Play.Tools: Ein Software-Framework zur prototypischen Umsetzung kontextsensitiver mobiler Anwendungen als Unterstützung von Innovationsprozessen

Christian Schüller[1], Bernhard Doll[1], Wolfgang Wörndl[2]

[1]UnternehmerTUM GmbH
Lichtenbergstr. 8
85748 Garching
schueller@unternehmertum.de,
doll@unternehmertum.de

[2]Technische Universität München
Institut für Informatik
Boltzmannstr. 3
85748 Garching
woerndl@informatik.tu-muenchen.de

Zusammenfassung: Speziell im Bereich der Entwicklung vermarktungsfähiger und innovativer mobiler Anwendungen werden hohe Anforderungen an Innovations- und Gründerteams gestellt. Dieser Beitrag stellt das Software-Framework Play.Tools vor, welches die prototypische Umsetzung und anschließenden Markttest kontextsensitiver mobiler Anwendungen als Unterstützung von Innovationsprozessen erleichtert. Bausteine des Frameworks sind Basis-Softwarekomponenten für Anwendungen auf mobilen Endgeräten, ein datenbankgestütztes Serversystem für die Bereitstellung von Geschäftslogik und syndizierten Inhalten sowie ein Anwendungsempfehler. Mittels des Frameworks können in kurzer Zeit Prototypen erstellt werden, um gerade in frühen Entwicklungsphasen Feedback beteiligter Stakeholder im Innovationsprozess, insbesondere von Kunden, bezüglich Kundennutzen und Kundenakzeptanz einzuholen. Dabei können aus Produktideen unternehmerische Chancen entwickelt bzw. Markt- und Technologieunsicherheiten besser abgeschätzt werden.

1 Einleitung

Trotz oder gerade wegen dramatischer Innovationssprünge in den letzten Jahren stellt die Entwicklung vermarktungsfähiger und innovativer mobiler Anwendungen eine große Herausforderung an Softwareentwickler dar. Die Gründe dafür sind unter anderem in unzureichenden Standards, fehlerhaften Programmierschnittstellen und extrem kurzen Produktlebenszyklen bei mobilen Endgeräten und der dazu passenden Software zu finden [Sch01, HLO05, Eix06]. Speziell der Zugriff auf systemnahe Funktionen mobiler Endgeräte, die Positionsbestimmung und die Entwicklung intuitiv zu bedienender Benutzerschnittstellen sind noch immer sehr aufwendig, weil bis dato entsprechende Software-Frameworks fehlen, die diese Funktionalitäten geeignet kapseln.

Dies betrifft insbesondere die Entwicklung von Context-Aware-Applications, also Anwendungen, die sich an die jeweilige Umgebung des Benutzers anpassen [Dey01, KL03, Pru06]. Hier sind durch die hohe Interdisziplinarität des Faches und die Vielzahl der in der Mobilfunkindustrie notwendigen Prozessbeteiligten entlang der gesamten Wert-

schöpfungskette für Softwareentwickler besonderes große Herausforderungen zu erwarten. Ein weiteres Problem besteht nach der Entwicklung in der Bereitstellung der Anwendung über geeignete Vertriebskanäle. Aufgrund einer Vielzahl an Softwareportalen mit einer nicht überschaubaren Vielfalt an angebotenen Programmen für mobile Endgeräte ist es für potenzielle Kunden schwierig, die für ihn in seinem Kontext interessante Anwendung in den vorhandenen Angeboten zu finden.

Gerade bei Innovationsvorhaben ist neben den erwähnten technologischen Schwierigkeiten auch mit Marktunsicherheiten zu rechnen. Die neuartige mobile Anwendung muss so entwickelt werden, dass für den Kunden ein erkennbarer Mehrwert entsteht, den er bereit ist, mit einem angemessenen Preis zu bezahlen. Sowohl für innerbetriebliche Innovationsteams als auch für Gründerteams ist es daher besonders wichtig, Bedürfnisinformationen, also Informationen über die impliziten wie expliziten Wünsche und Präferenzen von Kunden [Tho03, Ern01] aber auch anderen Prozessbeteiligten, in die Entwicklung einer mobilen Anwendung einzubinden. Eine Möglichkeit diese Bedürfnisinformationen in frühen Phasen des Innovationsprozesses zu ermitteln, ist die Entwicklung von Prototypen, um so greifbare wie nicht greifbare Aspekte des Mehrwerts und des späteren Nutzergefühls der mobilen Anwendung kommunizieren und von Kunden bewerten lassen zu können.

Zur Überwindung dieser inhärenten Technologie- und Marktunsicherheiten bei der Entwicklung kontextsensitiver mobiler Anwendungen, wurde bei der UnternehmerTUM GmbH, dem Zentrum für Unternehmertum an der TU München, das Software-Framework „Play.Tools“ entwickelt. Das Framework soll bei Innovationsvorhaben die prototypische Umsetzung von kontextsensitiven mobilen Anwendungen erleichtern, um Umsetzbarkeit und Kundenakzeptanz in frühen Phasen des Innovationsprozesses zu überprüfen und die Anwendung iterativ auf die technischen Anforderungen und Marktbedürfnisse besser anpassen zu können. Das Framework wurde in einer ersten Iteration implementiert und bereits bei einigen Innovationsprojekten zur Entwicklung kontextsensitiver mobiler Anwendungen eingesetzt. Die Erfahrung daraus fließt derzeit in eine verbesserte Version des Frameworks ein.

In diesem Artikel soll im weiteren Verlauf auf eine detaillierte Beschreibung der Motivation für die Entwicklung des Software-Frameworks eingegangen werden. Dann folgt in Kapitel 3 eine Beschreibung des Frameworks, wobei wir zunächst einen Überblick über die Architektur geben und dann die einzelnen Softwarebausteine näher vorstellen werden. In Kapitel 4 werden einige Beispielanwendungen beschrieben, die bereits entwickelt wurden und die Anwendbarkeit und Sinnhaftigkeit des Frameworks demonstrieren sollen. In Kapitel 5 und 6 werden abschließend verwandte Arbeiten diskutiert und eine kurze Zusammenfassung mit einem Ausblick gegeben.

2 Motivation für die Entwicklung von Play.Tools

Das Ziel des Frameworks Play.Tools ist es, die Komplexität bei der prototypischen Umsetzung und anschließender Markttests kontextsensitiver mobiler Anwendung als Unterstützung von Innovationsprozessen zu reduzieren. Dabei werden in einer modularen

Architektur die wesentlichen generischen Komponenten einer kontextsensitiven mobilen Anwendung (Benutzeroberfläche, Ortsbestimmung, Bereitstellung von syndizierten Geoinformationen, Persistenz usw.) gekapselt und den Beteiligten des Innovationsvorhabens in einem übersichtlichen Software-Framework zur Verfügung gestellt. Die Beteiligten des Innovationsvorhaben müssen sich daher nicht um technische Details der generischen Funktionsvielfalt kontextsensitiver mobiler Anwendungen kümmern, sondern können sich ganz auf die Entwicklung der Softwarebestandteile konzentrieren, die den Mehrwert für den Kunden ausmachen und letztlich entscheidend für den späteren Vermarktungserfolg sind [CM05]. Darüber hinaus bietet das System die Möglichkeit, über ein hybrides Recommender-System mobile Anwendungen unter Marktbedingungen auf Kundennutzen und Kundenakzeptanz zu testen. Dabei ist zu berücksichtigen, dass die Aufwände für Einarbeitung und Nutzung eines generischen Frameworks nicht die Zeit- und Qualitätsvorteile der Vorentwicklungen sowie die bekannten leistungssteigernden Effekte von Software-Prototyping [Boe84] aufwiegen. Der Aufbau des Frameworks wird ausführlich in Kapitel 3 beschrieben.

Die Entwicklung von Prototypen, verstanden als Annäherung einer geplanten Software-Anwendung entlang einer oder mehrere Interessensdimensionen [UE2004], spielen bei der Entwicklung vermarktungsfähiger Softwareanwendungen eine entscheidende Rolle: Wie bereits im einführenden Kapitel beschrieben, können Prototypen helfen, relevante Bedürfnisinformationen in frühen Phasen des Innovationsprozesses zu ermitteln, um so Anwendungen mit einem echten Mehrwert für Kunden entwickeln zu können [Dey01, KL03]. Dabei ist besonders wichtig, dass der Prototyp den besonderen Mehrwert und das Nutzergefühl der späteren mobilen Anwendung auf geeignete Weise zum Ausdruck bringt, um möglichst valides Feedback zu genau diesen Interessensdimensionen zu erhalten. Denn diese Dimensionen sind es, die bei der Entwicklung neuartiger mobiler Anwendungen meist das größte Unsicherheitspotential tragen. Dabei können zum einen ausgewählte Funktionsbereiche durch alle Ebenen der Anwendung hindurch implementiert (vertikales Prototyping) oder ein spezifisches Ende der Anwendung möglichst vollständig realisiert werden (horizontales Prototyping).

Entscheidend ist, dass zu Beginn des Innovationsvorhabens die umzusetzenden Teile des Prototyps so ausgewählt werden, dass maßgebliche Fragen bezüglich der bereits erwähnten Technologie- und Marktunsicherheiten im Laufe des Innovationsprozesses beantwortet werden können. Dabei können nicht nur explizite Präferenzen und Wünsche von Kunden abgefragt, sondern auch unbewusste Bedürfnisinformationen durch prozessbegleitende Kundenbefragungen und Kundenbeobachtungen während der Nutzung des Prototyps ermittelt und bewertet werden. Die Vorstellung ist also, dass durch die Umsetzung einer innovativen und damit für Kunden oft schwer vorstellbaren Anwendungsidee in einen Prototypen für den Kunden unbewusst ablaufende Bewertungs- und Entscheidungsprozesse bewusst gemacht werden können, um somit ganzheitlich die Produktidee bewerten zu können. Durch diese Rückkopplungsprozesse kann der Prototyp in mehreren Iterationen zu einer vermarktungsfähigen Anwendung weiterentwickelt und das Produktkonzept technologie- und marktbezogen abgesichert werden.

Das Framework ist in einen iterativen Entwicklungsprozess in Anlehnung an das Spiralmodell der Softwareentwicklung nach Böhm [Boe86] eingebunden, der speziell für

die Realisierung innovativer Prototypen entwickelt wurde und aus folgenden fünf Phasen besteht:

1. Analyze: Entwicklung und Verfeinerung der Geschäftsidee, Kundenbeobachtung und Technologierecherche
2. Design: Ermittlung der Kernfunktionen des Prototypen, Planung der Umsetzung und Entwurf des Software-Designs der Anwendung
3. Build: Umsetzung und fachlicher Test des Prototypen (mit Hilfe des Play.Tools Frameworks)
4. Play: Akzeptanztest des Prototypen durch Einholung von Kundenfeedback
5. Review: Auswertung des Akzeptanztests, Projektreview

Mit Durchlauf einer Iteration des Prozesses wird somit nicht nur ein Prototyp geplant und implementiert, sondern es wird auch Wert auf eine integrierte Überprüfung der Anwendung mit Unterstützung von Kunden gelegt. Es sind meist mehrere Iterationen notwendig, um aus einer Anwendungsidee eine vermarktungsfähige Anwendung zu erarbeiten. Der Fokus in diesem Beitrag liegt auf der Implementierung von Prototypen (Build) und deren Distribution (Play) mit Hilfe des Software-Frameworks Play.Tools.

Ein weiteres Problem für Innovations- und Gründungsteams ist die spätere Bereitstellung kontextsensitiver mobiler Anwendungen für das Einholen von Feedback durch Markttests. Viele kontextsensitive Anwendungen können erst dann ihren ganzen Mehrwert entfalten, wenn sich der Benutzer der Anwendung in einer spezifischen Umgebung bzw. Kontext aufhält. Dafür ist natürlich für die Validität der Markttests wichtig, möglichst viele (auch unbekannte) Nutzer an den Tests möglichst einfach beteiligen zu können, um unterschiedliche Kontextszenarien in den Test einbeziehen zu können. In unserem Projekt wird dazu eine Distributionsplattform aufgebaut, damit interessierte Nutzer leichter relevante Anwendungen finden können. Für die Auswahl geeigneter Anwendungen bietet unser Framework einen Anwendungsempfehler auf Basis eines hybriden Recommender Systems [Bur02] an (siehe Kap. 3.4). Dabei werden Erfahrungen von anderen Kunden berücksichtigt, um Empfehlungen auszusprechen. Beispielsweise kann damit ausgewertet werden, welche Anwendungen Benutzer in einem bestimmten Kontext (z.B. Aufenthalt am Bahnhof) verwendet haben, um diese dann anderen Nutzern in einem vergleichbaren Kontext zur Nutzung empfehlen zu können [Woj06]. Durch die Speicherung und Auswertung des Nutzungs- und Empfehlungsverhaltens in bestimmten Situationen können somit wertvolle Rückschlüsse auf Mehrwert und Akzeptanz bestimmter Anwendungen getroffen werden.

Der inhaltliche Fokus für die Entwicklung von kontextsensitiven mobilen Anwendungen mit dem Software-Framework Play.Tools ist auf den Marktbereichen M-Commerce und „Machine-to-Machine“ (M2M) Kommunikationssystemen gelegt.

3 Das Play.Tools Framework

3.1 Überblick über die Architektur

Für die Prototypenerstellung werden durch das Framework wichtige Basis-Funktionalitäten für eine Auswahl mobiler Endgeräte und ein zentrales Serversystem zur Verfügung gestellt, die die gesamte technische Komplexität speziell im Bereich der Darstellung von Geschäftsdaten, Datenbankzugriffen, Ortungsverfahren und der Kontextermittlung [SO05] kapseln. Weiter ist das System dahingehend ausgerichtet, dass vor allem kleine und „schlanke“ Anwendungen mit einer übersichtlichen Anzahl an Funktionen (sog. „Widgets“) effizient entwickelt werden können, die in ihrer Komplexität überschaubar und selbst auf einfacheren mobilen Endgeräten funktionsfähig sind. Beispiele hierfür sind Positionsbestimmung von Freunden (siehe 4.1), Anzeige von Wetterinformationen, „Virtual Post-it“ etc. Dies senkt bei den Innovations- und Gründerteams die Hemmschwelle, die für den Geschäftserfolg wichtigen Entwicklungen zu verfolgen und lenkt zudem die Aufmerksamkeit verstärkt auf Fragen des Kundennutzens, der Funktionsvielfalt und der Benutzerführung. Das Resultat sind Geschäftsideen mit einem für den Kunden wahrnehmbaren und erprobten Nutzen, einem funktionierenden Prototyp und einem Zeitvorteil bei der Entwicklung des finalen Endsystems.

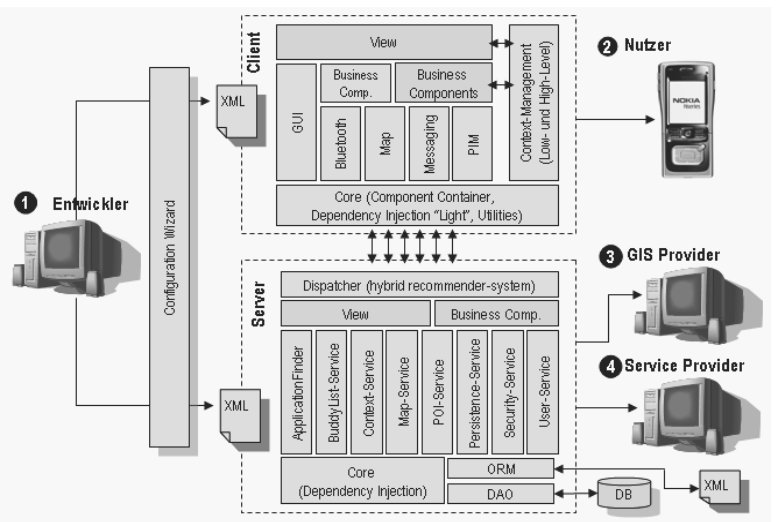

Abbildung 1: Architektur der Entwicklungsplattform

Die Play.Tools, bisher als Version 1.2 herausgegeben, bestehen aus drei zentralen Bausteinen:

(i) Basis-Softwarefunktionalitäten für mobile Endgeräte (Clients)
(ii) Datenbankgestütztes Serversystem
(iii) Distributionsplattform mit Empfehlungssystem für mobile Anwendungen

Die Architektur basiert dabei auf dem Konzept der komponentenbasierten Softwareentwicklung: Der Grundgedanke ist, mobile Anwendungen in wieder verwendbare Komponenten zu unterteilen, um möglichst wenig Code neu programmieren zu müssen. Den Aufbau und die bereitgestellten Komponenten kann man der Abbildung 1 entnehmen. Die grauen Felder stellen die von Play.Tools bereitgestellten Komponenten dar, die bei der Anwendungsentwicklung verwendet werden können. Schwarze Felder sind die von den Entwicklern zu erstellenden Funktionalitäten. Bei den Play.Tools handelt es sich um ein Komponenten-Framework, da in den Play.Tools definiert wird wie diese Komponenten über Schnittstellen miteinander kommunizieren (Interaction-Standard) und in dem Build-Prozess auch das Komponenten-Deployment (Composition-Standard) festgelegt ist.

Der Entwickler (Nummer 1, in Abbildung 1) hat die Möglichkeit über XML-Dateien (Properties) die benötigten Komponenten sowohl für den Client als auch den Server auszuwählen und zu konfigurieren sowie einfache GUIs zu erstellen. Derzeit muss der Entwickler hierfür noch die XML-Dateien per Hand editieren, ab der nächsten Version steht zu diesem Zweck ein auf Drag&Drop basierter Assistent bereit. Neben der Komponenten- und GUI-Konfiguration muss der Entwickler nur noch die Geschäftslogik für den Client und Server erstellen. Dabei kann der Entwickler serverseitig über eine einheitliche Schnittstelle auf externe Angebote von z.B. GIS Providern (3) bzw. Service Providern (4) zugreifen. Der Client wird über den Anwendungsempfehler (Dispatcher) dem potenziellen Kunden (2) angeboten. Im Folgenden werden die drei zentralen Bausteine der Play.Tools, die auch einzeln verwendbar sind, näher beschrieben.

3.2 Basis-Softwarefunktionalitäten (Client)

Die Basis-Softwarefunktionalitäten (vgl. Abbildung 1) bieten dem Entwickler ein klares Schnittstellenkonzept, um vor allem systemnahe Funktionen auf mobilen Endgeräten ansteuern und Benutzeroberflächen effizient aufbauen zu können. Über eine Abstraktionsschicht kann der Anwendungsentwickler bei der Implementierung von mobilen Anwendungen einfach und bequem auf Funktionen der Datenpräsentation (GUI), Netzwerkkommunikation (SMS, HTTP, SOAP), Benutzerauthentifizierung, Kontextermittlung (aus logischen und physikalischen Sensoren)/-verwaltung [Pru06] sowie auf Positionsdaten von Satellitennavigationssystemen (GPS) zugreifen. Von besonderer Bedeutung ist dabei der Einsatz der Markup-Sprache XML zur einfachen Erzeugung von Benutzeroberflächen auf dem mobilen Endgerät und Konfiguration der benötigten Komponenten. Entscheidend ist, dass die Komponenten einen geringen Speicherbedarf („footprint“) haben und auf mehreren mobilen Geräteplattformen lauffähig sind. Der Hauptfokus liegt dabei auf mobilen Betriebssystemen mit Unterstützung der Programmiersprache Java, wie z.B. SymbianOS 7/8, Nokia OS oder die PalmSource-Plattform. Diese Betriebssysteme werden von bekannten Handyherstellern wie z.B. Nokia, Sony Ericsson, Motorola, BenQ-Siemens, oder PalmOne verwendet, womit mehr als 85% der auf dem Markt verfügbaren Smartphones weltweit abgedeckt werden [Gar05].

Die Benutzung der generisch wieder verwendbaren Komponenten nach dem Blackbox-Prinzip zeigt folgendes Beispiel auf. Die Client-Komponente für Bluetooth-Verbindun-

gen (BluetoothComponent) kümmert sich um das Finden von Bluetooth-Geräten und baut zu diesen bei Bedarf eine Verbindung auf. Dazu ist die „Java APIs for Bluetooth Wireless Technology“ [SM05], eine API für diesen Zweck, innerhalb der BluetoothComponent gekapselt. Als Nutzer des Frameworks muss man sich keine Gedanken über diese API machen. Es reicht zu wissen, dass man über die Methode doDiscoverDevice() eine Suche anstößt und man, sobald die Suche beendet wurde, eine Rückmeldung mit Informationen zu den gefundenen Geräten und deren Diensten bekommt. Um den Lebenszyklus und Zugriff auf die einzelnen Komponenten kümmert sich der ComponentContainer. Dieser initialisiert die Komponenten bei Programmstart und gewährt über eine eindeutige Name/Instanz-Relation den Zugriff auf die Komponenten. Außerdem sorgt dieser bei Programmende für ein sauberes Beenden der einzelnen Komponenten. Dies ist notwendig, damit die von den Komponenten gehaltenen Ressourcen wieder freigegeben werden; es müssen z.B. offene Verbindungen geschlossen werden. Die dabei innerhalb einer Komponente ausgeführten Anweisungen können für jede Komponente verschieden sein. Will man über die von einer Komponente ausgelösten Ereignisse informiert werden, kann man sich bei ihr als ComponentListener registrieren (Listener-Pattern). Damit der ComponentContainer weiß, welche Komponenten er beim Start initialisieren soll, gibt es die Properties-Komponente. Diese liest ein XML-Dokument (Properties) ein, in der die benötigten Komponenten mit der dazugehörigen Klasse sowie die gewünschten Einstellungen definiert werden. Zusätzlich können Parameter definiert werden, die an die Klassen übergeben werden. Der Ablauf gestaltet sich folgendermaßen: Sobald das Programm auf dem mobilen Endgerät gestartet wird, erfolgt die Instanziierung des MainMidlets und die Properties werden eingelesen. Anschließend wird die statische Methode MainMidlet.startApp() aufgerufen. Dies hat zur Folge, dass der ComponentContainer initialisiert und für jede in den Properties angegebene Komponente eine neue Instanz erzeugt wird. Zudem wird die init()-Methode der Komponenten aufgerufen. Schließlich ist das Programm bereit und die eigentliche Geschäftslogik wird ausgeführt.

3.3 Datenbankgestütztes Serversystem

Als Gegenstück zu den Basis-Softwarekomponenten für mobile Endgeräte, syndiziert ein zentrales Serversystem multimediale Inhalte (wie Text und POI-Daten) in einem Datenbanksystem und stellt diese für den Abruf durch die Client-Anwendungen bereit. Dabei werden die Inhalte mit Positionsangaben angereichert, um auch die Entwicklung ortsgestützter Dienste zu ermöglichen. Die Architektur der J2EE-Applikation entspricht weitestgehend dem aus der Software-Entwicklung bekannten MVC-Konzept (Model-View-Controller) mit einer Software-Komponente für den effizienten Zugriff auf Datenbanksysteme (Persistenzschicht mit O/R-Mapping durch Hibernate und MySQL-Datenbank), einer Darstellungskomponente zur Aufbereitung der multimedialen Inhalte, für den Transport über das Mobilfunknetz und Darstellung auf dem mobilen Endgerät und mit einem generischen Framework zur Abbildung von Geschäftsprozessen. Zur Netzwerkkommunikation wird auf bewährte Protokolle wie HTTP und SOAP (Web Services) zurückgegriffen. Darüber hinaus bietet ein weiteres Modul den Anwendungsentwicklern eine homogene Schnittstelle an, um auf WebServices von z.B. POI- oder GIS-Providern einheitlich zugreifen zu können.

Kern der Server-Architektur ist das Spring-Framework (www.springframework.org), das die für unsere Zwecke benötigten Features, wie Dependency Injection[1] und aspektorientierte Programmierung[2], bereitstellt. Für das „Object-Relational-Mapping“ wird das Open-Source-Persistenzframework Hibernate (www.hibernate.org) eingesetzt.

3.4 Anwendungsempfehler (Dispatcher)

Die Anwendungsentwickler haben außerdem noch die Möglichkeit ihre mobilen Anwendungen, die mit Hilfe der Play.Tools erstellt wurden, bei der Vertriebskomponente, des Serversystems zu registrieren um diese, wie bereits beschrieben, unter Marktbedingungen testen zu können. Dafür werden die entwickelten Anwendungen in der Datenbank mit entsprechenden Beschreibungstexten und weitergehenden Informationen gespeichert. Die Entwicklungsumgebung bietet für diese Anwendungen einen dynamischen Installationsprozess an: Der Benutzer sieht auf seinem mobilen Endgerät automatisch eine Liste an Anwendungen, die für seinen aktuellen Kontext (wie sein Aufenthaltsort) von Interesse sein könnten. Beispiele dafür sind:

- vor dem Kino: Kinoplan, Kartenreservierung
- im Bahnhof: Fahrplan, Wetterinformationen des Zielorts etc.

Die Anwendungsliste wird mit Hilfe eines hybriden Recommender-Systems an die jeweiligen Interessen der Benutzer angepasst [Woj06, WSW07]. Ausschlaggebend hierfür sind die gespeicherten Profildaten sowie der aktuelle Kontext des jeweiligen Anwenders. Zu dem Kontext zählt auch das verwendete Endgerät. Die Liste beinhaltet somit nur Anwendungen, die auch auf dem mobilen Endgerät des Anwenders lauffähig sind. Hierfür ist in der Datenbank vermerkt, welche Anforderungen eine Anwendung an das Zielgerät hat und welche Endgeräte die benötigten Eigenschaften mitbringen. Der Anwender kann durch die Liste navigieren und Detailinformationen zu den Anwendungen einsehen. Zudem kann er die für ihn interessante Anwendungen herunterladen und installieren.

4 Beispielanwendungen

Es wurden bereits mehrere Anwendungen mit Hilfe des beschriebenen Frameworks entwickelt. Dabei wurde das Prinzip des Prototypings angewandt. Die Anwendungen konnten mit kleinen studentischen Teams im Rahmen eines Seminars an der TU München innerhalb kurzer Zeit entwickelt werden und zeigten somit insbesondere die Praxistauglichkeit des Konzepts und der Implementierung des beschriebenen Frameworks. Drei der entwickelten Anwendungen werden im Folgenden kurz vorgestellt.

---

1 Objekten werden Ressourcen zugewiesen, sie müssen nicht selbst danach suchen

2 Code ist frei von technischen Aspekten wie Transaktionen oder Sicherheit

4.1 FriendLocator

FriendLocator (www.friendlocator.biz) ist eine mobile Anwendung, die es Freunden, so genannten Buddys, ermöglicht, sich gegenseitig zu lokalisieren, um sich z.B. an einer bestimmten Position an einem unübersichtlichen Ort zu treffen. Diese Applikation verwendet hierfür eine Vielzahl der durch das Play.Tools-Framework bereitgestellten Komponenten, wie Benutzerverwaltung, Kartendarstellung und Positionsbestimmung. Für die Benutzung der Anwendung ist es zuerst notwendig, sich über das Webfrontend zu registrieren (Opt-In-Mechanismus).

Abbildung 2: Screenshots der Anwendung FriendLocator

Nachdem die Anmeldung erfolgreich abgeschlossen wurde, kann der Anwender seine eigene Buddy-Liste verwalten, d.h. nach neuen Freunden suchen und diese zu seiner Liste einladen oder auch Personen aus der Liste entfernen. Sobald der Benutzer eine eigene Buddy-Liste angelegt hat, kann er die gesamte Funktionalität des FriendLocator nutzen. Wie in Abbildung 2 dargestellt, meldet sich der Anwender zuerst über sein mobiles Endgerät am Buddy-List-Server (basierend auf den Serverkomponenten des Frameworks, vgl. Abschnitt 3.3) an. Nach der erfolgreichen Anmeldung wird die Verfügbarkeit eines GPS-Signals überprüft. Sobald die eigene Position ermittelt wurde, werden mittels eines WebServices eine Karte und die Positionen der Buddys auf das mobile Endgerät übertragen und visualisiert. Der Anwender kann den Maßstab der Karte beliebig anpassen und sich über die Karte bewegen.

4.2 MoNoTake

MoNoTake (MObile NOte Taker) ist eine Anwendung, die es Geschäftsleuten oder Reisenden ermöglicht, für sie wichtige Ereignisse oder Orte (Locations) auf ihren mobilen Geräten festzuhalten. Diese können zur späteren Nachbearbeitung bzw. zur Nachverfolgung des Tages- oder Reiseablaufs auf einen Server übertragen werden. Für einen bestimmten Speicherpunkt werden der Zeitpunkt (Timestamp), Positionsdaten (GPS), wenn verfügbar und gewünscht, ein Foto sowie ein kurzer Text abgespeichert. Ein Ereignis kann einer Kategorie zugeordnet werden und das mobile Gerät ermöglicht es dem Anwender die erfassten Ereignisse einzeln durchzugehen, um sich den Tages- oder Rei-

seablauf anzusehen. Zusätzlich zur mobilen Anwendung gibt es eine Serveranwendung, welche die Ereignisse über HTTP aufnehmen und speichern kann.

4.3 FindParty

Die Anwendung FindParty ermöglicht es Menschen, insbesondere Singles, neue Bekannte mit ähnlichen Interessen kennen zu lernen. Ein mögliches Anwendungsszenario sieht wie folgt aus: Zwei Freunde sind in einer fremden Stadt unterwegs, sitzen im Kaffee und suchen für den Abend noch eine Begleitung, um gemeinsam einen unterhaltsamen Abend zu verbringen. Sie starten also ihre FindParty-Anwendung und geben ihre Interessen und Hobbys ein, wie viele Personen sie suchen, deren Geschlecht und in welcher maximalen Entfernung sich diese zu ihnen befinden sollen. Die Anfrage wird zusammen mit der eigenen Positionsangabe auf dem Server gespeichert. Dadurch kann die Anfrage zum einen von anderen FindParty-Anwendern gefunden und zum anderen auch nach passenden Matches zu diesen gesucht werden. Der Serverdienst sucht anhand der Position nach möglichen Zielpersonen in Ihrer Nähe und bietet sie ihnen in einer Auswahlliste an. Die zwei Freunde können sich Details, wie Hobbys, Interessen, Alter und auch Bilder, zu den gelieferten Ergebnissen anzeigen lassen und eventuell eine Einladung an eine für sie interessante Person/Gruppe schicken.

5 Verwandte Arbeiten

Es gibt im Bereich Entwicklungsframeworks für mobile Anwendungen einige verwandte Arbeiten, welche in diesem Abschnitt diskutiert und mit unserem Ansatz verglichen werden:

[HH05] stellt einen zu unseren Komponenten vergleichbaren Ansatz dar. Diese Plattform ist aber eher für Forschungszwecke ausgerichtet, während Play.Tools die Innovations- und Gründungsförderung unterstützen soll. Ein weiteres ähnliches Framework wird in [WN06] vorgestellt, das aber recht stark auf das Umfeld des Nexus-Projekts zugeschnitten zu sein scheint. Die TraX Plattform [Mar06] ist ein eher middleware-orientiertes System, welches auch die Entwicklung von ortsbezogenen Diensten unterstützt und dabei den Fokus auf proaktive Anwendungen legt. MobCon realisiert die Idee eines „logical mobile container“, welcher eine Abstraktion für die Kernkomponenten einer mobilen Anwendung darstellt [CM05]. Der Schwerpunkt dieses Ansatzes ist die Codegenerierung aus einem abstrakten Modell in MIDP-Klassen. Unsere Play.Tools sind im Vergleich zu MobCon dafür umfassender, z.B. gibt es bei uns Klassen für Web-Services auf Client-Seite sowie die dargestellten Server-Komponenten. [SPM03] erläutert ein DRM-Framework für JME-Anwendungen, welches den Vertrieb von mobilen Java-Anwendungen mit Hilfe von Digital Rights Management (DRM) kontrollieren soll. Dies wäre als Ergänzung für unsere Serverkomponenten interessant und könnte als Abrechnungskomponente unsere Vertriebsplattform ergänzen. Der Ansatz ermöglicht dazu auch die Entwicklung eigener DRM-Anwendung auf Basis des zur Verfügung gestellten Frameworks.

6 Fazit

Die Entwicklung vermarktungsfähiger und innovativer mobiler Anwendungen stellt aufgrund unzureichender Standards, fehlerhaften Programmierschnittstellen und extrem kurzen Produktlebenszyklen bei mobilen Endgeräten und der dazu passenden Software eine große Herausforderung dar. Speziell bei der Einbeziehung von Kontextinformationen des Nutzers mobiler Anwendungen entstehen durch die hohe Interdisziplinarität des Faches und die Vielzahl der Prozessbeteiligten eine besondere Komplexität, die für Softwareentwickler sehr schwer zu beherrschen sind. Die hinzukommenden inhärenten Marktunsicherheiten von Innovationsvorhaben verschärfen noch einmal die Situation, die eine Anwendungs- und Geschäftsmodellentwicklung erschweren. Dies könnte zumindest teilweise erklären, warum bisher kontextsensitive mobile Anwendungen nicht kommerziell erfolgreich durchgesetzt werden konnten.

Daher wurde mit Play.Tools ein Software-Framework geschaffen, welches die prototypische Umsetzung und anschließende Markttests kontextsensitiver mobiler Anwendungen als Unterstützung von Innovationsprozessen vereinfacht und damit Technologie- und Marktunsicherheiten deutlich reduzieren kann. Das Framework stellt unter anderem Softwarekomponenten bereit, um generische Programmieraufgaben, wie die Ermittlung von Positionsangaben als Basis für ortsgestützte Anwendungen, zu kapseln. Des Weiteren wurde eine Server-basierte Infrastruktur für die Bereitstellung von syndizierten Geoinformationen (z.B. Kartenmaterial) entwickelt. Auf Seiten der Anwender ist die Schwierigkeit anzuführen, für den jeweiligen Kontext die passende und auf dem Endgerät lauffähige Anwendung zu finden. Hierfür wurde ein Anwendungsempfehler entwickelt, der den Anwendern eine Liste an Anwendungen anbietet, die zu seinem Kontext, also auch seinem Endgerät, passen.

Das Play.Tools Framework wurde bereits in einigen Innovationsprojekten erfolgreich zur Erstellung von Prototypen eingesetzt und ein signifikanter Zeit- und Qualitätsvorteil bei der Entwicklung mobiler Anwendungen festgestellt. Weitere empirische Untersuchungen dazu stehen allerdings noch aus. Dabei konnten auch einige Mängel in der bestehenden Version 1.2 des Frameworks identifiziert werden, welche derzeit in die Implementierung einer verbesserten Version von Play.Tools mit einfließen. Weitere Aspekte, an denen wir gerade arbeiten, sind die Generierung von High-Level Kontext aus Sensordaten und die Anbindung weiterer Sensoren (z.B. zur Bestimmung der Bewegung oder Beschleunigung eines Gerätes). Außerdem wird zurzeit eine Funktionalität für Instant Messaging auf mobilen Endgeräten sowie eine Komponente zur Positionsbestimmung über die Cell-ID des Mobiltelefons in das Framework integriert. Ein weiterer interessanter Punkt ist die Entwicklung eines Billing Systems für die Vertriebskomponente unseres Frameworks.

Literaturverzeichnis

[Boe84] Böhm, B.W.: Prototyping vs. Specifing: A multiproject experiment. In: IEEE Transactions in Software Engineering 10, Nr 3, 1984, pp. 290-302

[Boe86] Böhm, B.W.: A spiral model of software development and enhancement. In: ACM Sigsoft Software Engineering Notes 11, Nr 4, 1986, pp. 14-24

[Bur02] Burke, R.: Hybrid Recommender Systems: Survey and Experiments. In: User Modeling and User-Adapted Interaction 12, Nr. 4, 2002, pp. 331–370

[CM05] Cepa, V., Mezini, M.: MobCon: A generative middleware framework for Java mobile Applications. In: Proceedings of the 38th Hawaii International Conference on System Sciences, 2005

[Dey01] Dey, K.A., Abowd, D.G., Salber, D.: A Conceptual Framework and Toolkit for Supporting the Rapid Prototyping of Context-Aware-Applications. In: Human Computer Interaction, Volume 16, 2001, pp. 97-166

[Eix06] Eixner, T.: Entwicklungsprozess und Frameworks mobiler Anwendungen. Technische Universität Dresden, Lehrstuhl Multimediatechnik, 2006

[Ern01] Ernst, H.: Erfolgsfaktoren neuer Produkte: Grundlagen für eine valide empirische Forschung. Gabler, Wiesbaden, 2001

[Gar05] Gartner Dataquest (Quartal 03/04), In: VDI-Nachrichten, 18. Februar 2005, Nr. 7, S. 22

[Han06] Handango (Hrsg.) . Informationen über Handango. http://www.handango.de/GeneralHelp.jsp?siteId=8&CKey=germanHelp_GeneralQuestions&PKey=14 (Stand: 11.10.2006)

[HH05] Heutelbeck, D., Hemmje, M.: A Research Platform for Locations-based Applications. In.: Proc. 2.GI/ITG KuVS Fachgespräch – Ortsbezogenen Dienste und Anwendungen, Hagen, 2005

[HLO05] Hanhart, D., Legner, C., Österle, H.: Anwendungsszenarien des Mobile und Ubiquitous Computing in der Instandhaltung. In: Pousttchi, K. , Turowski, K. (Hrsg.): 5.Konferenz Mobile Commerce Technologien und Anwendungen, Gesellschaft für Informatik, Bonn, 2005

[KL03] Kuhn, J., Lehner, F.: Szenarien einer mobilen Zukunft. In: Pousttchi, K., Turowski, K. (Hrsg.): Mobile Commerce – Anwendungen und Perspektiven. Proceedings zum 3. Workshop Mobile Commerce, Universität Augsburg, Lecture Notes in Informatics, Gesellschaft für Informatik,, Bonn. 2003. pp. 130-142

[Mar06] Martens, J. et al..: Eine Plattform zur Unterstützung von proaktiven ortsbezogenen Mehrbenutzer-Anwendungen. In. Proc. 3. GI/ITG KuVS Fachgespräch - Ortsbezogene Anwendungen und Dienste, Berlin, 2006

[Pru06] Pruscha, P.: Kontext- und Profilverwaltung mobiler Anwendungen. Master-Thesis., Fakultät für Informatik, Technische Universität München, 2006

[Sch01] Schmidt, A. et. al.: Entwicklung von WAP-Anwendungen. In: Killat, U.; Lamersdorf, W. (Hrsg.): Kommunikation in Verteilten Systemen (KiVS), 12. Fachkonferenz der Gesellschaft für Informatik (GI) Fachgruppe "Kommunikation und Verteilte Systeme" (KuVS), Hamburg, 2001

[SM05] SUN (Hrsg.), Motorola: Java APIs for Bluetooth Wireless Technologie (JSR 82). Specification Version 1.1, 2005

[SO05] Senneset, T., Odegaard, F.: Context-Aware Services and Systems. In: TTM3 – Design of self-configuring systems, 2005

[SPM03] Santos, N., Pereira, P., Moura e Silva, L.: A Generic DRM Framework for J2ME Applications. In Proc. First International Mobile IPR-Workshop: Rights Management of Information (MobileIPR 2003), Helsinki, Finland, 2003

[Tho03] Thomke, S.: Experimentation matters: Unlocking the Potential of New Technologies for Innovation. Boston: Harvard Business School Press, 2003

[WN06] Wieland, M., Nicklas, D.: Ein Framework für kontextbezogene Anwendungen in der Nexus-Plattform. In. Proc. 3. GI/ITG KuVS Fachgespräch - Ortsbezogene Anwendungen und Dienste, Berlin, 2006

[Woj06] Wojtech, R.: Kontext- und Profilverwaltung mobiler Anwendungen. Diplomarbeit, Fakultät für Informatik, Technische Universität München, 2006

[WSW07] Wolfgang Woerndl, Christian Schueller, Rolf Wojtech: A Hybrid Recommender System for Context-aware Recommendations of Mobile Applications. In Proc. IEEE 3rd International Workshop on Web Personalisation, Recommender Systems and Intelligent User Interfaces (WPRSIUI'07), Istanbul, Turkey, 2007