NOTITIA

AdNovum

Bemerkenswertes von und über AdNovum

Friedlich vereint

J2EE und CORBA für komplexe verteilte Anwendungen

J2EE verlangt Experten

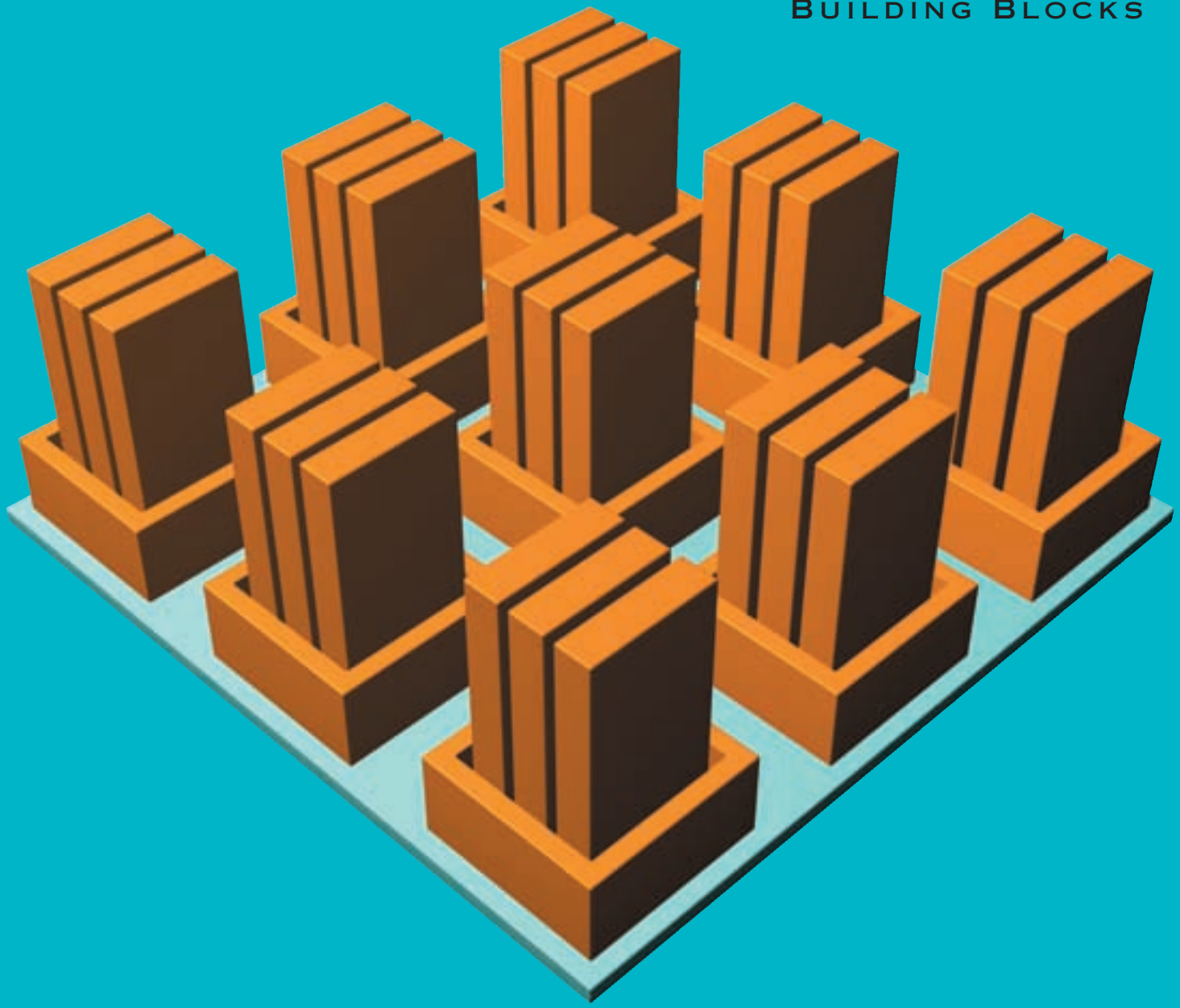
Chancen und Fallstricke produktiver J2EE-Lösungen

High-Grade Security im Finanzbereich

Sicherheit ist nach wie vor ein anspruchsvolles Thema

Frühling 2003, Nr. 4

Building Blocks



Liebe Leserin, lieber Leser

Building Blocks versprechen eine optimale Wertschöpfung von bereits getätigten Investitionen, weniger Risiken und kürzere Entwicklungszeiten, weil bereits Bestehendes wieder verwendet werden kann. Dabei geht leicht vergessen, dass Building Blocks an sich kein Qualitätsmerkmal sind, sondern ihre Vorteile nur zur Geltung kommen, wenn sie zu einem sinnvollen Ganzen zusammengesetzt werden. Die Aufgabe des Entwicklers ist es deshalb, die richtigen Elemente zu wählen

Friedlich vereint

In den letzten Jahren hat in der professionellen IT-Landschaft ein eindeutiger, anscheinend unaufhaltsamer Trend hin zu verteilten Architekturen gemäss den Richtlinien der Java Enterprise Edition, kurz J2EE, eingesetzt. In der Praxis bewährte Standards wie zum Beispiel die Common Object Request Broker Architecture der Object Management Group werden neuerdings von einigen Marktbeobachtern belächelt und totgesagt. Dieser Beitrag versucht, die Situation aus der Sicht der AdNovum zu beleuchten.

Von Stefan Wengi

In Bezug auf Architekturen von verteilten Systemen bietet J2EE grundsätzlich keine revolutionär neuen Ansätze. Es handelt sich vielmehr um einen guten Verschnitt bereits bekannter Konzepte und Muster, der dank einer weitgehend vom Markt getriebenen Spezifizierung relativ schnell realisiert werden konnte. Die meisten Elemente von verteilten Architekturen, sei dies nun CORBA, DCE (Distributed Computing Environment) oder COM (Component Object Model), finden sich in der einen oder anderen Form in J2EE wieder.

Hingegen wird erstmals eine einzige Sprache für die Implementierung von verteilten Systemen vorgeschrieben. Vielfach wird denn auch argumentiert, Entwickler seien gerade dank dem Einsatz von Java um ein Mehrfaches produktiver und die Anforderungen an die Qualifikationen dieser Entwickler müssten nicht so hoch sein wie zum Beispiel in einem C-/C++-lastigen Umfeld. In gewissen Bereichen widersprechen unsere Erfahrungen jedoch diesen Annahmen. Mit Java kann sicher in kürzerer Zeit mehr an applikatorischer Funktionalität realisiert werden, doch kann eine Programmiersprache nicht die systeminhärente Komplexität reduzieren. Das heisst konkret, dass der Entwickler bei seiner täglichen Arbeit mit Problemen wie Multithreading, Session State, Configuration, Scalability und anderem konfrontiert ist und sich permanent bewusst sein muss, dass sein Code in einem komplexen Umfeld betrieben werden muss. Dieses Bewusstsein verlangt weiterhin Profis in der Entwicklung, sonst riskiert man Misserfolge.

> Für Architekturen von verteilten Systemen bietet J2EE keine revolutionär neuen Ansätze.

CORBA als vermeintliche Legacy

Die AdNovum wird, nicht ganz ohne Grund, auf dem Schweizer IT-Markt häufig als eine Bastion der CORBA Middleware angesehen. Dies ist insofern richtig, als die AdNovum die treibende Kraft hinter den weltweit wohl ersten produktiven, sicheren CORBA-Installationen war. Dabei haben wir nicht nur applikatorischen Code geschrieben und zur Produktionsreife gebracht, sondern in Zusammenarbeit mit einem unserer Kunden auch gleich noch den ORB entwickelt. Vom dabei gewonnenen Know-how und Verständnis für die komplexen Zusammenhänge in verteilten Systemen werden wir auch in Zukunft profitieren. Sei es, wenn es darum geht, neue Architekturen zu verstehen oder zu entwerfen, sei es im Bereich der Analyse und des Troubleshootings nicht nur der Applikationen, sondern in allen Softwareschichten bis zum Betriebssystem.

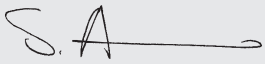
Koexistenz von CORBA und J2EE

Zurzeit laufen gegen 100 auf unserem ORB basierende Applikationen produktiv. Diese sind zu einem grossen Teil in C oder C++ geschrieben, in den letzten Jahren wurden aber auch vermehrt Services in Java realisiert. Schon in Anbetracht der Menge des produktiven Codes ist es illusorisch, eine

und auf die beste Art und Weise zu einem Ganzen zusammenzufügen. Dafür muss er in der Lage sein, die zur Verfügung stehenden Bausteine hinsichtlich ihrer Eignung für die anzustrebende Lösung beurteilen, richtig gewichten und die Reife neuer Technologien einschätzen zu können. Damit aus den Bauteilen und Architekturvorschlägen, die stetem Wandel unterliegen, brauchbare Lösungen entstehen, sind langjährige Erfahrung, umfassendes Fachwissen und das richtige Vorgehen nötig. Der Lösungsansatz darf nicht allein von den gerade die Diskussion beherrschenden technischen Trends diktiert werden.

Der Fortschritt der letzten Jahre hat es zudem möglich gemacht, dass nicht nur Lösungen aus einzelnen Komponenten zusammengefügt werden, sondern auch ursprünglich als eigenständige Applikationen konzipierte Lösungen selbst wieder zu Building Blocks werden können. Zum Beispiel, indem eine solche Anwendung als Web-Service in einer grösseren, umfassenden IT-Lösung eingesetzt wird. Damit haben Building Blocks eine neue Dimension erreicht, was wiederum neue Anforderungen an Entwickler und Software-Architekten stellt.

Stefan Arn



CEO AdNovum Informatik AG

Big-Bang-Migration aller Applikationen auf J2EE ins Auge zu fassen. Allerdings besteht immer wieder kurzfristig der Bedarf, einen Teil der Services aus neuen, auf J2EE basierenden Applikationen in bestehenden CORBA-Lösungen zu verwenden und umgekehrt.

In diesem höchst aktuellen Bereich von Integration und Migration kommt uns J2EE sehr entgegen, denn Teile von CORBA sind in der J2EE-Spezifikation enthalten.

> Schon in Anbetracht der Menge des produktiven Codes ist es illusorisch, eine Big-Bang-Migration aller Applikationen auf J2EE ins Auge zu fassen.

Tatsächlich geht Java schon seit langem sogar noch einen Schritt weiter, denn ein CORBA-konformer ORB ist Teil jedes ausgelieferten JDK. Die Idee dazu kam vor einigen Jahren in unserem Entwicklungsteam in Zürich auf, als man sich Gedanken über die Einbindung von aktiven, netzwerkfähigen Clients in einen Browser machte. Dank guten Kontakten zum damaligen Startup Netscape konnte sie von der AdNovum Software Inc. in Kalifornien eingespiesen werden.

In die Spezifikation von J2EE sind neben diesem Basis-API weitere Bestandteile von CORBA eingeflossen. So gibt es seit der ersten Spezifikation der Enterprise Java Beans ein Kapitel zum Thema Interoperabilität, in dem präzise beschrieben wird, wie ein EJB Interface auf die CORBA Interface Definition Language abzubilden ist. Entsprechende Compiler sind seit geraumer Zeit bei verschiedenen Herstellern verfügbar.

Auch auf der Protokollebene hat CORBA mit «RMI over IIOP» einen prominenten Platz in der Java-Welt erobert. Mit RMI-IIOP, einer Version der Remote Method Invocation für das Internet-Inter-ORB-Protokoll, kann die Interoperabilität von CORBA mit den leicht verständlichen Programmiereigenschaften von RMI kombiniert werden, da es die Kommunikation zwischen CORBA- und RMI-Komponenten erlaubt. Noch entscheidender dürfte aber die Verabschiedung von IIOPS mit CSIv2 (Common Security Interoperability, Version 2) als Standardprotokoll für die sichere Kommunikation zwischen CORBA-Klienten und Enterprise Java Beans sein. Auch wenn die Unterstützung dafür in den meisten Produkten noch zu wünschen übrig lässt, zeigt dies doch, dass Integration und Migration durch standardisierte Protokolle gefördert werden können.

> Im Bereich von Integration und Migration kommt uns J2EE sehr entgegen, denn Teile von CORBA sind in der J2EE-Spezifikation enthalten.

Die Integration eines bestehenden CORBA-Services in eine J2EE-Umgebung stellt zumindest theoretisch dank den unterstützten APIs und Protokollen keine Herausforderung dar. Der Entwickler schreibt dazu ganz normalen CORBA Client Code, und der J2EE Application Server übernimmt die Kommunikation.

Kritisch wird es erfahrungsgemäss immer dann, wenn eine Security-Integration erforderlich ist. Dies insbesondere, weil diesbezüglich die Unterstützung durch die auf dem Markt erhältlichen Produkte oft noch zu wünschen übrig lässt.

Ein bisschen anders sieht die Integration aus, wenn ein bestehender CORBA-Service auf EJB-Technologie migriert wird. Weil das Mapping von EJB Interfaces auf CORBA IDL nur unidirektional definiert ist, muss in diesem Fall praktisch immer der Code im Client angepasst werden. Ausnahmen wären Services, bei deren Interface-Definition bereits auf EJB Compliance geachtet wurde; allerdings ist dieses Szenario doch eher theoretischer Natur.

Neben der Integration von bereits produktiven Services stehen wir in der Praxis auch immer wieder vor dem Problem, neue, dazugekaufte Komponenten eines Drittherstellers in eine Applikation einbinden zu müssen. Handelt es sich dabei nicht um Java Code, steht man vor der Wahl, diesen mittels Java Native Interface oder irgendeiner anderen Form von Kommunikation einzubinden.

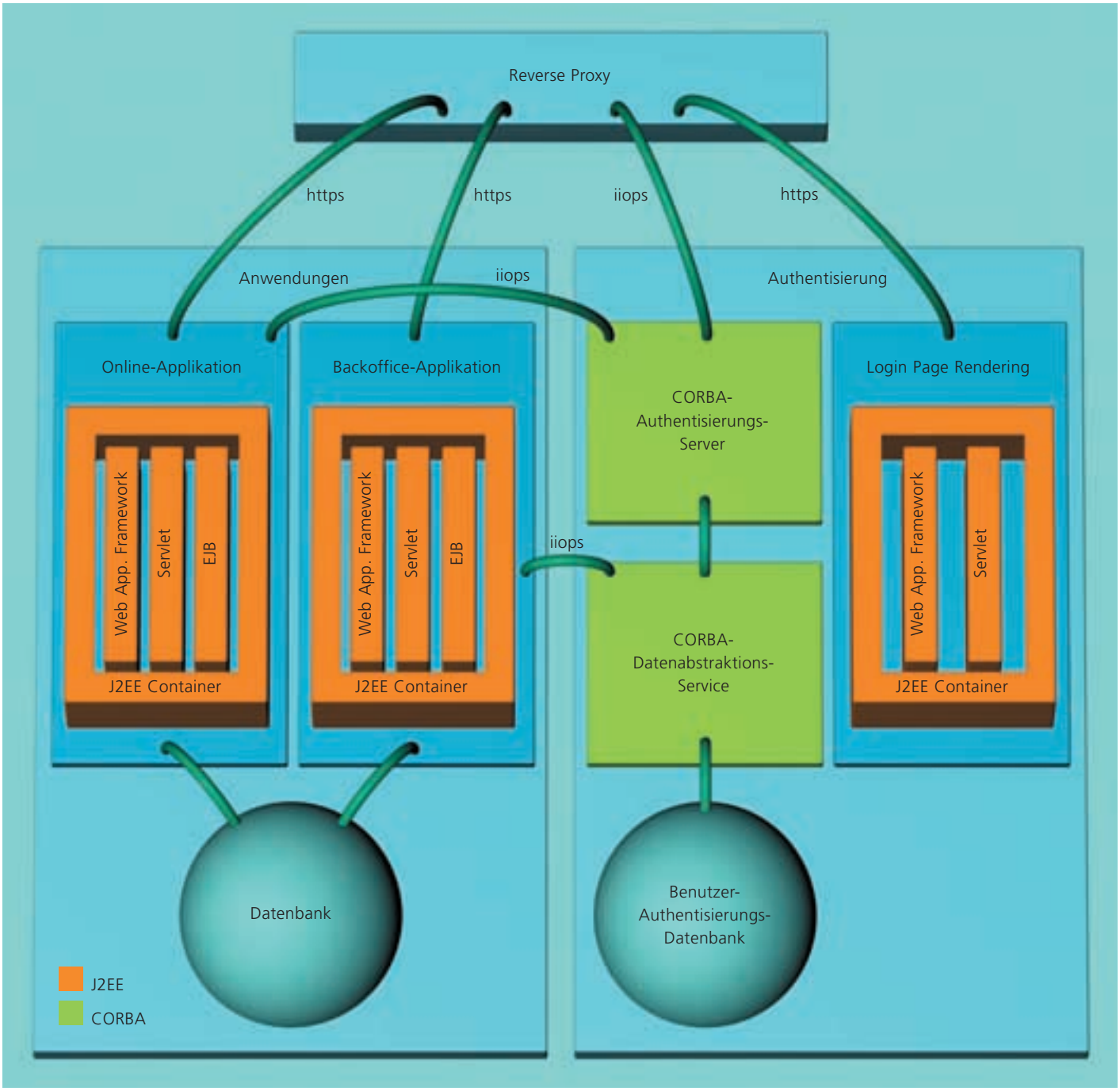
Mit JNI hat man in der Vergangenheit nicht nur gute Erfahrungen gemacht. Die im Vergleich zu Pure Java Calls massiv schlechtere Performance ist das eine; schlimmer ist jedoch, dass man damit eine nur bedingt vertrauenswürdige Komponente in den Java-VM-Prozess lädt. Programmierfehler in einer solchen Drittkomponente können sich fatal auf die Stabilität der Java Virtual Machine auswirken.

Die Analyse von Abstürzen, die auf solche Fehler zurückzuführen sind, bedürfen einer fundierten Kenntnis der Materie und sind in der Regel äusserst anspruchsvoll. Besonders herausfordernd stellt sich die Situation dar, wenn mehrere Fremdkomponenten über JNI eingebunden werden und man unter diesen nach derjenigen suchen muss, die den Fehler verursacht hat. In Anbetracht der Tatsache, dass auch heute noch ab und zu Abstürze von isolierten Java Virtual Machines zu beobachten sind, empfehlen wir, vor allem im applikatorischen Bereich Alternativen zur Einbindung über JNI zu evaluieren.

Eine solche Alternative stellt sicher die Bereitstellung einer Komponente als CORBA-Service dar. CORBA ist dank seinem Multi-Language-Support geradezu prädestiniert für solche Lösungsansätze. Der Fremdcode läuft damit isoliert in einem eigenen Prozess, und Fehler sind dank diesem Design viel einfacher zu lokalisieren. Bei Bedarf steht die Komponente über das Netzwerk auch anderen

Stefan Wengi

Stefan Wengi ist letzten Juni nach dreijähriger Tätigkeit als CTO der AdNovum Software Inc. im kalifornischen Silicon Valley nach Zürich zurückgekehrt und hat hier Matthias Loepfe als CTO der AdNovum Informatik AG abgelöst. In der zur Firmengruppe gehörenden AdNovum Software Inc. war der diplomierte Informatik-Ingenieur ETH für den Aufbau und Unterhalt der technischen Infrastruktur sowie für Software-Entwicklung vor allem im Middleware-Bereich verantwortlich. Seit seiner Rückkehr nach Zürich befasst er sich in erster Linie mit Architekturdefinitionen und Systemdesign im verteilten Umfeld.



Die Grafik zeigt eine mögliche J2EE-/CORBA-Integration am Beispiel einer sicheren Web-Applikation auf Basis der Nevis-Web-Architektur mit einem Reverse Proxy.

Applikationen zur Verfügung und kann besser skaliert werden. Als Nebeneffekt erhält man dank dieser stärker entkoppelten Service-Architektur eine gewisse Unabhängigkeit von Release-Änderungen einzelner Komponenten.

Betreibbarkeit als Schlüssel

Ein erfolgreiches Projekt ist ein produktiv eingeführtes Projekt. Das ist unsere Maxime, der wir prinzipiell alles andere unterordnen. Der Wert für unsere Kunden tendiert gegen Null, wenn wir zwar eine wunderschöne, akademische Lösung erstellen, diese aber in der Produktion nicht betreiben können. Dies bleibt sich gleich, unabhängig davon, ob eine Architektur nun auf CORBA oder J2EE basiert.

Für den Betrieb von J2EE-Applikationen sind fundierte Kenntnisse des verwendeten Containers unerlässlich. Das entsprechende Know-how muss sowohl bei den Entwicklern als auch bei den Release-Ingenieuren und den für den Betrieb verantwortlichen Personen vorhanden sein. Damit aber nicht genug: J2EE Application Server laufen immer in einer Java Virtual Machine, die selbst wiederum auf einem bestimmten Betriebssystem aufsetzt, das der Virtual Machine Ressourcen wie Memory, Network Connections oder ein File-System zur Verfügung stellt. Der Betrieb solcher Container ist nur dann erfolgreich, wenn man die darunter liegenden Softwareschichten im Griff hat. Die dafür nötigen Kenntnisse haben wir uns in Jahren intensiver Arbeit an Kundenprojekten angeeignet. Nicht zuletzt aus diesem Grund können wir in der Praxis immer wieder beweisen, dass wir auch im Betrieb und Troubleshooting von J2EE-basierten Anwendungen ein äusserst kompetenter Partner sind.

J2EE verlangt Experten

Tobias Murer über Chancen und Fallstricke beim Einsatz von J2EE-basierten Lösungen im produktiven Umfeld.

Interview: Thomas Schönfelder

NOTITIA: Die Software-Plattform J2EE scheint sehr en vogue zu sein. Was sind die Gründe dafür?

Tobias Murer: Aus technischer Sicht entscheidend sind sicherlich die Verwendung von Java als Basistechnologie und das einfache J2EE-Programmiermodell. J2EE bietet eine übersichtliche Kombination von standardisierten APIs (Application Programming Interfaces) und ergänzenden Plattformdiensten, die – zumindest auf den ersten Blick – die Komplexität von verteilten Systemen vor dem Anwendungsentwickler zu verstecken vermögen.

Daneben existiert eine ausgewogene Palette von J2EE-Standards, die einerseits die effiziente Realisierung von Applikationssystemen zulässt, andererseits aber auch den Wettbewerb unter den Plattformanbietern ermöglicht. Diese J2EE-Standards decken auch organisatorische Aspekte ab, was eher neu, aber auch äusserst relevant für serverseitige Plattformtechnologien ist. Dazu gehören das Komponenten- und «tier»-orientierte Programmiermodell, ausgewiesene Rollen mit fest umrissenen Aufgabenbereichen in J2EE-Projekten, aber auch betriebsrelevante Themen wie Deployment, Management und Monitoring.

> « Im Bereich der Plattformanbieter treten vermehrt Interoperabilitäts- und Integrationsaspekte in den Vordergrund. »

Beeindruckend ist vor allem das von der Industrie und der Open-Source-Gemeinde getriebene J2EE-Momentum mit seiner raschen Evolution und Adaption der offenen J2EE-Standards. Vereinzelte fixierte Standards, die sich als nicht ganz ausgereift erweisen, bilden die unvermeidbare Kehrseite der schnellen Entwicklung.

Welches sind die Schwerpunkte und Trends dieser J2EE-Evolution?

Einen Schwerpunkt bildet die Erweiterung der J2EE-Plattform um Web-Services. Sehr interessant und erfreulich ist aber auch, dass sowohl die Betreibbarkeit der J2EE-Plattform wie auch ergänzende Standards für J2EE-Plattformanbieter im Fokus stehen. Beides sind Hinweise auf den fortgeschrittenen Reifeprozess, den J2EE als standardisierte, sichere, verfügbare, erweiter- und betreibbare Plattform durchläuft.

Kritische Aspekte der Betreibbarkeit wie Deployment, Ressourcenverwaltung, Management und Monitoring von Plattformen und Anwendungen wurden im Java-Umfeld in der Vergangenheit zu stark vernachlässigt und typischerweise an das darunter liegende Betriebssystem delegiert. Erfreulicherweise sind jedoch sowohl im Bereich Java-Laufzeitumgebung (Java Virtual Machine) wie auch bei den J2EE-Standards verschiedene Anstrengungen erkennbar, um die erwähnten Mängel mittelfristig zu beheben. Bei einem kürzlichen Besuch der SunLabs (Sun Research Laboratories) in Palo Alto konnten wir uns zum Beispiel über ein interessantes Projekt zu optimiertem Ressourceneinsatz informieren. Es geht dabei um die Frage, wie sich Anwendungen in derselben Java-Laufzeitumgebung gegenseitig schützen lassen und wie die Komponenten der Laufzeitumgebung von verschiedenen Benutzern gemeinsam genutzt werden können.

Im Bereich der Plattformanbieter treten vermehrt Interoperabilitäts- und Integrationsaspekte in den Vordergrund. «Provider»-Konzepte werden festgelegt, um die Integration einer konkreten J2EE-Plattform in heterogene, kundenspezifische Umgebungen zu vereinfachen. Ein Beispiel dafür sind Providerstandards im Sicherheitsbereich, welche die standardmässige Integration von spezifischen Authentisierungs- und Autorisierungskomponenten ermöglichen.

Welche Bedeutung hat J2EE für Anwender im Kontext der gegenwärtigen wirtschaftlichen Rahmenbedingungen?

J2EE umfasst die notwendigen Technologien, um bestehende IT-Systeme investitionsschonend modernisieren und Betriebskosten

Tobias Murer

Tobias Murer, ETH-Software-Ingenieur, hat in der AdNovum die Aufgabe, das Know-how im Bereich Java Engineering und speziell J2EE voranzutreiben. Seit seiner früheren Arbeit als Post Doc im Bereich Software Engineering in Communities bei den Sun Research Laboratories in Palo Alto beschäftigt er sich intensiv mit Java und J2EE.

typischerweise im Finanzumfeld zum Einsatz kommen, ist und bleibt das Know-how der Beteiligten. In dieser Hinsicht hat sich auch mit J2EE nichts verändert, was meines Erachtens häufig unterschätzt wird. Für das Scheitern wird dann oft zu Unrecht die J2EE-Technologie verantwortlich gemacht.

Welches sind die kritischen Aspekte beim Einsatz von J2EE?

1. Fundamental ist zu wissen, wie eine verteilte Architektur und entsprechende Anwendungen realisiert werden.
2. Das J2EE-Programmiermodell und die entsprechenden APIs müssen den Standards gemäss korrekt verwendet werden. Den Verlockungen von Java muss widerstanden werden.
3. Eine optimale Integration und ein performanter Betrieb sind nur mit fundierten Kenntnissen des konkret eingesetzten J2EE-Plattformprodukts und des darunter liegenden Betriebssystems möglich.
4. Um früh Probleme identifizieren zu können, ist ein Vorgehen in geeigneten Iterationen unumgänglich.

Weshalb schätzen Sie das Know-how im Bereich verteilter Architekturen als entscheidend ein?

Es ist wichtig zu verstehen, dass sich die systembedingte Komplexität von verteilten Anwendungen trotz eines abgerundeten Angebots von APIs auch mit J2EE nicht eliminieren lässt. Die Probleme, die beim Erstellen, Integrieren und Betreiben einer sicheren, skalier- und verfügbaren verteilten Architektur und von entsprechenden Anwendungen auftreten, bleiben weiterhin bestehen, unabhängig davon, ob J2EE oder eine andere Technologie verwendet wird.

Wie alter Wein in neuen Schläuchen übernimmt und ergänzt J2EE viele etablierte Konzepte von alternativen Technologien. Die J2EE-Plattform ist dabei allerdings ein gelungener Evolutionsschritt im Bereich Plattformen für verteilte Architekturen. J2EE erlaubt den Anwendungsentwicklern, dank pragmatisch gewählter APIs und Diensten von verschiedenen Komplexitätsaspekten zu abstrahieren. Für die rasch auftauchenden anspruchsvolleren Probleme ist aber trotzdem das erwähnte, weitreichende und fundierte Know-how Match entscheidend.

> «Entwickler mit wenig Java-Know-how, aber Erfahrung im Bereich verteilter Architekturen haben mehr Potenzial, sich zu J2EE-Experten zu entwickeln, als solche mit umgekehrten Voraussetzungen.»

Entwickler mit wenig Java-Know-how, aber Erfahrung im Bereich verteilter Architekturen haben mehr Potenzial, sich zu J2EE-Experten zu entwickeln, als solche mit umgekehrten Voraussetzungen.

gezielt senken zu können. J2EE ist vor allem dank der Plattformunabhängigkeit und der verfügbaren Abstraktionskonzepte für die Weiterentwicklung bestehender, heterogener Systeme in kontrollierten, überblickbaren Iterationen sehr gut geeignet.

Aber auch durch Investitionen in ganz neue Projekte können rasch beachtliche Resultate erzielt werden, denn mit J2EE lässt sich mit beschränktem, aber optimalem Einsatz geeigneter Mittel in kurzer Zeit sehr viel erreichen.

Welches sind die Erfolgsfaktoren?

Absolut zentral für eine erfolgreiche Realisierung komplexer, verteilter IT-Systeme, wie sie

Sie sprachen von Versuchungen, denen es zu widerstehen gilt ...

Ja. Es ist wirklich nicht einfach, nicht in die Falle zu tappen: Dank Java und den vermeintlich einfachen J2EE APIs kann rasch und effizient Software geschrieben werden. Dies liegt vor allem auch daran, dass Java systembedingt gewisse schwierig zu identifizierende Fehler gar nicht zulässt. Symptomatisch ist daher die schon von mehreren Entwicklern geäusserte Aussage, dass sie normalerweise nie die Dienste eines Debuggers für die Fehlersuche in Anspruch nehmen müssen.

Eine Aussage, die bei der Verwendung anderer Technologien schwer denkbar ist.
Diese Vorteile von Java haben aber auch negative Auswirkungen. Mit mittelmässigem Know-how lässt sich schon in kurzer Zeit einiges realisieren. Das notwendige, fundierte und erfolgsentscheidende Wissen um die korrekte Verwendung des J2EE-Programmiermodells und die entsprechenden APIs fehlt leider gelegentlich, wird aber meist nicht als fehlend erkannt. Der Versuchung, das Projekt trotzdem ohne ergänzendes Fachwissen durchzuziehen, sollte widerstanden werden.

Eine weitere häufig beobachtete und problematische Folge der positiven Eigenschaften von Java ist die in unserer Gilde weit verbreitete Unart, dass sich jeder mit einem eigenen Framework verwirklichen und für die Ewigkeit erhalten möchte. Die durch die Verwendung von Java gewonnene Zeit scheint häufig damit verbracht zu werden, möglichst generische Mikroarchitekturen mit möglichst vielen Abstraktionen und Schichten zu realisieren. Danach werden die fehlende Performanz der realisierten Anwendung und die mangelnde Beherrschung der Komplexität oft der J2EE-Technologie angerechnet. Auch hier gilt es, der Versuchung zu widerstehen und einfache, schlanke Lösungen zu realisieren. Erst bei Bedarf sollten im Rahmen einer Iteration per Refactoring weitere Abstraktionen eingebracht werden.

Was wird mit einem iterativen Vorgehen bezweckt?

Bei J2EE-Projekten gilt es, früh mit Hilfe von entsprechenden Iterationen Unsicherheiten und damit Risiken zu minimieren.
Typischerweise liegen die Unsicherheiten weniger bei der Realisierung der Applikationslogik, da sich Probleme in diesem Bereich meist durch Fleissarbeit lösen lassen. Kritische Unsicherheiten liegen bei J2EE-Projekten vielmehr in den Bereichen Integration, Performanz, Sicherheit, Skalierbarkeit und, was vielfach unterschlagen wird, Betreibbarkeit.
Entsprechende Probleme lassen sich häufig, falls überhaupt, nur mit viel Aufwand beheben und bedrohen daher den Erfolg des Projektes. Wird iterativ vorgegangen und die Aufmerksamkeit dabei schon früh auf die oben erwähnten Bereiche gerichtet, können solche existenziellen Probleme schon in den ersten Iterationen angegangen werden.

Was bedeutet J2EE für die AdNovum?

Wir konnten uns in den letzten Jahren beim erfolgreichen Bau, der Integration und dem Betrieb von sicheren, verteilten Anwendungen und entsprechenden Infrastrukturkomponenten einiges an Erfahrung und Know-how erarbeiten. In jüngster Zeit sind die meisten unserer Projekte im J2EE-Umfeld angesiedelt. Der dadurch verfügbare breite und für die J2EE-Herausforderungen optimale Mix an Wissen ermöglicht uns, erfolgreich sowohl J2EE-Anwendungen wie auch Plattformkomponenten zu erstellen und integrieren.

> « Die Vorteile von Java können sich auch negativ auswirken. »

Im Moment sind wir zum Beispiel daran, den Single-Signon-Schutz von Nevis Web nahtlos über einen heterogenen Bereich von Anwendungsservern zu realisieren. Wir halten uns dabei an die aktuellen und antizipierten J2EE-Standards und ergänzen Aspekte, die durch diese Standards nicht abgedeckt werden.
Der Erfolg der bisher realisierten Projekte motiviert uns, weiter stark auf J2EE zu setzen und wie gewohnt unabhängig und an vorderster Technologiefront unser J2EE-Know-how zu pflegen und umzusetzen. ■

High-Grade Security im anspruchsvollen Finanzbereich

Die Sicherheit von IT-Lösungen ist heute insbesondere im Finanzdienstleistungssektor mehr denn je ein brennendes Thema, das allerdings vor allem in Fachkreisen wahrgenommen wird. Die Probleme und Lösungsansätze sind hinlänglich bekannt. Aus den vielen heute verfügbaren Bauteilen ein funktionierendes Gesamtsystem zu schaffen, ist aber nach wie vor eine Herausforderung.

Von Martin Vögeli

Da sich Sicherheitsvorkehrungen in einem modernen IT-System einerseits durch eine Zugriffsschicht etablieren lassen – im Internetbereich unter dem Stichwort «Reverse Proxy» und vermehrt auch «authentisierender Reverse Proxy» bekannt – und sich andererseits durch alle Schichten und Komponenten des dahinter liegenden Software-Systems ziehen, sind für integrative Aufgaben umfangreiches Know-how auf allen Ebenen und eine etablierte Infrastruktur (Middleware) nötig, um ein gut funktionierendes Ganzes zu erreichen.

Um dies praktisch zu verdeutlichen, sollen nachfolgend die vielfältigen Aspekte eines Kundenzugriffs aus Security-Sicht beschrieben werden, wie er in von der AdNovum realisierten IT-Systemen erfolgen kann.

Netzwerkzugriff: Ein Kunde greift über den Browser auf die E-Banking-Applikation seines Finanzinstituts zu. 128-Bit-Verschlüsselung ist dabei heute zwar eine Selbstverständlichkeit, muss bei den eingesetzten Komponenten allerdings korrekt konfiguriert werden. Aussergewöhnlich hingegen ist, dass bei Lösungen der AdNovum der Private Key des Server-Zertifikates das über PKCS#11 eingebundene «Hardware Security Module» (HSM) nicht verlässt, was verhindert, dass die Identität des Servers bei der Übertragung gestohlen werden könnte.

High-Grade Authentisierung: Das Finanzinstitut verlangt ein Login, um die Identität des Kunden zu ermitteln. Die Authentisierung erfolgt mittels eines Challenge-Response-Verfahrens unter Verwendung eines Security Device oder mit Hilfe von X509-User-Zertifikaten.

> Für integrative Aufgaben braucht es umfangreiches Know-how auf allen Ebenen und eine etablierte Infrastruktur.

Sessionzuordnung und Autorisierung: Nach dem Login wird der Zugriff auf den J2EE Application Container erlaubt. Benutzeridentität und globale Sessioninformationen werden signiert über verschlüsselte Netzwerkverbindungen geschickt und vom Empfänger geprüft. Bei positiver Validierung wird der Benutzer grundsätzlich für den Zugriff auf die Applikation autorisiert. Da die Benutzeridentität zusätzlich auch über die standardisierten J2EE-Schnittstellen zur Verfügung steht, ist gewährleistet, dass der Benutzer auf seine Daten zugreifen kann. Die Sessionzuordnung wird nach Möglichkeit ohne Cookies und URL Rewriting realisiert.

Schreibende Transaktionen: Der Benutzer ändert Daten über die Applikation. Hier kommen Autorisierungsfunktionen, die Verschlüsselung sensibler Daten in der Datenbank und Auditing zum Zug.

Single Signon: Der Benutzer verwendet zum Beispiel die in die Benutzeroberfläche integrierte Suche. Diese Anwendung läuft zwar auf einem anderen Applikationsserver, per Single Signon ist der Kunde aber bei dieser Applikation automatisch authentisiert.

Content-Kontrolle: Der Benutzer greift innerhalb der sicheren Applikation auf einen unsicheren Link zu. Durch Content Rewriting im Reverse Proxy wird der Link umgeschrieben und über den Reverse Proxy bezogen. Dieser kann den Inhalt auf Scripts und Code hin

Zertifikate und Security Devices

Ein Zertifikat kann als digitales Äquivalent zu einem Pass angesehen werden. Mit Zertifikaten kann über das Netzwerk die Identität von Individuen oder Unternehmen ohne die Eingabe von Benutzernamen und Passwörtern verifiziert werden. X509 ist ein die Syntax und Semantik von Zertifikaten definierender Standard.

Beim Public-Key-Verfahren besitzt jeder Teilnehmer ein Schlüsselpaar, einen privaten und einen öffentlichen Schlüssel. Eine Meldung wird mit dem öffentlichen Schlüssel des Empfängers chiffriert und kann nur mit dem privaten Schlüssel des Empfängers dechiffriert werden. Da nur öffentliche Schlüssel übertragen werden, ist kein geheimer Kanal für die Übertragung nötig.

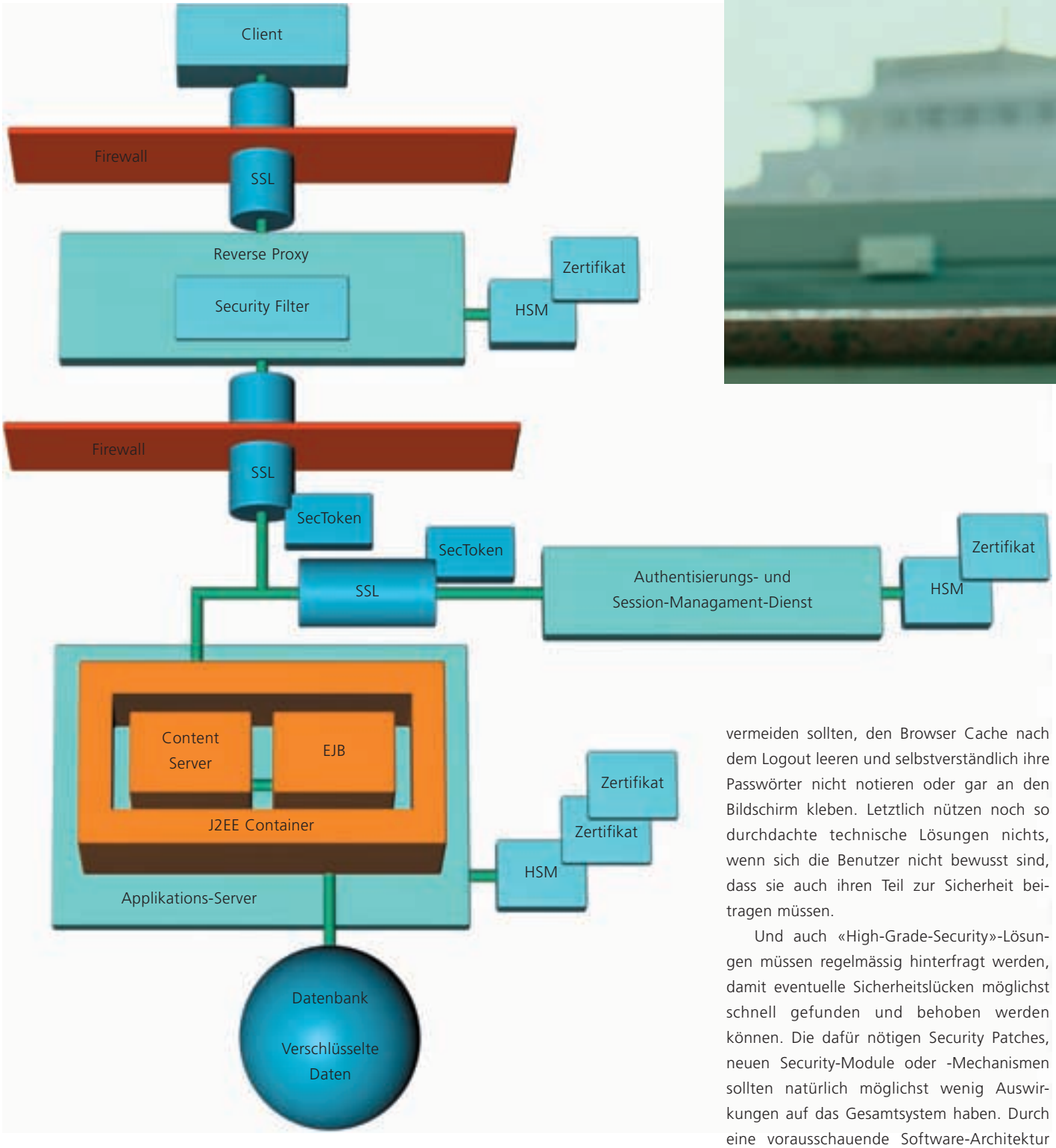
Eine Public-Key-Infrastruktur (PKI) umfasst weitere Komponenten, die sicherstellen, dass der Besitzer eines öffentlichen Schlüssels richtig authentisiert wird. Dazu gehört beispielsweise eine Zertifizierungsinstanz, die die digitalen Zertifikate generiert, eine Registrierungsinstanz und ein Zertifikate-Server, auf dem die Zertifikate verwaltet werden und für PKI-Anwender abrufbar sind.

Im Gegensatz dazu bedingen Challenge-Response-Verfahren mit Security Devices eine Interaktion des Benutzers. Dabei gibt der Authentisierungsserver einen zufällig generierten Code (Challenge) aus, auf den der Benutzer mit einem passenden Gegenstück antworten muss (Response). Dieses Gegenstück wird durch ein Security Device aufgrund der Challenge errechnet.

Da jede Response nur für einen Zugriff gilt – eventuell auch nur für eine begrenzte Zeitspanne – und der Benutzer sowohl das Security Device als auch ein Passwort braucht, um es zu aktivieren, ist dieses Verfahren Lösungen mit herkömmlichen Passwörtern überlegen.

überprüfen und gegebenenfalls den Zugriff verweigern. Cross-Window Scripting, das heisst, ein Abhören des Inhalts oder gar eine Manipulation durch andere Browserfenster, zumindest durch in der eigenen Applikation verlinkte Seiten, kann so vermieden werden.

Unsichere Links haben natürlich nichts in einer hochsicheren Applikation zu suchen. Die Problematik taucht aber beispielsweise im Single-Signon-Bereich auf, wenn mit Cross-Selling gearbeitet wird, und kann nur durch Überwachung des applikatorischen Content abgesichert werden. Ausserdem entbinden technische Möglichkeiten nicht davon, die Benutzer für die Schwachstellen von Internet-Applikationen zu sensibilisieren. Dazu gehört beispielsweise, dass Kunden das Surfen auf anderen Sites während E-Banking-Sessions



Beispiel einer gesicherten Applikations-Server-Infrastruktur

vermeiden sollten, den Browser Cache nach dem Logout leeren und selbstverständlich ihre Passwörter nicht notieren oder gar an den Bildschirm kleben. Letztlich nützen noch so durchdachte technische Lösungen nichts, wenn sich die Benutzer nicht bewusst sind, dass sie auch ihren Teil zur Sicherheit beitragen müssen.

Und auch «High-Grade-Security»-Lösungen müssen regelmässig hinterfragt werden, damit eventuelle Sicherheitslücken möglichst schnell gefunden und behoben werden können. Die dafür nötigen Security Patches, neuen Security-Module oder -Mechanismen sollten natürlich möglichst wenig Auswirkungen auf das Gesamtsystem haben. Durch eine vorausschauende Software-Architektur lassen sich solche Auswirkungen weitgehend

Martin Vögeli

Martin Vögeli ist als praxisgeschädigter AdNovum-Dinosaurier mit allen Aspekten von verteilten Systemen vertraut. Nach dem ETH-Informatikstudium begann er bei der AdNovum mit Systemmanagement-Projekten und arbeitete sich über CORBA und Web Middleware zu kundenspezifischen Applikationen mit Fokus auf Integration durch. Auf Grund seines breiten technischen Wissens befasst sich Martin Vögeli heute mit Software-Architektur- und Security-Fragen sowie der Weiterentwicklung der Nevis-Plattform.

isolieren, beispielsweise durch die Etablierung einer authentisierenden Zugriffsschicht wie in Single-Signon-Lösungen. Aus Security-Sicht steht dabei nicht der (unquantifizierbare) Kundennutzen von Single-Singon-Mechanismen im Vordergrund, sondern die isolierte, spezialisierte und flexible Handhabung von Sicherheitsmechanismen.

Software entwickeln in der Schweiz?

Bausteine für den erfolgreichen Aufbau von Software-Unternehmen: Der Case AdNovum und das Schweizer Software-Unternehmertum.

Von Simon Grand und Matthäus Urwyler
Research Center RISE, Universität St. Gallen HSG

Im Gespräch mit amerikanischen Wissenschaftlern, Unternehmern und Managern hört man immer wieder die erstaunte Frage: Gibt es in der Schweiz wirklich erfolgreiche und international wettbewerbsfähige Software-Unternehmen? Es gibt sie, und sie eröffnen spannende Einblicke in die Fragestellung, wie in der Schweiz und allgemein qualitätsorientierte, wissensintensive High-Tech-Unternehmen erfolgreich aufgebaut werden können.

Basierend auf einer Case Study zur AdNovum, aus dem Vergleich mit Case Studies zu Ergon und Netcetera und vor dem Hintergrund der wissenschaftlichen Forschung zu High-Tech-Unternehmertum, technologischer Innovation und strategischem Management identifizieren wir Bausteine für den erfolgreichen Aufbau von Software-Unternehmen. Die hergeleiteten Bausteine erstaunen vielleicht, weil sie auf den ersten Blick dem Common Sense zum Thema Unternehmertum nicht unbedingt entsprechen, aber sie werden den Herausforderungen von Unternehmen gerecht, die Software-Entwicklung als einen wissensintensiven Engineering-Prozess verstehen, der auf nachhaltige Wertschöpfung für den Kunden ausgerichtet ist und zugleich permanent durch wissenschaftlich-technologische Innovationen verändert und herausgefordert wird.

> Ein klarer Fokus ist wesentlich, damit die beschränkten finanziellen, personellen und zeitlichen Ressourcen optimal eingesetzt werden können.

Baustein 1: Fokus und Flexibilität durch technologische Fähigkeiten

Eine zentrale Herausforderung für Unternehmen, die in einem innovativen Umfeld tätig sind, ist die Etablierung und Festigung eines klaren Geschäftsfokus. Ein klarer Fokus ist wesentlich, um die beschränkten finanziellen, personellen und zeitlichen Ressourcen optimal einsetzen und die neu entstehenden geschäftlichen und technologischen Möglichkeiten in ihrer Relevanz beurteilen zu können. In den Worten von Stefan Arn: «Wir haben ein klares Verständnis davon, was unser Core Business ist und was nicht.» Dies bedeutet eine Konzentration auf Aufgaben, die sich aus den eigenen Fähigkeiten ergeben; eine zu starke Orientierung an externen technologischen und geschäftlichen Möglichkeiten kann zu einer Verwischung des Fokus führen. Es geht darum, die bestehenden Fähigkeiten angesichts neuer Trends und Möglichkeiten nicht ständig neu zu definieren, sondern systematisch zu erweitern und zu vertiefen.

Baustein 2: Wissensaufbau durch Lernen aus Erfahrung

Das konzeptionelle und praktische Wissen im Bereich Software Engineering ist für Unternehmen wie die AdNovum die zentrale strategische Ressource. Dieses Wissen etabliert und erneuert sich nicht von selbst, sondern ist das Ergebnis einer systematischen Wissensentwicklung, die als Kernprozess eines erfolgreichen Software-Unternehmens gesehen werden kann. Das bedeutet, dass wichtige Erfahrungen permanent in Wissen übersetzt, entwickelt und gefestigt werden müssen: «... wir haben diverse Sachen gemacht, bei denen wir lernen mussten, dass es doch nicht unser Core Business ist. Es hat zwar Änderungen gegeben, wir haben aber nie versucht, noch etwas Immobilien zu kaufen oder noch etwas Hardware zu verkaufen.»

Baustein 3: Wissensentwicklung durch Kommunikation und Innovation

Erfahrungen stehen dem Unternehmen und den Projekten nur dann zur Verfügung, wenn sie systematisch in organisatorisches Wissen übersetzt werden: Das bedeutet, dass das etablierte Wissen im Unternehmen permanent kommuniziert und begründet werden muss, damit es seine Gültigkeit im Tagesgeschäft behaupten kann.

Baustein 4: Ressourcenbeschaffung und Ressourcenallokation

Die laufende Entwicklung von neuem organisatorischem Wissen und die Überprüfung von bestehendem Wissen setzen zeitliche, finanzielle und personelle Ressourcen voraus. Es ist eine entscheidende Erkenntnis des strategischen Managements, dass die Quellen dieser Ressourcen einen Einfluss auf die Strategie des Unternehmens haben: die

> Erfahrung muss systematisch in Wissen übersetzt und permanent entwickelt und gefestigt werden.

Mitarbeiter, die bereits lange im Unternehmen arbeiten, spielen als Wissensträger und Wissenseinbringer eine wichtige Rolle, damit der Software-Entwicklungsprozess in allen Projekten einen möglichst hohen Qualitätsstandard erreicht. Diese Aufgabe übernimmt in der AdNovum zu einem bedeutenden Teil eine Gruppe von Senior-Entwicklern, «... die von einem Projekt ins nächste springen und versuchen, ihre Spuren zu hinterlassen. Nicht, indem sie einfach stillschweigend das Problem schnell lösen und wieder gehen, sondern indem sie dazu beitragen, dass die Leute dabei auch etwas lernen.»

Investoren, weil sie die wesentlichen finanziellen Ressourcen zur Verfügung stellen, die zentralen Kunden, mit denen ein Software-Unternehmen Lösungen realisiert, und die Technologiepartner, die massgeblich an der Ausrichtung des Unternehmens beteiligt sind. Für AdNovum, Ergon und Netcetera war es dabei immer klar, dass eine fundierte technologische Ausrichtung mit der Abhängigkeit von Investmentzielen externer Geldgeber nicht unbedingt vereinbar ist. Daher waren sie

> Anspruchsvolle Projekte haben die Weiterentwicklung des bestehenden Wissens wesentlich gefördert.

in den Boomjahren Venture Capital und Initial Public Offering und insgesamt Fremdfinanzierungsmöglichkeiten gegenüber kritisch eingestellt. Dagegen haben zentrale Kunden durch die Vergabe von anspruchsvollen Projekten wesentlich dazu beigetragen, dass das bestehende Wissen und die bestehenden Kompetenzen weiterentwickelt wurden.

Baustein 5: Strategien der Legitimation und Umwertung

Die grosse Herausforderung für jedes Software-Unternehmen besteht darin, dass die Qualität der Software per se für den Kunden nicht sichtbar ist, sondern nur durch ihr Funktionieren fassbar wird. Damit bekommt sie in der Wahrnehmung des Kunden nicht immer den Wert, den sie in der Optik des Software-Entwicklers verdient. Dazu kommt, dass Software-Entwicklung ex ante durch eine hohe Unsicherheit gekennzeichnet ist: «Software schreiben ist a priori nur glauben». Das heisst: «Wenn es darum geht, etwas Neues zu machen, dann geht es primär darum, ob der Kunde uns glaubt oder glauben kann, dass wir es können».

Schlussfolgerungen: Gemeinsamkeiten und Einzigartigkeit

Die erfolgreichen unternehmerischen Strategien von AdNovum, Ergon und Netcetera zeigen eine erstaunlich grosse Übereinstimmung in der technologischen Ausrichtung, aber auch bezüglich der zentralen Prozesse des Wissensaufbaus, der Wissensentwicklung und der Grundhaltung bezüglich der Ressourcenbeschaffung und -allokation.

Andererseits finden wir aber auch, dass jedes Unternehmen seinen spezifischen Weg finden muss, diese Bausteine zu etablieren. Eigentumsstrukturen, Entscheidungsprozesse und das Wachstum werden in allen drei Firmen unterschiedlich organisiert. Entscheidender ist, dass die Antworten auf die technologischen und unternehmerischen Herausforderungen konsistent sind und alle Bausteine für den Erfolg abbilden.

Das Research Center for Innovation, Strategy and Entrepreneurship (RISE)

Das Research Center for Innovation, Strategy and Entrepreneurship (RISE) an der Universität St. Gallen HSG untersucht in enger Zusammenarbeit mit führenden Konzernen und Unternehmern sowie internationalen Forschungsinstitutionen wie ETH Zürich, MIT und Wharton, wie und unter welchen Bedingungen Forschung und Innovation in High-Tech-Industrien zu erfolgreichen neuen Unternehmen und Business-Aktivitäten führen.
Nähere Informationen finden Sie unter: www.riseonline.net.

Impressum

Herausgeber:
AdNovum Informatik AG
Corporate Marketing
Röntgenstrasse 22
CH-8005 Zürich
Telefon 01 272 61 11
Telefax 01 272 63 12
E-Mail info@adnovum.ch
www.adnovum.ch
Verantwortlich für diese Ausgabe:
Thomas Schönfelder, Head of Services
Redaktion:
Barbara Stammler
Dorina Mayrhofer
Gestaltung und Realisation:
Rüegg Werbung, Zürich
Fotografie:
Matthias Auer, Zürich