Die Vorteile von Java im Outputmanagement

Auch wenn die Wahl der Programmiersprache von Anwendungen eigentlich nur eine technische Entscheidung für ein fachliches Problem ist, so hat ihre Verwendung doch auch immer fachliche Konsequenzen. Dies fällt besonders in Bereichen wie Innovationssicherheit, Anpassungsgeschwindigkeit, Wartungsaufwand und Ressourcenverbrauch auf. Dabei gibt es auch für das Outputmanagement einige Faktoren zu berücksichtigen.

Meistens wird der verwendeten Programmiersprache bei der Wahl einer Software keine entscheidende Rolle beigemessen. Sie wird im Rahmen der technischen Umsetzung für ein fachliches Problem ausgewählt. Zu Unrecht, denn die Wahl der Sprache hat sehr wohl Konsequenzen für verschiedene fachliche Bereiche – sowohl für die IT-Abteilung, die für die Lauffähigkeit der Anwendungen verantwortlich ist, als auch für die Fachabteilungen, die mit den Anwendungen arbeiten sollen.

Es gibt es einige Missverständnisse und Gerüchte über die Vor- und Nachteile von Programmiersprachen. In den letzten Jahren wird immer mehr moderne Software in der Programmiersprache Java entwickelt – sowohl außerhalb des Outputmanagements (zum Beispiel werden Apps für Android-Smartphones in Java entwickelt, und für große Softwarehäuser wie IBM oder Oracle ist Java die strategische Programmiersprache) als auch in der Outputmanagement-Branche (icon Systemhaus GmbH mit DOPiX, levigo solutions mit jadice und SET mit der neuen POSY-OutputFactory). Tatsächlich bietet der Einsatz von Java viele Vorteile gegenüber anderen Programmiersprachen.

Geschwindigkeit von Java

Eine der wichtigsten Eigenschaften von Software ist die Geschwindigkeit. Im Outputmanagement dürfen SLAs von Dienstleistern nicht durch die eingesetzte Software gefährdet oder gar das Jahresendgeschäft in der Produktion unnötig in die Länge gezogen werden.

SET (2)

Als einer der ursprünglichen Architekten der neuen POSY-OutputFactory von den zahlreichen Vorteilen von Java überzeugt: Hendrik Leder, Technical Consultant der SET GmbH

Interessanterweise wächst entgegen dem allgemeinen Trend der Kommunikationsbedarf zwischen Unternehmen und Endkunde und damit auch die produzierten Mengen bei vielen Großunternehmen. Die Situation wird durch die vermehrte elektronische Kommunikation und die höhere Erwartung der Endkunden an die Geschwindigkeit der Kommunikation zusätzlich verschärft.

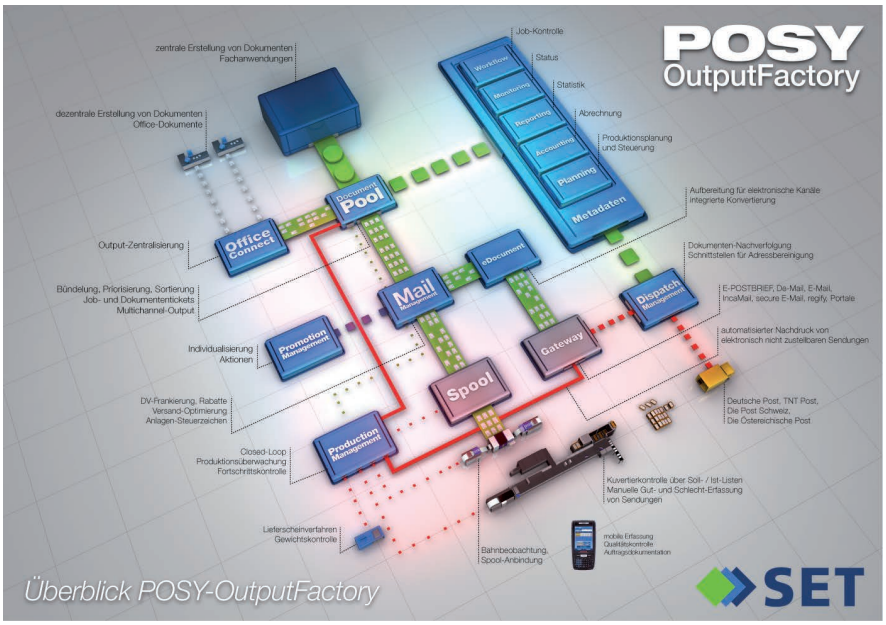
Gerade um die Geschwindigkeit von Java-Anwendungen gibt es viele Mythen und Vorurteile. Bei deren Aufklärung hilft ein Blick auf die Geschichte von Java. Bei den ersten offiziellen Versionen lag der Fokus auf einer größtmöglichen Funktionsvielfalt. Da die Performance der ersten Java-Anwendungen den Entwicklern und Anwendern nicht genügte, wurde der Fokus bereits ab der Version 1.2 stärker auf Stabilität und Geschwindigkeit gelegt. Insbesondere das Swing-Toolkit für Oberflächen-Entwicklung war auf normalen Arbeitsplätzen viel zu träge, und in den Augen von Anwendern entwickelte sich schnell die Meinung: Java ist langsam.

Mit der offiziellen Version von Java 1.3 im Mai 2000 wurde das bis dahin recht langsame Swing-Toolkit optimiert, so dass die Oberflächen seit dem Jahr 2000 durchaus mit anderen Oberflächen mithalten konnten, was wiederum das subjektive Empfinden der Lansamkeit von Java verringert hat.

Doch Java ist nicht nur aufgrund seiner Oberflächen-Frameworks etwas Besonderes. Den Kern von dessen Architektur bildet die Tatsache, dass eine Java-Anwendung in einen plattformunabhängigen Zwischen-Code, genannt Byte-Code, transformiert und anschließend auf unterschiedlichen Plattformen durch den Einsatz von Java Virtual Machines (JVMs) in den nativen Maschinencode übersetzt wird. Es gibt dabei verschiedene JVMs, neben den offiziellen Versionen von Oracle (früher: Sun) auch spezialisierte JVMs, zum Beispiel von IBM für die Plattform IBM z/OS. Zusätzlich gibt es in den JVMs Just-in-Time-Compiler, die einen oft genutzten Code nach Bedarf optimieren, damit die Laufzeit der Java-Anwendung schneller wird.

Mit der aktuellen Java-Version 7 wurden weitere Performance-Optimierungen hinzugefügt. Dabei ist Java nicht generell schneller oder langsamer als Programmiersprachen wie C, COBOL, C++. Es gibt Systemoperationen, in denen Java schneller ist als andere Systeme, bei anderen sind diese Java überlegen. Heutzutage kommt es allerdings nicht mehr auf solche Systemoperationen oder „Mikro-Optimierungen" an; das gesamte System muss schnell sein.

Moderne Softwaresysteme sind sehr umfangreich, und die Optimierung der Systemarchitektur bietet deutlich mehr Potenziale als die Optimierung einzelner Funktionen. Durch Mikrooptimierung kann man eine Optimierung im zweistelligen Prozentbereich erreichen,



Die neue POSY-OutputFactory basiert zu 100 Prozent auf Java.

durch Architekturoptimierungen jedoch drei- oder vierstellig. Dafür muss man aber die Architektur einer Software sehr strukturiert designen und regelmäßig Refactorings (aufräumen und an neue Anforderungen anpassen) durchführen. Mit Java kann man die Architektur einfach strukturieren und so mehr Optimierungspotenziale und eine schnellere Entwicklungsgeschwindigkeit erreichen.

Entwicklungsgeschwindigkeit mit Java

Die Geschwindigkeit, mit der Entwickler eine Software programmieren, mag auf den ersten Blick für die Endanwender unwichtig sein, bei genauerer Betrachtung ist dies aber ein ganz entscheidender Punkt. Ein Schlüsselmoment für einen Anwender ist, wenn eine Software in einem Worst-Case-Szenario nicht funktioniert, die Produktion stillsteht und die Software erweitert oder korrigiert werden muss. Durch den Einsatz von Java kann zwar nicht garantiert werden, dass eine Software fehlerfrei ist – das wird Software wohl nie sein –, aber Fehler können schnell behoben werden. Durch Java in Verbindung mit modernen Softwareentwicklungs-Methoden wie Scrum kann man darüber hinaus die Fehlerwahrscheinlichkeit im Vergleich zu herkömmlicher Softwareentwicklung deutlich reduzieren.

Java bietet für die Lokalisierung von Fehlern sehr aussagekräftige Fehlermeldungen, so genannte Stacktraces. Sie zeigen dem Entwickler genau, an welcher Stelle das Programm abgestürzt ist. Für andere Programmiersprachen müssen Entwickler solche Mechanismen explizit nachprogrammieren.

Eine höhere Entwicklungsgeschwindigkeit ist nicht nur für das Identifizieren und Lösen von Fehlern hilfreich. Bei neuen Projekten oder durch den Einsatz neuer Technologien, wie zum Beispiel neuer Hardware im Druckzentrum oder neuer Anwendungen mit benötigten Schnittstellen, kann sie zu einer schnelleren Umsetzung interner Projekte beitragen. Die weltweite Community von Java-Nutzern hat in den letzten Jahren außerdem dafür gesorgt, dass es kaum eine Programmiersprache gibt, für die es so viele frei einsetzbare Software-Frameworks gibt, die auch in Unternehmensapplikationen eingesetzt werden dürfen. Diese Frameworks (die bekanntesten von ihnen sind inzwischen durch die Apache Software Foundation gebündelt) helfen Softwareherstellern, bestimmte Probleme standardisiert zu lösen. Dazu zählen kleinere Frameworks zum Loggen aller Operationen bis hin zu offenen Webservern. Der große Vorteil dieser offenen Frameworks ist die große Verbreitung und die damit verbundene hohe Qualität. Diese Systeme sind in allen Branchen weltweit im Einsatz und müssen höchsten Sicherheitsbestimmungen genügen. Ein weiterer Vorteil für die Erhöhung der Entwicklungsgeschwindigkeit sind die eingesetzten Entwicklungsumgebungen. Für die Java-Community wurden sehr mächtige Entwicklungsumgebungen wie Eclipse entwickelt, die die Arbeit der Programmierer vereinfachen.

Die SET GmbH hat im Laufe der letzten 20 Jahre verschiedene Programmiersprachen evaluiert und dabei die Erfahrung gemacht, dass größere Softwarebestandteile mit Java mehr als doppelt so schnell entwickelt werden können wie mit den meisten anderen Programmiersprachen inklusive C und C++.

Plattformunabhängigkeit

Ein entscheidender Vorteil ist natürlich die Plattform-Unabhängigkeit von Java: Entwickler programmieren einmalig ihren Code, und dieser ist auf allen Systemen lauffähig. Doch welchen Vorteil bietet das für den Anwender? Sollte beispielsweise eine Unternehmensrichtlinie einen Plattformwechsel vorschreiben, ist die Portierung bei Nutzung von Java-Anwendungen deutlich einfacher. Wenn Konfigurationen in der POSY-OutputFactory erstellt werden, so spielt es keine Rolle, ob die Prozesse später auf einem Linux-, einem Windows-, einem AIX- oder einem IBM z/OS-Mainframe ablaufen. Beim Wechsel zwischen den Plattformen entsteht daher für die POSY-OutputFactory nur der einmalige Installationsaufwand; die Konfigurationen können genauso weiterverwendet werden. Das bedeutet auch, dass ein individueller Code für bestimmte Plattformen nicht nötig ist. Deren größter Nachteil ist oft eine schlechtere Qualität durch einen erhöhten Kostenaufwand für die zahlreichen Tests. Sobald ein Betriebssystem besonders optimierte Funktionen anbietet, die in einer Java-Anwendung genutzt werden sollen, wird dies durch spezielle Java-Bibliotheken ermöglicht.

Reduzierte Fehlerquellen

Durch die Architektur von Java mit der JVM können einige Fehlerquellen umgangen werden. Sobald Anwendungen innerhalb eines Application-Servers laufen, wie es für größere Systeme Standard ist, wird bei einem Systemabbruch nur die einzelne Java-Anwendung abbrechen und sich beenden. Bei der Verwendung von Software, die mit C oder C++ ge-

schrieben wurde, kann der Abbruch einer Anwendung zum Abbruch des gesamten Application-Servers führen – der potenziell viele andere Systeme beinhaltet, die gegebenenfalls überhaupt nichts mit dem Outputmanagement zu tun haben. Für die IT-Sicherheit ist daher die Verwendung von C- oder C++-Anwendungen riskanter.

Java-Anwendungen im Serverbereich sind zusätzlich sicherer gegen Attacken von Unbefugten. In Java gibt es keine Möglichkeit, auf den nativen Speicher zuzugreifen; dadurch werden Buffer-Overflow-Attacken vermieden. Diese Sicherheit entlastet die IT-Abteilungen.

Eine weitere Fehlerquelle von C- und C++-Anwendungen wird in Java durch einen internen Mechanismus komplett ausgeschlossen: In vielen Programmiersprachen muss sich der Entwickler darum kümmern, welchen Speicher eine Anwendung im Arbeitsspeicher halten und welcher Speicher dort wieder freigegeben werden soll. In Java ist der Mechanismus „Garbage-Collector" implementiert, eine automatisierte Speicherbereinigung. Dadurch wird die Wahrscheinlichkeit von Memory-Leaks, also einem permanent ansteigenden Speicherverbrauch, deutlich verringert.

Anwendungen im Outputmanagement

Systeme im Outputmanagement stellen, je nach System, höchst unterschiedliche Anforderungen an die einzusetzende Programmiersprache: In der Dokumentenerstellung durch Systeme wie DOPiX werden insbesondere CPU-intensive Operationen benötigt, während in der Prozessoptimierung durch POSY insbesondere I/O-lastige Operationen entscheidend sind. Für beide Zwecke bietet Java besonders optimierte Bibliotheken, die eine Entwicklung mit Java honorieren.

Im Outputmanagement gibt es ganz unterschiedliche Anwendungen. Je mehr davon eingesetzt werden, umso schwieriger ist es, einen kompletten Überblick zu gewinnen. Daher ist es das Ziel, verschiedene Anwendungen so zu integrieren, dass ein Überblick einfacher wird. Auch für diesen Fall bietet Java einige Vorteile. Durch ein großes Portfolio an Standardschnittstellen, welche die Programmiersprache von Haus aus mitbringt, ist eine Integration verschiedener Systeme möglich, insbesondere von Anwendungen in technische Monitoring-Systeme, da die standardisierten Logging-Frameworks gleichartig genutzt werden können. Darüber hinaus bietet Java mit JMX (Java Management Extensions) eine Technologie, um das Verhalten von Systemen zu überwachen und die Kommunikation zwischen verschiedenen Java-Anwendungen zu erleichtern.

Neben der fachlichen Anwendung von Systemen im Outputmanagement gibt es auf der anderen Seite die IT-Abteilung, die für die Maintenance der Anwendungen zuständig ist und für die Java entscheidend sein kann. Durch die hohe Verbreitung ist Java in vielen Konzernen im IT-Bereich schon zu einem Standard geworden. Dies führt dazu, dass das Know-how der IT-Abteilungen im Java-Umfeld stetig zunimmt. Architekten für Systeme beschäftigen sich nur noch mit Java-Themen wie den einzusetzenden VMs, anstatt sich darum zu kümmern, die verschiedenen Systeme unter einen Hut zu bekommen. Dies führt zu Economies of Scope mit geringeren Personalkosten im IT-Bereich und spezialisiertem Wissen.

Für die IT-Abteilungen spielen nicht nur die Personalkosten, sondern auch die Ressourcen-Nutzung von Anwendungen eine Rolle. Diese ist insbesondere im Großrechner-Umfeld relevant. Unter IBM z/OS hat sich IBM eine Besonderheit einfallen lassen: Mit dem Spezialprozessor zIIP entstehen zur Laufzeit von Java-Anwendungen keine Kosten.

Entscheidung für die Zukunft

Sobald eine Entscheidung für oder gegen eine neue Anwendung gefällt wird, wird der ROI betrachtet. Während dieser relativ einfach berechnet werden kann, ist dies bei der Investitionssicherheit komplizierter. Die Investition in eine neue Software muss in der Zukunft gesichert sein. Auch dafür bietet Java einige Vorteile.

Java ist eine moderne Programmiersprache, die sich inzwischen in sämtlichen Bereichen, sogar im Embedded-Bereich, also in sehr Hardware-nahen Prozessen, ausgebreitet hat. Fast sämtliche Ausbildungsstätten, von Universitäten über Fachhochschulen bis zu Berufsschulen, lehren Java. Das führt dazu, dass es keine Personalknappheit an Entwicklern gibt, wie derzeit bei C-, C++- oder COBOL-Entwicklern. Auch deshalb tauschen heute viele Unternehmen ihre Software komplett gegen moderne in Java entwickelte Software. Wer ein System anwendet, dem eine veraltete Programmiersprache zu Grunde liegt, riskiert, dass dieses nicht langfristig unterstützt wird. Dagegen kann man sich durch den Gang der Entwicklung darauf verlassen, dass die Weiterentwicklung für Java-Anwendungen sicher ist und diese Programmiersprache zukunftsweisend eingesetzt werden kann.

Zusammenfassung

Viele Vorteile haben Java als Programmiersprache zu einem De-facto-Standard guter Softwareentwicklung gemacht. Eine Java-Oberfläche mit einer C- oder C++-Anwendung im Hintergrund erreicht jedoch nicht die Vorteile einer 100-prozentigen Java-Anwendung.

Die SET GmbH hat zu Beginn der Implementierung der neuen POSY-OutputFactory verschiedene Programmiersprachen evaluiert und wurde von den massiven Vorteilen von Java klar überzeugt. Diese Entscheidung hat sich jetzt nach knapp fünf Jahren als richtig erwiesen. Die Vorteile wie die Schnelligkeit des Systems, zügige Umsetzung von Kundenanforderungen, höhere Sicherheit, große Zukunftsfähigkeit und die strategische Positionierung von Java in den IT-Abteilungen der Unternehmen bieten den Endanwendern einen hohen Mehrwert.

Der Einsatz einer 100-prozentigen Java-Anwendung ist daher ein Indikator für ein modernes Softwaresystem. Dabei tragen Java-Anwendungen nicht nur in Konzernen und Großunternehmen, sondern auch in kleineren Unternehmen entscheidend zum langfristigen Unternehmenserfolg bei. Insbesondere die geringeren IT-Kosten durch Java sind ein Argument für alle Arten von Unternehmen.

Hendrik Leder

Weitere Informationen:
www.set-software.de

SET auf der Cebit:
Halle 3, Stand D34