Einführung in Dylan

Vortrag: *Andreas Bogk* <ich@andreas.org>
WWW: http://www.gwydiondylan.org/

Bericht: *Jens Ohlig* <jo@devcon.net>

So etwas wie eine ideale Programmiersprache scheint es nicht zu geben. Nehmen wir zum Beispiel die Typisierung von Variablen. Es gibt im Grunde zwei Konzepte: dynamisches Typing und statisches Typing. Beides hat seine Vor- und Nachteile: Bei Sprachen wie C muß ich explizit festlegen, ob ich Ganzzahl-, Fließkomma- oder Zeichendaten verwenden will. In Perl dagegen schreibe ich einfach die Variable, die Sprache kümmert sich dann um die Typisierung. Dieses Verfahren ist natürlich viel bequemer, wenn ich einfach nur schnell eine Idee hinprogrammieren will, auf der anderen Seite ist es natürlich verständlich, daß eine solche dynamische Typisierung nicht sehr performant ist. Der Compiler kann den Code nicht zur Compile-Zeit auf den Datentyp hin optimieren. Ein klassisches Dilemma, so scheint es.

Vorhang auf: In diese Problematik hinein kommt Dylan (Dynamic Language), eine relativ junge Programmiersprache, die ihre Wurzeln bei Apple hat. In Dylan kann man problemlos einfach so drauflosprogrammieren und muß sich nicht um Typisierung kümmern. Wenn das Programm dann allerdings auf Geschwindigkeit optimiert werden soll, benutzt man einfach statisches Typing. Das beste zweier Welten, sozusagen.

Ähnliche Konfliktfelder gibt es bei der Art der Sprache: Neben den bekannten imperativen Programmiersprachen wie Pascal und C gibt es noch objektorientierte Sprachen wie Smalltalk, C++ oder Java und zu guter letzt funktionale Sprachen, von denen LISP die bekannteste sein dürfte. Dylan ist auch hier wieder hybrid. Komplett objektorientiert hat es trotzdem Möglichkeiten zur funktionalen Programmierung. Die Syntax von LISP ist ja so eine Geschmackssache. "Es gibt zwei Arten von Leuten: Die einen stehen auf viele Klammern und die anderen hassen es. Die meisten hassen es", fasst Andreas Bogk zusammen. Dylan hat daher eine sehr angenehm zu lesende, Pascal-ähnliche Syntax.

Dylan verfügt aber auch noch über andere Features, die man von einer modernen Programmiersprache erwartet. Garbage collection, das automatische Entsorgen ungenutzter Variablen oder Objekte, kennt man aus Sprachen wie Perl oder Java und natürlich gibt es sie in Dylan auch.

Eine Sache, die man sich bei der objektorientierten Programmierung immer wünscht, ist die Trennung von Methoden und Objekten. "Wenn ich zwei Objekte ’Frau’ und ’Mann’ habe und die Methode ’Sex’ darauf anwenden will -- soll dann die Methode ’Sex’ zum Objekt ’Mann’ oder ’Frau gehören?" machte Andreas Bogk die Überlegungen der Dylan-Designer deutlich.

Weiterhin bietet Dylan durch sogenannte Slots nette Möglichkeiten, um Objekten andere Objekte zu übergeben. Während man typischerweise unter C++ selbst Methoden schreibt, um Member Variables zu setzen oder sie aus dem Objekt herauszuziehen, kann man in Dylan ganz bequem Slots definieren, und die "Getter"- und "Setter"-Methoden stehen automatisch zur Verfügung.

Soweit kurz zu den Features von Dylan, zu denen es durchaus noch mehr zu sagen gäbe. Auf der Implementationsseite sieht Dylan auch relativ vielversprechend aus. Das ursprüngliche Projekt bei Apple kam zwar nie über den Status einer technologischen Machbarkeitsstudie hinaus, aber es gibt noch andere Ansätze. Eine kommerzielle Implementation für Win32 ist fertig (näheres unter http://www.harlequin.com/) und erzeugt Code, der zu 99% so schnell ist wie C.

Eine freie Implementation von Dylan für Unix-ähnliche Betriebsysteme läuft unter dem Namen "Gwydion Dylan". Andreas Bogk ist selbst maßgeblich am Gwydion Dylan-Projekt beteiligt. Haufenweise Informationen hierzu gibt es unter http://www.gwydiondylan.org/. Ganz fertig ist der natürlich in Dylan selbst geschriebene Compiler noch nicht, es fehlen noch Details. Auch die Perfomanz des vom Compiler erzeugten Codes kommt bei weitem noch nicht an C heran, aber Andreas Bogk verspricht Programme, die etwa "um den Faktor 10 schneller als Perl laufen". Immerhin, ein Anfang. Dylan wird noch von sich hören lassen.