

Prozessintegration für die Software-Entwicklung

Eclipse ist weit mehr als nur ein kostenloses Java-Entwicklungssystem. So ermöglicht das Application Lifecycle Framework die Werkzeuge unterschiedlicher Hersteller zu koppeln und somit den Entwicklungsprozess flexibel zu steuern. *Andreas Naujack*



Andreas Naujack
ist Regional Sales Manager
für die Schweiz bei Serena
Software

Mit der im Herbst 2001 erfolgten Freigabe des Quellcodes der Java-Entwicklungsumgebung Eclipse ist IBM ein guter Schachzug gelungen. Wer weiss, was aus dem Projekt ohne die 40 Millionen Dollar Entwicklungskosten, die das Unternehmen gewissermassen als Anschubfinanzierung einbrachte, geworden wäre. Auf Basis des Open-Source-Konzepts war eine der wichtigsten Java-IDEs nun kostenlos verfügbar, und IBM als Mitinitiator und «Betreuer» des Projekts konnte sich im Markt der Java-Entwicklung besser positionieren, vor allem gegenüber Microsofts Visual Studio und Suns Netbeans.

Technisch stellt Eclipse zwar eine Weiterentwicklung der «Websphere Studio Workbench» dar, die wiederum ein Nachfolger der Entwicklungsumgebung Visual Age for Java war. Aber Eclipse sollte von Anfang an mehr sein als nur eine neue Open-Source-Version einer altbekannten Java-IDE. Eclipse sollte vor allem die Integration von Entwicklungstools verbessern und eine Plattform bereitstellen, die es beliebigen Softwareherstellern erlaubt, ihre Tools als Plugins einzufügen.

Kommerzieller Erfolg dank Open Source

Die Initiatoren des Projekts erhofften sich, dass so, mit «Best-of-Breed»-Werkzeugen unterschiedlicher Hersteller, die gesamte Breite der Software-Entwicklung besser abzudecken sei, als das einem einzelnen Hersteller gelingen könnte. Der Verkauf dieser Tools – und nicht mehr wie bisher der IDE selbst – sollte dann auch die kommerzielle Grundlage des Modells bilden.

Rund fünf Jahre nach dem Start von Eclipse lässt sich feststellen, dass beides funktioniert hat: Eclipse hat bei Herstellern und Anwendern breite Zustimmung gefunden. In vielen Unternehmen ist Eclipse heute verbindlicher Standard für die Software-Entwicklung. Es gibt mittlerweile rund 780 000 Eclipse-Plugins, die so gut wie alles abdecken, was für ei-

ne moderne, professionelle Software-Entwicklung benötigt wird. Und Eclipse wurde auch ein kommerzieller Erfolg, denn gerade weil Eclipse als Open-Source-Plattform verfügbar ist, können sich die Softwarehersteller über die offenen Schnittstellen problemlos in das Framework einbringen. Damit wurde ein ganz neuer Markt von beachtlicher Grösse geschaffen.

Abkehr von der Code-Orientierung

Anfangs folgte Eclipse allerdings einem Konzept, das im Kern bereits überholt war. Steigende Anforderungen an die Applikationen, immer komplexere Systemumgebungen und immer engere Zeit- und Kostenvorgaben hatten nämlich die Unternehmen zu der Erkenntnis geführt, dass die möglichst schnelle Erstellung von Programmcode auf Dauer nicht das A und O der Software-Entwicklung sein konnte. Moderne Software-Entwicklung muss vielmehr als ganzheitlicher Prozess verstanden werden, der mit dem Application Lifecycle Management (ALM) gesteuert wird. Der Programmcode, wie er klassischerweise mit Entwicklungswerkzeugen erstellt wird, bildet dabei nur eine Phase im Entwicklungszyklus der Applikationen ab, die Code-Erstellung ist demzufolge bloss ein Abschnitt innerhalb des ALM. Daneben haben insbesondere die Erfassung der Anforderungen, Modelldesign, Codierung, Test, Deployment sowie Betreuung und Betrieb ihren Platz. Alles zusammen bildet einen Gesamtprozess, dessen Phasen nahtlos ineinander greifen müssen.

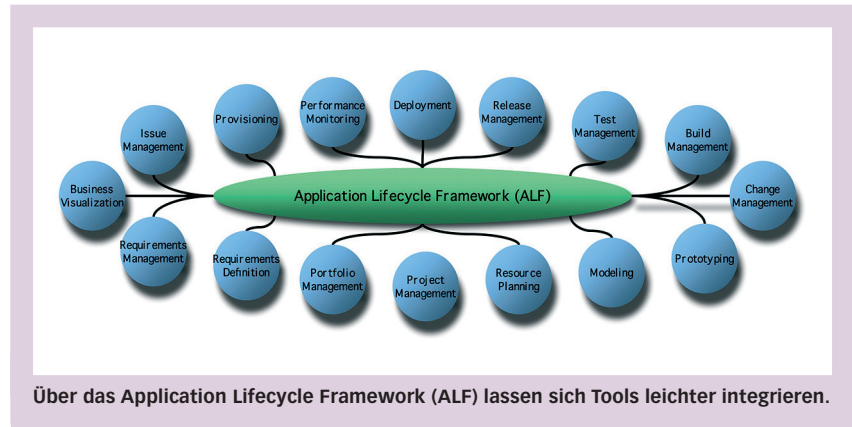
Auf diese veränderte Betrachtung der Software-Entwicklung hat sich Eclipse mit seiner Version 3 eingestellt. Eclipse ist seither nicht mehr eine Java-IDE, in die sich andere Tools als Plugins einklinken, vielmehr werden alle Funktionalitäten, also auch die Java-IDE, als Plugins zur Verfügung gestellt. Der gemeinsame Kern steuert die Kommunikation der Plugins und fällt vergleichswei-

se klein aus – die Features gehören alle in die Plugins. Diese Struktur verleiht Eclipse seine ausserordentlich grosse Flexibilität. Es ist tatsächlich eine hochintegrative Plattform für Software-Entwicklung, und hier liegt auch der technische Grund für die überaus breite Unterstützung für Eclipse, die von einer Vielzahl von Editoren bis zu Change und Configuration Management auf höchstem Niveau reicht.

Integration der Werkzeuge auf Prozessebene

Mittlerweile ist jedoch auch dieses Konzept an neue Grenzen gestossen. In der Praxis hat sich gezeigt, dass Eclipse mit der Steuerung des ALM überfordert ist. Zwar lassen sich einzelne Tools unterschiedlicher Hersteller über Plugins einbinden, aber sie arbeiten mehr neben als miteinander. Die Anwender finden in Eclipse also eher eine Art ALM-Portal, über das sich die jeweiligen Anwendungen ansprechen und Daten austauschen lassen. Die eigentlich beabsichtigte Integration der Werkzeuge auf Prozessebene fehlt jedoch. Dabei ist es gerade das, was Anwender vom ALM erwarten: Durchgängige Prozesse, die über alle Hürden von unterschiedlichen Systemen, Programmiersprachen, Herstellern usw. hinweg funktionieren – eine Grundvoraussetzung für eine wirklich umfassende Steuerung des Entwicklungsprozesses.

In dieser Situation hat sich das Open-Source-Konzept erneut bewährt. Eine echte Prozessintegration im ALM kommt nicht ohne eigene Infrastruktur aus, denn Dienste wie Workflow-Steuerung oder Sicherheitsfunktionen müssen für das ganze System einheitlich und verbindlich geregelt werden. Für diese zentrale Aufgabe waren herstellereigene Ansätze allerdings nur bedingt ge-



eignet, da die Kontrolle der Infrastruktur der Einstieg in eine Kontrolle des Marktes sein könnte: Niemand entwickelt gern Tools für ein System, in dem ein Mitbewerber die Regeln setzt. Auf Basis von Open Source stellt sich dieses Problem nicht.

ALF steuert Kommunikation zwischen den Applikationen

Offenbar haben die an Eclipse beteiligten Hersteller mittlerweile ein so grosses Interesse am Erfolg dieser Plattform, dass sie für deren Weiterentwicklung ihre besten Ressourcen einbringen. So erweitert das von Serena Software initiierte Application-Lifecycle-Framework-Projekt (ALF) Eclipse um eine Infrastruktur, mit der sich im Rahmen von ALM-Prozesse über unterschiedliche Werkzeuge hinweg steuern lassen.

Dafür musste das Eclipse-Konzept erneut überarbeitet werden. Im ALF werden nämlich die ALM-Komponenten nicht als mehr oder weniger statische Plugins eingebunden, sondern mittels eines Workflows lose gekoppelt. ALF stellt einen Workflow Manager zur Verfügung, der die Kommunikation zwischen den einzelnen Lösungen steuert.

Die beteiligten Tools müssen dafür nicht fest verknüpft sein, sondern lediglich mit dem Workflow Manager kommunizieren können. Der Workflow Manager nimmt beispielsweise eine im Requirements Management bearbeitete Anforderung auf, initiiert einen Workflow, der vielleicht eine Zuständigkeit oder Freigabe überprüft, und gibt diesen Workflow bei Erfüllen definierter Bedingungen als neues Event weiter, zum Beispiel an die Modellierung oder die Codierung.

Grosse Flexibilität dank loser Koppelung

Für die Kommunikation zwischen dem jeweiligen Tool und der ALF-Infrastruktur dient ein ALF-Adapter, den der Hersteller des Tools bereitstellen muss. Technisch wird die Zusammenarbeit der Einzelteile im Rahmen einer serviceorientierten Architektur (SOA) realisiert. Die Komponenten des ALM werden als Webservices entwickelt, womit die Konventionen der Kommunikation im Rahmen der Standards erfolgt – ohne Geheimnisse und ohne Fallstricke. Entscheidend ist dabei, dass ALF völlig unabhängig von bestimmten Tools und Herstellern arbeitet und ganz auf etablierten Open-Source-Komponenten beruht.

Durch diese lose Koppelung der Plugins im Rahmen eines Workflows ist ALF von sehr grosser Flexibilität. Egal ob vorhandene Tools von den Herstellern weiterentwickelt werden oder ob neue dazukommen, durch die Einbindung in den ALF-Workflow können sie alle unmittelbar am Entwicklungsprozess beteiligt werden und im Rahmen der definierten Vorgaben in ihn eingreifen. ALF ist damit ein entscheidender Schritt gelungen, um die technische Integrationsaufgabe einzelner Tools prinzipiell zu Gunsten der Fokussierung auf die eigentliche Umsetzung der Businessprozesse zu lösen.

