mehr zum thema:
www.4soft.de/leistungen/modellmanagement-und-modellierung.html

ZWISCHEN TOTALITARISMUS UND KLEINSTAATEREI: GROSSE MODELLE ADÄQUAT MANAGEN

Das Management großer, übergreifender Modelle hat starke Ähnlichkeit mit dem Management von Quellcode, aber es gibt es auch große Unterschiede. Das liegt vor allem daran, dass Modelle mehr Einsatzzwecke haben als Quellcode und insbesondere auch der Kommunikation zwischen Menschen dienen. Der Artikel stellt das Gebiet des Modellmanagements übergreifend dar und zeigt die wesentlichen Gemeinsamkeiten und Unterschiede auf.

▶ der autor

Klaus Bergner
(E-Mail: bergner@4soft.de)
ist einer der Gründer und Geschäftsführer von 4Soft. Seit vielen Jahren beschäftigt er sich mit Modellierung und der modellgetriebenen Softwareentwicklung.

Michael Kempf
(E-Mail: kempf@4soft.de)
arbeitet bei der 4Soft GmbH als Berater in den Bereichen Modellierung und Requirements-Engineering. Sein Spezialgebiet ist das modellgetriebene Anforderungsmanagement (Model-Driven RE).

Siegfried Schäfler
(E-Mail: schaefler@4soft.de)
arbeitet bei der 4Soft GmbH als Berater in den Bereichen Modellierung und Modellmanagement. Aktuell ist er bei einem Großunternehmen als Modellmanager tätig und organisiert die Konsolidierung konkurrierender Modelle für zentrale Produktstammdaten.

Modelle erfüllen vor allem zwei Zwecke:

- Sie stellen komplexe Zusammenhänge anschaulich dar und ermöglichen dadurch ein gemeinsames Verständnis von Menschen. Dies kann im Kontext eines einzelnen Teams oder Projekts geschehen, zur Koordination mehrerer Projekte oder zwischen Mitarbeitern unterschiedlicher Organisationen. Solche fachlichen Modelle können – zumindest in Teilbereichen – informell bleiben. Hier sind Anschaulichkeit und Verständlichkeit wichtiger als Korrektheit und Konsistenz im Detail.
- Sie dienen als Eingabe für Programme, beispielsweise als Input für Codegeneratoren oder zur Konfiguration und Steuerung von Systemen. Modelle dieser Art folgen strikten formalen Richtlinien und ähneln in dieser Hinsicht dem Quellcode von traditionellen, textuellen Programmiersprachen.

Häufig gehen beide Zwecke auch ineinander über: Ein Produktionsunternehmen kann beispielsweise über standardisierte fachliche Datenmodelle vorgeben, dass Produktstrukturen in Konstruktion und Produktion gleich aufgebaut sind. Häufen sich dann Produktionsfehler bei bestimmten Teilen, so lassen sie sich von den Arbeitern im Werke leichter zu den Konstrukteuren zurückspielen, da beide die gleichen Denkmodelle im Kopf haben und somit weniger Verständnisprobleme entstehen. Wenn dieser Prozess später durch Software unterstützt wird, erleichtern die aus den fachlichen Modellen abgeleiteten technischen Datenmodelle und Datenaustauschformate die softwaretechnische Integration der beteiligten Produktdatenmanagement-Systeme.

Modelle beider Arten haben wie Programmcode die Tendenz, aus kleinen Anfängen heraus immer weiter zu wachsen. Damit steigt die Gefahr, dass derselbe Sachverhalt an unterschiedlichen Stellen eines Modells oder in mehreren Einzelmodellen redundant und oft auch inkonsistent dargestellt wird. Soll das Wachstum nicht unkontrolliert ansteigen, müssen sowohl Modelle als auch Programmcode strukturiert und aktiv gepflegt werden. Viele der Prinzipien solcher *Clean Models* ähneln den Richtlinien für *Clean Code* (vgl. [Mar08]) – aufgrund der breiteren Einsatzszenarien von Modellen gibt es aber auch Unterschiede.

Einheitlichkeit um jeden Preis?

Ein kleines bis mittleres Modell in einem kleinen Team zu erarbeiten und konsistent zu halten, kann auf ähnliche Art und Weise erfolgen wie beim *eXtreme Programming (XP)*: Wenige Experten tragen die Verantwortung für das Modell gemeinsam und stimmen sich in einer offenen Arbeitsumgebung eng miteinander ab.

In solchen Szenarien fühlt sich aber meist niemand berufen, Modelle projekt- oder gar organisationsübergreifend zu vereinheitlichen. Einzelne Teams oder Projekte kommen so natürlich zu schnellen Erfolgen, die erstellten Lösungen sind aber untereinander nicht vergleichbar und repräsentieren häufig ganz unterschiedliche Sichtweisen auf an sich gleiche Sachverhalte. Eine Kopplung unterschiedlicher Systeme und der durch sie unterstützten Geschäftsprozesse ist auf dieser Basis nur mit hohem Aufwand möglich, weil die fachliche Grundlage erst erarbeitet und im Anschluss technisch und organisatorisch umgesetzt werden muss. So entsteht eine Art Kleinstaaterei mit vielen Sprachen und hinderlichen Schlagbäumen.

Um Standards und Richtlinien für das Modellmanagement zu etablieren und ihre Einhaltung zu überwachen, kann eine Organisation beispielsweise eine Stabsabteilung einrichten. Diese gibt verbindliche Standards vor, prüft alle neu entstandenen oder geänderten Modelle daraufhin, ob sie den geltenden Standards entsprechen, und integriert die Modelle schließlich in ein übergreifendes Gesamtmodell.

Allerdings darf diese Governance-Funktion nicht zu einem lähmenden Flaschenhals werden. Ein zu zentralistischer Ansatz artet leicht in einen bürokratischen Totalitarismus aus, der alles und jedes regelt, dadurch aber die termingerechte Umsetzung dringend benötigter Funktionalität behindert.

Föderalismus als Lösungsstrategie

Eine vernünftige Lösung liegt in der Regel irgendwo zwischen diesen Extremen: Standardisierung und Vereinheitlichung sind kein Selbstzweck, sondern werden dort vorangetrieben, wo sie aktuell den größten Nutzen versprechen.

Ein Unternehmen, das dieser Maxime folgt, legt zunächst einmal die Ziele für das Modellmanagement fest:

- Welche Zwecke verfolgen die zu erarbeitenden Modelle?
- Wie formal und konsistent müssen sie jeweils sein?
- Welchen Nutzen bringt eine Vereinheitlichung und Standardisierung und an welchen Stellen sollte man sie angehen?

Daraus wird eine Strategie abgeleitet, die verpflichtende Standards für spezifische Modelle setzt. Sie könnte beispielsweise festlegen, dass Modelle für bestimmte Stammdaten (z. B. für Produktstrukturen und Teilenummern) sowie die daraus abgeleiteten XML-Datenaustauschformate organisationsübergreifend verwendet werden müssen. Wo derartige Vorgaben nicht erforderlich sind, haben die einzelnen Geschäftsbereiche und Projekte die Freiheit, eigene Wege zu gehen und auf diese Weise handlungsfähig zu bleiben.

Um bei den politischen Vergleichen zu bleiben, entspricht dieses Vorgehen dem Föderalismus: Zentralisierung und Vereinheitlichung da, wo sie im Sinne aller Nutzen stiftet, aber lokale Autonomie in allen anderen Belangen. Standardisierung bzw. Heterogenität betreffen dabei nicht nur den Inhalt der Modelle, sondern auch Fragen der Organisation, der Methodik, der Modellarchitektur (ein oder wenige zentrale Modelle versus viele verteilte Modelle) und der verwendeten Werkzeuge.

Der Modellmanager als Schlüsselperson

Um von Standardisierung und Vereinheitlichung im Modellierungsbereich profitieren zu können, sollte eine Organisation zunächst den Prozess der Modellierung regeln. Insbesondere sollte klar sein, in welchen Situationen die Erarbeitung und Nutzung von Modellen verpflichtend sind und wer für sie verantwortlich ist. Darüber hinaus sollte geregelt sein, wie Modelle und Richtlinien zur Standardisierung vorgeschlagen werden können, welche Gremien darüber entscheiden und wie verabschiedete Standards kommuniziert werden.

Außerdem muss die Organisation regeln, wie die Einhaltung verabschiedeter Standards vorangetrieben und überwacht wird. Das kann etwa durch einen oder mehrere zentrale *Modellmanager* geschehen, die einzelne Projekte und ihre Ergebnismodelle daraufhin überprüfen, ob sie sich an die vorgegebenen Standards halten.

Damit diese Rolle von den einzelnen Projekten möglichst gut akzeptiert wird, sollte der Modellmanager neben seiner unvermeidlichen Überwachungsfunktion auch Dienstleistungen erbringen, die dem jeweiligen Modellierungsvorhaben oder Projekt unmittelbar nutzen. Dazu gehören vor allem:

- Der Modellmanager berät das Projekt frühzeitig über Standards, Musterdatenmodelle und Stammdatenquellen, mit deren Hilfe das Projekt seinen Aufwand reduzieren oder die Qualität erhöhen kann. Weiterhin informiert er das Projekt über absehbare Entwicklungen bei Standards und beeinflusst damit strategische Entscheidungen.
- Der Modellmanager bringt sein Expertenwissen über bestehende Lösungen und Zusammenhänge ein und coacht die Projektbeteiligten entsprechend. Zudem sichert er die Qualität ihrer Projektergebnisse.
- Der Modellmanager gestaltet Interimslösungen so, dass Projekte einerseits arbeitsfähig bleiben, andererseits aber ein späterer Übergang zu einem Standard mit möglichst geringem Aufwand erfolgen kann.

Außerdem kann der Modellmanager Erfahrungen aus der Projektarbeit an den Standardisierungsprozess zurückspielen, um zusätzliche Standardmodelle und Modellierungsstandards zu erarbeiten oder bestehende Schwachstellen zu erkennen und zu verbessern. Idealerweise überprüft die Organisation regelmäßig, wie gut sich Standards und Richtlinien in der Praxis bewähren, korrigiert nicht tragfähige Standards und nimmt sie im Extremfall auch wieder zurück, um die Erarbeitung brauchbarer Lösungen nicht unnötig zu behindern.

Methodik und Konventionen

Um einheitliche Modelle zu erstellen, benötigt man eine einheitliche Methodik und klare Modellierungsstandards. Mit der Wahl einer Modellierungssprache wie der *Unified Modeling Language (UML)* ist es dabei – analog zur Wahl einer Programmiersprache in der Entwicklung – nicht getan. Genauso wie es Programmierrichtlinien (*Code Conventions*) gibt, benötigt man bei der Modellierung Modellierungsrichtlinien (*Model Conventions*). Sie sollten in Form eines Modellierungsleitfadens festgehalten und – soweit möglich – von Werkzeugen vorgegeben werden, beispielsweise über *UML Profiles*. Dabei sollte beispielsweise Folgendes geregelt sein:

Welcher Ausschnitt der Modellierungssprache wird für welchen Zweck genutzt?
Weniger ist hier meistens mehr: Die Beschränkung auf die unbedingt notwendigen grundlegenden Diagrammarten und Features erleichtert sowohl das gemeinsame Verständnis als auch die Entwicklung und Anpassung von Werkzeugen. Zur Spezifikation der Daten von Informationssystemen auf Fachkonzept-Ebene müssen Fachklassenmodelle beispielsweise keine Operationen haben, und für die Datentypen der enthaltenen Attribute reicht in der Regel eine kleine Auswahl von Standardtypen aus.

Welche Bezeichner und Namen sind zulässig?
Da Modelle immer auch der Kommunikation zwischen Menschen dienen, sind

verständliche und aussagekräftige Namen bei ihnen noch wichtiger als bei der Programmierung. Für fachliche Modelle ist deshalb oft Deutsch die Sprache der Wahl, für eher technische Modelle, die in Programmcode übersetzt werden, meist Englisch. Weiterhin sollte geregelt sein, welche Modellelemente überhaupt benannt werden müssen.

Welche Farben und Symbole kommen zum Einsatz?

Modellelemente lassen sich nach fachlichen oder technischen Kriterien markieren. Ein Beispiel hierfür ist die Wahl unterschiedlicher Symbole für *Model*-, *Controller*- und *View*-Klassen bei der Modellierung von Architekturen für Benutzungsoberflächen. Eine andere Möglichkeit ist es, den Kommunikationsaspekt von Modellen in den Vordergrund zu stellen und beispielsweise die noch nicht abgestimmten Modellelemente in rot darzustellen, um so den Prozess der Modellierung selbst zu unterstützen. Bei farbcodierten Modellen sollte man allerdings immer darauf achten, dass die Farbe nicht das einzige Unterscheidungskriterium ist, damit die Modelle auch für Menschen lesbar bleiben, deren Farbsehen beeinträchtigt ist.

Welche Modellierungsmuster kommen zum Einsatz?

Für codenahe Modelle überschneidet sich dieses Thema mit der Wahl von eleganten und effizient umsetzbaren Entwurfsmustern bei der Programmierung. Für die Analysemuster bei fachlichen Modellen stehen aber üblicherweise andere Aspekte – insbesondere hohe Lesbarkeit und Ausdrucksmächtigkeit (vgl. [Fow98]) – im Vordergrund.

Welche Anforderungen an Vollständigkeit und Konsistenz gibt es?

Insbesondere bei Modellen, die der Kommunikation und Abstimmung dienen, ist es (im Gegensatz zu Quellcode) durchaus möglich und sinnvoll, Lücken oder Inkonsistenzen in oder zwischen Modellen zumindest temporär zuzulassen. Die Forderung nach einer vollständigen Modellierung aller Details und der Konsolidierung aller Inkonsistenzen mit anderen Modellen würde die Konzentration auf den fachlichen Kern erschweren und damit die Klärung der wesentlichen Zusammenhänge behindern.

Modelle strukturieren

Ohne eine klar strukturierte Softwarearchitektur aus abgegrenzten Teilsystemen und Komponenten endet die Entwicklung großer Softwaresysteme im Chaos. Entsprechend benötigen große Modelle eine Modellarchitektur aus klar abgegrenzten Teilmodellen.

Wie bei Softwarekomponenten auch ist dabei das Prinzip des *Loose Coupling* – *Strong Cohesion*, also die Bildung von Teilmodellen mit starkem innerem Zusammenhang und möglichst schmalen Schnittstellen zur Außenwelt, anzuwenden. Problematisch sind dabei nicht so sehr Modellentitäten (wie zum Beispiel einzelne Klassen oder Anwendungsfälle), sondern vielmehr die Beziehungen zwischen diesen. Ein gutes Beispiel hierfür ist die Verwaltung von Assoziationen in Klassenmodellen: Zwar unterstützt das Metamodell von UML die klare Zuordnung einer Assoziation zu einem Modellpaket – aus der graphischen Darstellung von Assoziationen in Diagrammen wird diese aber nicht ersichtlich. Zudem regeln die meisten Modellierungswerkzeuge die Zuordnung eher willkürlich und ändern sie beispielsweise eigenmächtig, wenn der Modellierer die (eigentlich von diesem Thema ganz unabhängige) Navigationsrichtung der Assoziation ändert. Ohne große Sorgfalt bei der Modellierung entstehen hier schnell verwirrende Spaghetti-Modelle, in denen jedes Paket über Assoziationen zu seinen Elementen von vielen anderen Paketen abhängig ist. Insbesondere zyklische Abhängigkeitsstrukturen sollten, wenn irgend möglich, vermieden werden.

Klar strukturierte Paketstrukturen bieten – ganz analog zu klar strukturierten Softwarearchitekturen – viele Vorteile: Teilmodelle lassen sich leichter verstehen und wiederverwenden, da man bei der Nutzung eines Teils nicht immer alle anderen, über Abhängigkeiten verbundenen Teilmodelle „anzieht". Außerdem ist es leichter möglich, das Modell zu warten und fortzuentwickeln, da weniger Abhängigkeiten berücksichtigt werden müssen. Da die Mechanismen zur verteilten Zusammenarbeit der meisten gegenwärtigen Modellierungswerkzeuge auf der Granularität von Paketen arbeiten (ein Bearbeiter kann typischerweise ein oder mehrere Pakete zur Bearbeitung sperren und sie dann ungestört von anderen Bearbeitern ändern), erleichtert eine klar definierte Paketstruktur auch die Verteilung von Verantwortlichkeiten und die Weiterentwicklung der Modelle. Schließlich ist eine derartige Strukturierung Voraussetzung für den Export von Teilmodellen, beispielsweise zur Kommunikation mit benachbarten Projekten oder Organisationen.

Modelle erweitern und konsolidieren

Modellmanagement ist kein rein organisatorisches Problem. Klare Abhängigkeitsstrukturen, wie im vorigen Abschnitt beschrieben, unterstützen auch einen modellierungstechnischen Ansatz zum Modellmanagement: Hiernach wird das Gesamtmodell in einen klar definierten, übergreifend abgestimmten Kern und mehrere darauf aufsetzende Erweiterungen aufgeteilt. Letztere entstehen in einzelnen, teilweise parallel laufenden Projekten und werden zu bestimmten Zeitpunkten vom Modellmanager auf Gemeinsamkeiten analysiert. In einer nachfolgenden Konsolidierungsphase extrahiert der Modellmanager diese Gemeinsamkeiten aus den Erweiterungsmodellen, abstrahiert sie und integriert sie in den gemeinsamen Kern. Dieses Vorgehen ähnelt stark dem Entwurf von Frameworks bei der Programmierung: Auch hier entstehen Frameworks meistens, indem Gemeinsamkeiten mehrerer Systeme erkannt und in einer allgemein verwendbaren Form implementiert werden.

Abbildung 1 zeigt das beschriebene Vorgehen an einem Beispiel: Die linke Hälfte der Abbildung enthält ein abgestimmtes Kernmodell mit Paketen wie Produktion und Vertrieb, das durch die beiden Erweiterungsmodelle aus Projekt A und Projekt B erweitert wird. Projekt A führt eine neue Klasse Zubehör mit den Attributen zbNummer (Zubehörnummer) und bezeichnung ein. Projekt B modelliert hingegen eine Klasse Sonderzubehör mit dem Attribut *name*.

Die rechte Hälfte von **Abbildung 1** zeigt die Situation nach erfolgter Konsolidierung: Der Modellmanager hat mit den jeweiligen fachlichen Ansprechpartnern geklärt, dass ein Sonderzubehör aus der Sicht von Projekt B inhaltlich dem *Zubehör* aus Sicht von Projekt A entspricht. Alle Beteiligten haben sich auf den Konsolidierungsvorschlag geeinigt, eine gemeinsame Klasse Zubehör mit den Attributen zbNummer und name in das Kernmodell aufzunehmen und diese im Paket Vertrieb anzusiedeln.

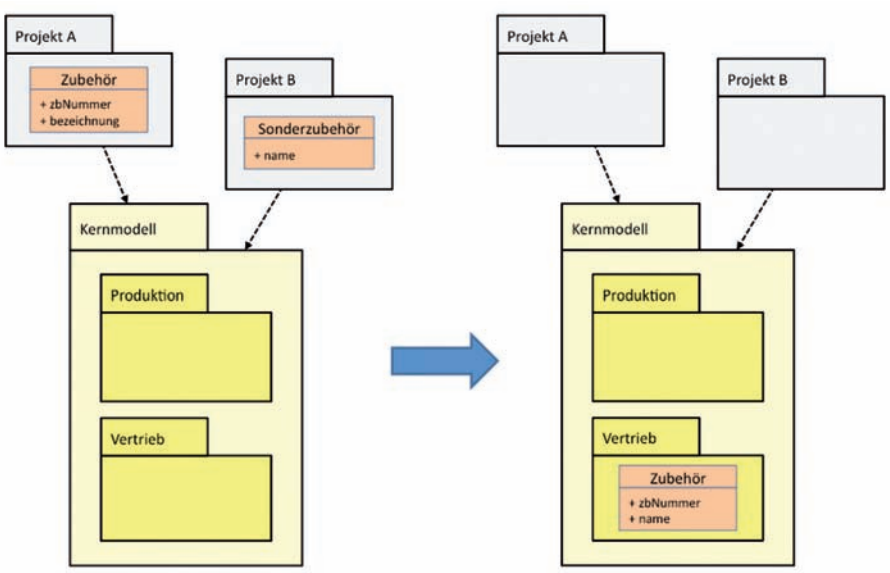

Abb. 1: Konsolidierung redundanter Modellerweiterungen.

Die Konsolidierung von verschiedenen Erweiterungsmodellen kann in der Praxis nur von Hand erfolgen und erfordert Expertenwissen. *Diff*- und *Merge*-Werkzeuge unterstützen den Experten aber immerhin dabei, schnell Unterschiede zu erkennen. Zudem können sie Vorschläge anbieten, wie sich Erweiterungen zusammenführen lassen.

Fachliche und technische Modelle

Über die Aufteilung großer Modelle in Pakete hinaus hat sich für die Strukturierung im Großen eine schichtenweise Strukturierung bewährt, beispielsweise in die folgenden drei Schichten (siehe auch Abbildung 2):

- Auf der obersten Ebene stehen übergreifende *abstrakte Modelle* grundlegender fachlicher Zusammenhänge. Ein Beispiel für derartige Modelle sind fachliche Klassenmodelle der grundlegenden Geschäftsobjekte, die in einer Organisation als Vorgabe gelten. Sie werden von einem kleinen Team von Standardisierern erarbeitet und definieren ein gemeinsames Vokabular, auf dem alle Abteilungen und Geschäftsprozesse der Organisation aufbauen können.
- Auf der mittleren Ebene stehen *fachliche Modelle* für konkrete Geschäftsprozesse oder Systeme. Im Vergleich zu den abstrakten Modellen der oberen Ebene enthalten sie viele weitere nötige Details, die zur vollständigen Spezifikation der Fachlichkeit entsprechender IT-Systeme erforderlich sind. Die Modellelemente verweisen über Trace-Beziehungen auf die Elemente der oberen Ebene, um anzuzeigen, dass sie die entsprechenden Konzepte für einen speziellen Kontext konkretisieren.
- Auf der unteren Ebene stehen *technische Modelle* von IT-Systemen, die wiederum aus den fachlichen Modellen der mittleren Ebene abgeleitet wurden: manuell oder über automatische Modelltransformatoren und Generatoren im Rahmen einer *Model Driven Architecture (MDA)*.

Da jede Ebene einen eigenen Zweck verfolgt, gelten jeweils spezielle Modellierungskonventionen. Gegebenenfalls gibt es sogar für eine einzelne Ebene unterschiedliche Konventionen, beispielsweise wenn verschiedene IT-Systeme jeweils andere Entwicklungsansätze nutzen. Die Situation ist dann in gewisser Hinsicht analog zur Nutzung mehrerer Programmiersprachen für unterschiedliche Komponenten eines großen IT-Systems.

Die Modell-Landkarte

Jede Ebene ist in sich in unterschiedliche Pakete aufgegliedert. Teilweise können sich

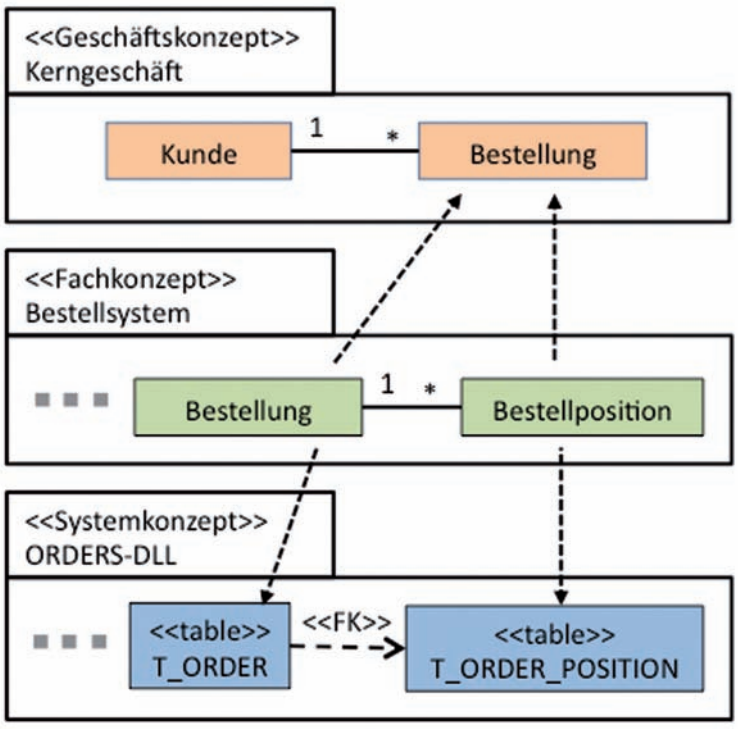

Abb. 2: Das Drei-Schichten-Modell.

		Geschäftsobjekte			
		Kunde	Bestellung	Zubehör	...
Systeme	Bestellsystem	CRU	CRUD	R	...
	Gewährleistungs-system	R	RU	R	...
	CRM-System	CRUD	R	R	...
	...	...	...	...	...

Abb. 3: Modell-Landkarte der Geschäftsobjekte und Systeme.

diese entsprechen (z. B. wenn ein fachliches Modell eines Systems auf der mittleren Ebene direkt zu einem entsprechenden technischen Modell auf der unteren Ebene korreliert), teilweise können sie voneinander unabhängig sein (z. B. wenn auf der oberen Ebene nur wenige Fachklassen vorhanden sind, die sinnvollerweise in einem einzigen Paket modelliert sind).

Aus der Struktur aus Paketen und Schichten lassen sich *Modell-Landkarten* ableiten, die der Modellmanager verwaltet. Diese können vielfältigen Zwecken dienen:

- Im Kontext eines einzelnen MDA-Projekts macht eine Modell-Landkarte den Fortschritt im Projekt von der fachlichen Modellierung einzelner Komponenten bis hin zur Implementierung sichtbar.
- In Rahmen eines größeren Maßnahmenprogramms aus mehreren Einzelprojekten in einem Großunternehmen kann eine Modell-Landkarte dazu dienen, die vorhandenen Modelle der einzelnen Projekte zu identifizieren, ihre Bearbeitung zu koordinieren und eine Wiederverwendung über die einzelnen Projekte hinweg zu erreichen.
- Organisationsübergreifend kann eine Modell-Landkarte dazu dienen, die Bebauungsplanung in der Organisation zu koordinieren, da sie anzeigt, welche Systeme welche grundlegenden Geschäftsobjekte implementieren und welche Lücken und Redundanzen es dabei gibt.

Abbildung 3 zeigt beispielhaft eine Modell-Landkarte für den zuletzt aufgeführten Fall. Die rote Achse stellt die Geschäftsobjekte dar, die grüne Achse die Systeme (siehe auch Abbildung 2). Aus der Landkarte lässt sich ablesen, auf welche Geschäftsobjekte ein System zugreift. Die Art des Zugriffs ist unter Verwendung der *CRUD*-Abkürzungen dargestellt, die für den erzeugenden (*Create*), lesenden (*Read*), ändernden (*Update*) und löschenden (*Delete*) Zugriff stehen.

Werkzeugunterstützung

Die effiziente Entwicklung großer Modelle erfordert adäquate Methoden und Werkzeuge. Beide müssen eine eng verzahnte, funktionierende Einheit bilden: Die Definition der Werkzeugkette ist also der Methodendefinition nicht nachgelagert, sondern beide Aufgaben sind eng mit einander verwoben. In der Regel muss sowohl die Methodik an das Werkzeug angepasst werden, als auch umgekehrt das Werkzeug an die Methodik.

Gerade bei großen Modelle spielen Ablage, Versionierung und Zugriffskontrolle eine wichtige Rolle. Besonders wichtig ist eine Suchfunktion, um wiederverwendbare Modelle zu finden und in eigene Modelle einzubinden. Datenbankgestützte Modell-Repositorys moderner Modellierungswerkzeuge unterstützen diese Aufgaben ebenso wie das parallele Bearbeiten von Modellen durch mehrere Modellierer.

Grundsätzlich reicht aber in vielen Fällen auch ein dateibasiertes Arbeiten aus. Voraussetzung dafür ist, dass das verwendete Modellierungswerkzeug es erlaubt, einzelne Modellpakete in separate Dateien auszulagern und in andere Modelle einzubinden. In diesem Fall ähnelt das Arbeiten stark dem Vorgehen bei Quellcode-Dateien, mit dem Unterschied, dass eine Modellzusammenführung zweier Versionen eines Modells wesentlich schwieriger ist als eine textbasierte Zusammenführung zweier Quellcode-Dateien. Häufig wird deshalb mit exklusiven Sperren auf einzelnen Paketen gearbeitet.

Die Basisfunktionen für Verwaltung und Bearbeitung bilden die Grundlage für höhere Funktionen, die den Prozess der Modellierung von der Erarbeitung und Abstimmung bis hin zur Veröffentlichung unterstützen. Dazu gehören insbesondere:

- Eine automatisierte Prüfung von Modellen erlaubt es, grundlegende Fehler und Inkonsistenzen in Modellen zu finden – eine Funktionalität, die bei Quellcode zu großen Teilen vom Compiler und zusätzlich von statischen Codeanalyse-Werkzeugen wie „FindBugs" (vgl. [Sou09]) erbracht wird. Aufgrund der unterschiedlichen Einsatzzwecke von Modellen müssen derartige Prüfwerkzeuge konfigurierbar sein, damit nicht ein abstraktes fachliches Modell mit den Detailprüfungen für technische Modelle konfrontiert wird. Besonders wichtig für die Arbeit mit großen Modellen ist eine automatisierte Prüfung der Abhängigkeitsstruktur zwischen den Paketen.
- Die Erzeugung von Dokumenten ist vor allem zur Abstimmung von Modellen essenziell. Modellierungswerkzeuge sind geeignet, um Modelle zu erstellen und darin zu browsen – nicht jedoch um sie zu lesen oder einem Review zu unterziehen. Für diesen Zweck ist ein Dokument nötig, in dem ein spezifischer, für den jeweiligen Zweck ausgewählter Ausschnitt der im Modell verteilten Informationen in sequenzieller, gegliederter Form zum Lesen aufbereitet ist. Im Idealfall können Review-Anmerkungen in diesem Dokument dann wieder in das Modell zurückgespielt werden, wo sie von den Modellierern abgearbeitet werden. Die getätigten Änderungen sollten dann wiederum den Reviewern in herausgehobener Form im Dokument präsentiert werden, damit diese nicht immer das gesamte Dokument lesen und von Hand auf Änderungen prüfen müssen.
- Die Generierung von Code oder anderen Entwicklungsartefakten gemäß einem MDA-Ansatz (vgl. [And04]) bildet die Schnittstelle von der Modellierung hin zur Entwicklung. Da hier die Anforderungen noch vielfältiger sein können als bei der Erzeugung von

Dokumenten, bieten Werkzeuge typischerweise offene Schnittstellen an oder unterstützen standardisierte Datenaustauschformate wie XMI, über die entsprechende Zusatzwerkzeuge auf die Modelldaten zugreifen können.

Außerdem bieten viele Werkzeuge Mechanismen zur domänenspezifischen Modellierung. Damit lassen sich z. B. methodenspezifische Diagrammtypen mit vordefinierten Elementen erstellen, was viele Fehlerquellen eliminiert und damit die Modellierung effizienter gestaltet.

Zusammenfassung

Das Management von Modellen weist starke Ähnlichkeiten mit dem Management von Quellcode auf: Mit der Erstellung und Ablage von Modellen allein ist es nicht getan. Größere Modelle erfordern neben einer definierten Methodik und geeigneten Werkzeugen eine adäquate Organisation sowie eine gut strukturierte Modellarchitektur, wenn sie längerfristig wartbar bleiben sollen. Da die Freiheitsgrade bei Modellen höher sind als bei Quellcode, ist das Modellmanagement insgesamt eine (noch) anspruchsvollere Aufgabe. ■

Literatur & Links

[And04] A. Andresen, Komponentenbasierte Softwareentwicklung mit MDA, UML 2 und XML, Hanser Verlag, 2004

[Fow98] M. Fowler, Analysemuster: Wiederverwendbare Objektmodelle, Addison-Wesley 1998

[Mar08] R. Martin, Clean Code: A Handbook of Agile Software Craftsmanship, Prentice Hall International 2008

[Sou09] SourceForge.net, FindBugs - Find Bugs in Java Programs, siehe: http://findbugs.sourceforge.net/