FOKUS: SOFTWARE-ENTWICKLUNG

Was bringt konkreten Nutzen?

In Programmiererkreisen werden derzeit viele neue Ansätze diskutiert – von prozessorientierten Business Mash-Ups über Rule-Modellierung bis zum Domain Specific Modelling. Welche Trends sind zukunftssicher?

→ VON DANIEL LIEBHART

Die Verbesserung der Produktivität ist seit Jahren eines der wichtigsten Ziele unserer Branche. Seit dem 1975 erschienen Klassiker des Software Engineerings «The mythical Man Month» von Fred Brooks versuchen Methodiker und Toolhersteller Werkzeuge bereitzustellen, die bei der effizienten Software-Entwicklung helfen sollen. 15 Jahre später hat der Autor in seinem vernichtenden Artikel «No silver bullet» zu erklären versucht, aus welchen Gründen die erwartete Produktivitätssteigerung doch nicht eingetreten ist. Die hohe Komplexität, die fehlende Konformität, die Form- und Unsichtbarkeit der Software sind die Hauptgründe. Heute – rund 20 Jahre später – versprechen uns Toolhersteller und Methodiker mittels graphischer Programmierung, domänenspezifischen Sprachen und umfangreichen Entwicklungssuiten wiederum signifikante Steigerung der Produktivität. Ob diese Programmiertrends die Erwartungen erfüllen, hängt von Ihrem Einsatz ab.

Trend 1 GRAPHISCHE PROGRAMMIERUNG FÜRS BUSINESS

Die graphische Programmierung basiert auf einer sehr einfachen Idee: Eine Software wird mittels einem Modellierungstool graphisch gestaltet. Das Tool generiert aus der graphischen Darstellung ausführbaren Code, welcher entweder durch eine spezielle Engine oder durch vorgängig konventionelle Kompilierung direkt auf dem Rechner ausgeführt werden kann. Es gibt eine Vielzahl solcher graphischen Programmierumgebungen. Heute sind vor allem zwei Arten verbreitet: Instrumente zur Bereitstellung von sogenannten Business Mash-Ups und Werkzeuge für ausführbare Prozesse und Regeln.

Die Programmierumgebungen für Business Mash-Ups eignen sich für einfach Anwendungen im Bereich Office-Processing – zum Beispiel für Workflows wie die Erfassung und Bearbeitung einer Reklamation samt allfälliger automatisierter Antworten an den Kunden und Ablage im Archivierungssystem. Business Mash-Ups gehen davon aus, dass die Funktionalität bestehender Systeme zur Bereitstellung solcher Abläufe aufgerufen werden kann. Der Ablauf selbst wird jedoch nicht im bestehenden System realisiert, sondern als schlanke separate Anwendung umgesetzt. Sie sind der Ersatz für die vielen «Under-the-table»- Anwendungen, die in Fachabteilungen aus der Not heraus in Word, Excel-Makros oder Access-Datenbanken entstanden sind. Sie erleichtern die tägliche Arbeit, sind schnell bereitgestellt und müssen nicht von ausgebildete Entwicklungsingenieure erstellt werden.

«Ein gut eingespieltes, erfahrenes Team arbeitet bis zu 15 mal produktiver»
Daniel Liebhart

Die meisten Hersteller dieser Tools, beispielsweise die Firma Serena mit Business Mashups, IBM Lotus Mashups, Tibco Spotfire oder auch Outsystems mit ihrer Agile Plattform stellen Werkzeuge zur Verfügung, die eine direkte Umsetzung einfacher Abläufe in graphischer Form erlauben. Zudem können einfache Webseiten als User Interface aus vordefinierten graphischen Elementen zusammengestellt und mit den Abläufen verbunden werden (siehe auch Beitrag auf S. 40). Diese Werkzeuge haben sich in der Praxis nur punktuell durchgesetzt, da sie nicht auf standardisierten Mechanismen basieren und ihr Einsatz beschränkt ist. Sobald die Anwendung umfangreicher wird, müssen Erweiterungen gebaut werden. Was wiederum den Einsatz konventioneller 3-GL-Programmierumgebungen erfordert. Da ist ein vollständiger Ersatz durch eine ausgewachsene Anwendung in einer standardisierten Programmiersprache naheliegend und oft auch nicht sehr viel aufwändiger.

Trend 2 PROGRAMMIERUNG VON PROZESSEN & REGELN

Eine völlig andere Art von graphischer Programmierumgebung sind die Werkzeuge für ausführbare Prozesse und Regeln. Sie arbeiten mit formalen und meist standardisierten Sprachen als Grundlage. Beispiele solcher Sprachen sind BPEL, BPMN oder XPDL für ausführbare Prozesse und BRML, SWRL oder RuleML für Regeln.

Die Modellierung erfolgt mittels graphischer Elemente, die den Sprachkonstrukten exakt entsprechen. Sie werden meist im Kontext einer auf SOA-basierenden Grundarchitektur eingesetzt und decken den dynamischen Teil der Business Logik, also die Steuerung der Abläufe, einer Anwendung ab. Der statische Teil ist in Form von Services, die von den ausführbaren Prozessen aufgerufen werden, realisiert.

Jeder der grossen Hersteller hat in seinem Portfolio die entsprechenden graphischen Programmierumgebungen. Diese eignen sich für eine grosse Anzahl von Einsatzgebieten. Ob nun die Abbildung von Validierungsregeln, die Steuerung eines Datenaustausches, die Umsetzung der Meldungsverteilung in einem Enterprise Service Bus oder die Gesamtsteuerung der Anwendung – ausführbare Prozesse und Regeln sind immer im Spiel.

Graphisch modelliert werden diejenigen Teile der Anwendung, die sich im Verlauf des gesamten Lebenszyklus am häufigsten ändern. Änderungen an Abläufen und an der Steuerung einzelner Prozessschritte durch Regeln sind mittels graphischer Tool sehr viel einfacher und schneller zu realisieren als mit konventionellen Programmiertechniken. Die entsprechenden Programmierwerkzeuge werden in naher Zukunft zum Grundwerkzeug jeder Entwicklerin und jedes Entwicklers gehören.

Trend 3 DOMAIN SPECIFIC LANGUAGES

Domain Specific Modelling (DSM) ist der Überbegriff im Bereich Software Engineering, der für die Generierung von Code aus einem abstrakten Modell steht. Die Kernkomponente von DSM ist die Domain Specific Language (DSL), auch als 4-GL-Sprache bezeichnet. Eine DSL ist eine Sprache, die für eine definierte Problemdomäne entwickelt wird. Sie enthält Elemente, die typischerweise den Grundbausteinen des abzubildenden Fachbereiches entsprechen. Die Begriffswelt der Domäne wird übernommen, um Anwendungen basierend auf diesen Grundbausteinen modellieren zu können. Den Elementen einer DSL werden graphische Repräsentationen zugewiesen, anhand derer die zu erstellenden Anwendungen modelliert werden. Die standardisierte Ausprägung von DSM ist MDA (Modell Driven Architecture), die den Code auf Basis von UML-Profilen (Unified Modelling Language) generiert.

Sowohl die Gestaltung einer eigenen DSL als auch der Umgang mit den entsprechenden Werkzeugen und Standards ist sehr komplex und in der Praxis daher nicht sehr verbreitet, obwohl die Produktivität durch die Generierung von Code direkt aus dem abstrakten Modell – welches aus Sprachelementen aus der Problemdomäne aufgebaut ist – erheblich gesteigert werden könnte.

Trend 4 MÄCHTIGE 3-GL-TOOLSUITEN

Die beide heuten am meisten verwendeten Programmiersprachen sind Java und C#. Für diese beiden Industriestandards stehen Entwicklungs-Frameworks zur Verfügung, die auf den Bau von Unternehmensanwendungen ausgelegt sind. Entsprechende Frameworks für mobile oder auch Embedded-Systems-Anwendungen sind ebenfalls verfügbar. Zur Verbesserung der Produktivität haben verschiedene Hersteller umfangreiche Toolsuiten bereitgestellt, die den ganzen Entwicklungsprozess unterstützen. Diese Suiten werden immer mächtiger. Sie sind mit einer Vielzahl von Features ausgestattet, angefangen von Test- und Deployment-Unterstützung über integrierte graphische Tools für GUIs, Data Mapping, Modellierung bis zu Werkzeugen für verteilte Anwendungsentwicklung und Application Lifecycle Management.

Diese Suiten gehören heute schon zum grundlegenden Instrumentarium in der Entwicklung. Allerdings müssen sie, um die gewünschte Produktivitätssteigerung zu erreichen, auch entsprechend eingesetzt werden.

FAZIT: AUF DEN PROGRAMMIERER KOMMT ES AN

Die heutigen Programmiertrends werden das Problem der mangelnden Produktivität der Individualentwicklung nicht abschliessend lösen. Software bleibt komplex, muss auf ständig sich ändernden Plattformen lauffähig bleiben, verändert sich dauernd und ist nicht sichtbar. Die etablierten 3-GL-Programmiersprachen Java und C# behaupten sich mit immer vollständigeren und mächtigeren Entwicklungsumgebungen. Aber auch die graphischen Werkzeuge für die Programmierung von Workflow- und Rule-Engines haben sich etabliert und werden künftig zum festen Bestandteil der Software-Entwicklung. Im Gegensatz dazu bleiben Business Mash-Ups und Domain Specific Languages wohl auf Nischen beschränkt.

Wie viel produktiver nun mit den neuen Programmiertrends gearbeitet werden kann, ist leider statistisch nicht bewiesen. Ganz sicher ist jedoch, dass auch angesichts der neuen Trends und Werkzeugen eine wichtige universelle Wahrheit bestehen bleibt: Je besser der Entwickler oder die Entwicklerin, desto produktiver wird programmiert. In einer Vergleichstudie wurde belegt, dass ein gut eingespieltes und erfahrenes Team bis zu 15 mal produktiver arbeitet als andere Teams. Diesen Steigerungsfaktor kann kein noch so innovativer Programmiertrend erreichen. ←

BILD: FOTOLIA

Daniel Liebhart ist Dozent für Informatik an der Hochschule für Technik in Zürich und Solution Manager der Trivadis AG. Er ist Autor von «SOA goes real» und Co-Autor verschiedener Fachbücher