Domain Driven Development

Markus Völter, voelter@acm.org, www.voelter.de

Kontext

Softwareentwicklung ist zu teuer. Dafür gibt es viele, viele Gründe, ich möchte hier nur einen herauspicken: Wir versuchen zu viele verschiedenartige Probleme mit denselben Werkzeugen zu lösen! In der Nicht-Software Welt gibt es Dinge wie Werkzeugmacher und Automatisierungstechnik. Für eine bestimmte Problemstellung (Schreinerei, Autobau, Hausbau) werden für die Problemstellung passende Werkzeuge bzw. Fertigungsstraßen erstellt, die es dann erlauben, effizient Produkte in der betreffenden Domäne zu erstellen. Ich denke, dass dieser Ansatz auch in der Softwareentwicklung eine zunehmende Bedeutung erlangen wird; letztlich versteckt sich genau dies hinter dem Begriff Domain-Driven Development.

Aufgaben

Was bedeutet dies konkret? Die Softwareentwicklung wird sich viel stärker nach den Domänen ausrichten. Unternehmen werden für ihre Domäne passende Werkzeuge kaufen, fertigen lassen oder selbst erstellen. Tools und IDEs wie wir sie heute kennen werden auf den "Metalevel" verlegt: es wird also Tools geben, um Domänenspezifische "Produktionsinfrastrukturen" erstellen zu können.

Die Aufgaben der Entwickler werden sich stärker differenzieren in

- Entwickler/Hersteller die Tools bauen, um Domänenspezifische Produktionsinfrastrukturen effizient erstellen zu können.
- Domänenexperten/-architekten werden mit diesen Tools die "Produktionsinfratruktur" für die betreffende Domäne aufsetzen
- Anwendungsentwickler werden damit konkrete Anwendungen (für die spezifische Domäne) bauen.

Tools

Um all dies zu ermöglichen, werden andere Tools zum Einsatz kommen. Modellgetriebene Softwareentwicklung [SV05], also die Idee, mittels domänenspezifischer Sprachen Sachverhalte der Domäne kurz und prägnant zu beschreiben spielt dabei eine zentrale Rolle. Erste "Annäherungen" an solche Tools gibt es bereits heute. Dazu zählen Generatoren, IDE-Frameworks, anpassbare Modellierungstools. Diese Tools sind bereits sehr gut einsetzbar, bilden aber nur einen kleinen Teil der nötigen Toolkette ab – beispielsweise sind sie alle relativ schwach bzgl. der Erstellung von Editoren für die betreffende domänenspezifische Sprache (DSL) ganz zu Schweigen von Debuggern oder Refactoring-Unterstützung.

Zukünftige Werkzeuge zur Modellgetriebenen Entwicklung werden es erlauben, DSLs aus DSL-Bibliotheken zusammen zu setzen, dafür entsprechende Editoren zu erstellen und die damit definierten Modelle dann mittels Transformatoren und Generatoren in ausführbaren Code zu wandeln. Debuggen wird in diesen Tools auf Ebene der DSL möglich sein.

Die MDA wird in diesem Zusammenhang in zweierlei Hinsicht eine Rolle spielen. Zum einen werden viele DSLs auf UML + Profilen aufbauen (wenn der Toolsupport dafür endlich etwas besser wird). Zum anderen werden einige Aspekte (z.B. Metametamodell, Transformationssprachen) durch die MDA standardisiert werden.

Für technische Basisdomänen wie Enterprise, Distributed Realtime Embedded (DRE), GUI werden die betreffenden DSLs und Tools bereits fertig kauf- bzw. herunterladbar sein. Als Basis dafür werden die MDA Core Models dienen [OMG]. Firmen werden ihre eigenen Infrastrukturen auf diesen verfügbaren aufbauen und damit quasi einen Stack von {DSL, Editor, Transformationen, Pattform}-Tupeln aufbauen, die mit jeder Ebene spezifischer für die betreffenden Domäne werden.

Die Anpassung der IDEs für die betreffende Domäne wird aber nicht nur DSLs enthalten. Sie wird für die betreffenden Rollen im Entwicklungsprozess passende Sichten, Wizards, Tools, Assistenten und Guidelines umfassen.

Das alles ist heute schon teilweise Realität, also mitnichten nur Zukunftsmusik: Eric Evans Buch zum Thema Domain Driven Design [EE03] zeigt, wie man "domänennah" denkt, modelliert und implementiert. Microsoft macht mit dem Software Factories Ansatz [GS04] schon einen großen Schritt in diese Richtung. Integrierte, wenn auch proprietäre Meta-Case-Tools sind vorhanden [Meta]. IDE Frameworks gibt es beispielsweise in Form von Eclipse. Frameworks zur effizienten Erstellung von Generatoren sind verfügbar [OAW], die Anbindung an "Modellierungstools" wie Visio, bzw. die Erstellung von Editoren ist in Arbeit. Domänenspezifische IDE-Anpassungen werden – wenn auch mit mehr Aufwand als wünschenswert – bereits gemacht [RV05]. Das Thema "DSLs aus Bausteinen zusammensetzen" ist derzeit Thema auf den einschlägigen Konferenzen [ECOO].

Folgen?

Was bedeutet dies nun für uns als Entwickler? Zum einen, denke ich, wird es eine starke Spezialisierung geben im Bezug auf die drei oben geschilderten Rollen. Die Bedeutung von Offshoring wird zurückgehen, weil keine so großen Mengen an "Handarbeit" mehr anfallen. Die Schlüsselqualifikationen für Softwareentwickler werden sich aber nicht ändern (Abstraktionsvermögen, (mentale) Modellbildung). Um unsere Arbeitsplätze brauchen wir uns also keine Sorgen zu machen ☺

Referenzen

SV05 Stahl, Völter, Modellgetriebene Softwareentwicklung, dPunkt, 2005

EE03 E. Evans, Domain Driven Design, Addison-Wesley 2003

GS04 Greenfield, Short, Software Factories, Wiley, 2004

Meta Metacase Consulting, MetaEdit+, http://www.metacase.com

OAW http://www.openarchitectureware.org

ECOO ECOOP 2004 Workshop on Evolution and Reuse of Language Specifications for DSLs, http://www.ifi.uio.no/ecoop2004/

OMG http://www.omg.org/mda/specs.htm#Profiles

RV05 Rudorfer, Völter, Domain-specific IDEs in embedded automotive software, EclipseCon 2005