Leserbrief

Zum Editorial

(erschienen in OBJEKTspektrum 3/2015)

Lieber Herr Janning,

ich habe das Editorial des OBJEKTspektrum April gelesen und stimme mit der Konklusion zum Thema „Das Ganze ist mehr als seine Teile" (S > ∑(Teile)) grundsätzlich überein – nur reicht das? Auf dieser Basis kann man nur sagen: „Moderne Softwaresysteme sind wahrhaft komplexe Systeme".

Ein Vorschlag: Das Ganze IST die Summe seiner Teile, der Beziehungen zwischen den Teilen und deren Eigenschaften: S = ∑((Teile, Beziehungen), Eigenschaften). Moderne Softwaresysteme sind komplexe Systeme, deren Komplexität sich ergibt aus der Menge der Teile (viele) und der Menge der Verbindungen (Beziehungen) (sehr viele) zwischen den Teilen sowie der Beschreibung der (spezifischen, vielen) Eigenschaften. Was heißt viele? Mehr auf jeden Fall als die Beteiligten (keine Superhelden) von Auge oder mit einem Sammelsurium von Zeichnungen und Beschreibungen festhalten können. Die Menge an Daten *über* ein Software-System (also Informationen, Datenbanken, Programme, Dokumente, Anforderungen, Scripts, involvierte Personen ...) – also Metadaten – geht leicht in den Bereich von mehreren hunderttausend. Ingenieure, die komplexe Bauwerke erstellen, tun dies, indem sie (meist mit CAD) die Teile und ihre Zusammenhänge erfassen (in einer Datenbank) und sie in vielfacher Form und Ausschnitten (als Listen, Zeichnungen, Beschreibungen) visualisieren und dokumentieren. Das tun sie nicht nur bei der Entwicklung des „Produkts", sondern für den gesamten Lebenszyklus und erreichen damit auch die Langlebigkeit ihrer Produkte.

Kürzlich brach bei einer Schublade meiner Küche (Baujahr 2002) die Aufhängung. Also rief ich beim Hersteller an und versuchte mein Problem zu schildern. „Einen Moment bitte, so jetzt habe ich die Pläne vor mir. Um welche Schublade geht es denn?"

Selbstverständlich erstellen auch wir Pläne – nehmen Sie nur als Beispiel das (konzeptionelle) Datenmodell. Dieses wird mit einem Tool als Zeichnung erstellt. Es ist zwar ein zentrales Artefakt der Entwicklung, aber getestet mit Testdaten wird es nicht, nein es wird mit dem relationalen Bügeleisen flach in Tabellen gebügelt und es entstehen ein Datenbank-Schema und eine Datenbank. Mit dieser Datenbank, ersten Programmen und Testdaten wird getestet. Tests führen zu Erkenntnissen und in der Folge zu Änderungen am Datenbank-Schema – aber das Datenmodell wird nicht nachgeführt. Damit ist das ursprüngliche Datenmodell bereits in der Entwicklung veraltet, gar nicht zu sprechen von der wichtigsten Lebensphase des Produkts, in der es genutzt, geändert, weiterentwickelt wird. Dieses Vorgehen treffen wir in allen Bereichen der Anwendungsentwicklung: Ein Modell wird erstellt, daraus wird Code oder Codeschnipsel erzeugt, die manuell nachbearbeitet werden – mit ein Grund, weshalb unsere Systeme an Code-Depositas leiden – sie sind einfach zu fett. Diese Form des „model-driven" Vorgehens ist durchaus nett in einem ersten Schritt, hilft aber im zweiten nicht mehr und schon gar nicht in der Produktionsphase.

Meine Behauptung kann ich durch viele Erfahrungen und Zitate stützen. Eine davon stammt von José Blakely (Architekt für Datenbanken bei Microsoft) aus seinem Paper „Making the conceptual model real" – „Most significant applications involve a conceptual design phase early in the application development lifecycle. Today, many people interpret ‚conceptual' as ‚abstract' because the conceptual data model is captured inside a database design tool that has no connection with the code and the logical relational schema used to implement the application. The database design diagrams usually stay ‚pinned to a wall' growing increasingly disjoint from the reality of the application implementation with time."

Wir erstellen immer umfangreichere, immer komplexere Systeme – aber wir beherrschen die Systeme im Grunde nicht. Wir erstellen zwar Pläne – aber die Bemerkung sei erlaubt – die Informationen *über* diese Systeme („Stücklisten" und „Montagepläne") beherrschen wir nicht. Das ist kein „Engineering".

Ich möchte nicht falsch verstanden werden – ich verstehe unter „Software Engineering" nicht das „Big Design Up Front" und mir gefällt das Bild vom emergenten Architektur-Entwurf der Natur sehr. Die Natur konnte das leisten, weil sie die Baupläne sorgfältig in der DNA aufgezeichnet hat und über viele Jahre über Mutationen der DNA experimentiert (= getestet) hat. Wie managen aber wir die DNA unserer Systeme, wie sollen wir diese entwickeln, testen, speichern, verwalten, visualisieren? Die konventionellen Mittel wie Files in Directories, Scripts, isolierte Pläne reichen wohl nicht. Für die Verwaltung solcher komplexen Strukturen ist auch eine relationale Datenbank schlicht nicht geeignet – wir benötigen eine direkte Abbildungsmöglichkeit für komplexe Strukturen. Bevor ich wegen Beleidigung des relationalen Modells (der Otto-Motor der Informatik) exkommuniziert werde, möchte ich kurz meine Sicht darlegen.

Wir heben die strukturelle und physische Trennung von Modell und Instanz-Datenbank auf. Wir verwalten (auch große) Modelle und kongruent mit den Modellen auch die Instanz-Daten auf der Basis des Entity-Relationship-Modells ...

Was kann man damit machen? Man kann eine DNA aufbauen und den Entwicklungsprozess ebenso unterstützen wie die Nutzung und Pflege von Anwendungssystemen. Sicher, zwar stehen wir noch immer am Anfang, aber es ist zumindest ein erfolgversprechender Schritt zu professionellem Software Engineering. Darüber würden wir gerne reden.

Dr. Reinhold Thurner, Küsnacht (Schweiz)