

COMPUTERWOCHE

NACHRICHTEN ♦ ANALYSEN ♦ TRENDS



Das Vorbild ist der Maschinenbau

Nach wie vor werden zu viele Entwicklungsprojekte erfolglos abgebrochen. Ein Ausweg liegt darin, die Softwareproduktion an den Ingenieursdisziplinen zu orientieren.

VON MARTINA MAIER*

Stellen Sie sich einmal vor, Sie sind mit dem Auto unterwegs. Sie wollen Täler und Flüsse überqueren. Doch von zehn Brücken sind fünf entweder noch im Bau – obwohl sie längst fertig gestellt sein sollten – oder drohen bei einer Überquerung einzustürzen. An zwei weiteren Brücken wurden die Arbeiten im Rohbau abgebrochen. Lediglich drei Brücken können Sie gefahrlos überqueren.

Absurd? Im Brückenbau sicherlich. Doch in der Softwareerstellung immer noch Alltag. Dabei sind es keineswegs die Hightech- oder die besonders innovativen Projekte, die scheitern, sondern Vorhaben rund um alltägliche Aufgaben wie Bestellabwicklung oder Gepäckabfertigung.

Kein Wunder, dass in Management-Kreisen das schlechte Preis-Leistungs-Verhältnis in der Softwareentwicklung beanstandet wird. Da lockt die Offshore-Entwicklung. Die Zielvorgabe könnte beispielsweise lauten: Im nächsten Jahr entwickeln wir mindestens die Hälfte unserer Projekte außerhalb unserer Grenzen. Doch Offshore-Entwicklung ist keine wirkliche Alternative. Sie löst die strukturellen Defizite der klassischen Softwareentwicklung keinesfalls. Im Gegenteil: Offshore-Projekte erzeugen zusätzliche Komplexität. Durch Verteilung der Aufgaben und schwierigere Kommunikation erhöht sich die Fehleranfälligkeit. Die Offshore-Debatte lenkt vom eigentlichen Brennpunkt ab: der fehlenden Qualität.

Die klassische Entwicklung versagt bei der Qualität

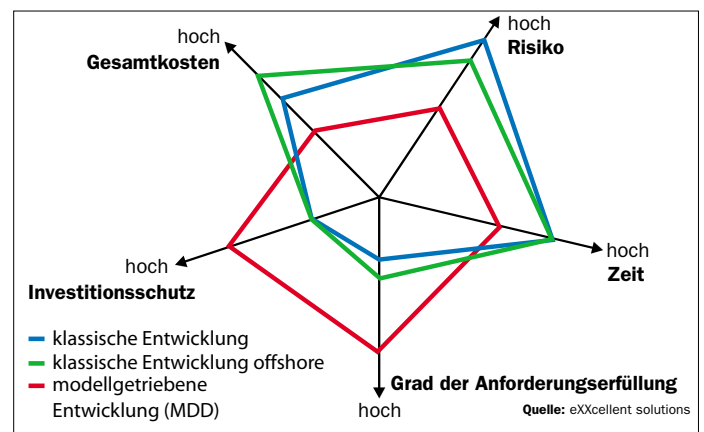
Damit sind wir bei den unbequemen Fragen angelangt: Was ist Softwarequalität? Wie kann sie gemessen und bewertet werden? Die fünf Merkmale:

Kosten, Zeit, Investitionsschutz, Anforderungserfüllung und Risiko sind Dreh- und Angelpunkt der Softwarequalität. Die klassische Entwicklung schneidet in keinem der genannten Qualitätsmerkmale besonders gut ab.

Nach wie vor werden im Entwicklungsprozess die fachlichen Anforderungen nicht systematisch strukturiert und zu wenig auf Richtigkeit, Widersprüchlichkeit, Vollständigkeit und Wirtschaftlichkeit überprüft. Die Beschreibungen sind sprachlich nicht eindeutig und lassen viel Spielraum für Interpretationen. Selbst adäquate Mittel wie UML-Diagramme dienen oft nur als Word-Ersatz. Der Entwickler setzt die qualitativ unzureichenden Artefakte in Codierung um und interpretiert sie zwangsläufig nach seinem Wissen. Anders als im Brückenbau, in dem detaillierte Planungen den Ausführenden wenig Spielraum für Interpretationen lassen, liegt hier in der Softwareentwicklung eine Bruchstelle.

Zudem fehlen die Instrumente, Anforderungsänderungen in die Entwicklung zu übertragen. Sattdessen werden gewünschte Modifikationen verteilt in Protokollen, E-Mails und Request-Tracking-Tools beschrieben oder gleich im Code nachgezogen. Bei Tests und in der Abnahme ist dann nicht mehr klar, was eigentlich getestet werden muss und welche finalen Entscheidungen zur

Vorteile für modellgetriebene Entwicklung



Die verschiedenen Vorgehensweisen der Softwareentwicklung erfüllen die Qualitätskriterien in unterschiedlichem Maß.

Funktionsweise des Systems getroffen wurden.

Die Arbeit der Entwickler lässt sich kaum kontrollieren

Damit wird die Bewertung des Projektfortschritts zum Problem. Zwischen dem Wissen in den Köpfen einzelner Fachexperten und dem Code gibt es keine Zwischenstufe, keine Abstraktionsebene, die dabei helfen könnte, die Arbeit der Entwickler zu beurteilen und notfalls korrigierend einzugreifen. Die Unsicherheit manifestiert sich in überdimensioniertem Projekt-Controlling, das nur scheinbar den Projektstatus widerspiegelt. Das Qualitätsmerkmal „Risiko“ gerät außer Kontrolle.

Schließlich ist der Softwarebetreiber abhängig von Fachexperten und Entwicklern, die als Einzige die Zusammenhänge verstehen. „Never touch a running system“ ist dann der in der Branche wohlbekannte Endstatus des Systems. Keine befriedigende Lösung, wenn das Unternehmen auf neue Her-

ausforderungen reagieren muss.

Diese Probleme verschärfen sich. Denn der Trend geht zu integrierten, immer umfangreicheren und komplexeren Lösungen. Unternehmen funktionieren schon lange nicht mehr ohne Software, aber auch aus dem Alltag ist sie kaum noch wegzudenken. Ist beispielsweise der Speicher in einem modernen Auto defekt oder gelöscht, ist auch der beste Mechaniker der Welt machtlos und kann mögliche Fehler nicht mehr diagnostizieren.

Verdeutlicht der Brückenbau die Notwendigkeit einer engen Verzahnung und Rückkopplung von Entwicklung, Planung und Umsetzung, so hilft der Blick auf die industrielle Fertigung, wenn wir nach Vorbildern für Softwareprojekte suchen.

Im Maschinenbau werden die gestellten Anforderungen zunächst zu CAD-Plänen verarbeitet. Diese bieten Kunden und Konstrukteuren eine Möglichkeit, die Umsetzung der Anforderungen zu überprüfen und zu korrigieren. Die Pläne werden anschließend, zumeist automatisiert, in Steuerprogramme für CNC-Maschinen transformiert, die dann ihrerseits die Einzelteile der künftigen Produkte plangenaue herstellen. In einem weiteren Schritt werden die Zwischenprodukte auf einer Fertigungsstraße zum Endprodukt zusammengesetzt. Entscheidend ist dabei, dass die einzelnen Produkte stufenweise vom grafischen Design in das konkrete Einzelteil überführt und immer wieder auf die Einhaltung der vorgegebenen Anforderungen überprüft werden. Treten Abweichungen auf, so kann jederzeit gezielt gegengesteuert werden.

Fertig- und Halbfertigprodukte verwenden

Die zukunftsweisende industrielle Softwareentwicklung verläuft analog. Das heißt, wir erhalten eine höhere Automatisierung, eine höhere Spezialisierung und verwenden auch Fertig- und Halbfertigprodukte. Durch mehrere untereinander integrierte Entwicklungsschritte entsteht Transparenz, und wir können dadurch die Herstellung steuern und wesentlich flexibler gestalten.

Verfolgt man die einschlägigen Konferenzen und Fachzeitschriften, so dominieren aspektorientierte Programmierung, Service-orientierte Architekturen, verschiedene Frameworks, Model Driven Architecture und Model Driven (Software) Development die Diskussion. Jeder Ansatz hat seine einzigartigen Stärken. Doch für sich allein

genommen ist keiner umfassend genug, die Probleme der klassischen Softwareentwicklung zu lösen.

Der fundamentalste Fortschritt wird in einer modellgetriebenen Softwareentwicklung liegen. Sie beinhaltet wesentliche Prinzipien ingenieurmäßigen Vorgehens. In ihr lassen sich zudem aspektorientierte Ansätze, Architekturkonzepte und Frameworks effizienzsteigernd einbinden. Die Zukunft der Softwareentwicklung lebt letztlich von der gelungenen Integration und der richtigen Mischung der Ansätze.

Die Zukunft gehört der modellgetriebenen Entwicklung

In der modellgetriebenen Entwicklung entsprechen den oben genannten CAD-Plänen grafische Modelle. Diese können auf verschiedener Detaillierungsebene und unter designtechnischen, funktionalen oder fertigungstechnischen Blickrichtungen betrachtet werden. Anhand dieser Pläne können Fachexperten überprüfen, ob die formulierten Anforderungen adäquat umgesetzt sind. Designer können die Modelle unter Zugriffs- und Mengengerüstaspekten betrachten und die Umsetzung in Designmodelle steuern.

Statt der CNC-Maschinen lassen sich Generatoren verwenden. Sie werden mit den grafischen Modellen gefüttert und schaffen automatisch den ausführbaren Code. Der Vorteil: Der Generator hat weder Vorlieben noch Eigenheiten, die sich im Code niederschlagen, macht keine Flüchtigkeitsfehler und ist rasend schnell.

Trennung von Fachlichkeit und Technologie

Diese Vorgehensweise trennt Fachlichkeit und Technologie. Sie löst damit eines der strukturellen Probleme klassischer Softwareentwicklung: die Verschmelzung beider Aspekte im Programmcode. Ändern sich die fachlichen Anforderungen, wird lediglich das Fachmodell angepasst. Der Generator bleibt unangetastet und erzeugt den neuen Code.

Erfahrungswerte zeigen, dass sich der erstellte Generator bereits in einem einzigen Projekt amortisieren kann. Verwendet ihn ein Unternehmen für mehrere Vorhaben, entstehen signifikante Kostenvorteile. Ändern sich technische Anforderungen, wird lediglich der Generator angepasst und mit dem technikunabhängigen Wissen in Form von Fachmodellen versorgt.

Die schrittweise Verfeinerung von Modellen hin zur angestrebten Soft-

warelösung bedingt ein klar definiertes Vorgehen und die Unterstützung durch geeignete Softwarewerkzeuge. Ohne diese kann nicht gewährleistet werden, dass die Transformationen so bewältigt werden, dass Konsistenz und Vollständigkeit der Informationen gewahrt bleiben.

Änderungen stellen großes Projektrisiko dar

Klar heißt keinesfalls starr. Der klassischen Entwicklung wird von den Anhängern der agilen Entwicklung oft zu Recht vorgeworfen, dass die verwendeten Prozesse zu unflexibel sind. Immer wieder gibt es Projekte, wo trotz des Wissens über Qualitätsmängel in der Spezifikation die notwendige Korrektur nicht vorgenommen wird, weil diese Prozessphase bereits abgeschlossen oder gar vom Kunden abgenommen ist. Das Gleiche gilt für Änderungen. Kontrolliertes Änderungs-Management ist essenziell, da sich Änderungen in Projekten nicht verhindern lassen und eines der größten Projektrisiken darstellen. Allerdings sollte agile Entwicklung nicht von einem einzelnen „genialen“ oder „talentierten“ Designer erwartet werden. Stattdessen werden die „best practices“ instrumentalisiert und allgemein nutzbar gemacht.

Die Vorteile der an den Ingenieursdisziplinen orientierten, modellbezogen-generativen Softwareherstellung sind zunächst struktureller Art: Fachlichkeit und Technologie werden getrennt, die Erstellung der Software erfolgt stufenweise und überprüfbarer. Der Erfüllungsgrad der Anforderungen kann bereits früh nachvollzogen und damit erheblich besser sichergestellt werden.

Durch die Automatisierung verringern sich die Produktionszeiten. Die Kosten sinken teilweise erheblich – vor allem bei einer wiederholten Nutzung des Generators. Die fertige Software kann länger eingesetzt und flexibler angepasst werden.

Nur durch eine umfassende und kritische Auseinandersetzung mit den Schwächen klassischer Softwareentwicklung wird die Zahl der gescheiterten Projekte sinken. Die Konzentration auf den am einfachsten zu vergleichenden Faktor, den Stundensatz, wird keinen Fortschritt und keine Lösung bringen. Diskussionen über Offshore-Entwicklung bleiben weiterhin spannend – aber nur im Kontext einer Qualitätsdebatte. (hv) ◆

*DR. MARTINA MAIER ist Geschäftsführerin der eXXcellent solutions gmbh in Ulm.