Anforderungen in mobilen Geschäftsprozessen und ihre Auswirkungen auf die Architektur mobiler Systeme

Volker Gruhn, André Köhler
Universität Leipzig
Lehrstuhl für Angewandte Telematik / e-Business
Klostergasse 3, 04109 Leipzig
{gruhn, koehler}@ebus.informatik.uni-leipzig.de

Abstract: Die Unterstützung mobiler Arbeiter mit mobilen IT-Lösungen kann erhebliche Verbesserungen in den mobilen Geschäftsprozessen eines Unternehmens erzeugen. Die wesentliche Eigenschaft eines solchen mobilen Systems ist die Fähigkeit, sich mittels Mobilfunk mit einem zentralen Server zu verbinden, um z.B. auf Kundendaten zuzugreifen. Die Häufigkeit und der Ort der Nutzung, die benötigte Datenaktualität, Anforderungen an die Benutzeroberfläche und vieles mehr sind zentrale Aspekte, die bei der Entwicklung passender Systemarchitekturen berücksichtigt werden müssen. Dieser Beitrag stellt vier typische Architekturen für mobile Systeme vor und beschreibt einige ihrer wichtigsten Eigenschaften. Darüber hinaus werden häufige Anforderungen aus mobilen Geschäftsprozessen vorgestellt und die aus ihnen resultierenden Implikationen für die genannten Architekturen diskutiert.

1 Einleitung

Seit der Verfügbarkeit breitbandiger Mobilfunknetze sowie rückläufigen Kosten für mobile Endgeräte ist die Nutzung mobiler Anwendungen zu einer interessanten Möglichkeit in verschiedenen Bereichen geworden. Unternehmen mit einer großen Zahl an mobilen Arbeitern (z.B. Servicetechniker, Vertriebsmitarbeiter, häusliche Pflegedienste) können mobile Anwendungen nutzen, um Zugang zu Unternehmensanwendungen und -datenbanken am POS (Point of Service) zu erhalten. Damit wird eine besser Koordination mobiler Arbeiter, schnellere Aufgabenzuweisung, die Vermeidung fehleranfälliger Medienbrüche, sofortiger Zugriff auf Kundendaten und vieles mehr möglich [GKK05], [NSS05], [GK06].

Die Architektur eines mobilen Systems bewegt sich häufig innerhalb eines Spektrums zwischen Always-Online-Systemen mit browserbasierten Clients auf der einen Seite und Offline-Systemen mit Fat-Clients und gelegentlicher Synchronisation mit einem zentralen Server auf der anderen Seite. Welche Softwarearchitektur am besten zu einer bestimmten mobilen Aufgabe passt, hängt vor allem von den Anforderungen aus dem Geschäftsprozess ab. Die Häufigkeit des Ortswechsels, die Wahrscheinlichkeit der Netzverfügbarkeit, Anforderungen an die Datenaktualität, Updatemechanismen, Synchronisationsmechanismen und vieles mehr spielen eine entscheidende Rolle. Darüber hinaus hängen die Kosten für die Entwicklung eines mobilen Systems stark von dem gewählten Architekturtyp ab.

Weil diese Sachverhalte von besonderer Bedeutung im Entscheidungsprozess von Softwarearchitekten und IT-Projektmanagern sind, werden in diesem Beitrag einige der häufigsten Typen mobiler Softwarearchitekturen erläutert und deren Vor- und Nachteile besprochen. Darüber hinaus werden einige typische geschäftliche Anforderungen an mobile Systeme identifiziert und ein Auswahlschema erläutert, mit Hilfe dessen es möglich ist, die passende Softwarearchitektur in Abhängigkeit einer gegebenen Menge an geschäftlichen Anforderungen zu bestimmen.

Dieser Beitrag ist wie folgt aufgebaut: Abschnitt 2 gibt einen Überblick über die relevante Literatur. In Abschnitt 3 werden die vier wesentlichen Typen von Softwarearchitekturen für mobile Systeme eingeführt, ihr Aufbau wird mit Hilfe von Komponentendiagrammen erläutert. Die wesentlichen Merkmale der Architekturen sowie ihre Vor- und Nachteile werden erläutert. In Abschnitt 4 werden typische Anforderungen an mobile Systeme aus mobilen Geschäftsprozessen heraus identifiziert und deren Bedeutung für die benannten Softwarearchitekturen beschrieben. Abschnitt 5 gibt eine Zusammenfassung.

2 Literaturüberblick

Die Veränderungen, die sich für die Disziplin der Softwareentwicklung im Umfeld mobiler Systeme ergeben, werden in [RPM00] besprochen. Die Autoren kommen dabei zur wichtigen Aussage, dass ”mobility represents a total meltdown of all stability assumptions [...] associated with distributed computing”. Es wird ein ausgezeichneter Überblick über die Softwareentwicklung für mobile Systeme gegeben, wobei insbesondere auf existierende Probleme wie Modellierung, Algorithmen, Anwendungen und Middleware eingegangen wird. Der vorliegende Beitrag adressiert einige der genannten Problemstellungen.

In [SC03] wird ein Architekturmodell vorgestellt, mit dem wichtige Komponenten mobiler Agenten beschrieben werden können. Die Dialoggestaltung für mobile Systeme ist Gegenstand in [BDN+05]. Die Autoren beschreiben ein Plattform, mit der das Rapid Prototyping einer mulitkanalfähigen, multimodalfähigen und kontextsensitiven Anwendung unterstützt werden kann. Es wird gezeigt, wie diese Plattform genutzt wurde, um ein Touristen-Informationssystem zu entwickeln.

Eine ganze Reihe von Arbeiten beschäftigt sich mit Softwarearchitekturen und anderen technischen Aspekten mobiler Systeme. Ein Beispiel dafür ist [DG03], die eine 3-Schichten-Architektur für verteilte und mobile Systeme präsentieren. [Tsa03] zeigt einen Ansatz für die Modellierung und Performanceanalyse mobiler Multimediasysteme, die mit Hilfe von stochastischen Petrinetzen durchgeführt wird. Der Autor fokussiert auf die Überprüfung der optimalen Performance, die unter Einhaltung bestimmter Vorgaben zum Quality-of-Service in Abhängigkeit gegebener Designparameter erreichbar ist. In [ITLS04] wird ein Ansatz zur Modellierung von Softwarearchitekturen für mobile verteilte Systeme vorgestellt. Diese Modellierungsmethode zielt auf die Verifikation der Korrektheit von funktionalen und nicht-funktionalen Eigenschaften des mobilen Systems.

Die Autoren in [AYBG02] berichten über ein Projekt, in dem eine Systemarchitektur entwickelt wurde, welche die Implementierung von adaptiven mobilen Anwendungen verein-

fachen soll. Dabei wird die zeitliche, räumliche und persönliche Mobilität berücksichtigt. Ein Vergleich verschiedener Architekturen für mobile Systeme wird in [KLH00] gegeben. Es werden Client-Server-Modelle, agentenbasierte Client-Server-Modelle und Modelle für mobile Agenten berücksichtigt. In [vTJ05] wird diskutiert, wie der Ansatz serviceorientierter Architekturen bei der Entwicklung mobiler Systeme berücksichtigt werden kann. Es wird gezeigt, wie mobile Services miteinander kombiniert werden und wie die Bereitstellung dieser Services durch die Systemarchitektur realisiert wird.

Die Ergebnisse aus [BGHS05] sind von besonderer Bedeutung für den vorliegenden Beitrag. Die Autoren geben, basierend auf einer Analyse von Nutzer- und Gerätemobilität, einen Überblick über verschiedene Typen von Softwarearchitekturen für mobile Systeme. Typische Anforderungen an mobile Systeme, die sich aus den Besonderheiten mobiler Geschäftsprozesse ergeben, werden z.B. in [GS04] vorgestellt.

3 Softwarearchitekturen für mobile Systeme

Bei der Analyse des Kommunikationsverhaltens eines mobilen Systems ist deren Architektur von besonderer Bedeutung. In Anlehnung an [BGHS05] können hauptsächlich vier verschieden Architekturtypen unterschieden werden. Bei einer Always-Online-Architektur kommuniziert die Anwendung ausschließlich online mit einem zentralen Server. Diese Architektur ist typisch für webbasierte Systeme. Eine Hybrid-Architektur kann die Vorteile von Offline- und Online-Architekturen zum Teil vereinen: Wenn ein Mobilfunknetzwerk verfügbar ist, kann die Kommunikation mit dem zentralen Server online erfolgen. Falls kein Netzwerk verfügbar ist, kann die mobile Anwendung auch offline genutzt und später mit dem zentralen Server synchronisiert werden. Weiterhin kann eine Offline-Architektur zum Einsatz kommen, bei der ein mobiler Anwender eine gelegentliche Synchronisation mit einem zentralen Server durchführt. Eine vollständige Offline-Architektur kann genutzt werden, wenn keine Kommunikation über ein Netzwerk stattfinden soll. Da die meisten Vorteile durch den Einsatz von IT-Systemen in mobilen Umgebungen aber gerade erst durch die Möglichkeit der Vernetzung entstehen, wird diese Architektur nicht weiter berücksichtigt.

Im Folgenden werden mit Hilfe einer komponentenbasierten Sicht auf diese Architekturen deren wesentliche Eigenschaften erläutert. Der Fokus liegt dabei auf der Beschreibung der wichtigsten Komponenten sowie einer Darstellung ihrer Verteilung. Diese Beschreibungen sind somit nicht vollständig, sondern müssen im Einzelfall entsprechend den konkreten Anforderungen erweitert werden. Es werden darüber hinaus keine Angaben zur technischen Realisierung (Einsatz von Middlewareprodukten u.ä.) gemacht.

3.1 Webbasierte Always-Online-Architektur

Innerhalb dieser Architektur (s. Abbildung 1) arbeitet der Client permanent always-online über ein Mobilfunknetzwerk. Die Präsentationsschicht ist vollständig auf dem Client mit

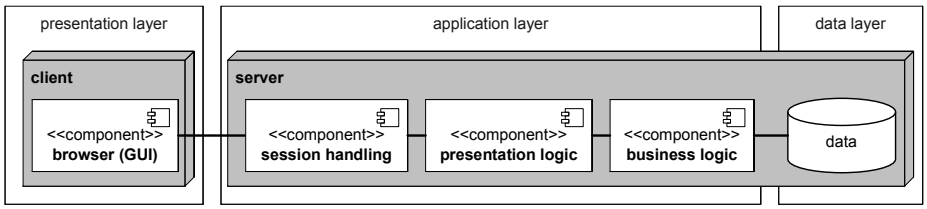

Abbildung 1: Webbasierte Always-Online-Architektur

Hilfe einer Browser-Komponente (GUI) realisiert. Die Anwendungsschicht ist vollständig auf dem Server realisiert und enthält eine Session-Handling-Komponente sowie Komponenten für die Präsentations- und Geschäftslogik. Die Datenschicht enthält die benötigten Datenbanken und ist ebenfalls vollständig auf dem Server realisiert. Der größte Vorteil dieser Architektur ist es, dass nur ein Browser auf dem Client benötigt wird. Somit erlaubt die Systemarchitektur die Integration einer großen Bandbreiten von Client-Systemen, unabhängig von Betriebssystemen oder anderen Client-Eigenschaften. Weiterhin sind keine Update- oder Synchronisationsmechanismen nötig, da die Anwendungs- und die Datenschicht komplett serverseitig realisiert sind. Alle Clients arbeiten auf derselben zentralen Datenbank und nutzen hochaktuelle Daten sowie die aktuellen Präsentations- und Geschäftslogiken. Der Aufwand für die Administration solcher Systeme ist vergleichsweise gering. Der größte Nachteil einer solchen Lösung ist es, dass immer ein Mobilfunknetzwerk zur Verfügung stehen muss, da ansonsten der Client nicht arbeitsfähig ist. Ich einigen Gebieten kann es jedoch vorkommen, dass eine Abdeckung mit einem solchen Netzwerk nicht gegeben ist. Darüber hinaus stellen heutige Mobilfunknetzwerke, verglichen mit LAN-Infrastrukturen, nur eine geringe Bandbreiten für die Datenübertragung zur Verfügung. Dies führt häufig zu einer unbefriedigenden Performance des Clients. Weiterhin kann beim Design der Benutzerschnittstelle von browserbasierten Systemen nur auf die beschränkten Darstellungsmöglichkeiten von HTML zurückgegriffen werden. Darüber hinaus kann die gleichzeitige Verbindung vieler Clients mit einem zentralen Server hohe Anforderungen an dessen Performance und Verfügbarkeit stellen.

3.2 Rich-Client Always-Online-Architektur

Mit dieser Architektur (s. Abbildung 2) wird ein Nachteil webbasierter Always-Online-Architekturen vermieden: Durch die Verwendung eines Rich-Clients müssen beim Design der Benutzerschnittstelle nicht mehr die durch HTML gegebenen Einschränkungen berücksichtigt werden, da bei dieser Art von Clients der volle Umfang an Interaktionselementen zur Verfügung steht. Darüber hinaus können Teile der Präsentationslogik auf den Client verschoben werden, um damit Performance- und Kostenvorteile durch eine Reduzierung des übertragenen Datenvolumens zu erzielen. Die Präsentationsschicht ist vollständig auf dem Client realisiert, der die GUI und die Präsentationslogik enthält. Zusätzlich enthält der Client einige Elemente aus der Anwendungsschicht, zum Beispiel eine Updatekomponente für die Präsentationslogik und eine Sessionhandling-Komponente.

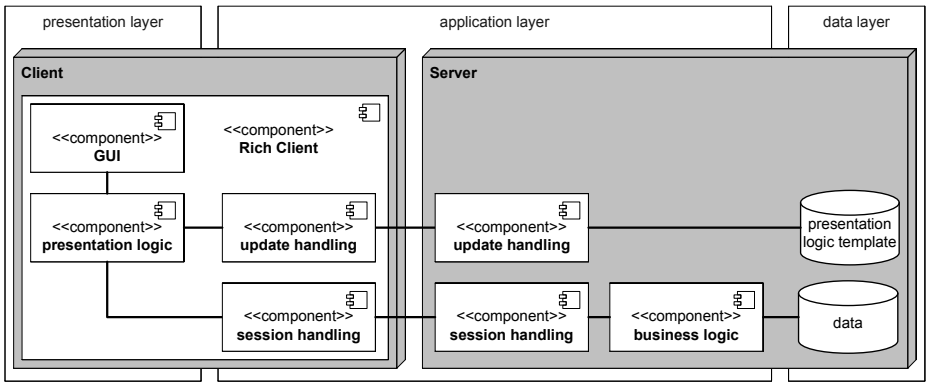

Abbildung 2: Rich-Client Always-Online-Architektur

Beide Komponenten haben jedoch noch ein Gegenstück auf dem Server, auf dem sich auch die Geschäftslogik (Anwendungsschicht) befindet. Die Datenschicht ist vollständig auf dem Server realisiert.

Die hauptsächlichen Vorteile dieser Architektur sind die Verwendung eines Thin-Clients mit den Möglichkeiten zur Oberflächengestaltung und das reduzierte Datenvolumen, das transportiert werden muss. Weiterhin können alle Clients auf derselben zentralen Datenbank arbeiten und aktuelle Daten sowie die jeweils neueste Geschäfts- und Präsentationslogik nutzen. Der Administrationsaufwand für solche Systeme ist vergleichsweise gering. Weiterhin kann durch die Verwendung von Rich-Clients eine teilweise asynchrone Kommunikation realisiert werden, die in Verbindung mit der Sessionhandling-Komponente auch kurzeitige Offlinezeiten überbrücken kann. Der Nachteil solcher Lösungen ist die Notwendigkeit zur Nutzung eines Mobilfunknetzwerks, da der Client sonst nicht funktionsfähig ist. Eine solche Situation kann in bestimmten Gebieten auftreten. Weiterhin sind Komponenten für die Session- und Updatemechanismen nötig. Darüber hinaus entstehen auch die bereits bei der Always-Online-Architektur geschilderten Probleme (Bandbreite des Netzes, Performance des Clients, Performance des Servers).

3.3 Rich-Client Hybrid-Architektur

Die beiden zuvor beschriebenen Architekturen haben einen wesentlichen Nachteil: Falls kein Mobilfunknetzwerk verfügbar ist, steht dem mobilen Arbeiter die Anwendung nicht zur Verfügung. Die Rich-Client Hybrid-Architektur vermeidet dieses Problem (s. Abbildung 3). Das Ziel dieser Architektur ist es, einen Client für das vorrangige Arbeiten online zur Verfügung stellen, aber es ebenso zu ermöglichen, den Client im offline-Fall weiter zu benutzen. Der Client besteht aus einer GUI und einer Komponente für die Präsentationslogik. Auf dem Client befinden sich zusätzlich Komponenten für die Geschäftslogik, für das Sessionhandling, das Updatehandling sowie für Synchronisationsaufgaben (Anwendungsschicht). Weiterhin wird eine Datenbank (Datenschicht) benötigt. Die

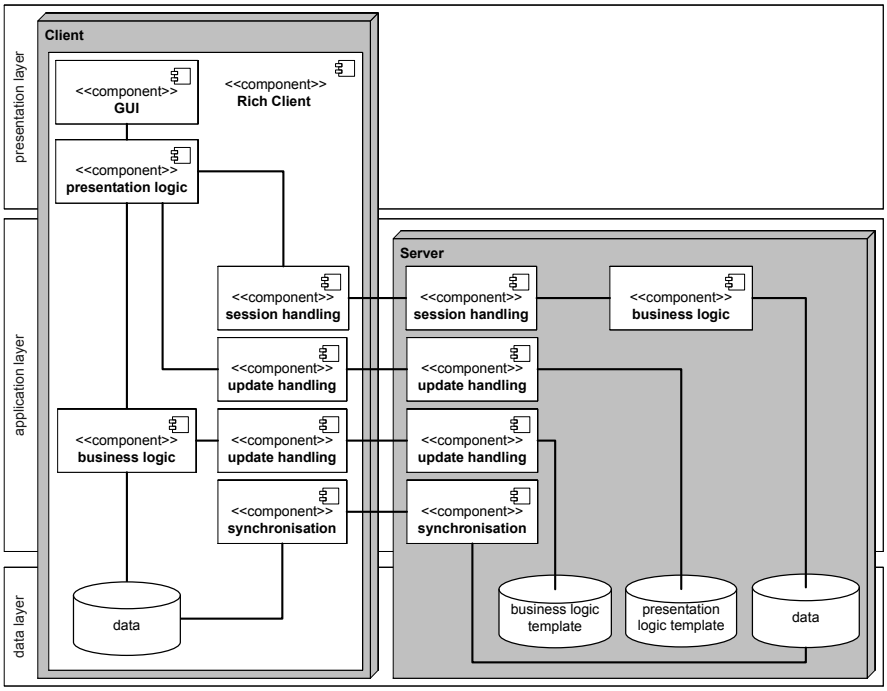

Abbildung 3: Rich-Client Hybrid-Architektur

Komponenten für die Geschäftslogik, für das Sessionhandling, das Updatehandling sowie für Synchronisationsaufgaben werden ebenso auf dem Server benötigt (Anwendungsschicht). Die serverseitige Datenschicht besteht aus den Vorlagen für die Geschäfts- und Präsentationslogik sowie ggf. anderen Datenbanken.

Im Normalfall arbeitet die Anwendung always-online und nutzt dabei die Geschäftslogik und die Datenbanken des Servers (wie im Rich-Client Always-Online-Szenario). Im Falle des Verbindungsabbruchs ist ein Weiterarbeiten auf dem Client möglich, da dieser nun seine eigenen Komponenten anstatt der severseitigen verwendet. Damit bei einer späteren Rückkehr der Netzwerkverbindung Client und Server auf den gleichen Datenstand gebracht werden können, ist ein Synchronisationsmechanismus nötig. Der größte Vorteil dieser Architektur ist die Möglichkeit, always-online zu arbeiten, aber dennoch, im Falle des Verlusts der Netzwerkverbindung, offline arbeiten zu können. Durch die Verwendung eines Rich-Clients stehen alle Möglichkeiten zur Gestaltung der Benutzerschnittstelle zur Verfügung, das übertragene Datenvolumen kann reduziert werden. Es gelten weiterhin die bereits genannten Vorteile (zentrale Datenhaltung, Zugriff auf zentrale Geschäfts- und Präsentationslogik im Normalfall, Möglichkeiten zur asynchronen Kommunikation etc.). Der größte Nachteil dieser Architektur wird durch ihren größten Vorteil erkauft: Es ist ein Synchronisationsmechanismus mit Konfliktlösungsstrategie nötig, um den offline erfassten und veränderten Datenbestand mit dem Server zu synchronisieren. Es gelten weiterhin die bereits oben genannten Nachteile (geringe Bandbreiten, Netzabdeckung, Performance

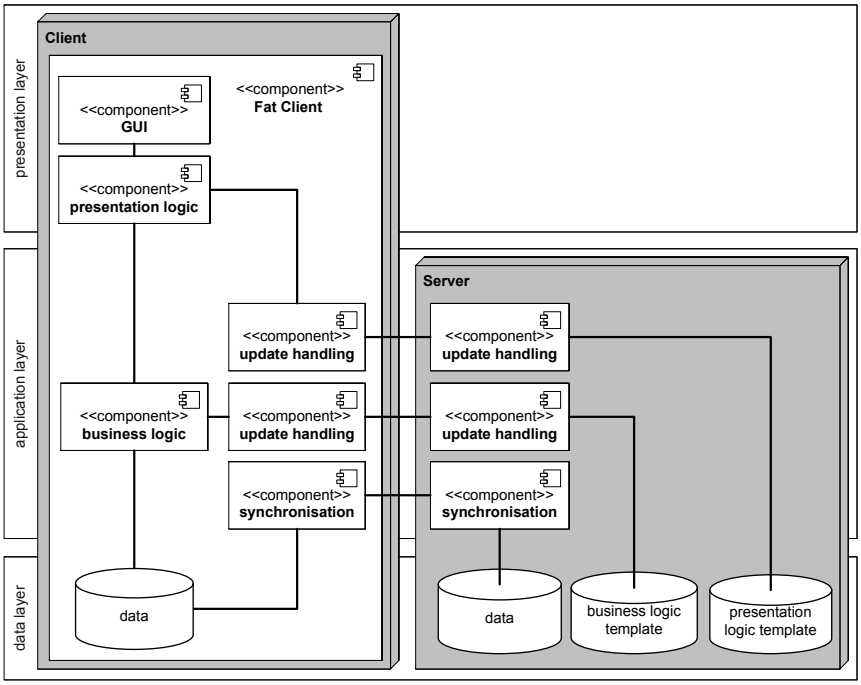

Abbildung 4: Fat-Client Offline-Architektur

des Clients und des Servers). Aufgrund der Komplexität des Clients ist der Administrationsaufwand als sehr hoch einzuschätzen.

3.4 Fat-Client Offline-Architektur

Die Fat-Client Offline-Architektur weist große Ähnlichkeiten zu der zuvor beschriebenen Rich-Client Hybrid-Architektur auf. Es wird jedoch nicht die Möglichkeit geboten, unter Nutzung eines Mobilfunknetzes online zu arbeiten (s. Abbildung 4). Der mobile Arbeiter verwendet den Client stattdessen immer offline, so dass sich dort die gesamte Anwendung befindet. Die erfassten oder veränderten Daten, die auf dem Client gespeichert sind, werden gelegentlich an festen Synchronisationspunkten (Büro, Homeoffice) mit dem Server synchronisiert. Auf diese Weise muss auch das Update für die Komponenten für Geschäfts- und Präsentationslogik durchgeführt werden.

Der hauptsächliche Vorteil dieser Architektur ist die Unabhängigkeit von einem Netzwerk. Die Nutzung der Anwendung ist daher ist fast allen denkbaren Umweltsituationen möglich. Damit entstehen auch keine Performanceprobleme oder Kosten für die Datenübertragung am POS. Ein weiterer Vorteil ist die Möglichkeit zur uneingeschränkten Gestaltung der Benutzeroberfläche des Clients. Der größte Nachteil dieser Architektur ist die Notwendigkeit zur regelmäßigen Weiterentwicklung und Verbreitung der Komponen-

ten für das Updatehandling und die Synchronisation. Die Synchronisationsintervalle liegen vollständig im Entscheidungsbereich des Anwenders, so dass eine festgelegte Aktualität der Daten und Komponenten nicht sichergestellt werden kann.

3.5 Vergleich der Architekturen

Die webbasierte Always-Online-Architektur ist relativ klein und hat wenige Komponenten. Die Entwicklung und Administration eines solchen Systems dürfte, im Vergleich zu den anderen Architekturen, deutlich geringere Aufwände verursachen. Jedoch erkauft man sich mit dieser Architektur auch einige Probleme hinsichtlich Konnektivität und Usability des Systems. Die anderen drei Architekturen zielen unterschiedlich stark auf die Vermeidung dieser Probleme. Doch je stärker man versucht, diese Konnektivitäts- und Usabilityprobleme zu lösen, wächst der Entwicklungs- und Adminstrationsaufwand für ein solches System sehr stark an. Eine der Hauptaufgaben für IT-Projektmanager und Softwarearchitekten ist es deshalb, zu entscheiden, wie viel Aufwand für die Entwicklung und die Administration solcher Systeme vertretbar ist, gemessen an den Anforderungen aus den betroffenen Geschäftsprozessen. Der folgende Abschnitt widmet sich deshalb typischen geschäftlichen Anforderungen an mobile Systeme und erläutert, wie eine passende Softwarearchitektur abgeleitet werden kann.

4 Typische Anforderungen in mobilen Prozessen und deren architekturelle Implikationen

Im Folgenden werden typische Anforderungen an mobile Anwendungen beschrieben und hinsichtlich ihrer Bedeutung für die zu Grunde liegende Softwarearchitektur bewertet. Diese Anforderungen sowie deren konkrete Ausprägungen wurden in einem Forschungsprojekt in Zusammenarbeit mit der Inverso GmbH [inv] entwickelt. Die Ergebnisse sind in Tabelle 1 aufgelistet. Wenn eine Architektur dazu geeignet ist, eine Anforderung vollständig zu erfüllen, ist sie mit ‘+’ bewertet. Wenn die Architektur die Anforderung nur mit Einschränkungen erfüllen kann, ist sie mit ‘0’ markiert. Wird die Architektur hinsichtlich einer Anforderung als unzureichend eingestuft, erfolgt eine Markierung mit ‘–’.

4.1 Point of Service (POS)

Die örtlichen Gegebenheiten am POS sind von besonderer Relevanz für die Auswahl einer passenden Architektur. Wenn die Anwendung häufig an festgelegten Orten wie dem Homeoffice oder im Büro verwendet wird, sind alle Architekturen gut geeignet. Wenn die Anwendung unterwegs vor allem in städtischen Umgebungen genutzt wird, sind Mobilfunknetzwerke vermutlich meistens vorhanden. In diesem Fall wäre eine webbasierte oder eine Rich-Client Always-Online-Architektur in den meisten Fällen gut geeignet. Die an-

	Always-Online Webbasiert	Always-Online Rich-Client	Hybrid Rich-Client	Offline Fat-Client
Point of Service				
Büro	+	+	+	+
Stadtgebiet	0	0	+	+
ländliche Gegend	–	–	+	+
Datenaktualität				
Echtzeitdaten gewünscht	+	+	–	–
relativ hohe Aktualität nötig	+	+	+	–
geringe Aktualität akzeptiert	+	+	+	+
Redundanz des Quellcodes				
hohe Redundanz akzeptiert	+	+	+	+
geringe Redundanz akzeptiert	+	+	–	–
keine Redundanz gewünscht	+	–	–	–
Softwareverteilung				
Verteilung offline akzeptiert	+	+	+	+
Verteilung online akzeptiert	+	+	+	0
keine Verteilung gewünscht	+	–	–	–
Oberflächendesign und Interaktionstechniken				
sollen umfassend sein	–	+	+	+
kann HTML sein	+	+	+	+
Sicherheit				
dezentrale Organisation	+	+	+	+
zentrale Organisation	+	+	0	–

Tabelle 1: Anforderungen und deren architekturelle Implikationen

deren beiden Architekturen sind natürlich in allen Fällen gut einsetzbar. Wenn die mobile Anwendung vor allem in ländlichen Umgebungen eingesetzt wird, ist die Verfügbarkeit eines Mobilfunknetzwerks vermutlich nicht immer gegeben. In diesem Fall sind nur die Rich-Client Hybrid-Architektur oder die Fat-Client Offline-Architektur einsetzbar.

4.2 Datenaktualität

Die Anforderungen an die Datenaktualität haben ebenfalls großen Einfluss auf die Wahl der passenden Systemarchitektur. Wenn der mobile Arbeiter immer hochaktuelle oder Echtzeitdaten benötigt, kommt nur die Always-Online-Architektur in Frage. Die Rich-

Client Hybrid-Architektur kann verwendet werden, wenn die Anforderungen an die Datenaktualität weniger hart sind (wenn z.B. Vortagesaktualität ausreicht). Die Fat-Client Offline-Architektur ist nur einsetzbar, wenn die Anforderungen an die benötigte Datenaktualität nicht kritisch sind (einige Tage oder Wochen reichen aus).

4.3 Redundanz des Quellcodes

Die Architektur eines mobilen Systems beeinflusst auch die Redundanz des Quellcodes der Anwendung. Wenn die Geschäftslogik einer Anwendung im mehreren Kontexten verwendet wird (z.B. ein Berechnungsmodul, das sowohl client- als auch serverseitig eingesetzt wird), muss das Deployment und ggf. sogar die Entwicklung oft mehrfach erfolgen. Grund dafür sind unterschiedliche Systemanforderungen (Betriebssysteme, Rechenleistung, Funktionsumfang der Komponente etc.). Falls eine solche Redundanz vermieden werden soll (single source, single instance), kommt nur die webbasierte Always-Online-Architektur in Frage. Wenn eine geringe Redundanz akzeptabel ist (single source, multiple instances), kommt vor allem die Rich-Client Always-Online-Architektur in Frage. Wenn der Redundanzaspekt keine Rolle spielt, können sowohl die Rich-Client Hybrid-Architektur als auch die Fat-Client Offline-Architektur eingesetzt werden.

4.4 Softwareverteilung

Die Verteilung neuer Releases der mobilen Anwendung verursacht oft einen großen Aufwand. Wenn dieser vermieden werden soll, ist die Always-Online-Architektur auszuwählen, da mit dieser Architektur der Updateprozess auf die Severinstanz beschränkt werden kann. Wenn ein Online-Updateprozess der Clients akzeptabel ist, können die beiden Rich-Client-Architekturen zum Einsatz kommen. Falls die Wahl auf die Fat-Client Offline-Architektur fallen sollte, müssen hohe Kosten und organisatorische Aufwände einkalkuliert werden, da große Datenmengen transportiert werden müssen und dies oft nur über offline-Wege (z.B. via CD, DVD) möglich ist.

4.5 Gestaltung der Benutzeroberfläche und Interaktionstechniken

Sowohl die beiden Rich-Client-Architekturen als auch die Fat-Client Offline-Architektur bieten alle herkömmlichen Möglichkeiten zur Gestaltung der Benutzeroberfläche. Wenn die Anwendung nicht zu komplex ist und nur einfache Interaktionen vorgesehen sind (abbildbar in HTML), kommt auch der Einsatz einer webbasierten Architektur in Frage.

4.6 Sicherheit

Mit Hilfe von mobilen Anwendungen werden häufig vertrauliche Daten verarbeitet und übertragen, die eine Reihe von Sicherheitsvorkehrungen nötig machen. Aus Sicht des Administrators eines solchen Systems ist ein einfaches, gut steuerbares, zentrales Sicherheitsmanagement wünschenswert. Mit beiden Always-Online-Architekturen kann dies erreicht werden. Für die beiden anderen Architekturen werden darüber hinaus noch zum Teil aufwändige Sicherheitsmaßnahmen für den Client nötig.

5 Zusammenfassung

In diesem Beitrag wurden die vier typischen Softwarearchitekturen für mobile Systeme vorgestellt. Die Entscheidung, welche dieser Architekturen in einer konkreten Situation angewendet werden sollte, hängt einerseits von den Anforderungen aus dem betroffenen Geschäftsprozess, andererseits aber auch von den notwendigen Aufwänden für die Entwicklung und Adminstration des Systems ab. Zwischen beiden Aspekten gibt es starke, wechselseitige Abhängigkeiten. Aus den vorgestellten Ergebnissen wird außerdem deutlich, dass mit steigenden fachlichen Anforderungen, besonders hinsichtlich der Konnektivität und Usability, die Aufwände für die Entwicklung und Administration des Systems stark ansteigen. Unter Berücksichtigung dieser Ergebnisse kann keine generelle Empfehlung zur Verwendung einer bestimmten Systemarchitektur gegeben werden. Es ist vielmehr nötig, jede einzelne fachliche Anforderung separat zu bewerten und die jeweils beste Architektur zu bestimmen. Im Anschluss muss dann in einem Prozess der Abwägung ein Kompromiss gefunden werden. Die in diesem Beitrag dargestellten Ergebnisse können helfen, diesen Prozess zu unterstützen.

6 Danksagung

Der Lehrstuhl für Angewandte Telematik / e-Business ist gestiftet von der Deutschen Telekom AG. Die in diesem Beitrag dargestellten Ergebnisse wurde zum Teil in einem Forschungsprojekt in Zusammenarbeit mit der Inverso GmbH [inv] entwickelt.

Literatur

[AYBG02] Iara Augustin, Adenauer Correa Yamin, Jorge Luis Victoria Barbosa und Claudio Fernando Resin Geyer. ISAM, a Software Architecture for Adaptive and Distributed Mobile Applications. In *ISCC '02: Proceedings of the Seventh International Symposium on Computers and Communications (ISCC'02)*, Seite 333, Washington, DC, USA, 2002. IEEE Computer Society.

[BDN+05] Rudi Belotti, Corsin Decurtins, Moira C. Norrie, Beat Signer und Ljiljana Vukelja. Ex-

perimental platform for mobile information systems. In *MobiCom '05: Proceedings of the 11th annual international conference on Mobile computing and networking*, Seiten 258–269, New York, NY, USA, 2005. ACM Press.

[BGHS05] Matthias Book, Volker Gruhn, Malte Hülder und Clemens Schäfer. A Methodology for Deriving the Architectural Implications of Different Degrees of Mobility in Information Systems. In M.Mejri H. Fujita, Hrsg., *New Trends in Software Methodologies, Tools and Techniques*, Seiten 281–292. IOS Press, 2005.

[DG03] Schahram Dustdar und Harald Gall. Architectural concerns in distributed and mobile collaborative systems. *Journal on System Architecture*, 49(10-11):457–473, 2003.

[GK06] Volker Gruhn und André Köhler. Modeling Communication Behaviour of Mobile Applications. In *International Workshop on Exploring Modeling Methods in Systems Analysis and Design (EMMSAD)*, Proceedings of the CAiSE'06 Workshops, Seiten 509–520, Luxembourg, 2006. Presses Universitaires de Namur.

[GKK05] Volker Gruhn, André Köhler und Robert Klawes. Modeling and Analysis of Mobile Service Processes by Example of the Housing Industry. In Wil M.P. van der Aalst, Boualem Benatallah, Fabio Casati und Francisco Curbera, Hrsg., *Business Process Management*, Seiten 1–16. Springer LNCS 3649, 2005.

[GS04] J. Gebauer und M. J. Shaw. Success Factors and Impacts of Mobile Business Applications: Results from a Mobile e-Procurement Study. *International Journal of Electronic Commerce*, 8(3):19–41, 2004.

[inv] http://www.inverso.de.

[ITLS04] Issarny Issarny, Ferda Tartanoglu, Jinshan Liu und Francoise Sailhan. Software Architecture for Mobile Distributed Computing. In *WICSA '04: Proceedings of the Fourth Working IEEE/IFIP Conference on Software Architecture (WICSA'04)*, Seite 201, Washington, DC, USA, 2004. IEEE Computer Society.

[KLH00] Zhigang Kan, Jun Luo und Jianping Hu. The Design of a Software Technology Architecture for Mobile Computing. In *HPC '00: Proceedings of the The Fourth International Conference on High-Performance Computing in the Asia-Pacific Region-Volume 2*, Seite 762, Washington, DC, USA, 2000. IEEE Computer Society.

[NSS05] Fiona Fui-Hoon Nah, Keng Siau und Hong Sheng. The value of mobile applications: a utility company study. *Communications of the ACM*, 48(2):85–90, 2005.

[RPM00] Gruia-Catalin Roman, Gian Pietro Picco und Amy L. Murphy. Software engineering for mobility: a roadmap. In *ICSE '00: Proceedings of the Conference on The Future of Software Engineering*, Seiten 241–258, New York, NY, USA, 2000. ACM Press.

[SC03] Marthie Schoeman und Elsabé Cloete. Architectural components for the efficient design of mobile agent systems. In *Proceedings of the SAICSIT '03*, Seiten 48–58. South African Institute for Computer Scientists and Information Technologists, 2003.

[Tsa03] Tony Tsang. Modelling and performance evaluation of mobile multimedia systems using QoS-GSPN. *Wireless Networks*, 9(6):575–584, 2003.

[vTJ05] Do van Thanh und Ivar Jorstad. A Service-Oriented Architecture Framework for Mobile Services. In *AICT-SAPIR-ELETE '05: Proceedings of the Advanced Industrial Conference on Telecommunications/Service Assurance with Partial and Intermittent Resources Conference/E-Learning on Telecommunications Workshop (AICT/SAPIR/ELETE'05)*, Seiten 65–70, Washington, DC, USA, 2005. IEEE Computer Society.