NOTITIA

AdNovum

Bemerkenswertes von und über AdNovum

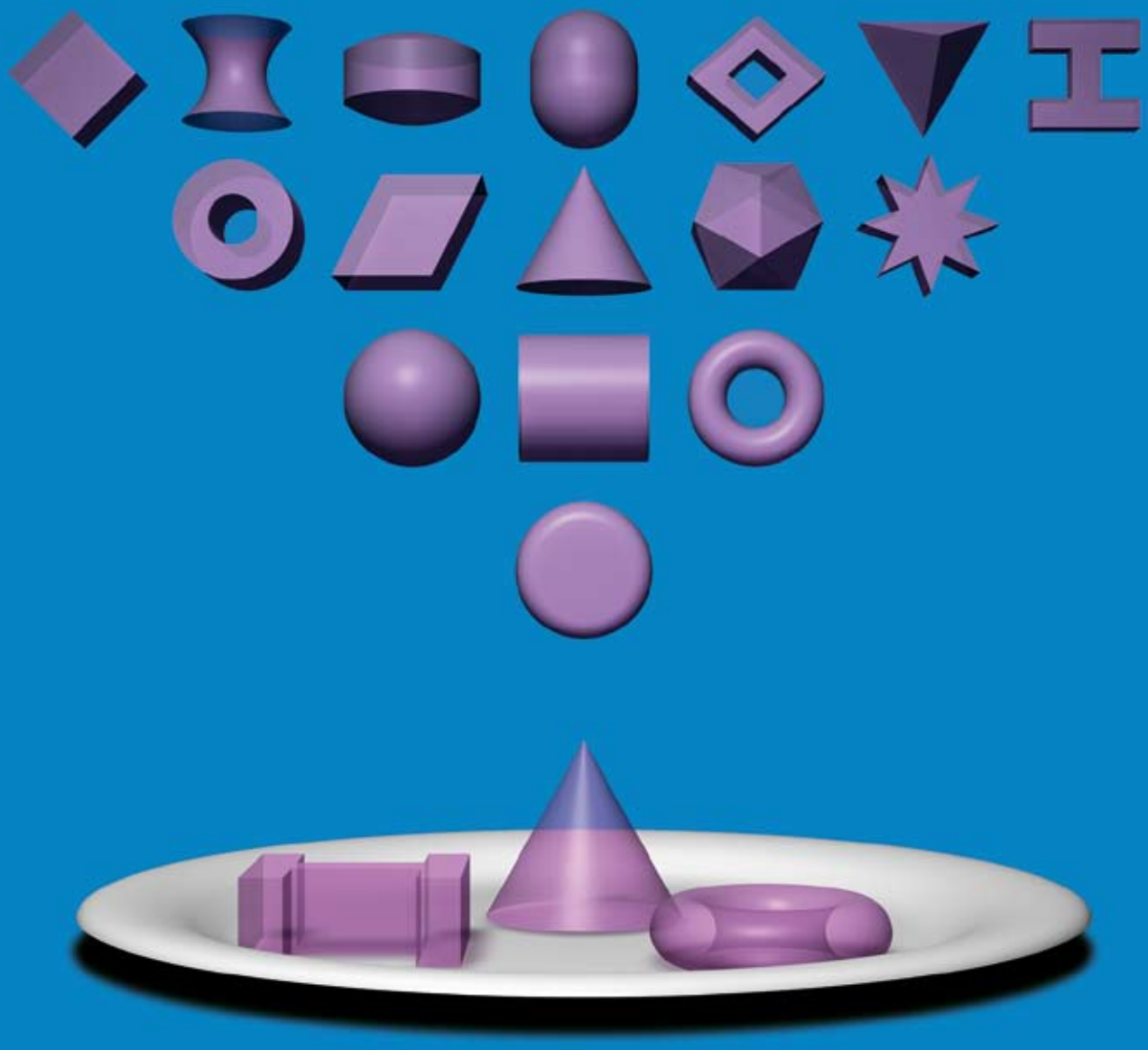
Individuell sourcen
Erfolgreich dank Schnittstellenmanagement

Verteilt entwickeln
Zusammenarbeit über Distanz meistern

Der Integrator
Schlüsselfigur zwischen Entwicklung und produktiver Einführung

Frühling 2007, Nr. 12

Sourcing à la carte




Gratian Anda


Ruedi Wipf

Individuell sourcen

Grössere IT-Projekte werden heute zunehmend auch auf feingranularer Ebene aufgeteilt und verteilt über weit entfernte Standorte entwickelt, mit entsprechend hohen Anforderungen an die Realisierung. Erfolgsentscheidend ist dabei das organisatorisch und technisch höchst anspruchsvolle Schnittstellenmanagement.

Von Martin Nokes

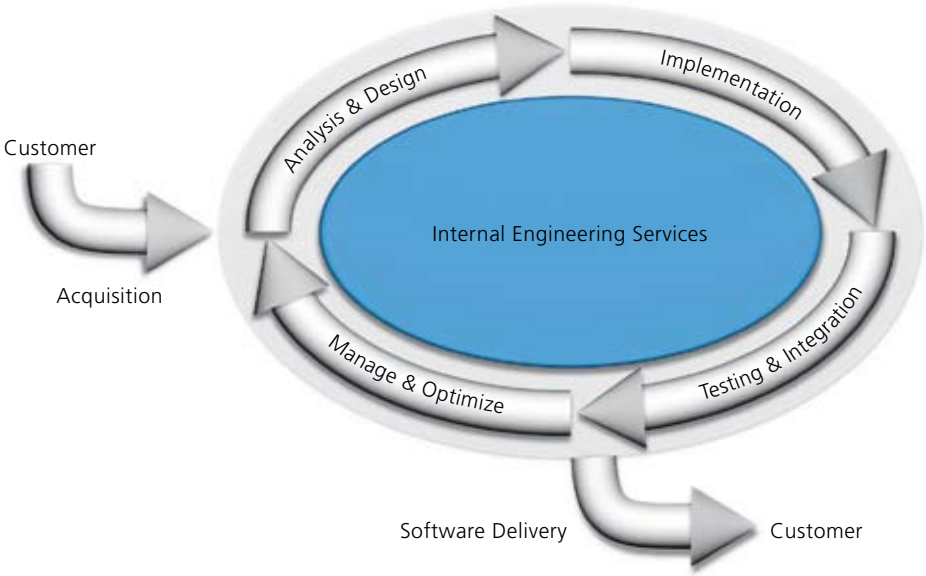
Obwohl seit über 50 Jahren Erfahrung in der Softwareentwicklung gesammelt wird, bleiben Informatikprojekte berüchtigt für Budgetüberschreitungen, Terminverzögerungen sowie Qualitätsprobleme bei der Einführung der Software – falls das Projekt nicht bereits vorher abgebrochen wird. Dazu kommt nun die globalisierte Arbeitsteilung mit den daraus resultierenden Komplexitätssteigerungen: Sourcing findet heute nicht mehr nur auf der Ebene von Branchen, Unternehmen oder Projekten statt, sondern direkt auf der Ebene einzelner Aufgaben oder Tranchen. Bei diesem «Sourcing à la carte» erbringen sowohl Auftraggeber wie externer Dienstleister ihre Projektleitungs- und Entwicklungsarbeiten in Teams an global verteilten Standorten. Die Anforderungen an die Projektrealisierung haben sich damit beträchtlich erhöht. Sie lassen sich jedoch meistern, wenn gewisse Punkte beim organisatorischen und technischen Schnittstellenmanagement rechtzeitig angesprochen und mit der notwendigen Aufmerksamkeit behandelt werden.

Organisatorische Schnittstellen

Welche Punkte speziell beachtet werden müssen und welche organisatorischen Schnittstellen sich ergeben, hängt vor allem davon ab, in welchen Projektphasen ein externer Sourcingpartner involviert ist und welche Aufgaben ihm übertragen werden, d. h. welche er übernimmt resp. übernehmen kann und will.

Die im Rahmen einer End-2-End-Umsetzung auftretenden Aufgaben und Aspekte lassen sich anhand des AdNovum-Software-Engineering-Prozesses illustrieren (s. Grafik).

Dieser umfasst, beginnend bei der Businessanalyse und technischen Spezifikation über die Implementation, Quality Assurance und Dokumentation, alle Phasen eines Softwareprojekts. Je nach Stadium werden Mitarbeiter mit verschiedenen Profilen benötigt: Business-analysten schreiben die Anforderungsspezifikation, Softwareingenieure implementieren



Software-Engineering-Prozess: Sourcingpotenzial in jeder Phase einer End-2-End-Umsetzung.

Liebe Leserin, lieber Leser

Wir begrüssen Sie im Namen des Verwaltungsrates und der neu bestimmten Geschäftsleitung von AdNovum. Bei unseren Kunden, Stefan Arn, der Führungscrew und allen Mitarbeitern möchten wir uns für die gute Zusammenarbeit und das Vertrauen in der Übergangsphase bedanken. Wir freuen uns auf die neuen Herausforderungen, und es ist unser ausdrückliches Bestreben, das Engagement und die Geschäftsbeziehungen der AdNovum in der angestammten Qualität mit hoher Kontinuität weiterzuführen, zu vertiefen und auszubauen.

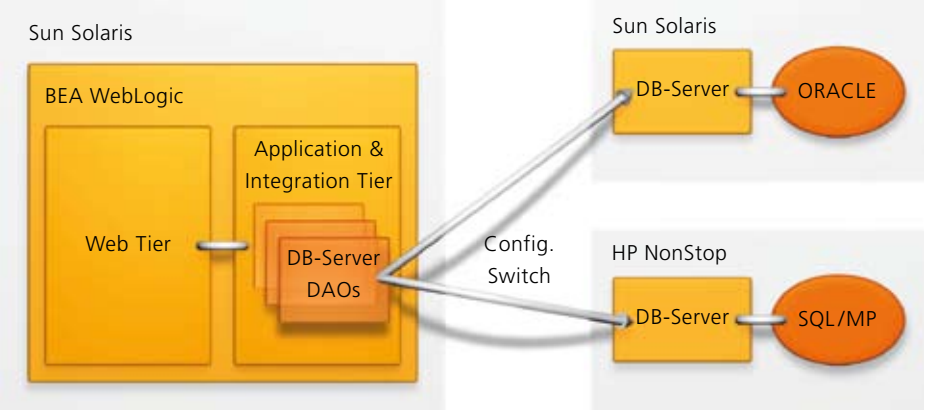
Die vorliegende Notitia befasst sich mit verteilter Applikationsentwicklung. Im einleitenden Artikel erläutert Martin Nokes, was es beim Sourcing von Informatikprojekten zu beachten gilt. Über seine Erfahrungen bei der multilokalen Entwicklung einer umfangreichen und komplexen Applikation berichtet uns im Interview Christian Siffert-Grob. Mit der zentralen Figur des Integrators befassen sich Andreas Leimbacher und Christian Widmer im Hintergrundartikel. Als Gastautor begrüssen wir Reto Brack von Sun Microsystems (Schweiz) mit seinem Beitrag über Solaris 10. Eine gute Lektüre wünschen Ihnen:

Gratian Anda
Verwaltungsratspräsident
der AdNovum Firmengruppe

Ruedi Wipf
Designierter CEO AdNovum Informatik AG

sie und Testingenieure übernehmen die Qualitätssicherung.

Die AdNovum als Sourcingpartner deckt mit ihrem Leistungsangebot sämtliche Aspekte einer End-2-End-Umsetzung ab. Bei grossen, verteilten und entsprechend komplexen Softwaresystemen, mit denen sich AdNovum hauptsächlich befasst, bestehen jedoch üblicherweise Schnittstellen zu Komponenten, die bereits existieren oder sich andernorts noch in Entwicklung befinden – beim Kunden oder bei einer weiteren involvierten Partei. Beispiele hierfür sind Datenbanken oder auf Mainframes betriebene Services.



Flexible Einbindung verschiedener Datenbank-Backends.

Technische Schnittstellen

Damit die entwickelten Komponenten später einwandfrei mit bestehenden produktiven Komponenten und Systemen kommunizieren, müssen die entsprechenden Schnittstellen auch innerhalb der Entwicklungs- und Testumgebung zur Verfügung stehen, bevor mit der Implementierung begonnen werden kann. Hierzu müssen die betreffenden Schnittstellen identifiziert und deren Ausprägungen analysiert werden, woraus sich verschiedene Optionen für deren Bereitstellung ergeben.

Ein netzwerkfähiger Service zum Beispiel kann über eine gesicherte Internet-Verbindung vom Kunden bezogen werden. Und ein simpler Backend-Service auf gängiger Hardware ohne weitere Abhängigkeiten lässt sich typischerweise auch ausserhalb der gewohnten Umgebung, also auch bei einem externen Sourcingpartner, problemlos installieren und betreiben.

> Grosse, verteilte Systeme enthalten meist Komponenten, die bereits existieren oder andernorts in Entwicklung sind.

Andere Ansätze sind erforderlich, wenn vertrauliche Daten oder Prozesse involviert sind oder wenn ein Service nicht mit vernünftigem Aufwand ausserhalb der Umgebung des Kunden repliziert werden kann. Hier sind Mock-ups (Simulationen) sowie Abstraktion und Ersatz gefragt. In einem Projekt, in dem Antworten einer Applikation aus deren Screen-Output gelesen und geparst werden mussten («Screen Scraping»), verarbeitete die AdNovum intern reguläre Dateien, in denen exemplarische Screens des Mainframes abgelegt waren, statt die Daten vom Netzwerk zu lesen. In diesem Fall machte es für die vorgelagerten Komponenten keinen Unterschied, ob die Daten aus einer internen Datei oder aus dem Netzwerk stammten.

In einem anderen Projekt mit einer HP-NonStop-Umgebung als Datenbank-Backend kaufte die AdNovum einerseits eigens ein solches System ein, um die Schnittstellen zur Verfügung zu haben, andererseits gestaltete sie den Integration Tier im Hinblick auf maximale Flexibilität so, dass auch andere Datenbanksysteme als Backend eingesetzt werden können (siehe Grafik).

Diese Lösung hatte zwei Vorteile: Ein Grossteil der Entwicklung konnte gegen eine

wesentlich kostengünstigere, auf einer Sun Solaris betriebenen Oracle-Datenbank durchgeführt werden. Dadurch musste das Non-Stop-System nicht die gesamte während der Entwicklung anfallende Last tragen, und es konnte ein weniger teures Modell angeschafft werden. Gleichzeitig vereinfacht die neue Architektur eine allfällige zukünftige Migration auf ein anderes Datenbanksystem. Auch hier spielt es für die übergelagerten Schichten resp. Tiers keine Rolle, welche Art von Datenbank im konkreten Fall eingesetzt wird.

Ist die Entwicklungs- resp. Testumgebung einmal in Betrieb, darf nicht vergessen werden, dass die von aussen bezogenen Komponenten laufend weiterentwickelt werden. Will man nicht Gefahr laufen, mit veralteten Versionen solcher Komponenten zu arbeiten, empfiehlt es sich, einen Prozess für deren Aktualisierung zu definieren. Dies können regelmässige Lieferungen der neuen Versionen oder Upgrades entlang eines Release Schedule sein. Zu beachten ist des Weiteren, dass alle Komponenten klar und eindeutig versioniert werden müssen.

Kundenstandards

Insbesondere grössere Unternehmungen mit eigenen IT-Abteilungen definieren häufig eigene Standards und Richtlinien, die auch für extern entwickelte Software gelten. Diese werden bei AdNovum nicht als Konkurrenz zu den eigenen entsprechenden Vorgaben verstanden, sondern als willkommene Ergänzung.

> Die Build- und Entwicklungsumgebung des Kunden sieht häufig ganz anders aus als jene der entwickelnden Firma.

Die Einhaltung der Standards und Richtlinien des Kunden stellt per se keine grössere Herausforderung dar. Wichtig ist es jedoch, vor der Implementationsphase abzuklären, ob solche Vorgaben existieren. Falls ja, sind die entsprechenden Informationen zusammenzutragen und innerhalb des Entwicklungsteams klar zu kommunizieren. Damit wird verhindert, dass eine vermeintlich fertige Applikation bei der Verifikation durch den Kunden durchfällt, weil beispielsweise eine unerlaubte Library eingesetzt wird.

Falls die Software nach Projektabschluss nicht von der entwickelnden Firma, sondern von der Informatikabteilung des Kunden oder einer Drittfirma zu warten ist, ist auch daran zu denken, dass die Build- und Entwicklungsumgebung des Kunden häufig ganz anders aussieht als jene der entwickelnden Firma. In der AdNovum wird diesem Umstand dadurch Rechnung getragen, dass alle Software der AdNovum ohne den Einsatz proprietärer Tools und Entwicklungsumgebungen wie zum Beispiel Eclipse gebaut werden kann. So besteht die Gewähr, dass sie auch in der Entwicklungsumgebung des Kunden bearbeitet werden kann. Ausserdem sind sämtliche externen Abhängigkeiten (typischerweise Pfade und Dateinamen) in Softwarekomponenten sauber konfigurativ festgelegt, was eine Umstellung wesentlich erleichtert.

Tests durch Kunden: möglichst früh

Je früher der Kunde «seine» Applikation testen kann, desto glücklicher ist er: Auch eine noch so perfekte Spezifikation ist abstrakter Natur, und häufig lassen sich insbesondere kleinere Detailfragen am besten «live» an der Applikation klären. Auch für den Partner, der die Software entwickelt, bringt dies Vorteile. Allfällige Missverständnisse können früh erkannt und korrigiert und Neuanforderungen, Ergänzungen und Präzisierungen des Auftraggebers rasch behandelt werden. Das Testing durch den Kunden muss bei der Projektplanung berücksichtigt und mit dem Kunden besprochen werden.

Innerhalb des iterativen Vorgehensmodells der AdNovum ist es oft sehr einfach möglich, weit vor dem eigentlichen Abschluss der Entwicklung Teile der Applikation zu testen. Dazu können Vorabversionen beim Kunden installiert oder auf dem System der AdNovum zur Verfügung gestellt werden. In letzterem Fall bietet AdNovum ihren Kunden die Option, die Applikation über einen gesicherten Internet-Zugang vom eigenen Standort aus zu testen. Damit kann der Entwicklungs-, Verifikations- und Change-Management-Prozess schneller und flexibler gestaltet werden .

> Eine Formalisierung des Prozesses im Change Management bedeutet nicht, dass Flexibilität verloren geht.

Integration beim Kunden

Ein wesentlicher, integraler Bestandteil des AdNovum-Projektmodells ist die Integration der Software in die Umgebung und die Systeme des Kunden. Nicht bei der Lieferung der Software, sondern erst bei deren erfolgreicher Einführung in der produktiven Umgebung des Kunden ist ein Projekt abgeschlossen.

Der Bedeutung dieser Projektphase wird unter anderem dadurch Rechnung getragen, dass in jedem Projekt ein AdNovum-Mitarbeiter die Rolle des Integrators übernimmt. Der Integrator zeichnet für das Zusammensetzen und die Einbettung der verschiedenen Komponenten in das Kundensystem verantwortlich (siehe Artikel «Der Integrator» auf Seite 9).

Change Management

Unabhängig davon, wie viele Phasen resp. Arbeiten von einer Entwicklerfirma übernommen werden, ist eine effiziente, simple und konsolidierte Verwaltung von Neuanforderungen und Bug Reports von zentraler Bedeutung. Je näher sich Auftraggeber und Entwickler sind, desto eher wird erfahrungsgemäss auf

Martin Nokes

Martin Nokes arbeitet seit 2001 für die AdNovum. Als im Frühling 2004 AdNovum Hungary gegründet wurde, zog er von Zürich nach Budapest, um bei deren Aufbau mitzuwirken.
In Ungarn arbeitet der dipl. Informatik-Ing. FH aktiv an der Umsetzung von Projekten mit und übernimmt als technischer Projektleiter Verantwortung für diese. In seiner Freizeit macht er Jagd auf die geschmeidigsten Gänselebern und den delikatesten Tokajer.

ein eigentliches Change Management verzichtet und ad hoc vorgegangen. Dies birgt jedoch offensichtliche Gefahren: Änderungen lassen sich nicht nachvollziehen, Abhängigkeiten werden übersehen, vermeidbare Doppelspurigkeiten nicht erkannt. Gleichzeitig bedeutet eine Formalisierung des Prozesses aber nicht, dass Flexibilität verloren gehen muss.

Die AdNovum verwendet als Change Management Tool eine einfache Applikation, auf die einerseits ihre Mitarbeiter, andererseits über die gesicherte Customer Zone der AdNovum-Website auch die Kunden Zugriff haben. Über diese Applikation ist jederzeit der Status aller Tasks ersichtlich und in welchem Release sie behandelt werden resp. erledigt worden sind. Der Kunde kann sich die in einem Release enthaltenen Änderungen als Liste ausgeben lassen und sie so einfach verifizieren.

Wartung und Support, Know-how-Transfer

Für die Zeit nach der Einführung des Projekts sind die Wartung und der Support zu gewährleisten. AdNovum bietet diesbezüglich ein langfristiges Commitment für die von ihr entwickelte Software inklusive vertraglicher Garantien.

Soll die Wartung des Projekts von der internen Informatikabteilung des Kunden übernommen werden, stellt sich die Frage nach dem Know-how-Transfer. Die für den Betrieb einer Applikation notwendigen Informationen lassen sich üblicherweise problemlos in entsprechenden Dokumentationen festhalten und über Schulungen vermitteln.

Es hat sich jedoch gezeigt, dass bei der Wartung, das heisst bei der Manipulation von bestehendem Code, Entwicklungserfahrung mit der Applikation von grossem Vorteil ist. AdNovum bietet ihren Kunden deshalb die Option, während der Entwicklungsphase Softwareingenieure zur AdNovum zu entsenden, damit sie aktiv am Projekt mitarbeiten, sprich mitcodieren können. Nach Abschluss des Projekts, wenn die Applikation zur Wartung an den Kunden übergeben wird, kehren diese Entwickler mit beträchtlichem Know-how in ihr Unternehmen zurück, wo sie die Betreuung weiterer interner Entwickler übernehmen können. Auch das umgekehrte Modell ist bereits angewendet worden: In manchen Projekten arbeiten nach der Einführung Ingenieure der AdNovum beim Kunden aktiv in dessen Entwicklungsteam mit, um den Know-how-Transfer sicherzustellen.

Martin Nokes, technischer Projektleiter, hat massgeblich am Aufbau der AdNovum Hungary mitgewirkt.

Fazit

Die hier behandelten Themen sind nicht neu und müssen bei jedem Softwareprojekt berücksichtigt werden; bei global verteilter Ausführung verschieben sich allerdings die Schwerpunkte. Insbesondere dem Management der verschiedenen Schnittstellen kommt eine grosse Bedeutung zu, und die dabei involvierten Prozesse erfahren häufig eine gewisse Formalisierung. In der Regel wird dies jedoch vom Kunden geschätzt, da nicht zuletzt auch er hiervon profitiert. Denn hat man die Schnittstellen unter Kontrolle und kann auf geeignete Teams zählen, lassen sich auch die komplexesten Projekte erfolgreich abwickeln.

Verteilt entwickeln

Christian Siffert-Grob im Interview zu den Herausforderungen bei der verteilten Entwicklung komplexer Applikationen.

Interview: Manuel Ott

NOTITIA: Sie haben als technischer Projektleiter bei der verteilten Entwicklung einer komplexen und umfangreichen Applikation mitgewirkt.
Welches waren die zentralen Herausforderungen?

Von Anfang an standen zwei synchron aktive Entwicklungsstandorte des Kunden fest, Zürich sowie ein Offshore-Standort, dazu kamen vier Teilprojekte in der AdNovum. Der gegebene Projektplan sah Early Feedback vor, konkret waren monatliche Milestones zu erreichen, mit Zwischenlieferungen für andere Entwicklungsteams. In der kurzen Gesamtlaufzeit von sechs Monaten war auf der Basis einer fast 700 Seiten starken Spezifikation eine relativ komplexe Businesslogik abzubilden. Gleichzeitig bestanden zwischen den Teilprojekten der verschiedenen Standorte hohe Abhängigkeiten.
Die Herausforderung war zuerst, die Planung und die Prozesse an diese Rahmenbedingungen anzupassen.

> «Organisatorische Sicherheit und Stabilität sind zentral, wenn Teams komplexe Aufgaben bewältigen.»

Zudem mussten wir im Projekt rasch einen geordneten Rahmen erreichen. Wir mussten den einzelnen Teams organisatorische Sicherheit und Stabilität geben, damit sie ihre teilweise komplexen Aufgaben in Ruhe in Angriff nehmen konnten.

Als anspruchsvoll bei der Realisierung erwiesen sich neben dem engen Terminplan auch die effiziente Kommunikation, die klare Definition der Verantwortlichkeiten, die Qualitätssicherung und das Konfigurationsmanagement.

Was mussten Sie bei der Terminplanung besonders berücksichtigen?

Bei verteilter Entwicklung besteht immer latent die Gefahr, dass der Administrations- und Planungsaufwand überproportional ansteigt.

Beispielsweise bestand die ursprüngliche Planung aus einer sehr detaillierten und langen Liste von Aufgaben. Diese war sicher wertvoll, aber der wesentliche Schritt war, einen Terminplan aller Teilprojekte inklusive aller Milestones auf einem einzigen A4-Blatt zu erstellen. Erst dann waren die Abhängigkeiten der Teilprojekte auf einen Blick erkennbar. Der Terminplan wurde so für alle verständlich und damit das Projekt steuerbar. Basierend auf Risikoüberlegungen und Abhängigkeiten legten wir so den Inhalt der Milestones neu fest. Die «A4-Planung» war zudem auch sehr einfach kommunizierbar. Jedes Team verinnerlichte sie. Dies führte zu klaren Erwartungen und mehr Sicherheit bezüglich der Anforderungen.

Wie bewältigten Sie die Abhängigkeit der Teilprojekte in der kurzen Projektlaufzeit?
Mit der vereinfachten Planung hatten wir eine ideale Flughöhe, um die Abstimmung unter den Teilprojekten vornehmen zu können.
Davon ausgehend priorisierten wir die einzelnen Arbeitspakete gemäss ihrer Zentralität im Projekt, ihrem Risiko und ihren Abhängigkeiten.
Die kurze Projektlaufzeit und die regelmässigen Lieferungen legten ein kaskadiertes Vorgehen nahe: Für jeden Monatsrelease wurde zuerst die technische Spezifikation erarbeitet, anschliessend wurde der Release implementiert. Ein Bein noch in der Umsetzung eines Themas spezifizierten wir jeweils bereits das nächste.

Dieses Vorgehen war etwas unkonventionell, sicher mit Risiken verbunden und nur dank der recht klaren Vorstellungen und Anforderungen des Auftraggebers machbar. Es hätte jedoch ein höheres Risiko bedeutet, die anderen Teilprojekte länger auf Kernteile des Systems warten zu lassen.

Das Konfigurationsmanagement bei verteilter Entwicklung ist sicher nicht trivial; wie sind Sie dies angegangen?
Mit dem Zusammenstellen eines Release begannen wir jeweils im Kern des Systems, wo keine weiteren Abhängigkeiten vorhanden waren, und setzten es fort über die weiteren beteiligten Komponenten und Teams.

Für jeden Zwischenschritt wurde gemeinsam mit allen Teilprojekten die Lieferform für Lieferungen zwischen Teilprojekten festgelegt – Code, Developer Package oder produktives Package. Am Ende der Kette lagen fertige, installationsbereite Packages vor.
Im Grunde genommen geht es um ein verteiltes Konfigurationsmanagement, bei dem die unterschiedlichen Kulturen der beteiligten Standorte in Einklang zu bringen sind. Klingt eigentlich einfach, es braucht aber doch einige Zeit, bis die Themen besprochen, akzeptiert und konsequent umgesetzt sind.
Ein Beispiel war die Häufigkeit der Releases. Im Sinne des «Early-Integration-Ansatzes» will man die Software während der Entwicklung häufig und bereits kurz nach Projektstart auf dem Zielsystem integrieren. Dazu ist der ganze Ablauf der Releaseerstellung durchzuführen, was jedes Mal einen gewissen Einschwingungsvorgang notwendig macht. Denn die Entwicklungsteams müssen sich untereinander synchronisieren, damit das System genau auf diesen Releasetermin stabil wird. Dieser Einschwingvorgang ist ausgeprägter, wenn die Änderungen an der Software noch bedeutend sind, also vor allem in der Hauptentwicklungsphase in der ersten Hälfte des Projekts. Es gilt also, bezüglich der Menge an Releases eine Balance zu finden. Zieht man zu oft Releases, lähmt man die Entwicklung durch die vielen Einschwingvorgänge.
Im beschriebenen Projekt haben wir in gut sechs Monaten etwa 70 Releases gemacht. Realisierbar war dies nur mit einer weitgehend vollständigen Automatisierung des Build- und Testprozesses. Die Abläufe haben sich durch die vielen Releases extrem gut eingespielt. Davon haben wir später in der Test- und Integrationsphase profitiert.

Was mussten Sie bei der Kommunikation beachten?

In einem Softwaresystem wird die Kommunikation weitgehend durch die verteilte Architektur bestimmt. Die Architektur bestimmt insbesondere, welche Schnittstellen und Kommunikationskanäle zwischen den Services etabliert werden müssen.
Wird ein solches System verteilt entwickelt, geschieht etwas Interessantes: Die Kommunikationsschnittstellen und -kanäle wie auch die Services widerspiegeln sich auf der organisatorischen Ebene. Man hat zwischen den einzelnen Teams die gleichen Kommunikationskanäle wie später im fertigen Softwaresystem. Dies kann man auf verschiedenen Ebenen und in verschiedenen Projektphasen beobachten. In der Entwicklung zeigt sich dies etwa bei API-Diskussionen und bei Testdaten, die untereinander ausgetauscht werden. Bei der Integration, beim Testen und Debuggen werden Analysen und Informationen zu Bug Fixes entlang dieser Kommunikationskanäle ausgetauscht.

> «Ein Telefonanruf hat eine ganz andere Qualität als eine Mail.»

Auf der Stufe der Projektleitung ist dafür zu sorgen, dass gezielt Kommunikationskanäle eröffnet werden, um durch kurze Wege klare und verbindliche Entscheidungen herbeizuführen. Dies wirkt sich direkt auf die spätere Stabilität der Schnittstellen aus.
Die Wahl des Kommunikationsmittels ist bedeutend: Greife ich zum Telefon, schreibe ich eine E-Mail, warte ich auf die nächste Sitzung? Ein Telefonanruf hat eine ganz andere Qualität als eine Mail, kann ganz anders ankommen; besonders in heiklen Phasen, wo man eh schon unter Spannung steht, ist es von Vorteil, einmal direkt anzurufen, statt eine Mail zu schreiben. Ich erachte es auch als Aufgabe des Projektleiters, hier beruhigend zu wirken und den Mitarbeitern die Verwendung eines weniger konfliktträchtigen Mediums nahezulegen. Das gegenseitige Verständnis auch über räumliche Distanzen hinweg, das aus dem persönlichen Kontakt entsteht, ist entscheidend für den Projekterfolg.
Insgesamt kann man sagen, dass soziale Faktoren für die Qualität der Software einen vergleichbar hohen Stellenwert haben wie das Software Engineering selbst.

Wie stellten Sie sich dem Problem der Verantwortungsdiffusion, speziell bei über-

Christian Siffert-Grob

Christian Siffert-Grob verantwortet seit dem Sommer 2006 den Java-Engineering-Bereich der AdNovum. Der ETH-Informatiker ist seit 1999 für die AdNovum aktiv und sammelte in zahlreichen Projekten Erfahrungen als Entwickler und technischer Projektleiter. Zum Ausgleich lüftet er seine Gedanken mit Mountain-Biken im Juragebirgszug aus.

geordneten Aufgaben und Aspekten?
An Projektstandsitzungen und bei anderen Meetings muss immer die Funktionsfähigkeit der Gesamtlösung das zentrale Thema sein. Findet der Austausch diesbezüglich offen und ehrlich statt und werden Verantwortlichkeiten entsprechend festgelegt, können einzeln funktionierende Softwarebausteine ein lauffähiges Gesamtsystem ergeben.

> «Die Teams müssen sich auf die Qualität der Komponenten verlassen können.»

Spezielle Aufmerksamkeit benötigt auch die Qualitätssicherung. Neben der Vorbereitung von fachlichen Tests sind die Testinfrastruktur und die Testdaten mit Technikern abzusprechen. Ist das Entwicklungs-Know-how verteilt, entsteht auch hier zusätzlicher Kommunikationsaufwand.

Wie viel Testing braucht es auf Stufe der Teilprojekte?
Um bei engen Releasezyklen an verteilten Standorten Softwarebausteine mit Abhängigkeiten entwickeln zu können, müssen sich die Teams auf die Qualität der jeweils anderen Komponenten verlassen können. Wir konnten die komplexe Businesslogik, die wir programmierten, nicht mit dem GUI testen, dieses wurde später in einem anderen Teilprojekt entwickelt. Basierend auf Risikoüberlegungen wurden Testpunkte vor und in der Businesslogik festgelegt.
Für die Qualitätssicherung setzten wir einen dedizierten Quality Assurance Engineer ein, der unabhängig von den Entwicklern alle Tests schrieb. Die Tests wurden vollautomatisch jede Nacht und vor Lieferungen ausgeführt. Die konsequente Qualitätssicherung hat sich für das Projekt gelohnt, war aber auch bereits Teil des rechtlichen Vertrags mit dem Kunden. Der Kunde hatte Einsicht in die technischen Testresultate, was einen gewissen Druck auf das Testing bedeutete. Umgekehrt wurde für den Kunden dadurch der Aufwand und der Nutzen des Testing transparent.

Für welche Art von Projekten lohnt sich eine verteilte Entwicklung?
Die Vergabe eines Teilprojekts an ein räumlich getrenntes Team bedingt eine relativ gute und genaue Spezifikation. Der Aufwand für deren Erstellung lohnt sich typischerweise nur für Projekte ab einer gewissen Grösse oder solche mit langer Laufzeit. Dabei spielt auch die Sprache des Entwicklungsteams eine Rolle: Je nachdem muss die Spezifikation übersetzt werden, was den Aufwand noch erhöht.
Die Architektur des Projekts muss zudem Sollbruchstellen aufweisen, entlang deren eine Aufteilung zu Vergabezwecken überhaupt Sinn macht.
Schliesslich kommt es auch darauf an, wie viel Flexibilität der Kunde erwartet, das heisst wie kurzfristig er während des Projektverlaufs eingreifen und Änderungen anbringen können will. Die Wissensübermittlung über räumliche Distanzen funktioniert anders als in einem Team im selben Raum; verteilte Entwicklung bedingt ein konsequenteres Vorgehen nach Phasenmodell und erschwert eine inkrementelle Entwicklung.

Spielen bei der Aufteilung in Teilprojekte und bei deren Vergabe neben der Architektur weitere Kriterien eine Rolle?
Wichtig sind auch die Skills an den verschiedenen Standorten: Was wird dort üblicherweise entwickelt, worauf sind die Entwicklungsteams spezialisiert? Ein erwähnenswerter Skill ist nicht nur das Business-Know-how, auch die Sprachkompetenz und die Teamfähigkeit tragen zum Erfolg bei.
Technische Risiken können ebenfalls einen Einfluss auf die Vergabe haben. Sind zum Beispiel Integrationsprobleme mit einem System am Firmensitz des Kunden zu erwarten, empfiehlt es sich, die entsprechende Anbindungslogik durch ein lokales Entwicklungsteam implementieren zu lassen.

Hat sich das Vorgehensmodell der verteilten Entwicklung aus Ihrer Sicht bewährt?
Im vorliegenden Fall war es ein Erfolg. Das Projekt hatte aus rechtlichen Gründen harte Deadlines. Durch die zusätzliche Ressourcenknappheit musste die Arbeit auf verschiedene Teams verteilt werden. Die Milestones wurden eingehalten und die Einführung verlief trotz des engen Terminplans problemlos.

Die globale Verteilung ist nicht nur in der Güterherstellung, sondern auch bei einzelnen Projekttasks der Softwareindustrie bereits Alltag. Nach meiner Erfahrung muss man ein Softwareprojekt wirklich explizit auf die Rahmenbedingungen einer verteilten Entwicklung zuschneiden, um der Wissensverteilung mit ihren Risiken Rechnung zu tragen: Die Planung, die Kommunikation zwischen den Teams, die Art der Gesamtprojektleitung und die Projektaufteilung, all das muss auf die verteilte Entwicklung ausgelegt werden, wie auch das Tailoring des Softwareentwicklungsprozesses und der Qualitätssicherung.

Der Integrator

In Kundenprojekten nehmen die Zahl involvierter Umsysteme und die Varianz benutzter Technologien kontinuierlich zu. Damit wird die Integration in der Softwareentwicklung noch bedeutender.

Von Andreas Leimbacher und Christian Widmer

Die technische Komplexität macht es immer anspruchsvoller, Applikationen produktionsnah zu entwickeln. Gebaut wird im J2EE-Bereich mit Entwicklungswerkzeugen, die einen Applikationsserver beinhalten, welcher sich vielfach anders verhält als sein produktives Pendant.

Konfigurieren statt programmieren

Aktuelle Technologien wie J2EE-Applikationsserver erlauben es, Applikationen zu entwickeln, ohne dass man das genaue Verhalten des Servers kennt und ohne dass man weiss, in welcher Umgebung der Code läuft. Wesentliche Aspekte der Applikation werden vom Programmcode in die Konfiguration ausgelagert, die applikatorische Logik wird vom Deployment entkoppelt und möglichst technologiefrei gehalten. Der Entwickler kann sich im Prinzip ganz auf die Logik konzentrieren und seine Komponenten entsprechend schneller entwickeln.

Die Implikationen dieses Konzepts werden jedoch gerne unterschätzt. Die grosse Anzahl involvierter Komponenten impliziert unzählige Schnittstellen mit unterschiedlichsten Technologien wie etwa J2EE SessionBeans, MessageBeans, WebService oder GIOP. Die damit verbundene Flut von Konfigurationsdateien und Deploymentdeskriptoren ist nur schwer fassbar. Dazu kommt, dass das Tooling zur Verifikation und für das Debugging von Konfigurationen bei Weitem nicht so mächtig ist wie die Debugging-Werkzeuge in der Entwicklung von Java Code. Proprietäre Herstellertechnologien komplizieren das Ganze zusätzlich.

Die Entkopplung von Entwicklung und Integration ist zwar effizienzsteigernd, birgt jedoch auch neue Risiken, da Entwickler für die Simulation von Fremdkomponenten oft nur begrenzt repräsentative Mock-ups zu Verfügung haben.

Organisatorische Komplexität

Die hohe Integration mit verschiedensten Umsystemen und die immer häufigeren Sourcingstrategien bringen es mit sich, dass an einem Projekt immer mehr Teams beteiligt sind, teilweise aus unterschiedlichen Kulturen und Zeitzonen. Dies bedingt eine entsprechende Qualität der Schnittstellenspezifikationen, die bei Änderungen der Anforderungen in letzter Minute kaum mehr zu erreichen ist.

Integration wird zur Notwendigkeit

Infolge der technischen und organisatorischen Komplexität entstehen Applikationen vielfach als lose Komponenten, die oft nur ungenügend gegeneinander getestet werden. Die Integration als Aufgabe bekommt damit eine grosse Bedeutung, muss doch letztlich aus all diesen Komponenten eine lauffähige Applikation gebaut werden, welche bezüglich Installierbarkeit, Performance, Wartbarkeit, Betreibbarkeit und Sicherheit hohen Anforderungen genügt.

> Der Integrator, nicht der Komponenten-Entwickler, baut die produktive Applikation.

Da die Applikation in einer Umgebung zusammengesetzt wird, welche sich von der Entwicklungsumgebung unterscheidet, bringt die Integration Aufgabenstellungen mit sich, mit denen die Entwickler der Einzelkomponenten nicht konfrontiert sind. Wie es dem einzelnen Entwickler bei grossen Projekten nicht mehr möglich ist, die Integration als Nebenjob «on the fly» neben dem Entwickeln zu gewährleisten, ist es dem Betreiber im Allgemeinen nicht mehr möglich, die Software ohne fachliche Unterstützung von Entwicklerseite zu verstehen und selbständig zu betreiben.

Die Rolle des Integrators

Die beschriebenen Trends in der Softwareentwicklung erfordern als logische Konsequenz den Einsatz eines «Integrators». Der Integrator übernimmt die Verantwortung für die Integration und überbrückt mittels Kommunikation und technischer Unterstützung die Distanz der Entwickler zum produktiven System. Er hat den Überblick über die involvierten Komponenten und die verwendeten Toolkits und Technologien und nimmt Einfluss auf das Technologiemanagement im Projekt.

Der Integrator nimmt in der Applikationsentwicklung eine zentrale Mittlerrolle ein. Im Unterschied zum reinen Architekten macht sich der Integrator auch «die Hände schmutzig», er setzt sich mit den praktischen Aspekten des Zusammenbaus der Applikation auseinander. Die produktive Applikation wird nicht vom Komponentenentwickler, sondern vom Integrator zusammengesetzt. Dazu muss er die Architektur und ihre Funktionsweise, das Verhalten und die Technologie der

Andreas Leimbacher

Andreas Leimbacher, El. Ing. HTL und CISSP-zertifiziert, befasst sich seit langem mit der Integration von Systemen und Komponenten. Protokolldetails interessieren ihn dabei gleichermassen wie das Erstellen von Architekturen. Derzeit begleitet er in der Rolle des Integrators J2EE-Projekte für verschiedene Kunden. In der Freizeit macht er gerne sein Motorrad für Touren in Südfrankreich oder Italien flott.

Christian Widmer

Auch Christian Widmer, Dipl. Ing. ETH, hat sich seit seinem Studium mit der Integration mannigfaltiger Systeme und Komponenten vertraut gemacht. Ursprünglich von der Betriebssystemprogrammierung herkommend beschäftigt er sich aktuell vorwiegend mit dem Erstellen und Integrieren von J2EE-Applikationen. Privat ist er am liebsten mit seiner Partnerin im Fernen Osten auf Reisen.

Applikationsserver und Schnittstellen und das produktive Deployment sehr genau kennen. Mit seiner ganzheitlichen und zugleich detaillierten Sicht schliesst der Integrator die Lücke zwischen Projektleiter, Entwicklern, Releaseingenieuren und Betrieb. Im Weiteren unterstützt der Integrator die Umsetzung der Sicherheitsarchitektur und führt bisweilen auch Performance Tuning durch.

Integration ab Projektstart

Zentral für das Gelingen der Integration sind die vorgängige Planung aller Integrationsschritte vom Beginn eines Projektes bis zur Installation in der Produktion und die Analyse der Risiken aus Integrationssicht wie Konfigurationsfehler und Kommunikationsprobleme. Im Rahmen der Vorabklärungen müssen insbesondere die Schnittstellen zu existierenden externen Systemen analysiert und die Entwickler beim Architekturdesign neuer Schnittstellen unterstützt werden. Bezüglich Mock-ups (Simulationen) ist abzuklären, wo sie verwendet werden dürfen und welchen Anforderungen sie genügen müssen. Dies ist der Zeitpunkt, mit den Testentwicklern die Integrationstests zu planen sowie das Lastverhalten abzuklären und Lasttests zu definieren.

Der Integrator arbeitet beim Bereitstellen der Entwicklungsinfrastruktur mit. Diese soll der produktiven Umgebung möglichst nahekommen und trotzdem kurze Entwicklungszyklen erlauben: So kann ein Teil der Integration – natürlich automatisiert – bereits während der Entwicklung durchgeführt werden.

Frühzeitig werden automatisiert Packages gebaut, welche dann auf dem Referenz- resp. dem Integrationssystem verifiziert werden. Diese enthalten vorzugsweise ein Applikationsgerüst mit allen Komponenten. Dabei ist zu diesem Zeitpunkt nicht relevant, ob die Komponenten logisch korrekt funktionieren.

> Behandelt man Installations- und Integrationsaspekte früh, bleibt Zeit für entsprechende Adaptionen.

Zusammen mit den Entwicklern definiert der Integrator auf der Basis der Kundenanforderungen die Konfigurationsparameter. Es ist auch im Interesse des Integrators, zu diesem Zeitpunkt die Weichen für eine einfache Installation zu stellen.

Early & Continuous Integration

Um unliebsamen Überraschungen vorzubeugen und die Teams vor Last-Minute-Aufwänden zu schützen, ist ein kontinuierlicher Austausch zwischen Entwicklern, Betrieb und Integrator notwendig. Das beste Hilfsmittel hierzu ist eine installierbare Applikation. Es geht lediglich darum, einen vertikalen «Durchstich» über alle Tiers zu haben, anhand dessen die Kommunikationswege und Interaktionen mit der produktiven Konfiguration getestet werden können.

Initial erstellen die Releaseingenieure die Packages in enger Zusammenarbeit mit dem Integrator. Die diversen Erkenntnisse und Resultate werden laufend den Entwicklern zurückgemeldet.

Der Integrator begleitet anfangs auch die Installation beim Kunden: So können sich die Systemadministratoren beim Kunden frühzeitig mit der Applikation vertraut machen,

betriebsspezifische Anforderungen definieren und sie via Integrator an die Entwickler kommunizieren.

Während und nach der Installation nimmt der Integrator erste Schnittstellentests vor und meldet Feedback über allfällige Fehler inklusive Hinweisen für die Fehlersuche an die Entwickler zurück. Durch das frühe Erkennen von Optimierungsbedarf hinsichtlich Installations- und Integrationsaspekten, aber auch Performancefragen bleibt für entsprechende Adaptionen noch genug Zeit.

Während der Entwicklungsphase wird die Applikation periodisch installiert. Dadurch wird sichergestellt, dass sie installierbar bleibt. Ein hoher Automatisierungsgrad ist dabei die Grundlage für Reproduzierbarkeit und Effizienz. Das beinhaltet auch eine Installation, die nur minimale manuelle Eingriffe benötigt. Auf Integration spezialisierte, automatisierte Tests helfen bei der Verifikation der installierten Applikation.

Typischerweise nehmen gegen den Projektabschluss die Integrationsaktivitäten ab, so dass der Integrator verwandte Arbeiten wie Lasttests durchführen kann. Auch macht ihn sein Know-how für Entwickler zu einem wertvollen Partner bei der Fehlersuche.

Der Integrator bei AdNovum

Aufgaben

- Schnittstelle zwischen Entwicklung, Release Engineering und Kundeninfrastrukturverantwortlichen
- Unterstützt den technischen Projektleiter bezüglich Systemarchitektur
- Unterstützt den leitenden Entwickler in codebezogenen Integrationsfragen
- Stellt frühzeitig und kontinuierlich sicher, dass die verschiedenen SW-Komponenten miteinander als Ganzes funktionieren
- Sorgt für die Optimierung der Betreibbarkeit (zusätzliche Tests, Tracing etc.)
- Unterstützt die Installation beim Kunden
- Überwacht das Verhalten der Applikation nach der produktiven Einführung
- Verantwortlich für Sicherheitsaspekte der Konfiguration und der Packages

Profil

- Senior Software Engineer mit mehrjähriger Projekterfahrung
- Guter Kommunikator
- Mit den verschiedensten Technologien auf allen Schichten des Software Stack vertraut
- Guter Analytiker und Troubleshooter

Erfolgreiche Risikominimierung

Der Erfolg der Continuous Integration lässt sich am folgenden Beispiel illustrieren:

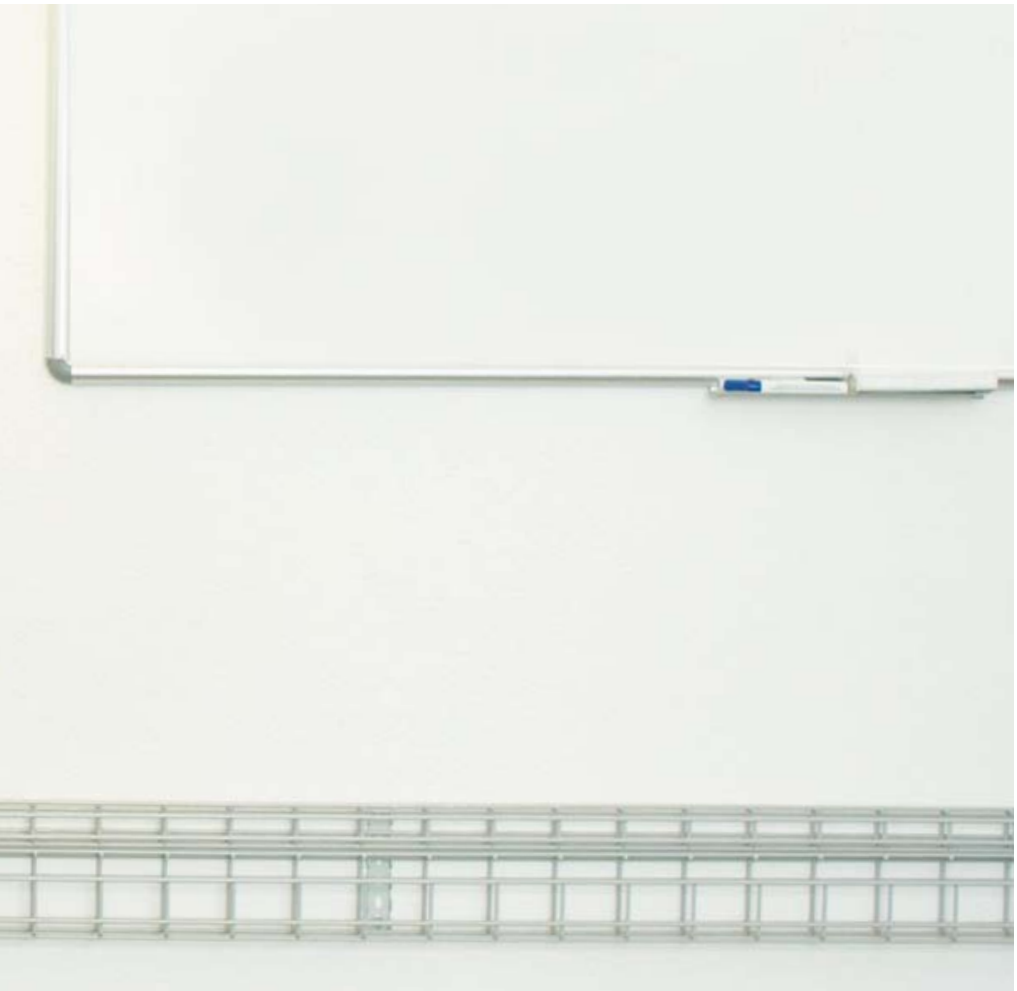
Ausgangslage war ein Softwareprojekt mit Entwicklung an vier verschiedenen Standorten mit unterschiedlichen Entwicklungsplattformen. Das System umfasste bestehende produktive Systeme, gegen welche integriert werden musste. Die Applikation war erst in der Entwicklungsumgebung lauffähig, d. h., es waren noch keine installierbaren Packages gebaut, und die meisten Entwickler arbeiteten mit Mock-ups.

AdNovum führte im Projekt Continuous Integration in zwei Stufen ein:

1) Den Entwicklern wurde vom Integrator eine Umgebung zur Verfügung gestellt, in der sie automatisiert produktionsnahe Integrationstests durchführen konnten, ohne sich um konkrete Integrationsaspekte kümmern zu müssen. Die meisten Entwickler waren jedoch weiterhin auf Mock-ups angewiesen.

2) Wöchentlich wurden Packages erstellt, beim Kunden installiert und automatisiert getestet.

Die zweistufige Continuous Integration war aus Sicht der Integration das stärkste Instrument zur Qualitätssicherung und Risikominimierung. In der Endphase konnte der Integrator sich dadurch auch um Unterstützung und Durchführung des Testings kümmern. Die Testphase wurde durch Tracing optimal unterstützt und Tester- und Applikationsverhalten konnten reproduziert werden. Der hohe Detaillierungsgrad beim Logging und Tracing erlaubte eine effiziente Fehleranalyse trotz verteilt arbeitender Teams. ■



Continuous Integration bei AdNovum

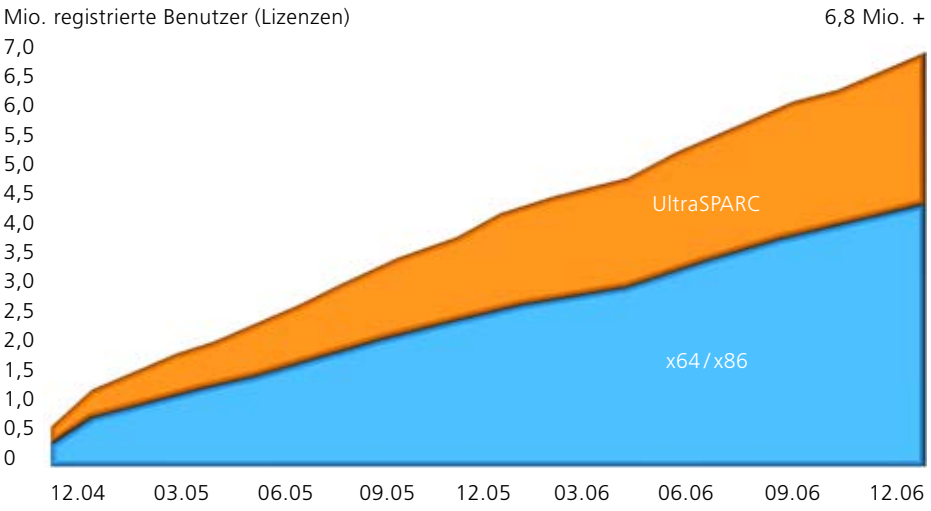
«Continuous Integration», ein Begriff aus der «XP eXtreme Programming»-Methodik, wird bei AdNovum seit langem gelebt. AdNovum hat eine starke Tradition im vollautomatisierten Bauen von Binaries und Packages. Alle Projekte werden mindestens nächtlich kompiliert und auf dem Zielsystem getestet.

Solaris 10 – ein Meilenstein

Solaris 10 und OpenSolaris definieren das Operating System neu als Service-Plattform. Traditionelle OS-Funktionalität wird mit Application Services und Identity Management erweitert.

Von Reto Brack, Sun Microsystems (Schweiz) AG

Für IT-Professionals, die sich mit Kostenreduktion beschäftigen müssen, ist Solaris 10 die erste Wahl. Mit seiner Verfügbarkeit auf x64-, x86- und SPARC-Systemen ermöglicht es die heute gefragte Standardisierung und Konsolidierung der IT-Infrastruktur für den Betrieb von Anwendungen in idealer Weise. Als einheitliches Betriebssystem kommt Solaris 10 auch Entwicklern entgegen. Sie können ihre Lösungen dank der «Sun source code compatibility guarantee» schnell und auf jeder gewünschten Systemplattform verfügbar machen. Eine einmal entwickelte Applikation ist zudem durch die einzigartige «Sun binary compatibility guarantee» geschützt. Aktuelle Entwicklungen können ohne Portierung und damit ohne zusätzliche Kosten auch auf künftigen Solaris-Versionen betrieben werden.



Um den unerwünschten Vendor Lock-In zu vermeiden, hat Sun Microsystems das Projekt OpenSolaris gestartet. Der Solaris Source Code wurde für die Open Source Community freigegeben. Somit sind die Innovation und die Weiterentwicklung für Solaris auf breiter Front gewährleistet.

Virtualisierung

Dank dem integrierten Container-Zonenkonzept bietet Solaris 10 die Möglichkeit der OS-Virtualisierung. Theoretisch können beliebig viele virtuelle Rechner auf einer Betriebssysteminstanz betrieben werden. Die Systemleistung bleibt dabei für die Applikationen nahezu konstant hoch. Neben dem einzigartigen Solaris-10-Konzept unterstützt Sun aber auch Virtualisierungstechnologien wie Resource Sharing, Logical Domains und Hard Partitioning.

Performance und Sicherheit

Solaris-10-Systeme skalieren bezüglich Anzahl Prozessoren und Netzwerkverbindungen nahezu linear und ohne Performanceverlust. Dazu kommen die Funktionen, die Verlässlichkeit sowie ein Sicherheitsniveau der Highend-Klasse. So unterstützt Solaris 10 auch umfangreiche und businesskritische Applikationen perfekt. Diese Eigenschaften gelten für die SPARC-Plattform ebenso wie für Systeme mit AMD-Opteron- oder Intel-Xeon-Prozessoren.

Dazu Larry Ellison, CEO Oracle: «With Solaris 10 Sun has delivered an open-source, cross-platform OS. And, it's impossible to ignore the significant market opportunity created by the incredible growth of Solaris 10 along with Sun's industry standard x64 and UltraSPARC-based systems. Solaris was the clear choice for our development platform.»

Optimierung

Mit Solaris 10 DTrace lassen sich Systeme in Echtzeit analysieren. Administratoren können so das Systemverhalten im laufenden Betrieb untersuchen und Fehler und Optimierungspotenzial ausfindig machen. Entwicklern erlaubt DTrace die gezielte Analyse neuer Applikationen und das Identifizieren und Beheben von Leistungsengpässen.

Marktakzeptanz

Für die Wahl eines Betriebssystems sind auch die Akzeptanz und die Verbreitung im Markt entscheidend. Seit Januar 2005 haben sich mehr als 6 Millionen Benutzer für eine Lizenz registriert. Damit wurde Solaris häufiger ausgeliefert als Red Hat Enterprise Linux, IBM AIX und HP-UX im gleichen Zeitraum zusammen.

Hinzu kommen die Anzahl der Anwendungsentwickler, die ein Betriebssystem unterstützen, und die Plattformen, auf denen es eingesetzt werden kann. Solaris 10 läuft auf mehr als 250 Systemen von Herstellern wie Dell, HP oder IBM. Kein anderes OS unterstützt so viele Plattformen. Seit kurzem setzt Intel Solaris als Betriebssystem für Intel-Xeon-basierte Server ein, auch dies ein deutliches Statement. Intel wird zudem die weitere Entwicklung von Open Solaris aktiv unterstützen. Solaris 10 ist kostenfrei verfügbar, dies erleichtert den Einstieg und lädt zum Ausprobieren ein. www.sun.com ■

Impressum

Herausgeber:
AdNovum Informatik AG
Corporate Marketing
Röntgenstrasse 22
CH-8005 Zürich
Telefon 044 272 61 11
Telefax 044 272 63 12
E-Mail info@adnovum.ch
www.adnovum.ch
Verantwortlich und Redaktion:
Manuel Ott
Gestaltung und Realisation:
Rüegg Werbung, Zürich
Fotografie:
Gerry Nitsch, Zürich