Dr. Horst Kargl

(E-Mail: horst.kargl@sparxsystems.eu)
beschäftigt sich seit 2000 mit OO-Modellierung. Bevor er 2008 zu SparxSystems wechselte, war er wissenschaftlicher Mitarbeiter an der TU Wien. In seiner Dissertation beschäftigte er sich mit der Integration von Modellierungssprachen. Seine Schwerpunkte sind Softwarearchitektur, Codegenerierung sowie die Anpassungs- und Erweiterungsmöglichkeiten von Enterprise Architect. Als Senior Consultant berät er Kunden bei Modellierungsfragen und Fragen rund um den Enterprise Architect. Er arbeitet immer wieder in Industrie- und Forschungsprojekten mit.

# Vom Entwerfen zum Kommunizieren einer Architektur

Die Rolle des Softwarearchitekten ist eine der schwersten im Produkt- und Projektgeschäft! Als Softwarearchitekt hat man die Aufgabe, eine Idee zu verwirklichen. Dabei liegt die Herausforderung darin, die oft vagen Anforderungen und unklaren Randbedingungen zu erfassen, dafür eine tragfähige Architektur zu entwerfen und diese dem Entwicklerteam so zu vermitteln, dass das Design und die Entscheidungen dahinter verstanden und akzeptiert werden. Der Architekt darf sich dabei nicht von „tollen" neuen Technologien verleiten lassen und muss auch die Anforderungen zwischen den Zeilen lesen und verstehen. Dabei ist der wichtigste Aspekt die Kommunikation in der Gruppe. Entscheidungen müssen leicht verteilbar und historisch nachvollziehbar sein.

## Ein Ansatz mit Schieflage

Als Architekt habe ich die ehrenvolle Aufgabe, ein System nach meinen Vorstellungen zu entwickeln. Natürlich studiere ich peinlich genau alle Kundenanforderungen und Spezifikationen, die ich bekommen habe. In denen steht alles, was ich wissen muss. Nachdem ich alle Informationen, die ich benötige, gesammelt habe, ziehe ich mich zurück, um Ruhe zu haben für meine Designentscheidungen. Ich besorge mir auch noch ein Buch über die neuesten Ansätze, ich will ja etwas schaffen, das auch in ein paar Jahren noch nicht out-of-date ist. Außerdem wollte ich mich schon immer mal mit den neuesten Ansätzen beschäftigen und dieses Projekt kommt mir da gerade recht.

Da ich als Architekt in diesem Projekt keine Entscheidungsbefugnis habe, versuche ich meinen Entwurf so perfekt wie nur möglich zu gestalten. Ich bin ein fortschrittlicher Architekt und werde meinen Entwurf daher natürlich mit einem Modellierungswerkzeug umsetzen. Die haben einige Vorteile gegenüber Zeichenwerkzeugen oder reinem Text. Ich habe mich für den Enterprise Architect entschieden [SPX] – der ist einfach zu bedienen, sehr flexibel und das Preis-Leistungs-Verhältnis stimmt.

Um nicht immer wieder gestört zu werden, arbeite ich, bis die Architektur steht, aus dem Home-Office. Um sicherzustellen, keine Daten zu verlieren, stelle ich mein Projekt unter Versionskontrolle. Da ich die anderen Projektbeteiligten nicht mit meinen vielleicht noch nicht ganz ausgereiften Entwürfen verwirren möchte, gebe ich ihnen noch keinen Zugang zu meinem Projekt. Erst wenn ich alles nachkontrolliert und die Architektur mit den Anforderungen quergecheckt habe, gebe ich mein Projekt frei und schicke meinen Kollegen einen Link zum Enterprise Architect-Projekt. Nach getaner Arbeit habe ich mir jetzt drei Wochen Urlaub verdient.

In der Zwischenzeit versuchen meine Kollegen, meine Architektur zu lesen und zu verstehen. Wir haben nur erfahrene Projektmitarbeiter, doch trotzdem haben sie erhebliche Schwierigkeiten die Struktur meines Modells und der Architektur zu verstehen. Ich habe nämlich einen neuen genialen Ansatz gefunden, den ich natürlich gleich angewendet habe. Meine Kollegen sind das Referenzmodell und die Modellierungsrichtlinien der letzten Projekte gewohnt. Mein neuer Ansatz ist ihnen allerdings völlig fremd.

Die wichtigsten Designentscheidungen habe ich sogar mit mehreren Diagrammen beschrieben und sehr detailliert ausgearbeitet. Dafür habe ich extra Validierungsregeln definiert, um sicherzustellen, dass mein Entwurf konsistent ist.

Als sehr engagierter Architekt habe ich bereits in der letzten Urlaubswoche meine Erkenntnisse, die ich aus dem Studium eines neuen Buches erlangt habe, in das Modell eingepflegt. Nun ist die Architektur noch genialer, als sie vorher schon war. Ich bin zufrieden.

Am Montagmorgen im Büro quillt mein Postfach bereits über. Meine Kollegen haben sich sehr viel Mühe gegeben, meine Entwürfe zu verstehen und umzusetzen. Sie haben sogar meine neue Struktur bereits nach zwei Wochen annähernd durchschaut. Leider sind ihnen einige meiner Designentscheidungen nicht ganz klar. Daher wurden teilweise Annahmen getroffen und andere Teile wurden liegengelassen.

Im ersten Meeting falle ich aus allen Wolken, da die aktuelle Implementierung noch auf Version 1 des Modelles basiert. Meine Änderungen wurden nicht eingearbeitet, obwohl ich sogar den Status aller geänderten Elemente auf „Modified" gesetzt habe. Das hätte ich wohl alles besser kommunizieren müssen. In den nächsten Wochen werde ich wohl einiges an Kommunikation nachholen müssen. Danach brauche ich dann sicher wieder Urlaub.

## Jetzt mache ich alles ganz anders

Aus dem zweiten Urlaub gestärkt und aus meinen Fehlern gelernt, mache ich mich an die Architektur des neuen Projektes. Damit das Projektteam einen besseren Gesamtüberblick hat, beginne ich gleich mit der Erstellung aller möglichen Use Cases, die ich aus den Anforderungen ableiten kann. Dabei unterscheide ich zwischen Business Use Case und System Use Case.

Ich vernetze alle Anforderungen mit den Use Cases, dann beginne ich mit der Struktur der Architektur. Nach jedem neu entworfenen Baustein in der Architektur (also alle 2-3 Tage) halten wir Modell-Reviews ab, um den aktuellen Entwurf zu diskutieren, alle Vor- und Nachteile aufzuschreiben und in das Modell einzupflegen. Hin und wieder dauert dieser Review auch den ganzen Tag, da kein Konsens über den Entwurf erreicht werden konnte und ein Abnahmekriterium des Reviews eine 100-prozentige Zustimmung zur Architektur ist. Alle Beteiligten müssen den Entwurf akzeptieren und mit ihrem „Blut" unterschreiben, um diesen genau so umzusetzen.

Nach einem Monat kommt neuerlich der Aufstand. Die Entwickler sind jetzt über das, was zu tun ist, vollständig informiert und auch der Auftraggeber ist voll im Bilde. Doch alle beschweren sich, dass nur noch Meetings abgehalten werden und keine Zeit mehr für die Entwicklung bleibt und diese dadurch sehr unter Druck gerät. Ich denke, diesmal habe ich es zu gut gemeint. Ich muss wohl den goldenen Mittelweg finden.

## Der goldene Mittelweg

Die oben beschriebenen Szenarien sind zwei von vielen, die das Leben schreibt. Der oder die eine oder andere wird sich sicherlich in einigen Punkten wiederfinden können. Es ist klar, dass Teile der Ansätze ganz brauchbar oder notwendig sind, es aber wichtig ist, den goldenen Mittelweg zu finden.

Natürlich ist es hilfreich, das richtige Werkzeug (Tool) zu verwenden. Doch nur das Werkzeug alleine bringt noch keinen Projekterfolg, es muss auch richtig eingesetzt werden. Im Folgenden möchte ich die nach unseren Erfahrungen wichtigsten Punkte aufzeigen.

### Mission-Critical-Anforderungen in Workshops herausarbeiten

Die Kundenwünsche und Anforderungen werden sorgsam dokumentiert [PohlSik09]. Dabei ist zu beachten, dass in erster Linie die Mission-Critical-Anforderungen herausgearbeitet werden. Diese bilden die Eckpfeiler der Architektur. Alle weiteren Architekturentscheidungen werden aufgrund dieser Anforderungen getroffen.

Dabei sollte beachtet werden, dass die Anzahl der Mission-Critical-Anforderungen nicht zu groß wird, um diese auch immer im Überblick zu behalten. Je nach Umfang und Größe des Projektes sind unterschiedliche Zahlen denkbar. Als Faustregel können Mission-Critical-Anforderungen um einen Faktor 10 bis 100 geringer sein als die übrigen Anforderungen.

Auch eine Unterscheidung sollte angedacht werden, um die verschiedenen Anforderungen leicht filtern zu können. Im Enterprise Architect können dafür z. B. Stereotypen und/oder Tagged Values verwendet werden. Ein zusätzliches Positionieren der kritischen Anforderungen in einem eigenen Paket ist sinnvoll. Durch die Verwendung von Stereotypen und Tagged Values können diese Anforderungen einfach gesucht und in verschiedenen Sichten angezeigt werden [Stark07] (siehe Abbildung).

Durch die Verwendung von Stereotypen ist es auch möglich, die grafische Notation der Elemente zu beeinflussen. Wird ein eigenes UML-Profil erstellt, ist es auch möglich, eigene Toolboxen und Diagramme zu definieren.

Ein wichtiger Aspekt für ein akzeptiertes und verständliches Modell ist die Nachvollziehbarkeit des Modellierten. Daher ist es anzuraten, bei wichtigen Artefakten auch Designentscheidungen festzuhalten und nicht nur den resultierenden Entwurf.

Im Enterprise Architect können für jedes Element weitere Metainformationen hinzugefügt werden. Bei Mission-Critical-Anforderungen können z. B. über *View Project Management* oder *Maintenance* zusätzliche Metainformationen zum Element strukturiert und leicht auffindbar abgelegt werden. Der Enterprise Architect *Model View* erlaubt die Konfiguration einer automatischen Benachrichtigung, wenn sich ein Sachverhalt am Modell geändert hat. Somit werden Entscheidungen für alle Benutzer nachvollziehbar.

Dabei ist es wichtig, alle Teammitglieder in die Struktur des Modells und in die Vorgehensweise einzuweihen! Eine eingeführte Struktur und Vorgehensweise sollte auch nicht leichtfertig geändert werden, um bei den Kollegen nicht unnötig Verwirrung zu stiften und somit die bereits akzeptierte und gelebte Struktur zu gefährden.

Können aus einem Projekt oder einer Iteration Verbesserungen (die „Lessons learned") abgeleitet werden, können diese natürlich in einem Folgeprojekt oder einer Folge-Iteration eingepflegt werden. Zuvor ist es allerdings wichtig, alle Teammitglieder ausreichend von den Änderungen in Kenntnis zu setzen und eventuell die neue Arbeitsweise in Workshops zu erproben.

### Architektur erstellen

Bei der Erstellung der Architektur ist darauf zu achten, dass sie die „passende" ist. Was ist aber nun „passend" für mein Projekt/Produkt? Das kann, muss und soll nicht eine Person alleine entscheiden, auch wenn sie der oder die erfahrenste Architekt/In ist. Dabei sind zwei Dimensionen zu bedenken:

1. *Gibt es noch Informationen, die ich nicht kenne, aber die die Architektur maßgeblich beeinflussen?* Nicht alle

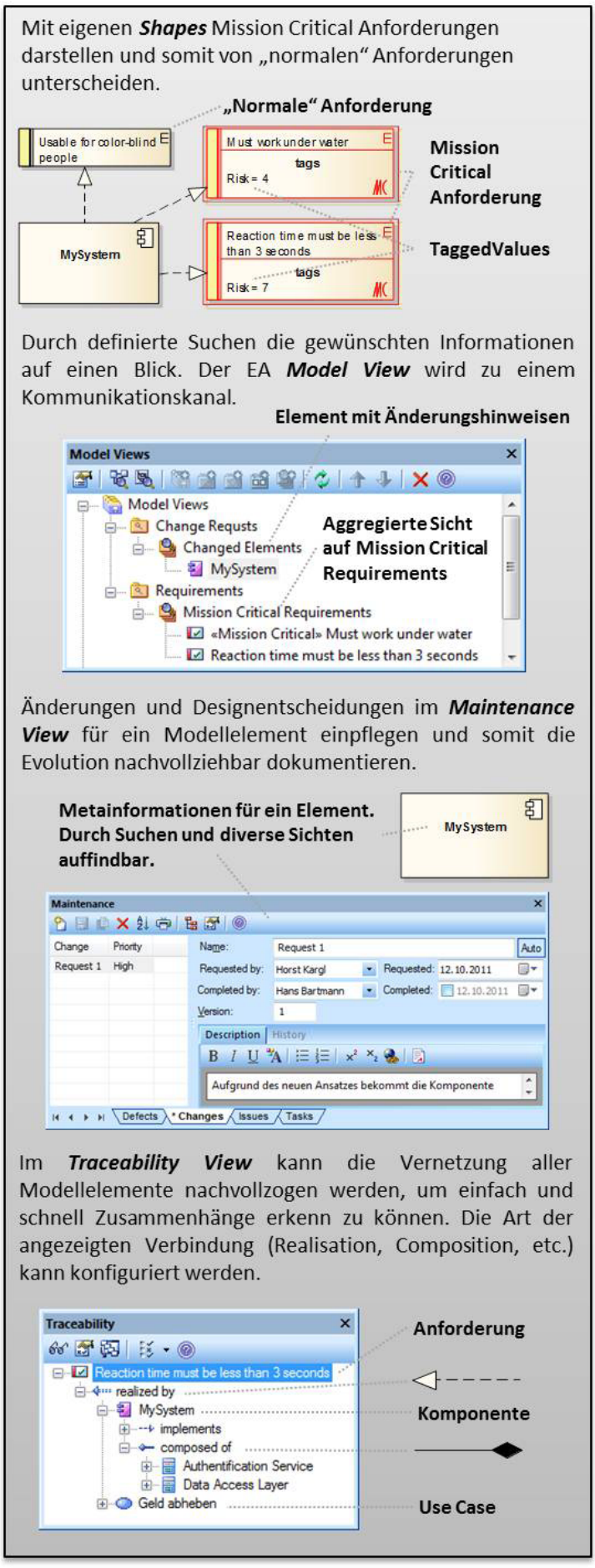


*Abb.: Verschiedene Sichten im Enterprise Architect*

Anforderungen sind vollständig und klar formuliert. Unumgänglich ist das Herausfinden von Mission-Critical-Anforderungen. Aber auch weitere nicht hinterfragte und nicht berücksichtigte Anforderungen können zu Zeit- und Geldlöchern werden.
Daher ist es anzuraten, Designentscheidungen zur Diskussion zu stellen. Eventuell treten Anforderungen erst ans Licht, wenn eine Architektur einen Freiheitsgrad einschränkt.

2. *Gemeinsames Verständnis und Akzeptanz schaffen:* Dies geschieht in der Regel automatisch, wenn Punkt 1 durchgeführt wird. Hat jeder den Architekturentwurf gesehen, verstanden und sich einbringen können, wird er/sie auch mit hoher Wahrscheinlichkeit hinter der vollendeten Architektur stehen. Beim „End-Review“ der Architektur, bei der Verbindlichkeiten eingegangen werden, kommt es zu weniger Diskussionsbedarf und Ablehnung bzw. zu blinder Zustimmung, welche die Probleme nur nach hinten verschiebt.

Bei der Umsetzung ist jedem Entwickler klar, worum es geht und warum die Architektur so aussieht und nicht anders, weil Designentscheidungen und Alternativen im Modell enthalten und dokumentiert sind. Dabei spielen Meetings eine große Rolle, dürfen aber das Tagesgeschäft nicht beeinträchtigen. Die Ergebnisse der Meetings müssen wohl strukturiert dokumentiert werden, um von jedem Teammitglied bei Bedarf schnell und einfach gefunden zu werden. Dafür sind geeignete Kommunikationskanäle zu definieren.

**Informationen bereithalten**

Keiner arbeitet für sich alleine. Selbst als Einzelunternehmer hat man zumindest Kunden, mit denen man sich abstimmen muss. In der Regel arbeitet man allerdings im Team. Das Team besteht meist aus Personen mit verschiedenen Vorkenntnissen, die verschiedene Rollen im Projekt spielen und somit auch verschiedene Interessen haben. Immer öfter sind diese Teams auch global verteilt und treffen sich nie oder nur sehr selten.

Daher ist es umso wichtiger, die „richtigen“ Informationen zu generieren und diese über die „passenden“ Kanäle zu verteilen. Welche Informationen für ein Team die „richtigen“ sind, kann nicht einfach so

gesagt werden. Dabei spielen die Vorkenntnisse und die Rolle der Personen eine große Rolle.

Als Faustregel kann allerdings gesagt werden, dass so wenig Informationen wie möglich definiert werden sollten, aber soviel als nötig sind. Zu viele Informationen führen dazu, dass sie an Wert verlieren, da auch unwichtigere Informationen neben höchst relevanten stehen. Schafft man es, den Mittelweg zu finden, wird jede verteilte Information im Modell zur wertvollen Dokumentation.

Der nächste wichtige Punkt ist der Inhalt der Information. Wird eine Architektur erstellt, ist es wichtig, die Adressanten zu kennen, um die Informationsdichte und das Abstraktionsniveau der Modelle den Bedürfnissen der Adressanten anzupassen. Es ergibt keinen Sinn, in die Werkzeugkiste der UML zu greifen und formal ausgeklügelte Modelle zu erstellen, die nur ein kleiner Kreis der Teammitglieder verstehen kann.

Natürlich ist es wünschenswert, dass die verwendete Sprache (UML, SysML, etc.) von allen Teammitgliedern fließend gesprochen wird. Ist dem allerdings (noch) nicht so, sollte man sich auf einen reduzierten Kern beschränken. Dies kann auch meist im verwendeten Tool eingeschränkt werden. Dadurch beschränkt man den Wildwuchs an Modellvarianten und den Interpretationsspielraum. Je reifer das Team bezüglich der Modellierung wird, desto mehr Modellierungsmöglichkeiten können natürlich in ein Projekt eingebracht werden.

**Kommunikationskanäle finden**

Nicht zu vernachlässigen sind natürlich auch die möglichen Kommunikationskanäle. Gern wird alles per Mail kommuniziert. Selbst wenn nur relevante Informationen per Mail ankommen, was wohl nie passieren wird, ist der Umfang der empfangenen Nachrichten wohl so groß, dass bald der Überblick verloren geht. Daher sollten für bestimmte Themen spezielle Kommunikationskanäle definiert und diese auch konsequent genutzt werden.

Verwendet man ein Modellierungswerkzeug, wie den Enterprise Architect, findet man verschiedene Möglichkeiten für Kommunikationskanäle. Einer der wichtigsten „Kanäle“ ist natürlich die Struktur des Modells. Diese sollte einem Referenzmodell entsprechen, das etabliert ist und von jedem Teammitglied gekannt und akzeptiert wird.

Dadurch wird die Gefahr abgewandt, dass vorhandene Informationen nicht gefunden werden. Zum Beispiel könnte es die Regel geben, dass das Verhalten eines Use Cases als Aktivität innerhalb des Use Cases definiert wird. Fehlt eine Aktivität unterhalb des Use Cases, kann davon ausgegangen werden, dass noch keine Aktivität definiert wurde und diese nicht gesucht werden muss.

Diese Regeln können auch implementiert und somit das Modell automatisch validiert werden. Dabei ist zu beachten, dass auch dokumentiert wird, was die Regel validiert und wie ein Fehler zu beheben ist. Werden die Regeln geschickt implementiert, können Hinweise zur Korrektur bei der Fehlerausgabe mitgeliefert werden.

Wenn Missverständnisse bei Designentscheidungen auftreten, da der Sachverhalt standardmäßig anders umgesetzt wird, aber durch eine konkrete Anforderung eine andere Architektur erforderlich ist, gehören diese Entscheidungen auch dokumentiert. Mindestanforderung ist dabei, die Anforderungen mit den betreffenden Designelementen zu verbinden, um eine nachvollziehbare Trace-Kette zu etablieren.

Im Enterprise Architect können Beziehungen verschiedenen Typs zwischen den Modellelementen gezogen werden. Diese Verkettung und Abhängigkeiten können nun leicht im *Traceability View* nachgesehen werden. Des Weiteren kann durch das Abringen von zusätzlichen Metainformationen (im *Maintenance View*) an einem Element erklärt werden, warum genau diese Architektur gewählt wurde. Auch Änderungen können so nachvollziehbar dokumentiert werden.

Durch zuvor definierte Positionen im Modell (wie dem *Maintenance View*, oder spezielle *TaggedValues*), an denen Informationen angebracht werden, können nun Suchen definiert werden, die eine dynamische Sicht auf das Projekt liefern. Der *Model View* im Enterprise Architect erlaubt es zusätzlich, eine automatische Benachrichtigung zu definieren, falls sich am Modell etwas geändert hat. Mit diesen Grundfunktionalitäten können wohl definierte Kommunikationskanäle aufgebaut werden (**siehe Abbildung**).

Ist dieses Vorgehen einmal definiert und etabliert, können Meetings auf das Notwendigste reduziert werden. Komplett entfallen, dürfen sie allerdings nicht. Der persönliche Kontakt und Austausch ist immer noch am wichtigsten. Dabei werden oft auch Punkte ausgetauscht, um die Kommunikationskanäle zu optimieren.

**Freiräume für den Entwickler schaffen**

Wird zum Beispiel die Architektur offshore umgesetzt oder handelt es sich um ein sicherheitskritisches System, ist es durchaus angebracht, die Architektur so konkret und detailliert wie nötig zu beschreiben, dass der Entwickler die Designentscheidungen nur noch in Code formulieren muss. Wird dieser Modellierungsaufwand betrieben, kann auch darüber nachgedacht werde, den Code aus den Modellen zu generieren.

Der Entwickler weiß in der Regel, wie er eine Aufgabe mit einer bestimmten Technologie umsetzt. Somit kann die Architektur auch an einem höheren Abstraktionsniveau stehen bleiben. Bei Entscheidungen über die Architektur, die Mission-Critical-Anforderungen betreffen, ist es oft angebracht, einige Entwickler ins Boot zu holen, um ihre Detailerfahrungen mit den zu verwendenden Technologien einzuholen. Dabei können sie Fragen aufwerfen, die die technologische Umsetzung der Architektur betreffen und in keiner Anforderung bedacht sind.

Dadurch können drei Fliegen mit einem Schlag erledigt werden:

1. Die Entwickler werden frühzeitig in die Architektur eingewiesen und verstehen diese. Sie können sich selbst einbringen und so die Qualität der Architektur verbessern und auf Umsetzungsprobleme hinweisen.
2. Die Entwickler, die an diesen Meetings beteiligt waren, werden zu Know-how-Trägern und werden die Architektur und deren Entstehung in ihrem Team weiter verteilen.
3. Die Abnahme der Architektur ist voraussichtlich nur noch ein formaler Akt, da von jeder Projektrolle zumindest eine Person im Entstehungsprozess involviert war und somit mit gu-

tem Gewissen die Architektur absegnen kann. Damit sind nicht nur die Entscheidungen für oder gegen einen Entwurf im Modell dokumentiert und für alle Personen zugängig, sondern einige Mitglieder der einzelnen Rollen, waren auch im Entscheidungsprozess eingebunden und tragen somit diese Entscheidungen in ihre Teams.

## Fazit

Eine robuste und langlebige Architektur zu entwickeln, ist nur die halbe Miete, sie muss auch umgesetzt werden. Dazu müssen allerdings alle beteiligten Personen am selben Strang ziehen und ein gleiches Verständnis der Problemstellung und der Lösung erlangt haben.

Mit Modellen als Sprache hat man schon den ersten Schritt getan. Um ein gemeinsames Verständnis zu bekommen, können Tools hilfreich sein, diese müssen allerdings auch richtig eingesetzt werden, um eine reibungslose und akzeptierte Kommunikation zu gewährleisten. ■

## Referenzen

**[Stark07]** ORDNUNG 2.0 – Hilfe für den Info-Schungel?, OBJECTSpectrum, 03/2007.
**[Grab09]** Auswirkungen von Architekturentscheidungen auf die Software-Performance, OBJECTSpectrum, Architekturen/2009.
**[PohlSik09]** COSMOD-RE: Verzahnung des Architekturentwurfs mit dem Requirements Engineering, OBJECTSpectrum, Architekturen/2009.
**[SPXBlog]** Sparx Systems – Central Europe Blog, blog.sparxsystems.at
**[SPX]** Sparx Systems – Central Europe Software GmbH, www.sparxsystems.de