Interaction with public displays

Nico Fremann
Institut für Geoinformatik
n_frem01@uni-muenster.de

Marc Jentsch
Institut für Geoinformatik
marc.jentsch@uni-muenster.de

Dennis Wilmsmann
Institut für Geoinformatik
dennis.wilmsmann@uni-muenster.de

ÜBERSICHT

Im folgenden Bericht stellen wir eine Übersicht über das Projekt „Interaction with public and private displays“ vor, dass im Sommersemester 2006 im Seminar „Location Based Services“ zusammen mit anderen Kurzprojekten entstanden ist. Das Projekt besteht aus der technischen Umsetzung eines Handy-basierten Client-Server Systems und der zugehörigen Benutzerstudie.

EINLEITUNG

Durch die stark fallenden Hardwarekosten gibt es immer mehr öffentliche Displays. Bei diesen entsteht das Problem, dass auf den Displays nicht genügend Platz ist, um ausreichend Informationen für alle vorbeikommenden Nutzer anzuzeigen, da sich die Interessen der einzelnen Nutzer stark unterscheiden können.
Aus diesem Grund wird im Folgenden ein Ansatz untersucht, wie die benutzerspezifischen Informationen auf den privaten Geräten, wie z.B. Handys, genauer aufgeführt werden können. Der Nutzer soll sich zu wechselnden Topics auf den öffentlichen Displays genauere Hintergrundinformationen zusenden lassen können. Zur Interaktion mit den Displays nutzt er sein Handy über eine Bluetooth Verbindung. Die Informationen werden ihm an eine Mailadresse gesendet, die er einmalig registrieren muss.
Für die zugehörige Benutzerstudie wird zum einen ein Scanner eingerichtet, der die Umgebung nach neu eingetretenen Bluetooth Geräten scannt. Über den Verlauf des Zeitabstandes zwischen Eintritt in den Bluetooth Bereich und Nutzen des Services versuchen wir eine Lernkurve für die Benutzung des Clients zu ermitteln. Zum anderen verteilen wir an die Nutzer gegen Ende des Testzeitraums Fragebögen mit spezifischen Fragen über ihre empirischen Erfahrungen mit dem System.

STICHWÖRTER

private displays, public displays, interaction, midlet, handy, bluetooth

VERSUCHSAUFBAU

Technische Voraussetzungen

Am Institut für Geoinformatik hängen aus einem früheren Studienprojekt zwei öffentliche Displays, genannt iDisplays. Diese werden genutzt, um zum einen allgemeine nützliche Informationen wie Mensapläne, Busfahrpläne oder Wetteraussichten zu verbreiten, zum anderen werden sie als „Schwarzes Brett“ verwendet. Hierbei werden in regelmäßigen Abständen Informationen aus den verschiedenen Fachbereichen angezeigt, beispielsweise werden Diplomarbeitsthemen angeboten oder es werden Hinweise auf Veranstaltungen wie das wöchentlich stattfindende GI-Forum angezeigt.
Wir konzentrieren uns auf den Teil „Schwarzes Brett“, da hier Hintergrundinformationen für einzelne Nutzer interessant sein können.
Unser Service soll als Java Dienst laufen. Dies hat mehrere Vorteile: Es ist relativ systemunabhängig. Wir besitzen die meiste Erfahrung mit dieser Programmiersprache. Und es sind am wenigsten Probleme beim Zusammenspiel mit dem Client zu erwarten. Denn auf dem Client muss ein Java-Midlet laufen, da es auf Handys kaum andere Möglichkeiten der Programmierung gibt. Windows Mobile Betriebssysteme sind dafür noch nicht verbreitet genug.
Die Steuerung läuft über Rechner, die direkt hinter den Displays angebracht sind. Diese sind nicht miteinander synchronisiert, haben aber Netzwerkzugang und damit auch Internetzugang und greifen über diesen auf dieselbe Datenbank zu. Ursprünglich lief auf dem einen Rechner ein Windows System und auf dem anderen ein Linux System. Wir haben uns entschieden erst einmal nur exemplarisch unseren Dienst für das Windows System zu entwickeln, da wir zum einen damit mehr Erfahrung haben, zum anderen die Bluetooth Unterstützung für Java auf Linux nicht so ausgereift ist.
Die angezeigten Informationen sind in der oben angesprochenen MySQL-Datenbank hinterlegt. Pro Eintrag sind für uns der angezeigte Text interessant, den man sich per Anfrage zuschicken können lassen soll. Ein Eintrag hat immer eine Überschrift, die in unserer Mail als Betreff verwendet wird. Desweiteren gibt es zu einzelnen Themen noch ein optionales Bild.
Außerdem steht uns ein weiterer Datenbankserver zur Verfügung, der allgemein für studentische Angelegenheiten genutzt wird. Hier protokollieren wir die gewonnenen Daten aus unseren Scanvorgängen und legen

die benötigten personenbezogenen Informationen der Nutzer ab, wie z.B. die Email-Adresse.
Unser Server, der die Anfragen der Handys entgegen nimmt, kann auf dem Displayrechner laufen, da dieser mit Bluetooth Hardware ausgestattet ist.
Als Client stehen uns die privaten Handys der Teilnehmer des Seminars „Location Based Services“ zur Verfügung. Das heißt, dass alle Seminarteilnehmer, die ein Java-fähiges Bluetooth Handy besitzen, dazu aufgefordert werden, unseren Service über einen Testzeitraum zu benutzen. Als kleinen Ansporn verlosen wir unter den aktivsten Nutzern zwei Kinokarten.

Organisatorische Rahmenbedingungen

Das Projekt ist für ein Semester vorgesehen, in dem einige Termine einzuhalten sind. Zu Beginn des Seminars ist ein Zeitraum von vier Wochen zur Planung und technischen Umsetzung veranschlagt worden. Es folgt ein Termin, an dem den Seminarteilnehmern das Projekt und eine Einführung in die Benutzung vorgestellt wird. Zum Abschluss ist eine Präsentation der Ergebnisse vorgesehen. Somit bleibt ein Zeitraum von vier Wochen für den Testlauf.
Die iDisplays werden bisher ausschließlich für Institut-interne Zwecke benötigt. Das hat den Vorteil, dass wir unser System direkt in das produktive Display System einbinden können. Allerdings wird am iDisplay System weiter entwickelt und es gibt im Institut einige andere Projekte, die ebenfalls direkt am produktiven System arbeiten. Das bedeutet, dass die Stabilität des Systems nicht immer gewährleistet sein kann.

VERSUCHSABLAUF

Technische Umsetzung

Zunächst gilt es eine passende API für Bluetooth zu finden. Ein erster Versuch das Microsoft .NET 2.0 Framework zu benutzen schlägt fehl, weil wir nur kommerzielle Bluetooth APIs finden. Wir entscheiden uns schließlich für die frei verfügbare Java-API Bluecove [1], da dieses das erste System ist, dass wir zeitnah lauffähig bekommen. Bluecove setzt auf nativen Methoden für den Microsoft Bluetooth Stack [4] auf, der mit dem Windows XP Service Pack 2 geliefert wird. Für den ebenso verbreiteten Widcomm Bluetooth Stack ist unser Dienst damit also nicht lauffähig.

Scanner

Um eine Lernkurve bei der Benutzung unseres Clients zu erhalten, benötigen wir einen Scanner, der die Umgebung nach vorhandenen Bluetooth Geräten scannt. Wird ein Gerät gefunden, wird vom Scanner die MAC-Adresse des Geräts ausgelesen um es eindeutig zu identifizieren. Aus Datenschutzgründen nehmen wir nur die Daten von Geräten auf, die sich bei unserem Service angemeldet haben.
Beim Scanner kommt es zu mehreren Problemen: Ein Scanvorgang dauert ca. 12 Sekunden. Diese Zeitdauer ist von der Hardware vorgegeben und kann nicht modifiziert werden. Das bedeutet, dass unsere Genauigkeit Zeitstempel zu erfassen nicht unseren Wünschen entspricht. Da wir aber davon ausgehen, dass der Aufenthalt im Scanbereich und Nutzung des Dienstes eher im Minutenbereich liegt, sehen wir diese Ungenauigkeit im Bereich des Erträglichen an.
Während eines Scanvorgangs kann die Bluetooth Hardware keine Anfrage entgegennehmen, da Bluecove den Zugriff blockiert [2]. Datentransfers über die serielle Verbindung sind zwar möglich, wenn bereits eine Verbindung besteht, aber ein Client kann sich nicht neu mit dem Server verbinden, während dieser scannt. Eine Lösungsmöglichkeit wäre es, zwischen den einzelnen Scanvorgängen ein Freiintervall einzubauen. In diesem könnten dann Verbindungen vom Client entgegengenommen werden. Das hat den Nachteil, dass die sowieso schon schlechte Scangenauigkeit weiter verschlechtert würde, wodurch die Daten für unsere Auswertung unbrauchbar wären. Außerdem wäre es weiterhin möglich, dass ein Nutzer während des Scans einen Verbindungsaufbau versucht, der dann fehlschlägt.
Als zweiten Lösungsversuch bringen wir am Rechner zusätzlich zur eingebauten Bluetooth Hardware einen zweiten Bluetooth Dongle an. Die Idee ist, den Scanner über die eine Hardware und den Server über die andere laufen zu lassen. Leider wird die zweite Hardware vom Betriebssystem nicht erkannt.
Deshalb trennen wir Server und Scanner auf. Der Scanner verbleibt auf dem Displayrechner. Der Server ist nun auf einem zusätzlichen Rechner ausgegliedert, welchen wir in einem angrenzenden Technikraum aufstellen. Dieser Rechner ist mit einem Klasse 1 [5] Dongle ausgestattet, dessen Empfangsradius durch Tür und Wand reicht, sodass sichergestellt ist, dass alle Anfragen auch entgegengenommen werden können.
Im Display Rechner ist eine Klasse 3 [5] Bluetooth Hardware eingebaut. Das bedeutet, dass wir einen Bereich von maximal 10 Metern um das Display scannen können. Ein höhere Reichweite ist auch nicht gewünscht, da davon auszugehen ist, dass Geräte, die weiter als 10m vom Display entfernt sind nicht das Display nutzen, da es nicht mehr gut genug zu erkennen ist. In unserem Fall wäre es bei einer höheren Reichweite gut möglich gewesen, „falsche“ Bluetooth Geräte zu erkennen, da sich in näherer Umgebung die Medienwerkstatt mit einigen anderen Bluetooth Geräten befindet.

Client

Bluetooth bietet mehrere Arten von Services. Der ObexPush-Service wird benutzt, um Objekte wie Termine, Kontakte oder Aufgaben zwischen Geräten auszutauschen.

Unsere erste Idee ist, dass der Nutzer in seinem Handy einen Kontakt mit der ID der anzufordernden Information anlegt und diesen per ObexPush an den Server schickt.
Die Alternative ist ein SerialPort-Service, der eine normale COM-Verbindung simuliert und über den einfache Datentypen ausgetauscht werden können.
Wir entscheiden uns schließlich für den zweiten Service, da dieser der Standard Bluetooth Service ist und somit leichter umzusetzen ist. Außerdem halten wir es für intuitiver ein Programm zu benutzen um Informationen anzufordern, als einen Kontakt anzulegen und damit zu interagieren. Nutzer empfinden es außerdem als nicht angenehm, Kontakte zwischen Ihren normalen Kontakten halten zu müssen, die nur aus Zahlen bestehen.
Als Client muss also ein Midlet erstellt werden, dass sich die Nutzer auf ihrem Handy installieren. Beim Midlet-Start wird automatisch versucht, zum Server eine Verbindung aufzubauen. Hier gibt es die Möglichkeit den Client die komplette Umgebung nach Bluetooth-Geräten scannen zu lassen, die SerialPort Dienste anbieten und sich anschließend mit diesen zu verbinden. Die andere Möglichkeit ist, die MAC-Adresse des Servers fest im Client zu hinterlegen und sich direkt mit dieser zu verbinden. Die erste Möglichkeit ist normalerweise die flexiblere, da der Client so für verschiedene Server verwendet werden kann. Da wir für unseren Test aber nur einen Server verwenden, entscheiden wir uns für die zweite Möglichkeit, da bei dieser der Verbindungsaufbau schneller ist.
Bei der ersten Benutzung muss der Nutzer seine Email-Adresse eingeben, die in der Datenbank mit der MAC-Adresse verknüpft wird. Da die MAC-Adresse bei jeder Anfrage automatisch mitgesendet wird, braucht sich der Nutzer dort später nicht mehr darum zu kümmern. Das System erkennt ihn automatisch und schickt die Informationen an die hinterlegte Mailadresse. Das Formular für die Eingabe der Mailadresse ist auf dem Midlet über einen Menupunkt zu erreichen. Hier kann auch die Mailadresse geändert werden. Nach Benutzen dieses Formulars erhält man vom Server eine Bestätigung im Erfolgsfall, in dem man seine eingegebene Mailadresse auch noch einmal überprüfen kann. Sollten Probleme auftreten, wie eine abgerissene Bluetooth Verbindung, erscheint eine Fehlermeldung. Das Formular unterstützt den User bei der Eingabe, indem nur für Mailadressen gültige Zeichen eingegeben werden können.
Das Hauptmenü besteht aus einem Eingabefeld, in dem die ID der Information eingegeben wird, die angefordert werden soll. Anschließend betätigt man eine Taste zum Absenden der ID. Zurzeit werden maximal zwei Themen gleichzeitig auf dem Display angezeigt, sodass nur eine 1 für das erste Thema respektive eine 2 für das zweite Thema eingegeben werden können. Auch hier bietet unser Client wieder Benutzerunterstützung durch beschränkte Eingabemöglichkeiten, Bestätigungs- bzw. Fehlermeldungen. Um die Bedienung möglichst einfach zu gestalten, steht dem Benutzer nur ein Eingabefeld zur Verfügung.

Server

Der Server nimmt die Anfragen vom Client entgegen. Die vom Client geschickte Anfrage enthält die oben genannten Ziffern 1 oder 2, die die Platzierung auf dem Display repräsentieren. Die Beiträge haben in der Datenbank eine eindeutige ID, die aber nicht diesen gesendeten Ziffern entspricht. Wir bekommen vom Admin der Displays einen Dienst gestellt, der per telnet Anfrage die Zuordnung Display-Ziffern zu Datenbank-ID auflöst. Der Text und das eventuell verfügbare Bild werden aus der Datenbank gelesen und per Mail an die hinterlegte Adresse verschickt. Jedes Thema hat einen Titel, der als Mailbetreff verwendet wird.
Der Server befindet sich die meiste Zeit im Wartezustand und wird bei eingehenden Anfragen aktiviert. Über ein Flag in der Anfrage entscheidet er, ob es sich um eine Anfrage nach einer Information oder um das hinzufügen bzw. ändern einer Mailadresse handelt.
Jede Anfrage wird in einem eigenen Thread verarbeitet, sodass mehrere Anfragen gleichzeitig bearbeitet werden können.
Zur Verfügbarkeitsüberwachung senden Scanner und Server zweimal täglich eine Status-Email.

VERSUCHSVERLAUF

Planung und Implementierung

Zu Beginn des Projektes sind die Ziele festgelegt worden. Das System soll konkret umgesetzt werden. Über die Daten des Scanners soll eine Benutzerstudie in Bezug auf eine mögliche Lernkurve ermittelt werden. Über Fragebögen wollen wir herausfinden, wie die Nutzerakzeptanz aussieht.
Wir erstellen einen Ablaufplan, der eine Übersicht der Prozessschritte wiedergibt. (s. Abbildung 1)
Bei der Implementierung arbeiten wir in 3 Teilbereichen. Client/Serverimplementierung, Scanner und Datenbankzugriff. Anfang Mai treffen wir uns um das System zusammenzufügen und zu testen. Dabei stellen wir das oben angesprochene Problem fest, dass Scannen und Verbindungsaufbau nicht gleichzeitig möglich ist.

Testphase

Zu Beginn der Testphase erklären wir den potentiellen Nutzern die Idee des Systems und die Voraussetzungen zur Benutzung. Die Verteilung soll über zwei verschiedene Wege funktionieren. Zum einen erstellen wir eine Homepage [3], auf der man sich den Client herunterladen und in Eigenverantwortung installieren kann. Zum anderen verteilen wir von unseren Rechnern die Midlets per Bluetooth, bzw. soll die Weitergabe von Handy zu Handy stattfinden.

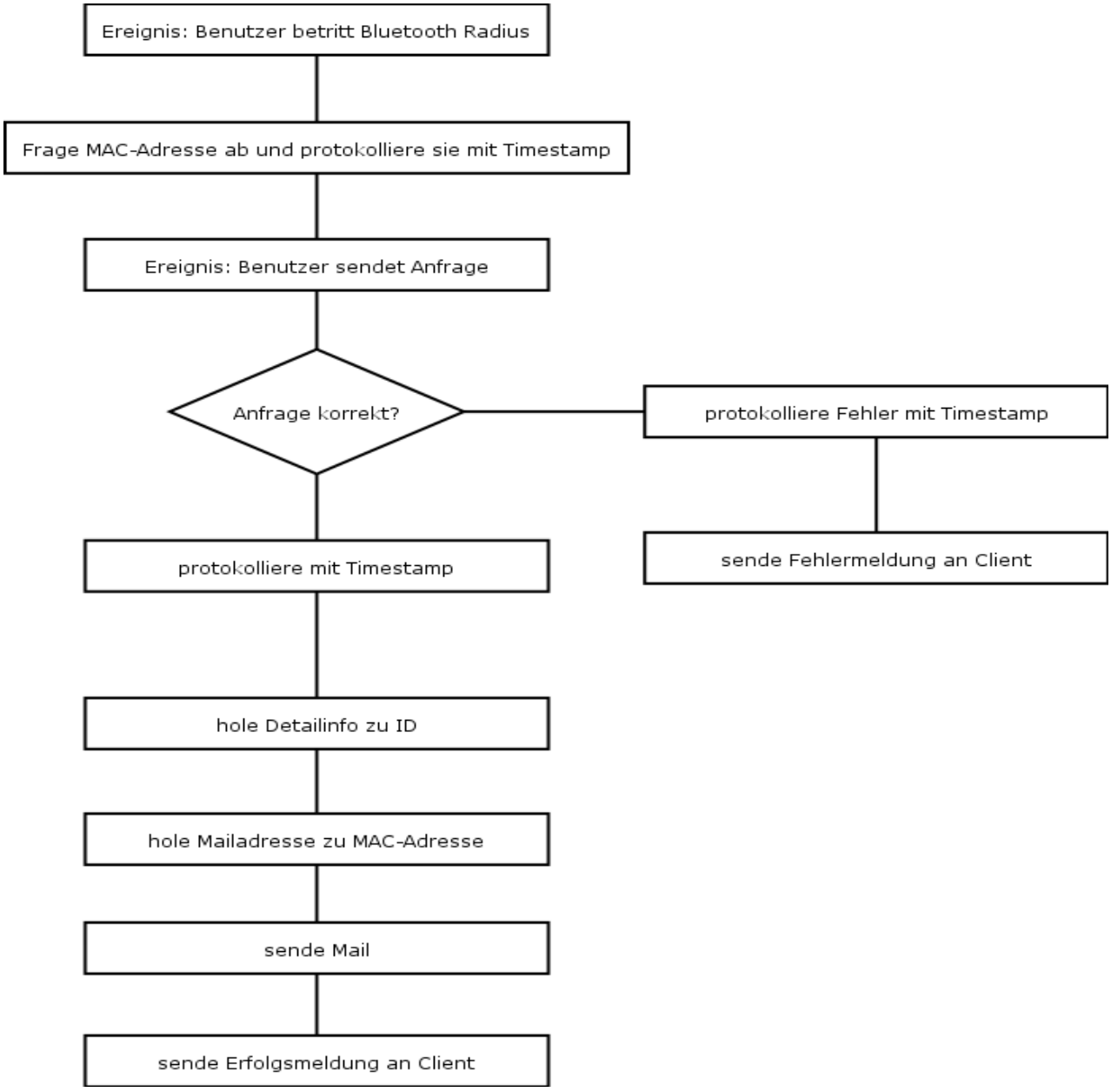

Abbildung 1: Ablaufplan der Prozessschritte

Hier haben wir das erste Mal die Möglichkeit das System auf einer Vielzahl verschiedener Handymodelle zu testen und stoßen auf eine Reihe von Problemen, die vor allem herstellerabhängig zu sein scheinen. (s. Tabelle 1)
Die Probleme können aus mehreren Faktoren resultieren. Zum einen unterstützen manche Handys nicht CLDC 1.1. CLDC [6] definiert die kleinstmögliche Konfiguration einer J2ME Laufzeitumgebung. Da wir keine CLDC 1.1 Funktionen nutzen, stellen wir noch eine CLDC 1.0 Version zur Verfügung.
Außerdem unterstützen manche Handys kein JSR-82, die Bluetooth Spezifikation für Java. Für das Siemens S55 müsste beispielsweise für die Implementierung das Siemens Wireless Toolkit benutzt werden, um Bluetooth steuerbar zu machen. Andere Handys wiederum führen nur digital signierte Midlets aus. Aus Zeitgründen führen wir die digitale Signierung nicht mehr durch.
Um das System unter möglichst realistischen Bedingungen zu testen, beschränken wir die Benutzung nicht auf die drei Anwesenheitstermine, sondern fordern die Nutzer dazu auf, es über einen Zeitraum von vier Wochen zu benutzen, während sie sowieso im Institut verkehren.

Zur Mitte des Testzeitraums funktionierte der telnet-Service, der die Zuordnung von angefragter Ziffer zu Datenbank ID liefert, aufgrund von Wartungsarbeiten nicht mehr, sodass leere Mails verschickt wurden, weil eine

Hersteller	Modell	Status
Sony Ericsson	K750i	Funktioniert ohne Einschränkung
Motorola	PEBL	Funktioniert ohne Einschränkung
Motorola	E770v	Funktioniert ohne Einschränkung
Nokia	6310i	Midlets lassen sich generell nur über spezielle Nokia Software installieren, danach funktionsbereit ohne Einschränkung
Nokia	6670	Midlets lassen sich generell nur über spezielle Nokia Software installieren, allerdings danach Client nicht ausführbar
Siemens	S65	Client lässt sich zwar ausführen, aber bei Verbindungsaufbau wird eine PIN angefordert. Dieses ist eine Siemens Sicherheitsvorkehrung. Die einzelnen Geräte müssen initial einen Schlüssel mit der Server-Hardware austauschen. Danach funktionsbereit ohne Einschränkung
Siemens	S55	Midlet lässt sich nicht starten
Samsung	SGH-D820 und andere Modelle	Midlet nicht als Applikation erkannt
XDA	Phone	Midlet lässt sich nicht starten

Tabelle 1: Übersicht der getesteten Handymodelle

leere Datenbank ID abgefragt wurde. Gegen Ende des Testzeitraums wurde ebenfalls an den Displays gearbeitet. Deswegen konnte unser Scanner dort nicht andauernd ausgeführt werden.
Zwei Wochen vor Ende des Testzeitraums senden wir einen Fragebogen an die registrierten Mailadressen, mit der Bitte, ihn ausgefüllt zurück zu schicken.
Ende Mai können wir Scannerdaten filtern und auswerten und haben bis auf eine Ausnahme die Fragebögen vorliegen.

ERGEBNISSE

Auswertung Scannerdaten

Die Datengrundlage ist zu gering um eine generelle Aussage zu treffen. Das hat mehrere Gründe. Zum einen haben wir mit nur sieben Testpersonen einen relativ geringen Nutzerkreis, da das Midlet auf vielen Handys nicht lauffähig ist. Desweiteren stellt sich heraus, dass das Scanintervall von 12 Sekunden doch ein wenig zu grob ist. So haben wir des Öfteren Daten erhalten, bei denen die Serververbindung zeitlich vor dem Eintritt in den Scanbereich liegt. Zuletzt ist während des Testzeitraums am Display-System gearbeitet worden, sodass über eine längere Zeit überhaupt keine Scandaten erhoben werden konnten.
Trotzdem sind einige markanten Auffälligkeiten zu nennen. Die Connect Time, also der Zeitabstand zwischen Eintritt in den Scanbereich und Verbindungsaufbau zum Server liegt zwischen 3 Sekunden und 1 Minute 43 Sekunden. Das kann einerseits bedeuten, dass der Verbindungsaufbau nicht sofort funktioniert hat.
Wahrscheinlich hält sich der Nutzer aber zuerst eine gewisse Zeit mit aktiviertem Bluetooth im Bereich auf, um das Display zu lesen, sich zu entscheiden welche Informationen er zugesendet bekommen möchte, um sich dann anschließend zu verbinden.
Die Request Time, also die Zeit zwischen Verbindungsaufbau und abschicken des 1. Request liegt zwischen 2 Sekunden und 51 Sekunden. Die meisten Request Times befinden sich unter 20 Sekunden. Das zeigt, dass die Anwendung schnell ist und sich dazu eignet nebenher ausführen zu lassen.
Anhang A, Tabelle 6 zeigt, dass eine sehr schnelle Lernkurve zu beobachten ist. Bei der ersten Benutzung liegt die Request Time noch bei 40 Sekunden. Danach bleibt sie konstant bei 7 bis 8 Sekunden. Das zeigt, dass das System nach der ersten Benutzung verstanden ist und dann sehr schnell verwendet wird.
Beim gleichen Beispiel wird mehrere Male die Mailadresse geändert. Hier pendelt sich der Zeitraum zwar nicht sofort auf einen konstanten Wert ein. Eine Verkürzung der Email Time ist aber auch hier deutlich sichtbar.

Auswertung Fragebögen

Es liegen uns nur 6 Fragebögen vor. Aufgrund der Antwortstruktur ist trotzdem eine repräsentative Auswertung möglich. (s. Tabelle 2)

Frage\Nutzer	1	2	4	5
Geschlecht	m	m	m	m
Studiengang	Geographie	Geoinformatik	Geoinformatik	Geographie
Alter	26	24	26	24
Hersteller	Siemens	Motorola	Motorola	Siemens
Modell	S65	E77v	PEBL	S65
Bluetooth bereits genutzt	ja	ja	ja	ja
regelmäßige Nutzung	nein	ja	ja	nein
Wofür	Headset, Datenübertragung	Datenübertragung	iDisplays, Adressen tauschen, Synchronisierung	Datentransfer vom/zum Laptop
Java bereits genutzt	ja	ja	ja	ja
regelmäßige Nutzung	nein	ja	nein	nein
Wofür	Rechner, Spiele	Programmieren fürs Studium	iDisplays, spiele	Spiel, Busfahrplan
Bedienung ersichtlich	ja	nein	ja	nicht sofort
Eigeninitiative oder Anleitung	beides	Anleitung	kurze Anleitung	Anleitung
Schwierigkeiten zu Beginn des Testraums	Anmeldung nur mit Code	nein	keine	ja
Schwierigkeiten zu Mitte des Testraums	kein Text in Emails	nein	keine	nein
Schwierigkeiten zum Ende des Testraums	keine	nein	keine	nein
Konzept sinnvoll	ja	ja	ja	ja
Programm in Zukunft nutzbar	vielleicht	ja	jein	ja
Verbesserungsvorschläge	Artikel sollten auch aufs direkt Handy geschickt werden können, nicht nur als Email z.B. Mensaplan, Veranstaltungen, etc		Kein Zusatznutzen momentan (ausser Erinnerung), Termine direkt in Terminkalender pushen wäre _sehr_ sinnvoll (evtl auch Pics Anhang an Email)	Größerer Funktionsumfang; uni-weit

Tabelle 2: Auszug der Fragebögenzusammenfassung

Die Testgruppe besteht zu gleichen Teilen aus Geoinformatikern und Geographen zwischen 24 und 27 Jahren. Der Großteil von Ihnen hat bereits die Java und Bluetooth Funktionalitäten ihrer Handys genutzt, allerdings nicht regelmäßig oder eher für Spielereien.
Obwohl die meisten Nutzer anfangs eine kurze Einweisung bekamen, wurde die Bedienung von der Hälfte der Befragten als nicht ersichtlich eingestuft. Dass die Probleme zum Ende des Testraums als weniger schwerwiegend eingestuft wurden, zeigt, dass die Verwendung schnell erlernbar ist.
Alle Befragten finden das Konzept sinnvoll und können sich vorstellen, es in Zukunft weiter zu benutzen. Es besteht vor allem noch der Wunsch nach Zusatzfunktionalitäten, beispielsweise sollen die Information auch direkt auf das Handy geschickt werden können, anstatt nur per Mail versendet. Und es besteht der Wunsch auch die übrigen Informationen des Displays, wie Mensapläne oder Wetter abfragen zu können.

OFFENE PUNKTE

Wie die Umfrage ergeben hat, besteht der Wunsch nach weiteren Funktionalitäten. Beispielsweise werden bisher nur die Informationen verschickt, die auf dem Display angezeigt werden. Es wäre auch denkbar, weitergehende Informationen zu verschicken.
Andere Wege der Kommunikation zwischen Client und Server können untersucht werden. Mit ObexPush wäre es denkbar, dass der Server einen angezeigten Termin direkt in den Kalender des Handys schreibt. Vielleicht ist auch die Kommunikation über SMS eine Alternative, da fast alle Handynutzer damit vertraut sind und außerdem keine Bluetooth Funktionalität vom Handy benötigt wird. In diesem Zusammenhang müsste geprüft werden, inwiefern Nutzer bereit sind, Geld für diesen Service zu bezahlen.
Es kann genauer untersucht werden, woher die Schwierigkeiten der Lauffähigkeit des Clients bei einzelnen Handymodellen kommen. Beispielsweise könnte eine Verbesserung durch digitale Signierung der Midlets erreicht werden.
Falls ein Scanner auch noch für andere Zwecke als die Datenaufnahme für Testuntersuchungen genutzt werden soll, wäre eine Möglichkeit zum gleichzeitigen Ausführen des Scanns und Entgegennahme von Anfragen sinnvoll. Hierzu wären Alternativen zu Bluecove ein Ansatz. Evtl. könnte eine eigene Hardware-nähere Lösung programmiert werden.
Ebenso ist nach Lösungen für Widcomm Bluetooth Stacks und alternative Betriebssysteme zu suchen.

FAZIT

Wir haben ein funktionierendes System entwickelt, dass es Bluetooth-fähigen Java-Handys ermöglicht, Interaktion mit einem öffentlichen Display herzustellen und über diese Verbindung dem Nutzer persönliche Informationen bereit zu stellen. Die Nutzerakzeptanz ist in der gesamten Breite der Testgruppe gut. Interesse und Erweiterungsvorschläge sind durchgängig vorhanden, sodass eine zukünftige Nutzung erwartet werden kann. Teilweise anfängliche Nutzungsschwierigkeiten lassen schnell nach.
Probleme sind vor allem durch wenig Erfahrung mit Bluetooth Entwicklungen entstanden. Zahlreiche APIs, die die Programmierung erleichtern, sind nicht vorhanden, vor allem nicht für verschiedene Plattformen. Außerdem machen unterschiedliche Funktionsweisen der Handymodelle Probleme. Hier gibt es zwar einheitliche Standards, die aber von den unterschiedlichen Herstellern von Handys nicht immer eingehalten werden und somit keine pauschalen Aussagen über Lauffähigkeit von J2ME Programmen auf allen Modellen zulassen.

QUELLENANGABEN

1. Bluecove Homepage
http://bluecove.sourceforge.net/.

2. Hilfeforum zu Bluecove
http://sourceforge.net/forum/forum.php?thread_id=1489997&forum_id=390056.

3. Projekthomepage
http://www.muckum-server.de/lbs/.

4. Microsoft Bluetooth Stack
http://www.heise.de/mobil/artikel/51923.

5. Bluetooth Klassen
http://de.wikipedia.org/wiki/Bluetooth#Reichweite.

6. CLDC Spezifikation
http://de.wikipedia.org/wiki/CLDC.