FRÜHER WAR NICHT ALLES BESSER

Programmieren im Wandel der Zeiten

Von Karlheinz Kronauer*

Seit der Erfindung des Computers sind über 1.000 Programmiersprachen entstanden und viele wieder verschwunden. Nur wenige Klassiker wie Basic oder Cobol haben die rasante IT-Evolution überstanden und werden heute in weiterentwickelten Versionen genutzt.

Noch vor 40 Jahren arbeiteten Programmierer meistens mit Sprachen der 3. Generation wie Cobol, Fortran und PL/1 – oder entwickelten direkt in der 2. Generation, der „Maschinensprache" Assembler. Direkt im Binärcode musste niemand – geschweige denn an Steckpulten wie in den Anfangszeiten. Aber dennoch waren nicht alle mit diesen Systemen glücklich, wie ein Zitat des niederländischen Informatikers und Turing-Award-Preisträgers Edsger Wybe Dijkstra aus dieser Zeit beweist: „Cobol verwirrt das Gehirn. Diese Sprache zu lehren, sollte deshalb als kriminelle Tat verfolgt werden."

Ähnliche Aussagen traf Dijkstra damals übrigens auch zu Basic, PL/1 und Fortran. Um die Software-Entwicklung generell zu vereinfachen, sollten Programmiersprachen der 4. Generation (4GL – „Fourth Generation Language") über eine hohe Abstraktionsebene verfügen. Mit diesen 4GL-Sprachen sollte es dann möglich sein, Anforderungen problemnah und in einer für den Menschen leicht verständlichen Syntax zu formulieren. In den siebziger Jahren gehörten Mainframe-Systeme zum Standard. Datenfernverarbeitung und Dialogverarbeitung waren für viele Betriebe noch exotisch anmutende Fremdwörter. Die wenigen Unternehmensapplikationen, die es gab, liefen im Batch-Betrieb ab, der Source-Code befand sich auf Lochkarten. Zudem waren kaum geeignete Editoren vorhanden, um interaktiv Programme zu entwickeln. Insbesondere die Entwicklung von Bildschirmmasken und Benutzeroberflächen steckte noch in den Kinderschuhen. Syntaxfehler wurden nicht bei der Programmerstellung, sondern erst bei der Kompilierung erkannt. War ein Programmcode endlich syntaktisch korrekt, folgten noch die „Linkage Steps", um das Programm zu vervollständigen. Erst danach konnte ein erster Test des eigentlichen Programmes erfolgen, was den Entwicklungsprozess extrem kompliziert machte.

> „Cobol verwirrt das Gehirn. Diese Sprache zu lehren, sollte deshalb als kriminelle Tat verfolgt werden."
>
> Turing-Award-Preisträger Edsger Wybe Dijkstra

Mit den 4GL-Programmiersprachen wurde dieser Arbeitsaufwand erheblich verringert: erste Programmeditoren kamen auf den Markt, und Programmergebnisse konnten sofort dargestellt werden. Für damalige Verhältnisse waren dies sensationelle Funktionen, die zu drastischen Produktivitätssteigerungen in der Software-Entwicklung führten.

Ein Beispiel liefert die Programmiersprache Natural der Software AG: Mitte der achtziger Jahre waren schon mehr als 1.000 installierte Natural-Anwendungen im Einsatz, die von über 500 Kunden weltweit entwickelt wurden. Einer der ersten Anwender war die Stadt Wien, die das erste Software-System mit der Datenbank Adabas und Natural in Europa entwickelten. Peter Leibl vom Rechenzentrum der Stadt Wien leitete damals das Projekt: „Wir entwickelten die erste Applikation in nur zwei Monaten – und konnten diese anschließend direkt in die Produktivumgebung übernehmen."

*Unser Autor Karlheinz Kronauer ist Director Product Management bei der Software AG

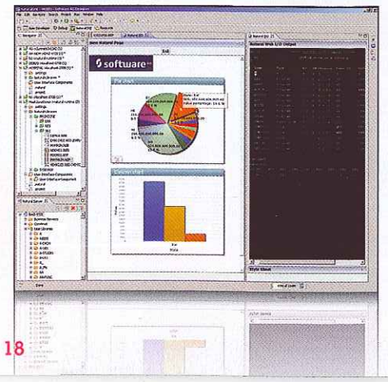
Der Trend zur Objektorientierten Programmierung ließ neue Sprachen wie Java erfolgreich werden. Auch 3GL- und 4GL-Sprachen erhielten entsprechende Spracherweiterungen.

Ebenfalls Mitte der neunziger Jahre erschienen 4GL-Sprachen mit der Idee, dass Entwickler einfach zu bedienende Windows-Entwicklungsumgebungen für die Programmentwicklung sowohl für Mainframes als auch Unix-, Linux- und Windows-Server nutzen konnten. Der Vorteil dieses „Single Point of Development"-Konzeptes: Software kann einmal entwickelt und dann auf den unterschiedlichsten Plattformen genutzt werden – eine Idee, die jetzt angesichts der Gerätevielfalt bei Smartphones und Tablet-PCs auch für mobile Anwendungen neue Aktualität gewinnt.

Mit der steigenden Verbreitung von Eclipse als Entwicklungsplattform – insbesondere für Java-Anwendungen – wurden die 4GL-Sprachen auch in Eclipse integriert. Damit sind Entwickler nun in der Lage, selbst ohne tiefgehende Kenntnisse der Mainframe-Programmierung für Großrechner moderne und webbasierende Geschäftsanwendungen zu entwickeln.

Aktuell arbeiten Unternehmen wie die Software AG (mit ihrer Entwicklungsumgebung „Natural One") daran, die Eclipse-Unterstützung weiter auszubauen. Die Generierung von Web-Services wurde beispielsweise weiter vereinfacht, ebenso die Entwicklung von „Rich Internet"-Applikationen auf Basis von Ajax. Dies ermöglicht die Weiterverwendung bestehender Programmlogik innerhalb von browserbasierenden Web-Applikationen. Auch einzelne Lösungen können nun direkt in die Entwicklungs-Umgebung eingebunden werden, um so die Integration von beliebigen Unternehmensanwendungen, unter anderem SAP-Lösungen, zu ermöglichen und SOA-Governance zu garantieren. Die Entwickler sind so in der Lage, Web-Applikationen und Services zu entwerfen, die auf allen wichtigen Systemplattformen ausführbar sind. ◆

Rich Internet Apps und Mainframe-Integration in einer Natural-One-Umgebung



SPECIAL
OKTOBER 2011
Eine Publikation von IT MITTEL STAND DV-Dialog

1911-2011

100 JAHRE IBM