Eclipse zur Domänenspezifischen Entwicklung

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

Eclipse ist nicht nur eine gute, mit „coolen“ Plugins erweiterbare Java IDE. Aufgrund der offenen Architektur und der zwar umfangreichen, aber gut strukturierten API ist es eine ideale Plattform, um domänen- oder projektspezifisch erweitert zu werden.

Was ist denn nun die „beste Java IDE“? Über dieses Thema kann man sich anscheinend trefflich streiten, vor allem die IntelliJ/Eclipse-Fraktionen tun dies ja sehr gerne und sehr hartnäckig. Meiner Meinung nach ist Eclipse die beste IDE – aber nicht, weil es das eine oder andere coolere Java-Editing Feature besitzt, sondern wegen der Tatsache, dass es sich bei Eclipse um eine IDE-Plattform handelt. Seine Mächtigkeit erhält Eclipse durch die Erweiterbarkeit. Diese Erweiterbarkeit beschränkt sich nicht darauf, dass man sich die neuesten und tollsten Plugins aus dem Web saugt (wobei ich zugeben muss, dass es da eine ganze Reihe nützlicher kleiner Helfer gibt!). Wirklich interessant wird es, wenn man sich für die eigene Projektarbeit eigene Plugins schreibt um die eigenen, projektspezifischen Abläufe zu vereinfachen. Darauf möchte ich in diesem Artikel kurz eingehen.

Hintergrund: Meine aktuellen Projekte verwenden alle den Ansatz der Modellgetriebenen Softwareentwicklung [Voe04]. Unter anderem geht es dabei darum, für eine bestimmte Domäne (beispielsweise Middleware für eingebettete Komponenten, Mobile Phones, J2EE) die Softwareentwicklung durch Automatisierung des Softwareproduktionsprozesses effizienter zu gestalten. Dies geschieht vor allem dadurch, dass man mit Domänenspezifischen Sprachen Sachverhalte der Domäne beschreibt, und diese dann mittels Generatoren in auf der entsprechenden Zielplattform lauffähige Artefakte (also in aller Regel Code) überführt. Eine der Haupttätigkeiten ist es also, sich eine entsprechende Toolkette für die Softwareentwicklung in der betreffenden Domäne zu erstellen. Natürlich spielen da Compiler und Generatoren eine zentrale Rolle; für diese ist eine GUI nicht so entscheidend, eine Ant-Integration ist da wichtiger. Was man aber auch benötigt, sind ggfs. Modellierungswerkzeuge/Editoren, die es erlauben, die betreffenden Spezifikationen oder Modelle für die Domäne zu erstellen. Im Rahmen meiner aktuellen Projekte erstellen wir diese Editoren in Form von Eclipse Plugins, sodass sie sich schön in Eclipse integrieren. Im Folgenden einige konkretere Beispiele.

Meine beiden aktuellen „Hauptprojekte“ befassen sich mit der Entwicklung von Middleware für eingebettete Systeme. Dabei verwenden wir zur Generierung den openArchitectureWare Generator [OAW]. Dieser ist in Java implementiert. Auch die Konfiguration des Generators geschieht zum Teil durch die Entwicklung von Javaklassen. Insofern ist die Java-Funktionalität von Eclipse hier sehr nützlich und wichtig. Der andere

Teil der Konfiguration des Generators besteht im Schreiben von Codegenerierungstemplates. Diese sind bei openArchitectureWare in einer eigenen, auf die Beschreibung solcher Templates optimierten Sprache zu schreiben. Der Generator umfasst ein Eclipse-Plugin welches es erlaubt, im Rahmen von Eclipse diese Templates zu erstellen (incl. Syntax Highlighting, Code Completion und Syntax Checking). Die Konfiguration des Generators durch Klassen und Templates kann also komplett und komfortabel innerhalb Eclipse erfolgen.

In beiden Projekten basiert die Zielplattform auf C/C++. Da auch im Rahmen Modellgetriebener Entwicklung noch Code von Hand geschrieben werden muss, und man den generierten Code debuggen können sollte, wird eine C/C++ IDE benötigt. Auch hier kann Eclipse punkten. Das CDE (C/C++ Development Environment) ist zwar vielleicht nicht die beste C/C++ IDE der Welt, aber für unsere Zwecke ist sie völlig ausreichend und integriert sich sehr schön mit der restlichen IDE.

Besonders spannend wird es allerdings bei der Modellierung. Modellierung findet bei uns in zweierlei Hinsicht statt: Zum einen modellieren wir das Domänenmetamodell; daraus werden dann die oben erwähnten Klassen zur Konfiguration des Generators generiert. Zum anderen erstellen wir Modelle unserer (zu generierenden) Anwendungen. Aus denen wird dann die eigentliche Zielanwendung generiert.

Die Modellierung des Domänenmetamodells geschieht mittels UML. Dafür kommen prinzipiell beliebige UML Modellierungswerkzeuge in Betracht, die XMI als Exportformat unterstützen. Aktuell verwenden wir Enterprise Architect. Wir planen aber sobald als möglich auf EclipseUML/Omomdo oder ein anderes, direkt in Eclipse ablaufendes UML Tool umstellen. Damit kann die Modellierung des Metamodells auch in „der IDE“ stattfinden.

Der spannendste Part ist sicherlich die Modellierung der Domänenspezifischen Modelle. Diese geschieht nicht mittels UML, sondern mit eigenen Domänenspezifischen Sprachen. Zunächst haben wir begonnen, spezifische XML Schemas/DTDs zu verwenden um die Modelle einfach als XML Dokument zu beschreiben. Auch dies konnte (mehr oder weniger) komfortabel in Eclipse geschehen, durch Verwendung des XMLBuddy Editors. Allerdings ist dieser Weg die Modelle zu beschreiben nicht besonders effizient und (end-)benutzerfreundlich. Viel besser eignen sich grafische oder formular-basierte Editoren. Allerdings ist deren Erstellung relativ aufwendig, und die Anpassung an die sich ständig weiter entwickelnden Modellierungssprachen fehleranfällig. Wir haben uns daher entschlossen, aus dem (in UML beschriebenen) Domänenmetamodell sowie zusätzlichen Beschreibungen die Editoren zu generieren. Für die formularbasierten Editoren setzen wir auf EclipseForms auf (das ist dieselbe API die auch für die Formulare des PDE verwendet wird). Für die grafischen Editoren verwenden wir GEF. Es werden basierend auf oben erwähnten Spezifikationen komplette Plugins erstellt, die es erlauben, bestimmte (Teil-)Modelle des zu erstellenden Systems in Eclipse zu beschreiben. Übrigens hat sich der Ansatz, die GEF Editoren zu generieren sehr gut bewährt, da die manuelle Entwicklung

mit GEF nicht gerade einfach ist … durch automatische Generierung kann man die dort entstehenden Schmerzen erheblich lindern.

Fazit

Aus meiner Sicht liegt die Stärke von Eclipse ganz klar in der Tatsache, dass man die IDE an seine eigenen Bedürfnisse anpassen kann. Dazu zählen eben nicht nur fertige, im Web verfügbare Plugins, sondern insbesondere auch selbst erstellte, die einem die Arbeit im eigenen, spezifischen Projektumfeld erleichtern. Für modellgetriebene Softwareentwicklung sind solche Tools essentiell. Eclipse stellt dafür die mit Abstand beste derzeit verfügbare Basis dar.

Literatur

[Voe04] Markus Völter, MDSD Tutorial, http://www.voelter.de/services/mdsd-tutorial.html

[OAW] openArchitectureWare Generator Framework, http://www.sourceforge.net/projects/architecturware