Anlage 2.3 zum Antrag auf Förderung eines Entwicklungsvorhabens

Antragsteller: consistec Engineering & Consulting GmbH
Antrag vom 23.02.2012 im Technologieprogramm Saar (TPS)

Bitte beachten: Eine detaillierte und für Dritte nachvollziehbare Darstellung des Projektes, der einzelnen Arbeitsschritte sowie des im Einzelnen damit verbundenen Zeit- und Kostenumfangs ist unabdingbare Voraussetzung für die Prüfung und Festsetzung der zuwendungsfähigen Kosten sowie der daraus resultierenden Zuwendung.

1. Projektbezeichnung

SYNC-BIB

2. Allgemeine Projektdarstellung (wesentlicher Inhalt, Zielsetzungen)

Zielsetzung des Vorhabens

Die consistec GmbH beabsichtigt die Entwicklung einer generischen, plattformunabhängigen Synchronisationsbibliothek für SQL-Datenbanken. Die Synchronisationsbibliothek soll folgende Leistungsmerkmale erfüllen:

1. Plattformunabhängiges Design zur leichten Anbindung verschiedener Betriebssysteme, Datenbanken und Programmiersprachen. Dabei soll mittels plattformunabhängiger Kommunikationsprotokolle auch - völlig transparent für den Anwender - ein gemischter Betrieb (z.B. Windows Server mit Microsoft SQL Server Datenbank und Android Client mit SQLite Datenbank) unterstützt werden.
2. Leichte Integrierbarkeit in existierende Anwendungen ohne Einschränkungen oder Änderungen von existierenden Datenbankschemas
3. Effiziente Unterstützung sehr großer Datenmengen durch Nutzung von Filtern und Datenbanktriggern (sofern von der genutzten Datenbank unterstützt)
4. Verschlüsselte Kommunikation und zertifikatsbasierte Authentifikation zur Sicherung schützenswerter Daten
5. Vollautomatische Erkennung der Datenbankschemas ohne manuelle Konfiguration
6. Eignung für die Anwendung in großen Unternehmen:
   - Zentrales Geräte- und Identitäts-Management
   - Skalierbarkeit: Gute Leistung auch bei tausenden Nutzern

Die beschriebene Projektidee basiert auf unseren umfangreichen Erfahrungen bei der Entwicklung von Lösungen zur mobilen Auftragserfassung und –bearbeitung sowie aus dem Wissen, dass verteilte Datenbestände in vielen Anwendungen synchron gehalten werden müssen.

Kontext & Motivation

Nutzer von IT-Systemen erwarten heutzutage, dass sie ständig und überall auf ihre Daten zugreifen können. Der einfachste Ansatz hierfür ist, auf die Daten eines zentralen Servers direkt per Internet zuzugreifen. Diese Methode, die Daten in der „Cloud“ zu speichern, hat jedoch neben potenziellen Sicherheitsproblemen auch den großen Nachteil, dass sie eine ständige Hochgeschwindigkeits-Internetverbindung voraussetzt, um mit den Daten zu arbeiten. Gerade bei Außendienstmitarbeitern ist jedoch häufig keine stabile oder performante

Internetverbindung möglich (man denke nur an einen Monteur, der in einem Heizungskeller mit schlechter Mobilfunkverbindung arbeiten muss, oder Tätigkeiten im Ausland, wo teure Roaming-Gebühren vermieden werden sollen). Über dieses klassische Beispiel hinaus gibt es viele weitere Anwendungsfälle in denen verteilte Datenbestände synchron gehalten werden müssen: Offline Anbindung von Community-Datendiensten, ERP- und CRM-Systeme, die über mehrere Standorte verteilt sind, moderne Backup-Lösungen für mobile Endgeräte (vgl. itunes von Apple) u.v.m.
Eine gute Alternative zum direkten Online-Zugriff auf die Daten stellt die regelmäßige Synchronisation der Daten dar. Dabei werden die benötigten Daten auf das mobile Gerät (Laptop, Mobiltelefon, iPad, …) kopiert und regelmäßig mit dem Datenbestand auf dem zentralen Server abgeglichen. Dieser Abgleich kann unterwegs per Internet oder bei größeren Datenmengen auch (z.B. einmal täglich) vor Ort über ein lokales Netzwerk erfolgen.
Bei der Umsetzung eines solchen Synchronisations-Szenarios für Anwendungsprogramme ergeben sich für Softwareentwickler regelmäßig die gleichen Anforderungen:

- Effiziente und selektive Datenübertragung, um die transportierten Datenmengen zu minimieren
- Sichere Zugriffssteuerung und Verschlüsselung der Datenübertragung
- Hohe Performanz, um auch größte Datenmengen und eine hohe Anzahl von Nutzern unterstützen zu können
- Sichere Erkennung und Behandlung von Konflikten durch den gleichzeitigen Zugriff mehrerer Anwender auf die gleichen Daten
- Unterstützung einer großen Anzahl von Geräten, Betriebssystemen, Datenbanksystemen und Programmiersprachen, um Anwender nicht auf eine bestimmte Plattform festzulegen (sog. Vendor-Lock-In)
- Extrem hohe Qualitätsanforderungen, da Verfälschungen der Daten unter allen Umständen vermieden werden müssen

Eine optimale technische Lösung dieser Probleme ist dabei intern höchst komplex, kann allerdings nach außen auf einige wenige, einfache Schnittstellen abgebildet werden. Hier bietet sich für den Entwickler von Anwendungssoftware die Verwendung von Software-Bibliotheken bzw. Frameworks an, die die komplexe Lösung eines Problems kapseln und in wiederverwendbarer Form zur Verfügung stellen. Ein und dieselbe Bibliothek kann dann von vielen Programmen genutzt werden und wird deshalb oft von Drittfirmen entwickelt und separat vertrieben. Die Nutzung solcher Bibliotheken erlaubt den Anwendungsentwicklern, sich bei der Entwicklung auf ihre jeweiligen Kernkompetenzen zu konzentrieren.
Im professionellen Umfeld werden in der Regel Datenbanken zur Speicherung der Informationen genutzt. Fast keine der existierenden Lösungen zur Synchronisation solcher Datenbanken erfüllen die oben genannten Anforderungen. Die wenigen Ausnahmen (z.B. das Microsoft Sync Framework) sind auf spezielle Betriebssysteme und Datenbanken beschränkt und behindern dadurch die Wahlmöglichkeiten des Anwenders deutlich. Insbesondere angesichts der immer stärkeren Nutzung von Smartphones, ultra-mobilen Laptops oder web-basierten Lösungen unter den verschiedensten Betriebssystemen (Android, Apple iPhone, Microsoft Windows, Linux, …) sind solche proprietären Ansätze ungeeignet.

3. Innovationscharakter im Vergleich zum Stand der Technik

Die Entwicklung eines generischen Synchronisationsframeworks für Datenbanken, das die oben beschriebenen Anforderungen erfüllt, erfordert die Lösung mehrerer komplexer Teilprobleme:

- Automatische Erkennung des Datenbankschemas und Replikation des Schemas vom Server auf die verschiedenen Client-Datenbanken
- Effiziente und skalierbare Änderungserkennung und bidirektionale Synchronisation der Änderungen
- Hohe Skalierbarkeit der Serverkomponenten, so dass auch hunderte zeitgleich zugreifende Nutzer, tausende Endgeräte und Millionen von Datensätzen unterstützt werden können
- Höchste Stabilität: die zu synchronisierenden Datenbanken dürfen auf keinen Fall in einen undefinierten oder illegalen Zustand gelangen, da sonst Datenverluste und -verfälschungen möglich sind
- Zuverlässige Absicherung der Systeme gegen elektronische Angriffe - auch für den Fall, dass Endgeräte gestohlen werden
- Cross-Plattform Unterstützung, so dass auch Kunden mit heterogenen Computersystemen Daten problemlos synchronisieren können
- Optimale Unterstützung der in der Regel sehr asymmetrischen Verteilung von Speicher- und Rechenkapazitäten: große Server mit äußerst leistungsfähigen Datenbanksystemen auf der einen Seite und Mobiltelefone mit geringem Speicherplatz, niedriger Rechenleistung und einfachsten Datenbankmanagementsystemen auf der anderen Seite
- Modulares Design, um mit geringem Aufwand die Unterstützung zusätzlicher Datenbanken, Betriebssysteme oder Programmiersprachen hinzufügen zu können
- Minimierung des Konfigurationsaufwands:
  - Automatische Erkennung des Datenbankschemas und Ablegen der notwendigen Verwaltungsdaten ohne Veränderung existierender Tabellen
  - Einfache und feingranulare Regelung des Verhaltens in Konfliktsituationen, d.h. wenn mehrere Anwender die gleichen Daten editieren

Es gibt bereits Lösungen für Teile der genannten Anforderungen auf dem Markt. Die einzig kommerziell verbreitete Lösung ist das Microsoft Sync Framework. Die Verwendbarkeit und der Nutzen des Microsoft Sync Framework ist ebenso wie die der anderen bereits existenten Lösungen durch die nicht vorhandene Unterstützung verschiedener Betriebssysteme und Datenbanken stark eingeschränkt. Diese können auf weit verbreiteten mobilen Plattformen (Smartphones, Tablets) nicht eingesetzt werden.

Die Erfüllung der zuvor genannten Designanforderungen ist einzigartig.

4. Angaben zur Wettbewerbssituation

bitte Quellenangaben, Internet-Adressen etc. hierzu bzw. beigefügte Unterlagen

Das zentrale Kriterium zur Abgrenzung zu Marktbegleitern ist die Unterstützung unterschiedlicher Betriebssysteme und Datenbanken durch ein komfortabel zu nutzendes Framework. Es gibt nur wenige Sync-Frameworks für SQL-Datenbanken, die dieses anvisierte Aufgabenfeld wenigstens ansatzweise abdecken.
Das verbreitete Microsoft Sync-Framework für SQL-Datenbanken unterstützt sowohl client- als auch serverseitig nur Microsoft-Komponenten und ist daher für eine betriebssystemübergreifende Synchronisierung mit verschiedenen Datenbanksystemen nicht geeignet. Bei Nutzung dieses Frameworks würde sich ein Softwareentwickler auf Microsoft Betriebssysteme und den Microsoft SQL-Server festlegen. (http://msdn.microsoft.com/de-de/library/bb902854.aspx)

Eine dem hier geplanten SyncFramework auf den ersten Blick ähnliche Lösung ist das Opensource Projekt SymmetricDS (http://symmetricds.codehaus.org/). Dieses unterstützt verschiedene Datenbanksysteme und nutzt wie die hier geplante Lösung ein optimistisches Replikationsverfahren. Allerdings hat die SymmetricDS Lösung auch client-seitig umfangreiche Abhängigkeiten an andere Bibliotheken; so setzt sie unter anderem einen vollständigen Webserver auf dem Client voraus. Dies stellt auf Mobilgeräten nahezu ein Ausschlusskriterium dar. Des Weiteren ist der Konfigurationsaufwand sehr groß und die Integration der Synchronisationsmechanismen in eigene Programme nur schwer möglich. Nicht zuletzt kann der Einsatz von SymmetricDS durch die verwendete Lizenz (LGPL) in manchen kommerziellen Anwendungsszenarien problematisch sein.

Eine weitere Synchronisationsbilbiothek ist im Software Development Kit (SDK) des Android Betriebssystems integriert. Ähnlich dem Microsoft SyncFramework sind dadurch andere (Client-)Betriebssystem von der Nutzung praktisch ausgeschlossen.

5. Darstellung des Funktionsprinzips bzw. der wesentlichen Unterschiede im Vergleich zum Stand der Technik (technische Beschreibung)
   bitte anschauliche Skizzen, Zeichnungen, Fotos etc. beifügen

   Architektur (siehe beigefügtes Diagramm)

Um eine optimale Flexibilität bezüglich der unterstützen Datenbanksysteme zu gewährleisten, soll die zu entwickelnde Synchronisationslösung modular um einige wenige zentrale Kernkomponenten herum aufgebaut werden.
Die zentralen Komponenten kapseln die Kern-Algorithmen zur Schema-Erkennung, Erkennung und Übertragung von Änderungen oder auch zur Behebung von Konflikten. Die relativ kompakten Module binden diese Kernkomponenten an die jeweiligen Datenbanksysteme an und verstecken dadurch deren Besonderheiten vor der Implementierung der Kernkomponenten. Dadurch können Anpassungen an weitere Datenbanksysteme mit wenig Aufwand umgesetzt werden.
Client- und Server-Komponenten sollen über ein einheitliches Internet-Protokoll kommunizieren, so dass sich auch Komponenten auf unterschiedlichen Betriebssystemen miteinander synchronisieren können.

Geplanter Lösungsansatz zur Änderungserkennung

Bei der initialen Synchronisation eines neuen Mobilgeräts besteht die hauptsächliche Schwierigkeit in der automatischen Erkennung und Replizierung des auf dem Server verwendeten Datenbankschemas. Die eigentliche Übertragung der Daten stellt dann keine große Herausforderung mehr da.
Im Vergleich zur initialen Synchronisation ist die spätere Erkennung, Zusammenführung und Übertragung von geänderten Daten deutlich anspruchsvoller.
Die Erkennung von Änderungen, die ein Client (d.h. ein Mobilgerät) an den zentralen Server übertragen muss, gestaltet sich dabei noch relativ einfach. Da es nur einen Server gibt, muss sich der Client lediglich merken, welche Datensätze er seit der letzten Synchronisation geändert hat.
In der Gegenrichtung hingegen muss der Server unter Umständen tausende Client-Geräte bedienen, so dass er sich nicht für jeden Datensatz merken kann, welcher Client

diesen in welcher Version vorliegen hat. Statt dieses server-basierten Ansatzes planen wir die Nutzung einer verteilten Lösung, bei der sich jeder Client mittels eines Zeitstempels merkt, wann er das letzte Mal Daten vom Server bezogen hat. Diesen Zeitstempel sendet er zum Server und erhält im Gegenzug die seither geänderten Daten. Der Server muss sich dadurch unabhängig von der Anzahl der Client-Geräte pro Datensatz nur ein einziges zusätzliches Feld merken (den Zeitpunkt der letzten Änderung) wodurch eine sehr gute Skalierbarkeit der Lösung sichergestellt ist.

Um Problemen mit verschiedenen Zeitzonen oder asynchronen Uhren auf den Endgeräten aus dem Weg zu gehen, wollen wir dabei keine „normalen“ Zeitstempel einsetzen sondern Revisions-Nummern, die den Zeitverlauf abstrakt abbilden.
Die Revisionsnummer wird bei jeder Änderung von Daten um 1 erhöht. Dadurch steht eine bestimmte Revisionsnummer für den Zustand der Daten zu einem bestimmten Zeitpunkt - die Revisionsnummer repräsentiert also gewissermaßen diesen Zeitpunkt. Dadurch, dass die Revisionsnummern aber unabhängig von Zeitzonen, Sommer-/Winterzeit oder asynchronen Uhren auf Client-/ Server-Seite sind, braucht man sich um solche Probleme nicht mehr zu kümmern und alle beteiligten Computersysteme haben eine eindeutige Vorstellung davon, welche Version der Daten durch z.B. Revision 4300 repräsentiert wird.

Auch die effiziente serverseitige Erkennung der geänderten Daten ist keineswegs trivial. Zwar aktualisiert das Synchronisationsframework bei jeder eigenen Änderung der Daten auch den oben genannten Zeitstempel. Allerdings kann (und soll) nicht verhindert werden, dass andere auf dem Server laufende Anwendungen im Sinne einer besseren Effizienz direkt in der Datenbank schreiben; bei solchen externen Zugriffen kann man sich nicht darauf verlassen, dass der Zeitstempel ebenfalls aktualisiert wird. Die Sync-Bibliothek muss dementsprechend eine Möglichkeit zur Erkennung solcher externer Änderungen implementieren. Als einfache und bei allen Datenbanksystemen nutzbare Basisvariante soll dazu eine Lösung umgesetzt werden, die anhand eines sogenannten Hash-Wertes bei jeder Synchronisation für jeden einzelnen Datensatz überprüft, ob sich dieser seit der Änderung des Zeitstempels geändert hat.

Hashwerte bestehen aus relativ kurzen Zahlen bzw. Zahlen-Buchstaben-Kombinationen (in der Regel ca. 30 Zeichen, z.B. d8e8fca2dc0f896fd7cb4cb0031ba249), die eine größere Menge Daten repräsentieren. Im geplanten Projekt repräsentieren die Hash-Werte z.B. den Inhalt einer Zeile einer Datenbanktabelle, der je nach Anwendungsfall hunderte Kilobytes oder auch noch größer werden kann. Die Hash-Werte werden aus den Daten mittels mathematischer Formeln so errechnet, dass möglichst immer für verschiedene Original-Daten verschiedene Hash-Werte ergeben (sogenannte Kollissionsfreiheit). Das funktioniert in der Praxis "erstaunlich" zuverlässig und man verlässt sich bei praktisch allen Methoden zur Verschlüsselung bzw. digitalen Signatur auf Hash-Werte.
Die Idee bei der Sync-Bibliothek besteht nun darin, dass die Hash-Werte genutzt werden, um zu erkennen ob ein Datensatz auf Client- und Serverseite die gleichen oder unterschiedliche Daten enthält. Dazu könnte man bei einem naiven Ansatz einfach den kompletten Datensatz über das Internet versenden. Der große Vorteil der Hash-Werte liegt jedoch in ihrer geringe Größe. Es erzeugt sehr viel weniger Netzwerklast, kleine Hash-Werte zwischen Server und Client zu übertragen, als jedes Mal die (großen) Original-Datensätze zu versenden.

Bei großen Datenbanken stellt die oben geschilderte Nutzung von Hash-Werten unter Umständen aber einen Performance-Engpass dar und soll daher – sofern durch das

Datenbanksystem unterstützt – durch eine trigger-basierte Variante ergänzt werden, die die Änderungen an den Zeitstempeln automatisch bei jedem Schreibzugriff auf einen Datensatz durchführt.
Die optimale Umsetzung und Kombination dieser Kern-Technologien (abstrakter Zeitstempel, Hash basierte Änderungserkennung, Nutzung von Datenbank-Triggern zur Optimierung) kann momentan nur skizziert werden und stellt einen der Forschungsschwerpunkte im Projekt dar.

Erkennung und Behandlung von Konflikten

Natürlich muss eine Synchronisationslösung auch mit der Situation umgehen können, dass mehrere Nutzer gleichzeitig versuchen, dieselben Datensätze zu bearbeiten. Eine Lösung dafür besteht darin, den Zugriff auf Datensätze, die ein Nutzer gerade bearbeitet, für alle anderen zu sperren – ein Ansatz der nur bei sehr kleinen Teams in der Praxis funktioniert. Bei der geplanten Synchronisationsplattform wollen wir hingegen auf eine sogenannte optimistische Zugriffssteuerung setzen. Dabei sind gleichzeitige Änderungen durch mehrere Nutzer erst einmal möglich und werden erst im Nachhinein bei der nächsten Synchronisation erkannt und aufgelöst. Da es für die Auflösung der Konflikte keine universelle Lösung gibt sondern diese immer von der konkreten Anwendung und Datenbankstruktur abhängt, wollen wir den Anwendern der Synchronisationsplattform über ein flexibles Interface verschiedene Lösungswege anbieten: angefangen von einfachen, festen Regeln (pro Datenbanktabelle) bis hin zu grafisch unterstützter, interaktiver Konfliktbehebung durch den Endanwender.

6. Schutzrechtsituation (erteilte Patente, Gebrauchsmuster, Schutzrechtsanmeldungen)
bitte ggf. in Anlage beifügen

Da es sich bei der SYNC-BIB um eine Softwarebibliothek handelt sind keine Schutzrechtsanmeldungen vorgesehen.

7. Erläuterung, in welchem Umfang sich das Projekt hinsichtlich Aufwand und Komplexität von sonstigen routinemäßigen Tätigkeiten / Projekten des Unternehmens abhebt

Ein Projekt in der Größenordnung von 2 Personenjahren bei den unten beschriebenen wirtschaftlichen Risiken durchzuführen, ist für consistec nicht möglich. Zur Durchführung des Projektes sind im Schnitt zwei Entwickler über einen Zeitraum von einem Jahr notwendig, die in diesem Zeitraum keine anderen betrieblich notwendigen Tätigkeiten in nennenswertem Umfang durchführen können. Insofern ist das beantragte Projekt außerhalb routinemäßiger Tätigkeiten einzuordenen.

8. a) Angaben zu den mit dem Projekt verbundenen erheblichen technischen Risiken

Aufgrund unserer langjährigen Erfahrung gehen wir davon aus, dass unsere Ideen grundsätzlich umgesetzt werden können. Dennoch bleibt ein erhebliches technisches Risiko, da die meisten unserer Lösungsansätze noch evaluiert werden müssen. Erst nach einer prototypischen Implementierung kann durch Lasttests die Skalierbarkeit und notwendige Performance für "große" Einsatzszenarien nachgewiesen werden. Des Weiteren muss durch umfangreiche Tests des Prototypen sichergestellt werden, dass die geforderte Robustheit gegenüber konkurrierenden Zugriffen vieler Nutzer tatsächlich durch den gewählten Synchronisationsalgorithmus erfüllt wird. Nach der prototypischen Implementierung könnten dann das Performance-Tuning und die Weiterentwicklung zu einem Produkt erfolgen.