Mittels der gewichteten Entscheidungsmatrix (siehe Abbildung 6) werden Design Entscheidungsprozesse kontextbezogen systematisiert und validiert. Die Bewertungsobjekte ① (Requirement × Design Element) werden zeilenbezogen jeweils in den ersten beiden Spalten der GEM aufgelistet. Die verschiedenen „Key Value Pairs“ haben ein bestimmtes Verhältnis untereinander, was ihre Gewichtung betrifft. Diese relative Gewichtung ② bestimmt sich aus dem Verhältnis zwischen den kontextbezogenen Relevanzgewichtungen pro Zeile (③ Gewichtungsvektor), in der die Kundenanforderung (Requirement) mit dem jeweiligen Design Element gekoppelt vorkommt unter zusätzlicher Berücksichtigung der jeweiligen Kundenpriorisierung. Bei der Priorisierung der Kundenanforderung handelt es sich um den sogenannten „Stakeholder Value“, der in TraCo als gegeben angenommen wird. Die

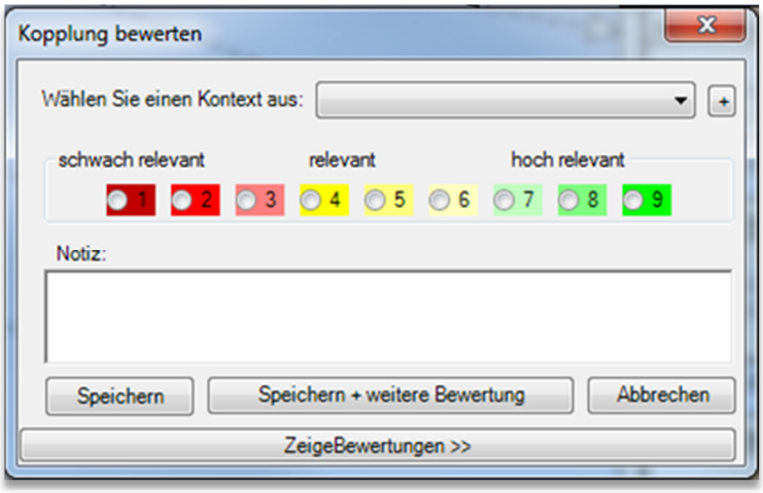

Abbildung 7 Relevanzgewichtung

Gewichtungsbewertung wird direkt vom Modellierer bestimmt und ausgeführt (siehe Abbildung 7). Sobald eine „Kopplung“ generiert wurde, wird der Modellierer automatisch vom System aufgefordert eine vordefinierten Kontext auszuwählen oder neu zu generieren und einen Wert (in Form von Punkten) zu vergeben. Die farblich unterlegte Punkteskala weist eine Bandbreite von 1 (schwach relevant) bis 9 (hoch relevant) auf. Durch den Range wird eine gesteigerte Sensitivität gewährleistet. Die Relevanzsemantik hinter dem Konzept bedeutet: je relevanter die gewählte Lösung in Bezug zu einem bestimmten Kontext ist, umso höher ist deren kontextueller und somit qualitätssichernder (z.B. Benutzerfreundlichkeit, Funktonalität, Kompatibilität) Effekt. Keine Gewichtung bedeutet, dass das Bewertungsobjekt keinen Nutzen im Kontext erfüllt. Jedes Kontextkriterium das zur Auswahl steht erhält eine eigene Spalte in der Matrix. Gegebenenfalls kann dadurch die Anzahl schnell anwachsen, nur so lassen sich die Kriterien aber sauber und differenziert betrachten.

Die Option "Validierung" ermöglicht dem Modellierer, den eingegebenen Kontext mit den bereits bestehenden Kontexten semantisch zu vergleichen. Existiert bereits ein eventuell gleichbedeutender Kontext, so erscheint ein entsprechender Hinweis (siehe Abbildung 8).

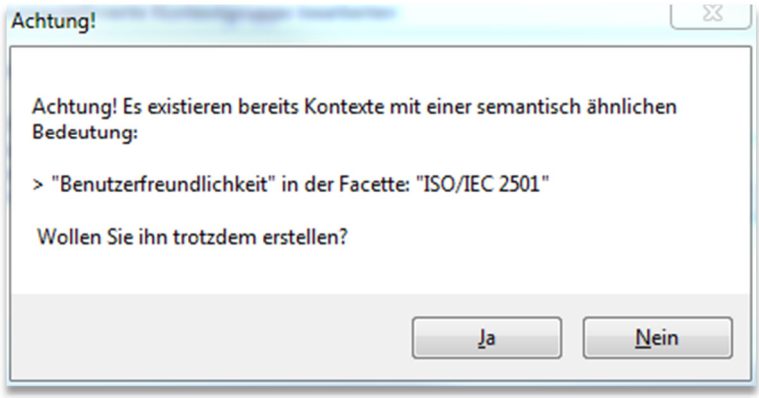

Abbildung 8 Semantische Validierung

Zudem werden die Relevanzgewichtungen herangezogen, um daraus Effektgrößen — wie beispielsweise den Erfüllungsgrad je Kontextkriterium — abzuleiten ④. Dabei werden die einzelnen Bewertungen pro Spalte herangezogen, um daraus eine präzise Gewichtung aller (qualitätsbezogener) Kontextkriterien als Summenprodukt zu ermitteln. Diese Effektgröße — angegeben in Prozent — ist ein Indikator dafür, welchen Anteil/Ausmaß ein bestimmtes Qualitätskriterium (z.B. Benutzerfreundlichkeit) im Modell einnimmt. Zusätzlich bieten Textfelder („optional“ in jeder Zelle der Matrix) die Möglichkeit einer Kurzbeschreibung zu den jeweiligen Bewertungen. Die GEM hilft dabei, Designentscheidungen mit Kontextbezug und deren Zustandekommen übersichtlich und nachvollziehbar zu gestalten.

Design Decision Traceability Grid

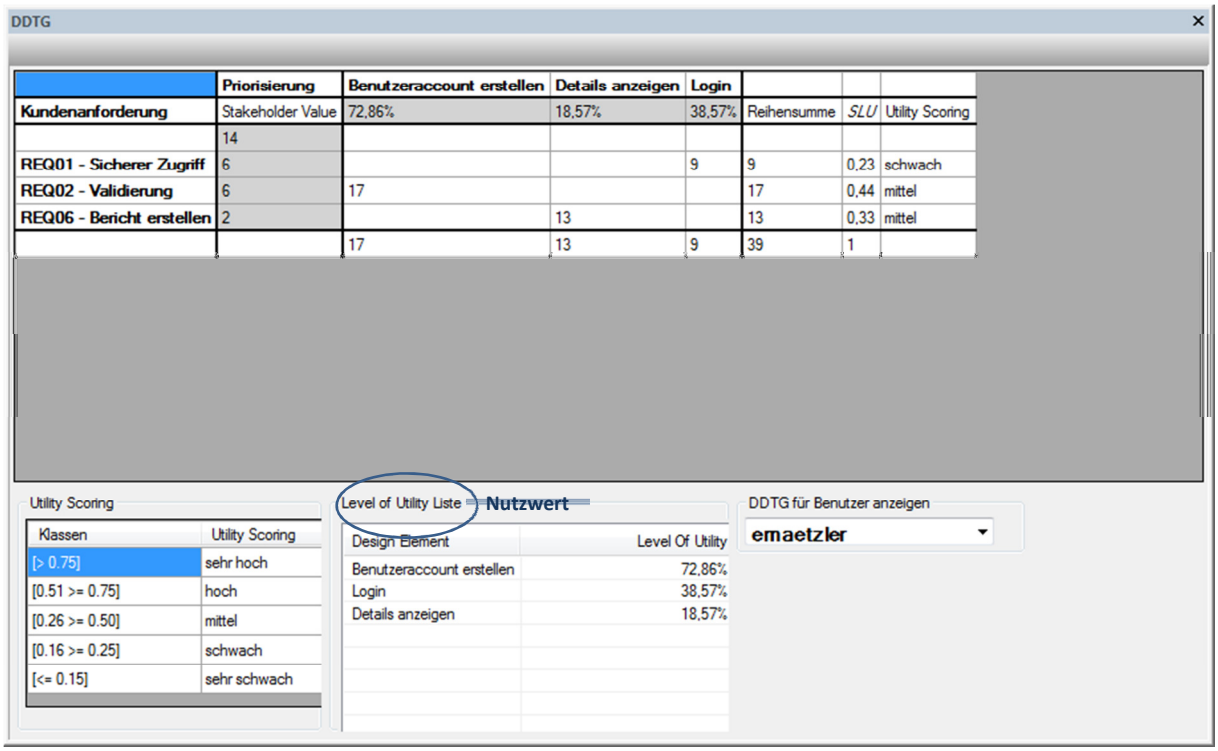

Abbildung 9 Design Decision Traceability Grid (DDT-Grid)

Die zentrale Aufgabe des Design Decision Traceability Grids (DDT-Grid) ist es, die Folgen einer Entscheidung zu visualisieren, um bei unerwünschtem Ergebnis so früh als möglich gegensteuern zu können, indem Handlungsalternativen erkannt und genutzt werden (TEMoR: Regulate). Die praktische Anwendung der präskriptiven Entscheidungstheorie unterstützt die aktiven Akteure im Designprozess bei der Überprüfung ihrer Entscheidungen (TEMoR: Monitor), um Architektur-Verletzungen schon in der Entstehung abzufangen (Prävention von Flickwerken). Das DDT-Grid dient dem gezielten modulbezogenen Monitoring von Designentscheidungen mit Blick auf die (tatsächliche) Verwendung des Systems (Anwendersicht). Im Fokus stehen dabei inhaltliche Qualitätsaspekte des fertigen Systems. Durch die Nachvollziehbarkeit und Sichtbarmachung des „Modelliererwissens“ lassen sich aus dem Grid, ebenso wie aus der vorangestellten gewichteten Entscheidungsmatrix, entsprechende Kennzahlen — Nutzwert pro Designkomponente — ermitteln. „Nutzen“ stellt ein entscheidungstheoretisches Konstrukt dar, das einen rein subjektiven Wert bezeichnet, daher wird Nutzen aus kognitiver Sicht in einem bestimmten Kontext messbar.

Der Nutzwert („Level of Utility“) errechnet sich linear additiv und bezieht sich direkt auf die Realisierungen/Kopplungen zwischen Kundenanforderungen und Designkomponenten. Dieser Wert, der den Anteil ausdrückt, den die Designkomponente an der Umsetzung zur Zielerfüllung (Erfüllung der vom Kunden gewünschten Eigenschaften eines Produkts) hat wird indirekt über die Gewichtungen der Bewertungsobjekte aus der gewichteten Entscheidungsmatrix berechnet. Dabei berechnet sich der Nutzwert aus der Multiplikation des relativen Teilnutzwerts (Summe der Relevanzgewichtungen) und der Priorisierung der Kundenanforderung als eine additive-multiplikative Verknüpfung. Der Wertebereich geht dabei von 0% (überhaupt nicht) bis 100% (unentbehrlich). Bei der Auswertung im „Utility Scoring“ werden die vom Kunden vorgegebenen Priorisierungswerte in Relation zum empirischen Relativ gestellt.

Im DDT-Grid geht es um die optimale Nutzenausprägung der Designelemente und nicht um deren maximale Ausprägung. Das bedeutet, erfüllt eine Designkomponente eine Kundenanforderung mit hoher Priorisierung ergibt das natürlich einen höheren prozentuellen Wert, als die Umsetzung einer Kundenanforderung mit niedriger Priorisierung. Mit Hilfe des Nutzwerts sind Rückschlüsse über die innere Softwarequalität des Architekturmodells hinsichtlich der Korrektheit und Notwendigkeit jeder (einzelnen) Designkomponente möglich. Zudem kann beobachtet werden, wie stark die Designkomponenten auf Änderungen der Kundenanforderungen (z.B. durch Löschen, Änderung der Priorisierung) reagieren, um daraus gegebenenfalls resultierende Entscheidungen und entsprechende Handlungsfolgen auf strategischer Ebene zu veranlassen. Somit dient das DDT-Grid der Überprüfung, ob und in welchem Ausmaß eine Designkomponente im Modell zur Erfüllung der Kundenanforderungen beiträgt. Daraus kann die Effektivität der Modellstruktur abgeleitet werden, um bei zeitlichen Engpässen die Entwicklung auf Komponenten mit höherer Relevanz zu konzentrieren.

Jeder Modellierer hat in der Grundeinstellung die Sicht auf seinen Part des Modells. In einer erweiterten Funktion wird — je nach Rechtevergabe — zusätzlich die Möglichkeit geboten die eigene Sicht auf den Modellierungsstatus anderer Modellierer auszuweiten im Sinne eines „Collaborative Reviewings“. Diese erweiterte Funktionalität schafft Transparenz von Designentscheidungen bzw. Modellierungsüberlegungen in Bezug auf das Anforderungsprofil eines Gesamtmodells. Die kumulierte Sicht auf das Modell dient einerseits der Visualisierung des Leistungsfortschritts (Umsetzungssicht) und anderseits der Risikoprävention.

In TraCo wird versucht Handlungen und deren Folgen nachvollziehbar zu machen, indem die kognitive Entscheidungsbildung im Modell explizit hinterlegt wird. Die Grundlage dafür bildet die jeweilige Entscheidungssituation in unterschiedlichen (z.B. qualitätsbezogenen) Kontexten. Implementiert wird das Verfahren mittels zweier „Controlling Instrumente“ — der gewichteten Entscheidungsmatrix und dem Design Decision Traceability Grid. In der aktuellen Version ermöglicht TraCo den Nutzern (Modellierer, Projektmanager), die Prüfung der Systemarchitektur (Designmodell) unter Berücksichtigung inhaltlicher Qualitätsaspekte aus Anwender- und Integrationssicht und dass zu jedem beliebigen Zeitpunkt (präventiv) während der Erstellung des Modells.

Fazit

Durch den Traceability Controlling (TraCo) Ansatz wird die vorhandene Repräsentation der Relationship Matrix um weitere Dimensionen erweitert. Neben der wichtigen Information, welche Artefakte zueinander in Beziehung stehen, wird auch der Grund dafür in Form einer Relevanzkennzahl, die sich auf einen Kontext bezieht, angegeben. Durch Aggregation dieser Kennzahlen im Design Decision Traceability Grid (DDT-Grid) wird ein neues Werkzeug geschaffen, das allen Stakholdern dazu dient, bisher verloren gegangenes Wissen sichtbar zu machen und somit in weitere Entscheidungen einfließen lassen zu können.

Die Umsetzung von TraCo im Modellierungswerkzeug Enterprise Architect versucht die Vergabe der Kennzahlen so einfach wie möglich zu gestalten. Dennoch ist die Motivation, über die Relevanz von Beziehungen nachzudenken, ein wichtiger Aspekt, dem in Form einer Praxisstudie nachgegangen wird. Das simple Vergeben eines Durchschnittswertes (z.B. 5) für alle Beziehungen würde spätestens im DDT-Grid auffallen und diskutiert werden müssen.

Die AutorInnen:

Dr. Horst Kargl beschäftigt sich seit 2000 mit objektorientierter Modellierung und Softwareentwicklung. Bevor er 2008 zu Sparx Systems Central Europe wechselte, war er wissenschaftlicher Mitarbeiter an der TU Wien und arbeitete in diversen Forschungsprojekten. Bei Sparx Systems beschäftigt er sich immer noch mit Forschung und Entwicklung, sowie mit Kundenprojekten und Produktentwicklung und gibt sein Wissen als Trainer und Consultant weiter.

Dipl.-Ing. Mag.rer.soc.oec. Dr.techn. Alexandra Mazak arbeitet derzeit als Forscherin für die Business Informatics Group an der Technischen Universität Wien am Projekt REAlist. Zuvor leitete sie in der Research Studios Austria Forschungsgesellschaft mbH (RSA FG) das erste „Junior Studio“ im Bereich Cognitive Engineering (CoE). Vor ihrer wissenschaftlichen Karriere sammelte Mazak 16 Jahre lang Erfahrung als Unternehmerin im Bereich IT-Entwicklung und Datenbank-Marketing.

Literaturverzeichnis

Großmann, M. (2010). Messbasierte Qualitätssicherungstechniken für die Steuerung von innerer Software-Qualität in der Praxis. *Workshop Software-Reengineering und Design for Future.* Bad Honnef.

Kniberg, H. (2007). *Scrum and XP from the Trenches.* C4Media Inc.

MacMillan, J., & Vosburgh, J.R. (1986). Software Quality Indicators. Report. Scientific Systems Inc Cambridge: Cambridge University Press.

Pohl, K. (2008). *Requirements Engineering: Grundlagen, Prinzipien,Techniken.* Heidelberg: dpunkt.verlag GmbH.

Pohl, K., & Rupp, C. (2011). *Basiswissen Requirements Engineering.* Heidelberg: dpunkt.verlag GmbH.

Schatten, A., Biffl, S., Demolsky, M., Gostischa-Franta, E., Östreicher, T., & Winkler, D. (2010). *Best Practice Software-Engineering.* Heidelberg: Springer.

Schön, D. A. (1984). *The Reflective Practitioner: How Professionals Think in Action.* Chicago: Basic Books Inc.

Steinpichler, D., & Kargl, H. (2011). *Enterprise Architect, project management with UML and EA.* Wien: SparxSystems Software GmbH.