Institut für Architektur von Anwendungssystemen
Universität Stuttgart
Universitätsstraße 38
D–70569 Stuttgart

Diplomarbeit Nr. 2991

Entwicklung eines Anwendungs-Konzepts zur Nutzung kontextbezogener Workflows in der Smart Factory

Michael Schäfer

Studiengang:	Softwaretechnik
Prüfer:	Prof. Dr. Frank Leymann
Betreuer:	Dipl.-Inf. Matthias Wieland
begonnen am:	01. Oktober 2009
beendet am:	02. April 2010
CR-Klassifikation:	H.4.1, H.3.4, C.5.3

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Verzeichnis der Listings

1 Einleitung

Unternehmen setzen immer mehr auf die Optimierung ihrer Geschäftsprozesse, um auf einem globalen Markt wettbewerbsfähig bleiben zu können. Die Rolle der IT hat sich dabei in den letzten Jahrzehnten stark gewandelt. Früher beherrschten monolithische Anwendungen die IT-Architektur der Unternehmen. Heute werden Geschäftsprozesse ganzheitlich betrachtet, z. B. vom Zulieferer bis zum Händlerregal. Dazu müssen die beteiligten IT Services durch eine zentrale Komponente überwacht und gesteuert werden. Diese Aufgabe übernimmt ein Workflow Management System. Den Hauptteil der Geschäftsaktivitäten erledigen allerdings Mitarbeiter. Damit auch deren Tätigkeiten vom Workflow Management überwacht werden können, gibt es in der Welt der Workflows das Konstrukt der Human Tasks. Die Benutzerschnittstellen zwischen Human Task und Mensch müssen dabei dem speziellen Einsatzgebiet angepasst sein, denn Mitarbeiter arbeiten selten ausschließlich im Büro an einem PC. Je nachdem, welche Arbeiten verrichtet werden müssen, wechselt der einzelne Mitarbeiter seinen Arbeitsplatz. Die Benutzerschnittstellen sind in der Regel nicht dafür ausgelegt, dass diese von unterwegs genutzt werden. Zusätzlich weisen mobile Tasks ganz spezielle Charakteristiken auf. So spielen die Position der Mitarbeiter und die zur Aufgabe benötigten Werkzeuge eine herausragende Rolle. Herkömmliche Workflow Management Systeme sind für Kontextdaten wie z. B. die räumlichen Positionen nicht ausgelegt. Es existieren zwar schon Konzepte, Kontextdaten in Workflows zu integrieren (Context4BPEL [WKNL07], Context-Integration-Processes [Hau09]), allerdings fehlt ein ganzheitliches Konzept zur flexiblen Bearbeitung von kontextbezogenen Arbeitsaufgaben. Die Anforderungen an ein solches System aus Benutzersicht ändern sich mit dem Kontext, in dem sich der Benutzer befindet. Diese Art von Aufgabenerledigung unterscheidet sich dadurch stark von herkömmlichen Human Tasks und der Bedarf für neue Konzepte schlägt sich vor allem in der Notwendigkeit einer Kartendarstellung für die Liste der Arbeitsaufgaben und der Verwendung einer mobilen App nieder.

1.1 Aufgabe

Die Konzepte von kontextbezogenen Tasks wurden bereits in [Lae09] ausführlich behandelt. Aufbauend auf diesen Erkenntnissen soll ein lauffähiger Prototyp entstehen. Dieser Prototyp soll in der Smart Factory zum Einsatz kommen und dort den Arbeiter bei der Störungsbehebung an einer Maschine unterstützen. Dem Mitarbeiter sollen die dabei anfallenden Einzelaufgaben und verfügbaren Betriebsmittel auf einer Karte angezeigt werden. Mit verschiedenen Anwendungen kann der Mitarbeiter mit diesen Umgebungsmodellobjekten interagieren und diese bearbeiten. Zusammengefasst soll durch den Prototyp ein Arbeiter

die Liste seiner Arbeitsaufgaben (Tasks) einfach erkennen und diese effizient abarbeiten können.

Im Detail sollen die Schnittstellen zwischen der Nexus-Plattform, welche das Umgebungsmodell verwaltet, und dem Workflow Management System, das die Arbeitsaufgaben verwaltet, konzipiert werden. Darauf aufbauend soll ein Anwendungskonzept entwickelt werden, damit die Benutzer die Human Tasks des Störungsprozesses mobil und flexibel bearbeiten können. Die Umsetzung des Konzepts soll durch eine App auf einem Google-Handy mit dem Android-Betriebssystem realisiert werden. Damit der Störungsprozess real vorgeführt werden kann, muss das fertige System vorkonfiguriert werden, z. B. mit den korrekten Geo-Daten.

1.2 Motivation

Unternehmen sind ständig bestrebt, Verbesserungspotential in ihren Prozessen zu erkennen und Optimierungen gezielt einzusetzen. Die Trendforscher von Gartner prognostizieren, dass mobile Anwendungen eins der zehn wichtigsten IT-Themen 2010 ist [WK10]. Hier gibt es viele Möglichkeiten Arbeitsabläufe zu beschleunigen. Ganz nach dem Motto „anytime and anywhere“ soll der Mitarbeiter über Smartphone und Laptop ständig im Kontakt mit seinem Büro stehen. Die Organisation der Aufgaben erledigt der Mitarbeiter bisweilen selbst. Dabei sind die technischen Mittel heutzutage längst gegeben, um den Arbeitsablauf von Mitarbeitern außer Haus zu optimieren. Die personalisierte Worklist ist nicht mehr gebunden an den PC, der im Büro des Mitarbeiters steht, da moderne Smartphones über schnelle Datenverbindungen wie z. B. UMTS mit dem Workflow-Server des Unternehmens verbunden sind und den Mitarbeiter so live über Aufgaben informieren können. So erfährt der Techniker über den Defekt einer Maschine, obwohl er sich nicht in der Nähe des Überwachungs-Terminals aufhält.

1.3 Herausforderungen

Die Schwierigkeiten dieser Arbeit liegen hauptsächlich darin, die bestehenden Systeme und Konzepte in eine ganzheitliche Lösung für die Smart Factory zu integrieren. Hinzu kommt die Neuentwicklung der mobilen App, da es für die kontextbezogene Aufgabenverwaltung bisher keinen mobilen Client gibt.

1.3.1 Datenintegration

Die Datenintegration stellt eine schwierige Aufgabe dar, da die Daten von zwei Systemen konsolidiert werden müssen. Das Umgebungsmodell enthält die Kontextdaten und das Workflow-System die Benutzerdaten sowie alle Daten über die Tasks. Die Daten sind in beiden Systemen unterschiedlich modelliert und stehen in keinem direkten Zusammenhang.

Über Referenzen und IDs lässt sich allerdings ein zusammenhängendes Netz aus den Daten ableiten.

1.3.2 Mobile Anwendungen

Die Entwicklung von mobilen Anwendungen ist von diversen Schwierigkeiten geprägt. Zentrale Punkte sind u. A. die Sicherheit, Bedienbarkeit sowie die Datenübertragung.

Die **Sicherheit** bei mobilen Anwendungen spielt eine sehr wichtige Rolle. Abgesehen von den Sicherheitsproblemen, die die Luftschnittstelle mobiler Datenübertragung mit sich bringt, bergen mobile Geräte die Gefahr, dass ihre Benutzer viel zu sorglos damit umgehen. Täglich werden zahlreiche Handys und Laptops in den Fundstellen abgegeben. Die Gefahr, dass unternehmenskritische Daten in die Hände von Kriminellen geraten, ist sehr hoch, sobald die Mitarbeiter die Geräte außerhalb der Räumlichkeiten des Unternehmens verwenden. Mobile Geräte sind aber gerade für den Einsatz unterwegs konzipiert, daher bedarf es spezieller Sicherungsmaßnahmen. Unternehmenskritische Daten müssen mit Passwörtern geschützt sein und die Daten sollten bestenfalls im Falle eines Verlust des Geräts auch nachträglich gelöscht werden können. Das geht z. B. bei BlackBerry-Geräten über die Killswitch-Funktion [Lim]. Sobald ein BlackBerry-Gerät sich mit einem Mobilfunk-Provider verbindet, wird überprüft, ob der Besitzer das Gerät als verloren gemeldet hat. Wenn ja, dann werden alle Daten gelöscht.

Die **Bedienbarkeit** ist ein wichtiges Kriterium mobiler Anwendungen. Benutzer sind bei der Benutzung mobiler Anwendungen normalerweise unterwegs und viel stärker abgelenkt als im Büro. Das heißt, dass die Benutzung öfter unterbrochen wird, als dies bei Desktop-Anwendungen der Fall ist. Die Informationen auf dem Bildschirm der mobilen Geräte müssen daher auf einen Blick zu erfassen sein. Auch die Bedienung sollte mit so wenigen Befehlen wie möglich auskommen.

Die **Datenübertragung** in mobilen Funknetzen ist kostspielig und nicht immer zuverlässig. Daten sollten also möglichst schon vor der Benutzung auf das mobile Endgerät geladen werden (Data-Hoarding). Außerdem kann die Datenmenge reduziert werden, indem die Daten nach ihrer Wichtigkeit gefiltert werden.

1.4 Aufbau des Dokuments

Kapitel 1 beginnt mit einer Beschreibung der Problemstellung für diese Arbeit. Daran schließt Kapitel 2, welches dem Leser die Grundlagen vermittelt, die für das weitere Verständnis wichtig sind. Anschließend wird das Anwendungsszenario „Störungsprozess“ in Kapitel

3 ausführlich beschrieben. Das Realisierungskonzept für den Prototyp, das u. A. die Architektur umfasst, wird in Kapitel 4 erklärt. In den Kapiteln 5 und 6 werden u. A. die Benutzerschnittstellen für die mobile App sowie den Web Client erläutert. Die besonderen Anforderungen und Konfigurationsmaßnahmen für die Installation in der Smart Factory sind in Kapitel 7 aufgelistet. Die Diplomarbeit schließt mit einer Zusammenfassung in Kapitel 8.

2 Grundlagen und verwandte Arbeiten

Dieses Kapitel gibt eine knappe Einführung in die Themen Workfow-Management-Systeme, Nexus und Mobile Computing. Außerdem werden die wichtigsten Konzepte für kontextbezogene Tasks beschrieben.

2.1 Workflow-Management-Systeme

Ein Workflow-Management-System (WfMS) [LR00] wird vornehmlich in Unternehmen eingesetzt, um Geschäftsprozesse zu überwachen und zu steuern. Dabei kann das WfMS viele einzelne Aktivitäten innerhalb der Prozesse beschleunigen oder sogar gänzlich automatisieren. Die Geschäftsprozesse werden mit Modellierungssprachen wie BPEL [OAS07] modelliert. Durch ein WfMS können diese dann ausgeführt werden.

2.1.1 Die drei Dimensionen von Workflows

Ein Workflow besteht aus den drei Dimensionen Werkzeug, Agent und Kontrollfluss. Der Kontrollfluss gibt an, zu welcher Zeit und nach welchen Kriterien Aktivitäten ausgeführt werden. Das sog. Staff-Query bestimmt, von welchem Agent die Aktivität ausgeführt wird und das Werkzeug gibt an, mit welchem Software-Programm die Aktivität bearbeitet wird. In Abb. 2.1 werden die drei Dimensionen am Beispiel eines Workflows für Bestellungen verdeutlicht (Quelle: [LR00]). Kontrollfluss, Staff-Query und Werkzeuge werden bei kontextbezogenen Workflows zusätzlich durch den Kontext bestimmt. Daher gibt es bei dieser Art von Workflows eine vierte Dimension, die den Kontext beschreibt. So hat z. B. die Position der Mitarbeiter Auswirkungen auf das Staff-Query (vgl. Abschnitt 2.4.1).

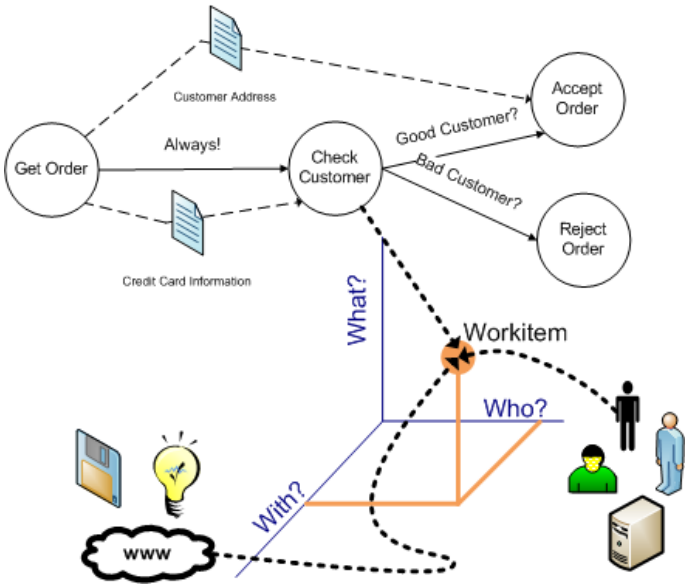

Abbildung 2.1: Drei Dimensionen von Workflows [LRoo]

2.1.2 Zuordnung von Tasks

Aktivitäten in einem Geschäftsprozess, die von einem Menschen bearbeitet werden sollen, werden als Tasks bezeichnet. Zu jedem Task gehört ein Staff Query [LRoo]. Das Staff Query gibt an, von welcher Rolle der Task bearbeitet werden soll. Jeder Mitarbeiter des Unternehmens wird nach seinen Fähigkeiten und seiner Position im Unternehmen einer oder mehreren Rollen zugeordnet. Das WfMS berechnet beim Erzeugen eines Tasks die Liste der Mitarbeiter, die für die Auführung des Tasks durch ihre Rollenzugehörigkeit in Frage kommen. Jedem in Frage kommenden Mitarbeiter wird dann ein Workitem auf dessen Worklist hinzugefügt.

2.1.3 Die Worklist

Über eine Worklist erfährt ein menschlicher Agent, welche Aktivitäten von ihm als nächstes auszuführen sind – die sog. Tasks. Das Füllen der Worklist mit Tasks geschieht entweder mittels Pull, Push oder Grab [LRoo]. Beim Pull-Verfahren obliegt es dem Benutzer, seine Worklist zu aktualisieren. Beim Aktualisieren kommen neu angelegte Tasks hinzu und die Tasks verschwinden, die bereits von anderen Benutzern gestartet wurden. Beim Push-Verfahren ist die Worklist immer aktuell. Neue Tasks werden sofort der Worklist des Benutzers hinzugefügt und Tasks, die von Benutzern gestartet werden, verschwinden von den Worklists der anderen Benutzer. Beim Grab-Verfahren meldet der Benutzer dem WfMS, ihm einen passenden Task zuzuweisen. Jeder Benutzer sieht immer nur einen Task und die Tasks werden nicht gleichzeitig von mehreren Mitarbeitern gestartet, wie es beim Push-Verfahren

möglich wäre. Nach Beenden des Tasks kann der nächste passende Task automatisch vom WfMS gestartet werden, so dass der Benutzer beschäftigt bleibt.

2.2 Kontextbasierte Systeme

2.2.1 Nexus

Der Sonderforschungsbereich 627 mit Namen Nexus beschäftigt sich mit der Frage, wie man Kontextmodelle in mobilen kontextbezogenen Systemen zugänglich macht. Die einzelnen Projekte decken viele interdisziplinäre Themen ab, wie z. B. Sensor-Netzwerke oder Mobile Computing.

Smart Factory Im Projektbereich D1-Smart Factory wird erforscht, welche Optimierungen sich bei der Verwaltung der Betriebsmittel in der Produktion durch Kontextdienste erzielen lassen [LCW08]. Die Dienste zur Lokalisation, Sensorintegration, der Kommunikation und der geometrischen Modellierung der Produktionsumgebung nehmen dabei eine wichtige Rolle ein. Durch die Ergebnisse dieser Forschung erhofft man sich Steigerungen in Qualität sowie Flexibilität und eine deutliche Reduzierung der Kosten. Ein einfaches Beispiel ist die Überwachung der Produktionswerkzeuge. Die Mitarbeiter können vom Umgebungsmodell zu jeder Zeit erfahren, wo sich das gesuchte Werkzeug befindet und ob es gerade verfügbar ist. Das spart Zeit und unnötige Kommunikation bei der Suche wird vermieden.

In [LW07] wird ein Kontextdatenmodell zur Erfassung aller Realwelt-Objekte in der Smart Factory beschrieben. Die Daten sind für die Umsetzung von kontextbezogenen Anwendungen notwendig, denn so kann beispielsweise der Zustand einer Maschine in den Workflow mit einbezogen werden, um Störungen automatisch zu erkennen. Das Datenmodell wird in dieser Arbeit verwendet.

Föderation aus Location Server und Spatial Model Server Die Plattform des Nexus-Projekts wird unterteilt in Spatial Model Server, die statische Objekte, wie z. B. Gebäude und Straßen, verwalten und in den Lokationsdienst, der mobile Objekte, wie z. B. die Benutzer selbst, verwaltet [Moto3]. Die Verteilung der Objekte auf die Server wird durch eine Föderationkomponente verborgen. Alle Objekte können über die Föderation mit der Query-Sprache AWQL abgefragt und mit AWML angelegt und bearbeitet werden. Durch die Verwendung der Föderation können die Daten räumlich große Gebiete abdecken, ohne dass es zu Performance-Einbußen kommt [GBH+05].

2.2.2 WebFeatureService

Zur Nexus-Plattform gibt es ein alternatives Konzept, welches vom Standardisierungsgremium OpenGISConsortium konzipiert wurde. Mit dem Standard GML (Geography Markup

Language, [gml07]) werden geographische Daten modelliert. Die GML-Daten werden über einen WebFeatureService verfügbar gemacht. Die Schnittstellen des WebFeatureService werden vom Standard vorgegeben und sind als WebService zu implementieren. Die genaue Implementierung liegt allerdings im Ermessen des Herstellers. Die Schnittstellen und Dienste des OpenGISConsortium sind im OpenGIS-Standard ähnlich definiert wie die Nexus-Dienste. In [Fla04] ist ein Wrapper entwickelt worden, mit dem beide System-Varianten miteinander genutzt werden können.

2.3 Mobile Computing

Mobile Endgeräte haben sich in den letzten Jahren zu immer leistungsstärkeren Geräten weiterentwickelt. Die Bandbreite mobiler Geräte ist sehr groß: Angefangen bei kleinen Mobiltelefonen, über SmartPhones bis hin zu Laptops. Allen mobilen Geräten ist aber gemein, dass sie mit knappen Akkulaufzeiten, wenig Speicher und schwacher Rechenleistung ausgestattet sind. Hinzu kommt das Problem mit der Internetverbindung. Zwar sind schnelle Datenverbindungen wie UMTS mittlerweile flächendeckend möglich, dennoch tragen die Eigenschaften mobiler Funknetze zu häufigen Verbindungsabbrüchen bei. Das Übertragungsmedium Luft ist nicht nur sehr störungsanfällig, sondern birgt auch etliche Gefahren bei der Sicherheit der Datenübertragung. Da mobile Endgeräte für gewöhnlich in jede Tasche passen müssen, sind die Möglichkeiten der Interaktion stark eingeschränkt. Oftmals hat ein Benutzer unterwegs nur eine Hand frei, mit der er das Gerät gleichzeitig halten und bedienen will. Diese Anforderung stellt hohe Ansprüche an das Design von mobilen Anwendungen, welche intuitiv und effizient bedienbar sein sollten.

2.3.1 Entwicklungsmodelle mobiler Anwendungen

Für die Entwicklung mobiler Anwendungen kommen mehrere Entwicklungsmodelle in Frage. Oftmals reicht es, ein schon vorhandenes **Web-Interface** für die Darstellung auf kleinen Endgeräten zu optimieren, da Smartphones heutzutage mit Internet-Browsern ausgestattet sind. Dabei ist aber der Verzicht auf große Bilder und komplexe Tabellen angebracht, da diese auf den kleinen Bildschirmen nur unzureichend dargestellt werden. Der Vorteil bei dieser Methode liegt darin, dass man keine Anwendung auf den mobilen Geräten installieren muss. Eine weitere Möglichkeit ist die Entwicklung eines **Thin Clients**. Beim Thin Client werden die Benutzeroberflächen z. B. mit JavaME [SMa] entwickelt und können so besser auf die Eigenheiten der mobilen Endgeräte abgestimmt werden. Die Daten der Anwendung werden hierbei nicht auf dem mobilen Gerät gespeichert, sondern immer von einem Server angefordert, sobald der Benutzer diese anzeigen möchte. Der **Rich Client** speichert die Daten im Gegensatz zum Thin Client lokal, damit diese auch ohne Verbindung zum Server abrufbar sind. Dieses Modell stellt aber hohe Anforderungen an die Speicherkapazität der Endgeräte und kommt somit nur bei Anwendungen mit geringem Datenaufkommen in Frage.

	J2ME	Android	Windows Mobile	iPhone
Hersteller	Sun Microsystems, Inc.	Google	Microsoft	Apple
Plattform-unabhängigkeit	Ja	Nein	Nein	Nein
Programmier-sprache	Java	Java	C++ / C#	Objective C
App Store	Nein	Ja	Nein	Ja

Tabelle 2.1: SDKs für mobile Anwendungsentwicklung

Für mobile Anwendungsentwicklung gibt es zahlreiche SDKs (Software Development Kits). Die SDKs unterscheiden sich vor allem stark bei den verwendeten Programmiersprachen und den Möglichkeiten für die Gestaltung der Benutzeroberflächen. Java ME (Java-Plattform, Micro Edition) ist eine mobilfunkherstellerunabhängige Laufzeitumgebung, die – wie der Name schon sagt – auf Java beruht. Das besondere an Java ME ist, dass durch die beiden Spezifikationen CLDC (Connected Limited Device Configuration, [SMb]) und MIDP (Mobile Information Device Profile, [SMa]) herstellerunabhängige Schnittstellen vorgegeben werden, gegen die Anwendungsentwickler ihren Code implementieren können. Die Verbreitung von Mobilfunkgeräten, die Java ME unterstützen, ist sehr hoch. Demgegenüber stehen SDKs, welche nur vom speziellen Betriebssystem des mobilen Geräts unterstützt wird. Für Mobilfunkgeräte, die das Android-Betriebssystem von Google haben, gibt es ein SDK, dass auf Java basiert. Die Entwicklung von Windows Mobile Anwendungen wird mit dem auf C++ und C# basierenden SDK von Microsoft bewerkstelligt. Beim SDK für das iPhone von Apple werden die Anwendungen in Objective C programmiert. Objective C basiert auf der Programmiersprache C und hat Spracherweiterungen für objektorientiertes Programmieren. Objective C ist aber nicht gleichzusetzen mit C++. Allen SDKs gemeinsam ist, dass sich die mobilen Anwendungen mit einem Emulator testen lassen und man zum Entwickeln kein Mobilfunkgerät besitzen muss. Android- und iPhone-Anwendungen lassen sich effizient über den App Store von Apple bzw. den Android-Market von Google vermarkten.

2.4 Verwandte Arbeiten

2.4.1 Konzepte für kontextbezogene Arbeitsaufgaben-Verwaltung

Grundlage für diese Diplomarbeit ist [Lae09], in welcher der Frage nachgegangen wird, welche Erweiterungen notwendig sind, damit kontextbezogene Arbeitsaufgaben vom WfMS verwaltet werden können. Dazu wurden die kontextbezogenen Arbeitsaufgaben analysiert und in BPEL-Prozessen realisiert. Die BPEL-Prozesse verwenden die Nexus-Plattform, um

Kontextdaten in die Workflows mit einzubeziehen. Ergebnis dieser Arbeit sind vier BPEL-Prozesse, die jeweils einen Typ von kontextbezogenen Arbeitsaufgaben implementieren. Die folgenden Abschnitte fassen die wichtigsten Konzepte von [Lae09] zusammen.

Die kontextbezogenen Arbeitsaufgabentypen In [Lae09] werden kontextbezogene Arbeitsaufgaben als Ausführung einer physischen Tätigkeit definiert, welche somit immer an einen Ort gebunden sind. Das Ergebnis dieser physischen Tätigkeit muss zudem mit Sensoren messbar sein und sich in einer Veränderung im Umgebungsmodell widerspiegeln. Kontextbezogene Arbeitsaufgaben werden in vier Basistypen unterteilt:

- Der **Query Task** stellt seinem Bearbeiter Fragen über einen Gegenstand der Umgebung. Beispiel: Zustand einer Maschine überprüfen.
- Der **Move Task** fordert seinen Bearbeiter auf, an einen bestimmten Ort zu gehen. Beispiel: Zu einer Besprechung gehen.
- Der **Transport Task** dient dazu, Gegenstände von einem Ort zu einem anderen Ort zu überführen. Beispiel: Anlieferung von Ersatzteilen.
- Der **Operate Task** fordert eine mechanische Arbeit an einem Objekt. Beispiel: Wartungsarbeiten an einer Maschine.

Ermitteln der potentiell zuständigen Mitarbeiter Da kontextbezogene Arbeitsaufgaben an einen Ort gebunden sind, bietet es sich an, nur solchen Mitarbeiter einen Task zuzuweisen, welche sich auch in der Nähe befinden. Die Filterung der Tasks, für die ein Mitarbeiter zuständig ist, richtet sich also bei kontextbezogenen Tasks nicht mehr nur nach der Rolle der Mitarbeiter sondern auch nach deren Position. Die Filterung der Tasks kann auf zwei Arten erfolgen: Bei der serverseitigen Filterung ist die Position dem WfMS bekannt und die Worklist des Mitarbeiters wird entsprechend um Tasks reduziert, bei denen der Mitarbeiter zu weit entfernt ist. Die serverseitige Filterung ist aber kritisch in Hinblick auf die Datenschutzbestimmungen in Deutschland, da Angestellte nicht bei ihrer Arbeit überwacht werden dürfen [JGH+06]. Unbedenklich ist die lokale Filterung mittels Kartenausschnitt (visuelle Filterung), da die Positionsdaten auf dem Endgerät des Mitarbeiters verbleiben. Allerdings entsteht bei dieser Methode ein erhöhtes Datenaufkommen.

Anzeigen der kontextbezogenen Arbeitsaufgaben Verglichen mit konventionellen Tasks haben kontextbezogene Tasks zusätzliche Daten aus dem Umgebungsmodell. Diese zusätzlichen Daten sollten in der Benutzeroberfläche dargestellt sein. Am besten eignet sich für die Positionsdarstellung eine Kartenansicht, in der die Tasks und Objekte mit Icons dargestellt werden. Wenn die kontextbezogenen Arbeitsaufgaben innerhalb von mehrstöckigen Gebäuden zum Einsatz kommen, ist die Visualisierung der Etagen unverzichtbar. Eine hilfreiche Zusatzfunktion ist die Anzeige von Routen, damit der Mitarbeiter immer auf dem kürzesten Weg zu seinem Ziel gelangt.

Kontextbezogene Aufgaben für mehrere Mitarbeiter Bei vielen Aufgaben, wie z. B. komplexen Reperaturen, werden mehrere Mitarbeiter benötigt. Solche Tasks müssen zeitlich koordiniert werden, damit für keinen Mitarbeiter unnötige Wartezeiten entstehen, solange der andere sich noch auf dem Weg zum Ort der Aufgabe befindet. Eine integrierte Terminplanung ist eine mögliche Lösung.

Anmerkung zum Move Task Die Sinnhaftigkeit des Move Task erschließt sich nicht aus Benutzersicht. Die Intention von [Lae09] für den Move Task war bestimmt nicht die Bewegung zum Selbstzweck, da ja auch das Beispiel mit der Besprechung angebracht wird. Allerdings kann man in solchen Fällen genausogut den Operate Task einsetzen, da der Beauftragte am Zielort eine Tätigkeit vollführen muss – und sei es nur die bloße Anwesenheit. Da die Mitarbeiter nicht überwacht werden dürfen [WLL+09], entfällt auch ein entscheidendes Kriterium, das in der Definition von kontextbezogenen Tasks verlangt wird, nämlich dass das Ergebnis der Aufgabe im Umgebungsmodell widergespiegelt wird. Das hieße allerdings, dass die Position des Mitarbeiters im Umgebungsmodell gespeichert wird und somit eine Überwachung stattfindet.

2.4.2 PerFlows

Persönliche kontextsensitive Prozesse werden in [UHW+09] als PerFlow bezeichnet. Dies sind Workflows für den persönlichen Alltag. Ein Benutzer modelliert seine alltäglichen Aufgaben in PerFlows und führt diese mit einem PerFlow-Manager aus. Ein PerFlow-Manager ist eine Art persönlicher, ubiquitärer Assistent [Hub08] und übernimmt Aufgaben wie z. B. das Vorheizen der Wohnung, wenn der Benutzer sich auf dem Nachhauseweg befindet. PerFlows eignen sich allerdings nur für den persönlichen Gebrauch und sind für eine kontextbasierte Aufgabenverwaltung in der Smart Factory nicht ausgelegt. Die PerFlows verschiedener Benutzer können nämlich nicht miteinander koordiniert werden.

3 Anwendungsszenario: Störungsprozess

Der Einsatz von Maschinen ist essentieller Bestandteil im Produktionsablauf von Unternehmen. Mitarbeiter werden vornehmlich zur Überwachung und Steuerung dieser Maschinen eingesetzt. In der Regel ist der Ausfall einer Maschine mit sehr hohen Kosten verbunden, weshalb Unternehmen viel in die Optimierung von Störungsprozessen investieren. Ein Störungsprozess koordiniert die Tätigkeiten, die zur Behebung einer Maschinenstörung notwendig sind. Im weiteren Sinne kann man zu einem Störungsprozess auch die Aktivitäten zählen, die mit einer Störung im Produktionsbetrieb einhergehen. So wird z. B. der Kunde darüber informiert, dass sich die Auslieferung seines Produkts verzörgert. Eine kontextbezogene Aufgabenverwaltung hilft nicht nur bei der Koordinierung des Störungsprozesses, sondern optimiert diesen auch, indem benötigte Ersatzteile und Werkzeuge durch das Umgebungsmodell in den Prozess mit eingebunden werden. Somit weiß der Mitarbeiter zu jeder Zeit, wo sich die benötigten Betriebsmittel befinden. Im folgenden ist der Störungsprozess im engeren Sinne gemeint.

3.1 Überblick

Abbildung 3.1 zeigt die groben Phasen des Störungsprozesses. Eine Störung wird erkannt und muss analysiert werden, bevor entsprechende Maßnahmen eingeleitet werden können. Die Ausführung der Maßnahmen wird protokolliert und zeitgleich überwacht, z. B. ob Zeitvorgaben eingehalten werden.

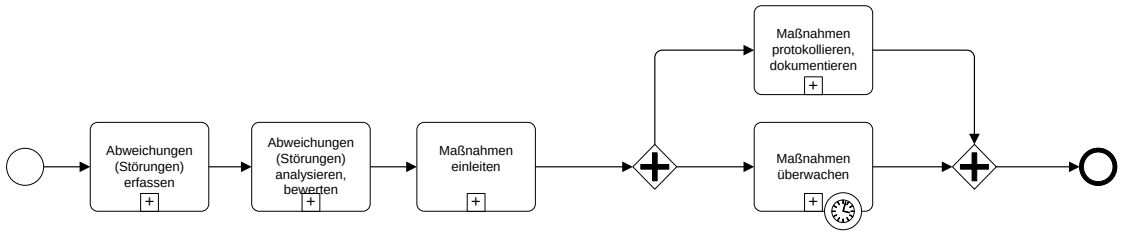

Abbildung 3.1: Der Störungsprozess im Überblick

3.2 Ablauf im Detail

In der Smart Factory tritt eine Störung auf und es wird ein Failure Event erzeugt. Der Störungsprozess wird gestartet und erhält als Parameter den Nexus Object Locator (NOL) des Failure Events (Abbildung 3.2). Die erste Aktivität im Störungsprozess besteht darin, den Fehler zu erfassen und anhand der NOL die Fehlerdaten vom Nexus-Umgebungsmodell anzufordern. Mittels der Fehlerdaten können die Kontextdaten der betroffenen Maschine sowie alle Störungsfälle, die in der Vergangenheit aufgetreten sind, abgerufen werden. Zeitgleich wird ein Subprozess gestartet, der Eskalationsmaßnahmen einleitet, falls die Störung nicht innerhalb eines definierten Zeitraums behoben ist. Zu den Eskalationsmaßnahmen gehört z. B., dass Vorgesetzte darüber in Kenntnis gesetzt werden, wenn Aufgaben nicht erledigt werden. Die Art der Eskalationsmaßnahmen richtet sich nach der Klassifikation des Fehlers. Je nachdem, wie gravierend sich die Folgen einer Störung für den Produktionsprozess auswirken, wird die Störung in die Fehlerklassen A, B oder C eingeordnet. Dabei entscheidet vor allem die benötigte Zeit zur Behebung des Fehlers über dessen Klassifikation.

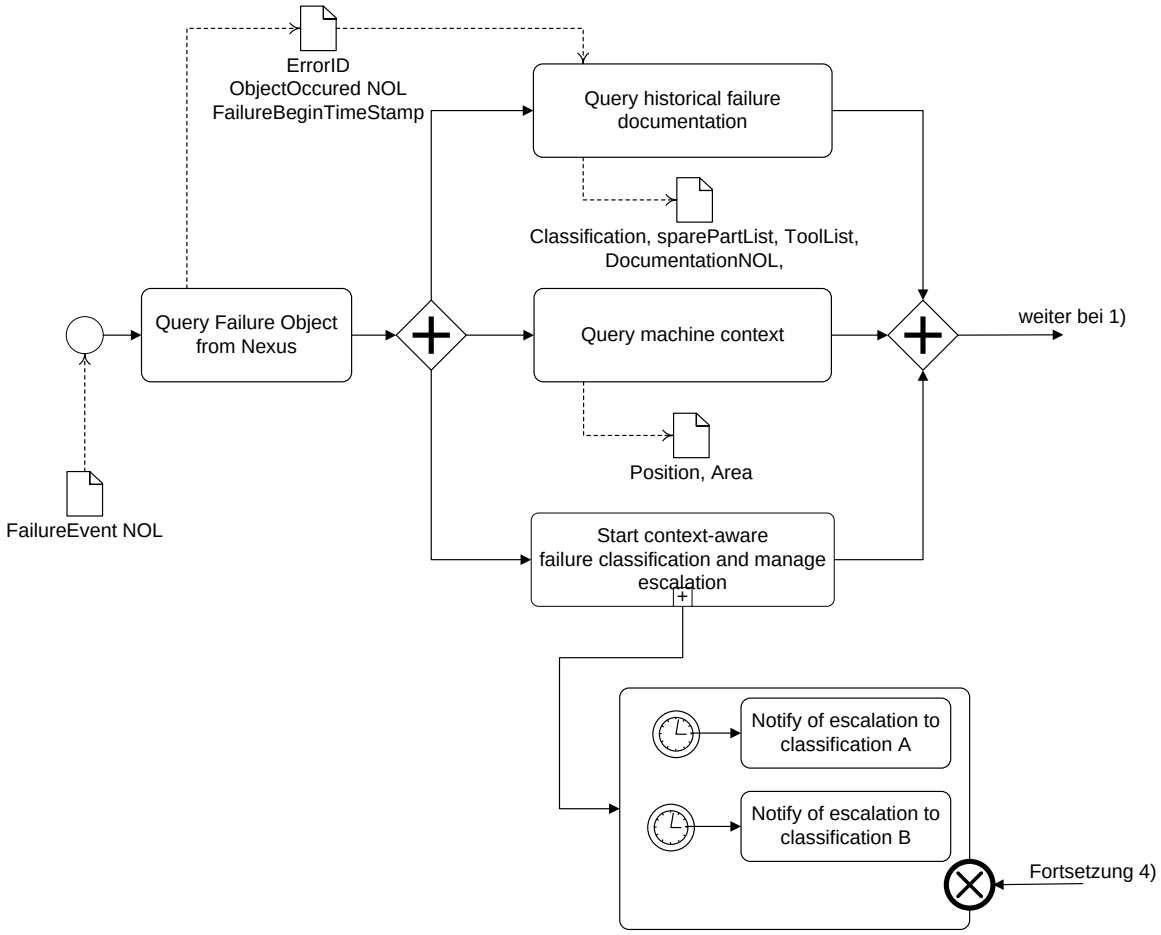

Abbildung 3.2: Der Störungsprozess (1/5)

Auf Basis der Analyse der historischen Störungsdaten lässt sich entscheiden, ob die Störung schon einmal in der Art aufgetreten ist (Abbildung 3.3). Ist dies der Fall, dann kann auf Grund der historischen Dokumentation ermittelt werden, welche Werkzeuge und Ersatzteile zur Behebung der Störung erforderlich sind. Für jedes einzelne Werkzeug und jedes Ersatzteil wird ein Transport Task als Subprozess erzeugt, damit die benötigten Ressourcen zur defekten Maschine gebracht werden. Wenn die Störung bisher unbekannt ist, wird eine neue Fehler-Dokumentation erzeugt.

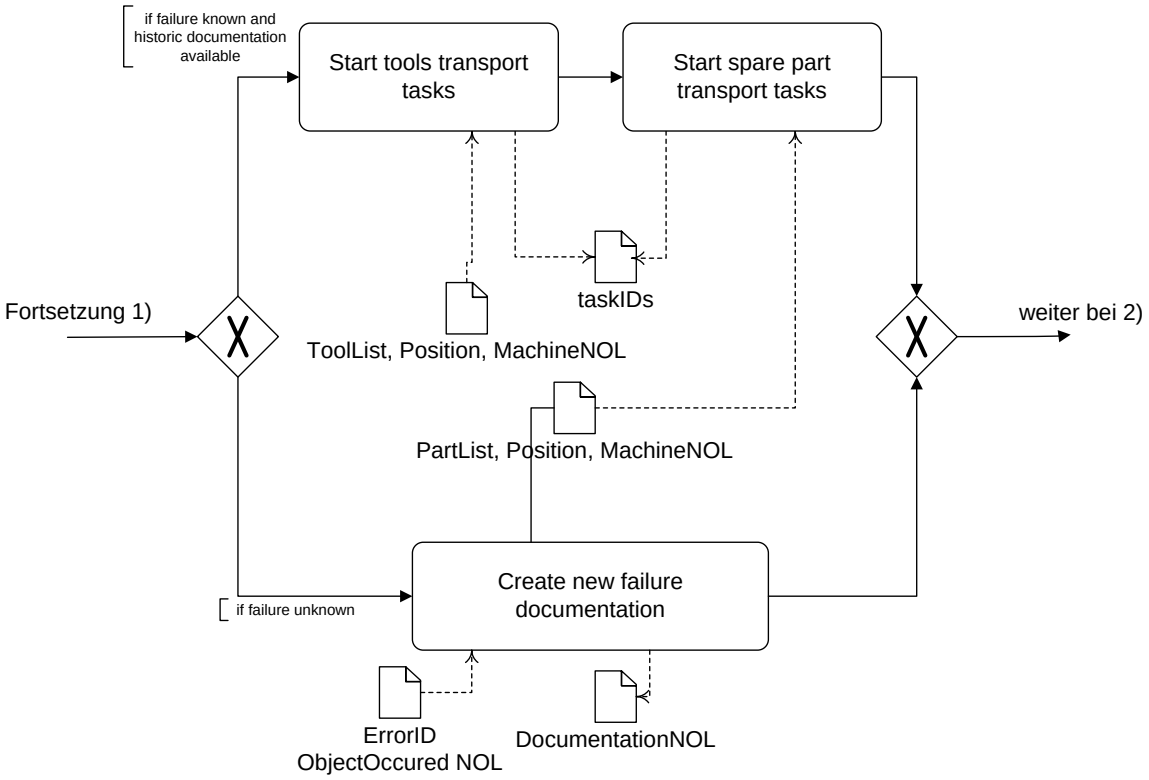

Abbildung 3.3: Der Störungsprozess (2/5)

Für die eigentliche Behebung der Störung wird ein Operate Task mit den Daten über Position und NOL der Maschine angelegt (Abbildung 3.4). Wenn eine historische Dokumentation vorhanden ist werden zusätzlich die NOLs der benötigten Ressourcen mit dem Operate Task verknüpft. Parallel zum Operate Task läuft ein Subprozess ab, der anhand eines in einen Handschuh integrierten RFID-Lesegeräts erkennt, welche Werkzeuge und Ersatzteile der ausführende Mitarbeier in die Hand nimmt. Die Liste der verwendeten Ressourcen ist für die Fehler-Dokumentation vorgesehen.

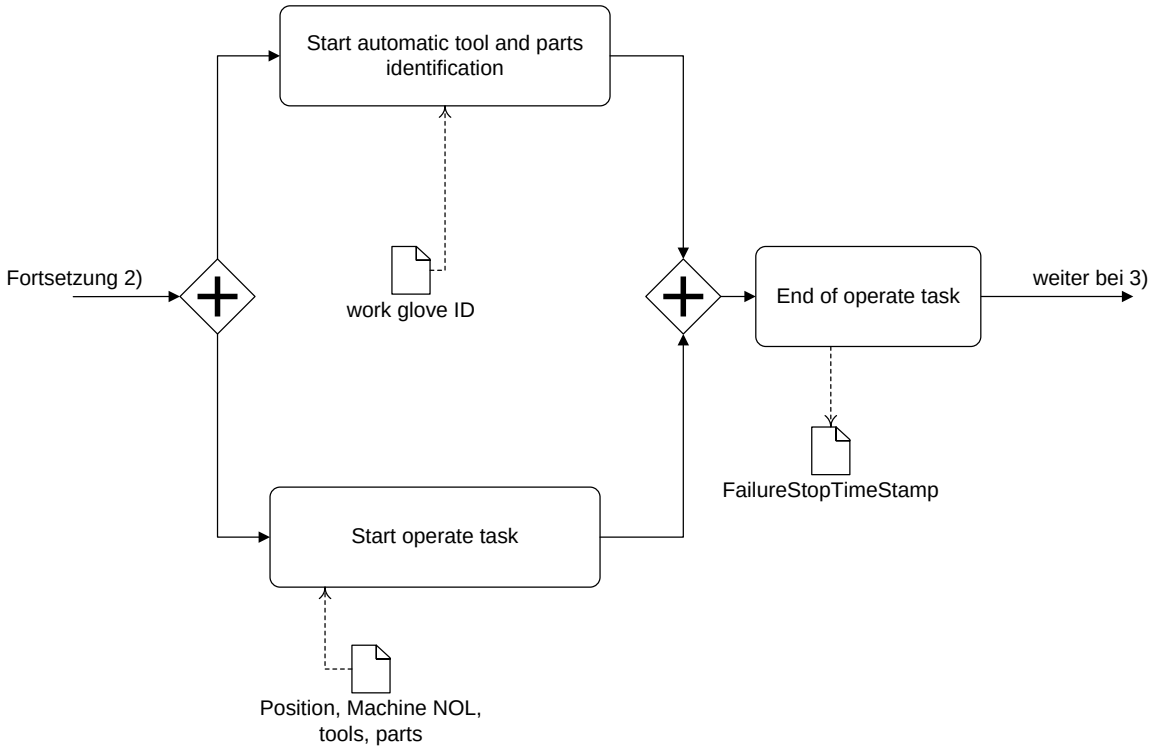

Abbildung 3.4: Der Störungsprozess (3/5)

Sobald der Operate Task erfolgreich beendet wurde, wird zum einen der Subprozess für den RFID-Handschuh gestoppt und zum anderen der Subprozess für die Eskalationsmaßnahmen über die erfolgreiche Behebung des Fehlers benachrichtigt (Abbildung 3.5). Der Zeitpunkt der erfolgreichen Fehlerbehebung wird für die Überprüfung der automatischen Klassifizierung des Fehlers verwendet. Außerdem errechnet sich daraus ein neuer Wert für Mean Time Between Failure und Mean Time To Repair. Die beiden Werte sind für die Produktionsplanung strategisch wichtig, weil sie Aufschluss darüber geben, wie oft bzw. wie lange der Produktionsprozess auf Grund von Fehlern unterbrochen werden muss. Falls es noch Tranpsort Tasks gibt, die nicht vom Mitarbeiter ausgeführt wurden, z. B. weil ein Ersatzteil nicht benötigt wurde, werden diese ausstehenden Tasks beendet. Falls es keine historische Fehler-Dokumenation gegeben hat, wird ein Query Task für den Mitarbeiter angelegt, damit er die Art des Fehlers manuell klassifizieren kann.

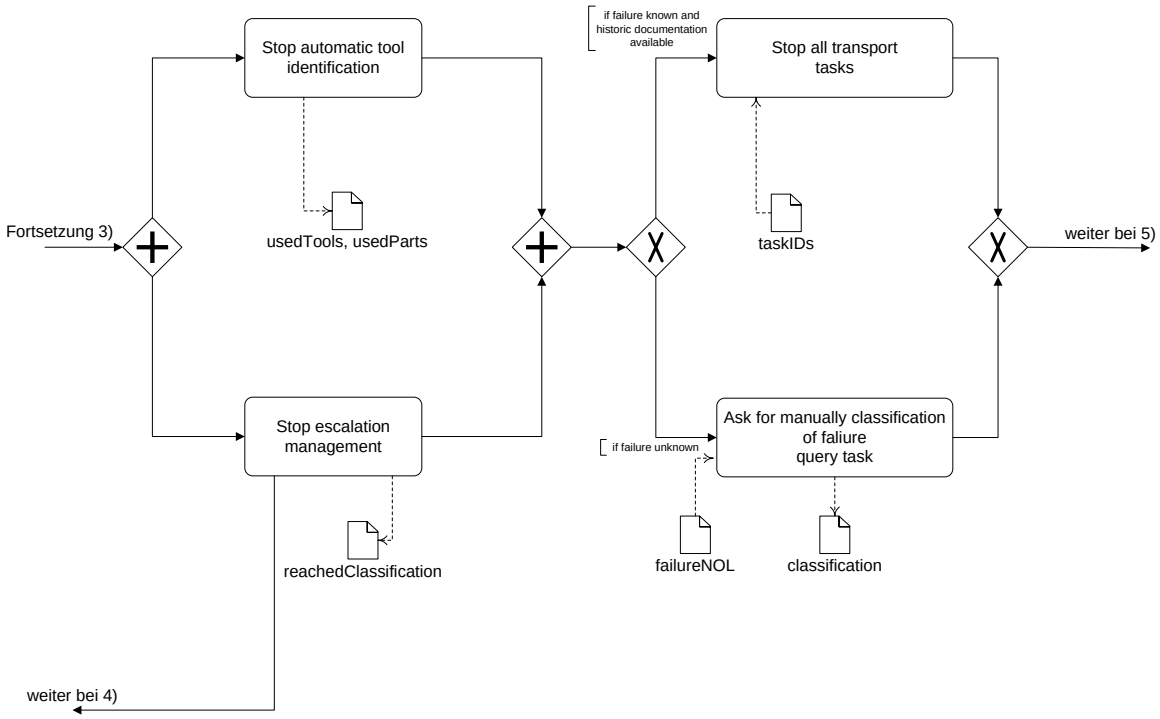

Abbildung 3.5: Der Störungsprozess (4/5)

Um den Störungsprozess abzuschließen, wird noch ein Query Task erstellt, damit der Mitarbeiter die automatisch erfasste Fehler-Dokumentation überprüfen kann (Abbildung 3.6). Wenn der Mitarbeiter den Inhalt der Dokumentation akzeptiert, wird diese in das Umgebungsmodell der Nexus-Plattform neu aufgenommen bzw. die historische Dokumenation im Nexus aktualisiert.

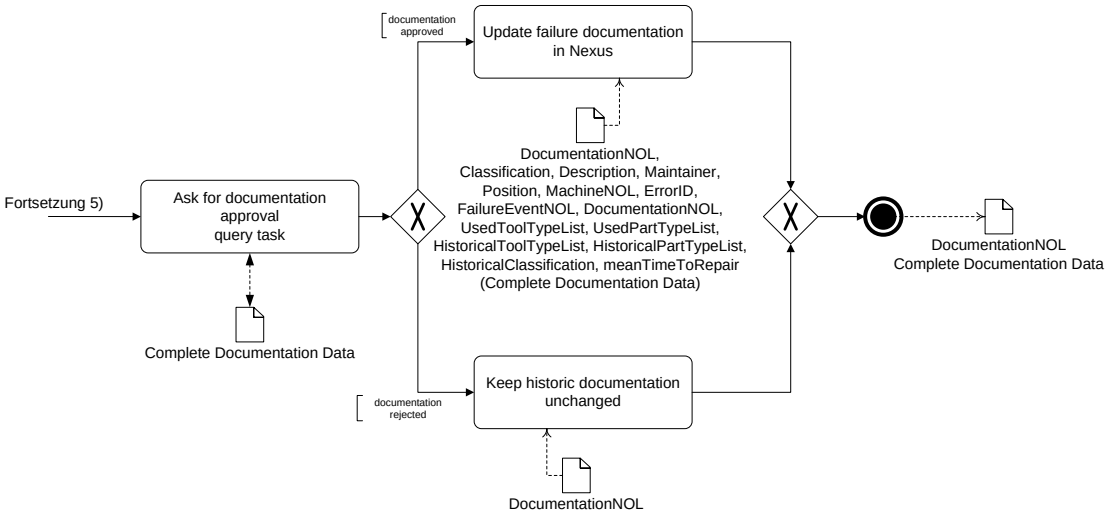

Abbildung 3.6: Der Störungsprozess (5/5)

3.3 Störungsprozess aus Benutzersicht

Im Störungsprozess kommen drei Arten der kontextbezogenen Tasks vor: Mehrere Transport Tasks, ein Operate Task und zwei Query Tasks. Für die Transport Tasks und den Operate Task ist eine mobile Lösung zwingend, denn die Aufgaben müssen vor Ort erledigt werden. Beim Query Task, der den Benutzer die automatisch erfasste Dokumentation prüfen lässt, ist die bereits vorhandene Unterstützung durch die Oracle BPEL-Arbeistliste auf Grund der Übersichtlichkeit der Datenmenge besser geeignet als die kompakte Task App. Wenn der Benutzer allerdings die Positionen der dokumentierten Ressourcen abrufen möchte, kann er dies bequem über den Web Client machen, der eine Kartenansicht enthält. Deshalb muss der Web Client von der Oracle BPEL-Arbeitsliste aus für den Benutzer erreichbar sein (z. B. über einen Link). Da der Störungsprozess nur ein Spezialfall für die Verwendung von kontextbezogenen Tasks ist, wird die Task App auch über eine Integration des Query Tasks verfügen, da der Query Task in anderen Szenarien durchaus vor Ort erledigt werden muss.

4 Realisierungskonzept

In diesem Kapitel findet sich eine Beschreibung des Realisierungskonzepts. Zunächst wird auf die Anforderungen für das Konzept eingegangen, um anschließend den Entwurf der System-Architektur vorzustellen.

4.1 Anwendungskonzept

Zuallererst muss die Anwendung „kontextbezogene Aufgabenverwaltung“ genauer eingegrenzt werden, um klarzustellen, welche Erweiterungen überhaupt notwendig sind. Im weitesten Sinne existiert die Anwendung für die kontextbezogene Aufgabenverwaltung bereits, wenn man die BPEL-Prozesse der Context Tasks als Anwendung betrachtet. Der Benutzer kann mit dem Workflow interagieren und zieht für sich den Nutzen aus der Logik der Geschäftsregeln, die in BPEL abgebildet sind. Genauso gibt es bereits Anwendungen, über die der Benutzer Informationen vom Umgebungsmodell anfordern kann. Es wird deutlich, dass es bisher aus Benutzersicht keine Anwendung gibt, die sowohl die Interaktion mit dem Workflow als auch den Datenaustausch mit dem Umgebungsmodell unterstützt. Hier bedarf es einer Integrationslösung auf der Ebene des Benutzers.

Die Anforderungen an eine solche Lösung sind aber abhängig von der Art der Verwendung und vor allem von der Situation des Benutzers. Ist er zum Beispiel gerade dabei einen Transport Task zu erledigen, so ist eine mobile Lösung unbedingt erforderlich. Der Vorteil einer mobilen Lösung ist zudem, dass mit modernen Smart-Phones der Kontext des Benutzers z. B. mittels GPS-Sensor ermittelt werden kann. Dem gegenüber steht die Situation des Benutzers, bei der er sich einen Überblick über die Aufgaben des bevorstehenden Tages machen möchte. Hier ist eine mobile Lösung auf Grund von den Beschränkungen, die mit mobilen Geräten einhergehen, weniger gut geeignet. Als gleichwertige Variante zur mobilen Lösung bietet sich ein Web Client an, da dieser den ganzen Bedienkomfort einer Desktop-Anwendung besitzt, zur Ausführung auf dem Zielrechner aber nichts installiert sein muss außer einem Internet-Browser. Der Web Client muss also nur auf dem Application Server installiert werden. Das erleichtert die Konfiguration und Wartung.

Die mobile Lösung wird für diese Arbeit im folgenden nur noch *Task App* genannt, da sie keine eigenständige Anwendung sondern vielmehr eine mobile Datenintegrationslösung auf Benutzerebene ist.

	Task App	Web Client
Installation	auf allen mobilen Geräten	einmal auf dem AS
Benutzeroberfläche	klein, kompakt	groß, detailreich
Mobilität	sehr mobil	eingeschränkt mobil
Plattformunabhängigkeit	kaum	erfüllt
Zusatzfeatures	Benutzerkontext abrufbar	–

Tabelle 4.1: Mobile App vs. Web Client

4.1.1 Automatisches Beenden von Tasks

Die Interaktionen mit Tasks beschränken sich nicht nur auf die Benutzer des Workflow-Systems, denn bei kontextbezogenen Workflows spielt die Nexus-Plattform auch eine aktive Rolle. So soll z. B. das Umgebungsmodell für einen Transport Task erkennen, wann ein Objekt seinen Zielort erreicht hat, und den Task dann beenden. Die Frage, welche Workflow-Rolle der Nexus-Dienst zugewiesen bekommt, kann nicht allgemein beantwortet werden. Unter Umständen muss zwischen mehreren Nexus-Rollen unterschieden werden, je nachdem, welche Rechte man den einzelnen Rollen zugestehen möchte. Wichtig hierbei ist vor allem die Nachvollziehbarkeit des kausalen Zusammenhangs zwischen Veränderungen im Umgebungsmodell und der dadurch initiierten Änderung im Workflow-System.

In den Statistiken der Workflow-Systeme können Benutzer einsehen, wie viele Tasks sie bereits erledigt haben. Solche Statistiken sind zum einen für die Mitarbeiter motivierend und bieten zum anderen den Vorgesetzten eine Möglichkeit, die Arbeitsleistung der Mitarbeiter zu beurteilen. Wenn das Nexus-System einen Task automatisch beendet, muss also nicht nur nachvollziehbar sein, welcher Mitarbeiter die zur Beendigung des Tasks notwendigen Veränderungen im Umgebungsmodell herbeigeführt hat, der erledigte Task muss auch der Statistik des Benutzers gutgeschrieben werden.

Die Nachvollziehbarkeit des eigentlichen Benutzers, der den Task erledigt hat, ist noch von anderer wichtiger Bedeutung. In Unternehmen muss oft aus rechtlichen Gründen nachvollziehbar sein, welcher Mitarbeiter Aktionen durchgeführt oder bestimmte Entscheidungen getroffen hat. Dies ist sehr kritisch, wenn es um Haftungsfragen geht. Je nachdem, ob die Konsequenzen bei Fehlverhalten gravierend sind, müssen die Namen der beteiligten Mitarbeiter zu einem späteren Zeitpunkt noch abrufbar sein.

Für die eigentliche Implementierung eines Services, der Tasks automatisch beendet, ist demnach entscheidend, ob ein Task bereits von einem Mitarbeiter angenommen wurde. Denn nur dann gibt es einen Mitarbeiter, der sich für den Task verantwortlich erklärt hat. Das automatische Beenden von Tasks, die noch nicht angenommen worden sind, sollte also unterbunden werden. Das manuelle Beenden eines Tasks durch einen Mitarbeiter ist hingegen unkritisch. Damit den Mitarbeitern die erledigten Tasks gutgeschrieben werden, bietet der TaskService von Oracle die Option, dass ein Benutzer, welcher über Administrationsrechte verfügt, sich in Vertretung von anderen Benutzern anmeldet. Alle getätigten Aktionen im

Workflow-System werden dann so behandelt, als hätte der andere Benutzer die Änderungen vorgenommen.

4.1.2 Position eines Tasks

Die Position eines Tasks ist dafür entscheidend, welche Mitarbeiter für die Erledigung des Tasks überhaupt in Frage kommen. Allerdings ist die Frage, wohin ein Task positioniert werden soll, nicht einfach zu beantworten. Für den Transport Task kommen z. B. mindestens drei Positionen in Frage: Die Startposition, an der sich die zu transportierende Ressource befindet, die Zielposition oder sogar die Position der Ressource selbst, die sich ändert, sobald der Mitarbeiter den Gegenstand transportiert. Selbst für den Operate Task kann die Position des Tasks nicht allgemein verbindlich festgelegt werden. Die Position der Ersatzteile, die für den Operate Task benötigt werden, kann nämlich auch ein wichtiger Faktor sein, welche Mitarbeiter den Task am ehesten zugewiesen bekommen sollten. In dieser Arbeit wird die Position der Tasks nicht aktualisiert, sondern verbleibt an einer Position.

4.2 Architektur

Abbildung 4.1 zeigt die wesentlichen Schnittstellen zwischen den verschiedenen Systemen, die zu einer kontextbezogenen Aufgabenverwaltung in der Smart Factory notwendig sind. Das Gesamtsystem besteht grob zusammengefasst aus den drei Komponenten WfMS, Nexus-System und SmartFactoryClient.

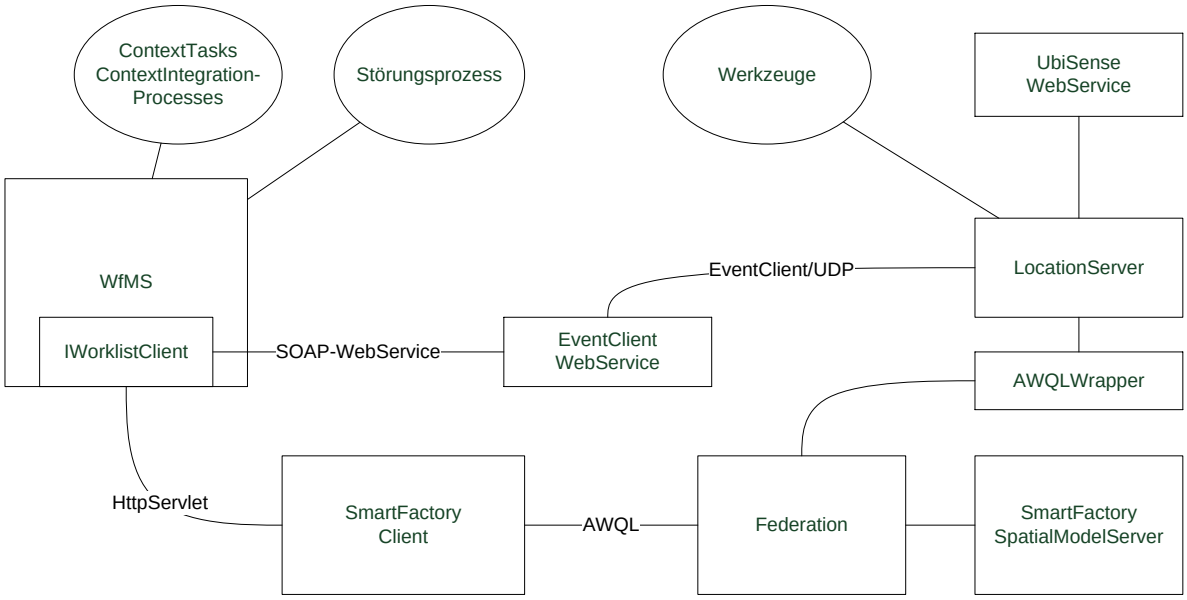

Abbildung 4.1: Gesamtentwurf

4.2.1 WfMS

Das WfMS beinhaltet einen Oracle Process Manager, der auf einem OC4J-Standalone-Server ausgeführt wird. Im Process Manager kommen die ContextTasks sowie die ContextIntegrationProcesses (CIPs) als BPEL-Prozesse zum Einsatz. Der Störungsprozess ist ebenfalls in BPEL modelliert und kann über den Process Manager gestartet werden. Über das Interface IWorklistClient können Tasks, die von den Prozessen instanziiert werden, sowohl abgerufen als auch beendet werden.

4.2.2 Nexus-Föderation

Die Föderation besteht aus dem Spatial Model Server für die statischen Daten des Umgebungsmodells sowie dem Location Server für Kontextdaten, die die Positionen der Nexus-Objekte abbilden. Der Location Server aktualisiert die Kontextdaten der Werkzeuge anhand der Positionsdaten, die vom UbiSense-System empfangen werden.

4.2.3 UbiSense

UbiSense ist ein Echtzeit-Positionierungssystem, das sich innerhalb von Räumen besonders gut eignet. In dem Raum werden zur Positionierung optimalerweise in jeder Ecke Sender befestigt. Die mobilen Objekte werden mit Tags versehen, welche auf die Funksignale der Sender antworten. Durch die unterschiedlichen Antwortzeiten können die Positionen der Tags und indirekt darüber die Position der mobilen Objekte berechnet werden. Das UbiSense-System ermöglicht dem LocationServer die Abfrage der Positionsdaten über einen Http-SOAP-WebService.

4.2.4 SmartFactoryClient

Der SmartFactoryClient steht sowohl für die mobile App als auch für den Web Client. Die Kontextdaten werden über die Föderation mittels AWQL abgefragt. Die Datenübermittlung der Queries und der Antworten erfolgt mittels SOAP. Der Zugriff auf die Tasks des WfMS wird mit einem Http-Worklist-Servlet ermöglicht. Der Client schickt einen simplen HttpRequest und erhält im HttpResponse die Liste der Tasks. Das Annehmen und Beenden der Tasks erfolgt genauso über das Http-Worklist-Servlet. Das ungewöhnliche Design, anstatt eines WebServices ein Http-Servlet zu verwenden, eignet sich besser, weil bei Verwendung eines WebService bei der mobilen App zuviel Rechenlast anfällt, da mobile Geräte den Anforderungen an Rechenkraft für XML-Serialisierung noch nicht gewachsen sind.

4.2.5 EventClient WebService

Der EventClient WebService überprüft die Kontextdaten der NexusObjekte und entscheidet wann die automatische Beendigung von Tasks erfolgen soll. Die Kommunikation zum WfMS wird über einen WebService ermöglicht.

4.3 Datenintegration

Für den Benutzer müssen die Daten aus zwei Systemen angezeigt werden. Zum einen benötigt der Benutzer die Informationen über die Tasks. Welche Tasks wurden ihm zugewießen und wie ist der Status der Tasks? Zum anderen benötigt er die Kontextdaten aus dem Umgebungsmodell. Wo befindet sich das Werkzeug, das für den Reperaturauftrag benötigt wird? Die mobile App sowie die Web-Anwendung benötigen die Daten vom WfMS und dem Location Server.

4.3.1 Datenabruf im Oracle Process Manager

Der Oracle Process Manager in der Version 10.1.3 beinhaltet eine Schnittstelle namens *IWorkflowServiceClient*, die sowohl als Enterprise Java Bean (EJB) als auch SOAP Web Service verfügbar ist. Über den *IWorkflowServiceClient* wird zuerst eine Authentifizierung des Benutzers vorgenommen. Anschließend erhält man über den *ITaskQueryService* alle Informationen über Tasks, zu denen der angemeldete Benutzer berechtigt ist. Über das *ITaskService*-Interface kann der Benutzer Tasks annehmen, verändern, delegieren und beenden.

Für das Abrufen der Task-Informationen und die Interaktion mit den Tasks ist als Teil dieser Diplomarbeit zuerst ein Servlet und später aus Teilen daraus ein Web Service entstanden (siehe Abb. 4.2). Der Einfachheit halber werden beim Http-Worklist-Servlet alle Informationen über die URL als Parameter mitgegeben (`http://server/WorklistServlet/?username=RSteven&password=welcome1&action=worklist`). Die Sicherheit spielt bei dieser Einfachlösung nur eine geringe Rolle und kann vernachlässigt werden. Für den reellen Einsatz dürfen die Benutzerdaten natürlich nicht unverschlüsselt übertragen werden. Das Http-Worklist-Servlet nutzt den IWorkflowServiceClient via lokaler EJBs und muss daher auf dem Anwendungsserver deployed sein, auf dem auch der Oracle Process Manager läuft. Das Servlet erzeugt je nach URL-Parametern einen HttpResponse mit einer Bestätigung für die durchgeführte Aktion oder der vollständigen Worklist für den Benutzer. Die Bestätigungen sowie die Worklist werden im XML-Format im HttpResponse übertragen.

Eine konsequente Verwendung von Web Services anstatt des Http-Servlets wäre hier sicher die bessere Lösung, da sowohl der Oracle Process Manager als auch der Nexus Location Server über Web Service-Schnittstellen verfügen. Allerdings bietet das Android SDK keine Unterstützung für Web Services an. Man müsste also entweder einen eigenen SOAP-Client entwickeln oder auf kSOAP [HS] zurückgreifen. kSOAP ist eine Bibliothek für SOAP-Web

Service-Clients in mobilen Java-Anwendungen. Aber kSOAP ist ursprünglich für Java ME entwickelt worden und der Einsatz bei Android hat sich als schwierig erwiesen.

Beim Anwendungsszenario Störungsprozess werden Tasks automatisch beendet. Damit das Beenden von Tasks leichter in BPEL modelliert werden kann, gibt es eine funktional abgespeckte Variante des Http-Worklist-Servlets als Web Service. Diese Variante bietet gegenüber dem IWorkflowServiceClient auf Web Service-Basis den Vorteil, dass die Schnittstelle wesentlich kompakter ist und der BPEL-Code für den Störungsprozess nicht unnötig kompliziert wird.

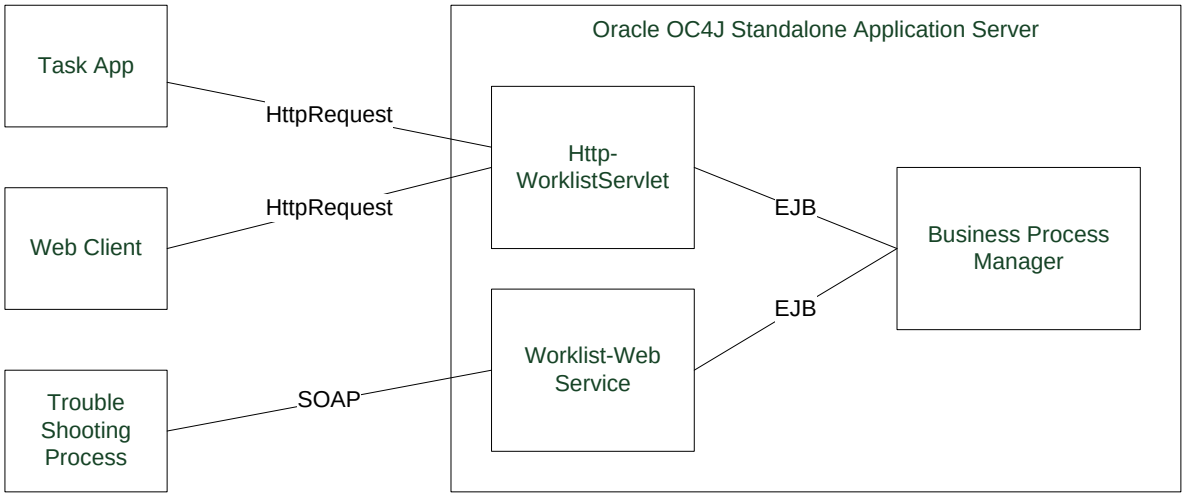

Abbildung 4.2: Datenabruf mittels Http-Worklist-Servlet bzw. Worklist-Web Service

4.3.2 Datenabruf im Nexus Location Server

Die Kontextdaten werden vom Nexus Location Server auf verschiedene Arten angefordert. In der Task App wird ein AWQL-Query an die Föderation geschickt, das alle Tasks anfordert. Anschließend werden aus der Antwort alle Nexus Object Locators (NOLs) extrahiert, welche von den Tasks referenziert werden. Die NOLs verweisen z. B. auf Werkzeuge oder Ressourcen, mit denen der Task in Zusammenhang steht. Danach wird von der Task App ein Query erzeugt, das alle Nexus-Objekte mit den eben extrahierten NOLs anfordert (Beispiel: Listing 4.1). In der Task App können also nur Tasks und die damit verbundenen Objekte angezeigt werden. Im Web Client hingegen will der Benutzer nicht nur mit Tasks interagieren, sondern einen Überblick über verschiedene Typen von Nexus-Objekten gewinnen. Deswegen spezifiziert der Benutzer alle Typen, die für ihn von Interesse sind. Dann wird ein AWQL-Query erzeugt, das alle Nexus-Objekte der angegebenen Typen anfordert. Da die Ergebnismenge sehr groß werden kann, reicht das Filterkriterium Nexus-Objekt-Typ allein nicht aus. Daher wird das Query zusätzlich mit einem Orts-Filter versehen, der dem vom Benutzer ausgewählten Kartenausschnitt entspricht (Beispiel: Listing 4.2). Es werden im Web Client also nur Nexus-Objekte angezeigt, die einen bestimmten Typ

haben und die sich innerhalb einer bestimmten Zone befinden.

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:awql="http://www.nexus.uni-stuttgart.de/2.0/AWQL"
  xmlns:nsas="http://www.nexus.uni-stuttgart.de/1.0/NSAS"
  xmlns:nscs="http://www.nexus.uni-stuttgart.de/1.0/NSCS"
  xmlns:sfcs="http://www.nexus.uni-stuttgart.de/SmartFactoryExtendedClassSchema">
  <soapenv:Header/>
  <soapenv:Body>
    <awql:awql >
      <awql:ecs>http://www.nexus.uni-stuttgart.de/SmartFactoryExtendedClassSchema</awql:ecs>
      <awql:restriction>
        <awql:or>
          <awql:equal>
            <awql:target>nsas:nol.nsas:value</awql:target>
            <awql:referenceValue>nexus:http://nemesis.informatik.uni-stuttgart.de:8080/
            spase/seife||0x7ec28ccd5e1b11dd975d001731c341e7/0x4bb799d5dd1e11d78d3508002
            0a23633</awql:referenceValue>
          </awql:equal>
          <awql:equal>
            <awql:target>nsas:nol.nsas:value</awql:target>
            <awql:referenceValue>nexus:http://nemesis.informatik.uni-stuttgart.de:8080/
            spase/seife||0x7ec28ccd5e1b11dd975d001731c341e7/0x54f4376464b611deb051c478f
            37a28fa</awql:referenceValue>
          </awql:equal>
            <awql:equal>
            <awql:target>nsas:nol.nsas:value</awql:target>
            <awql:referenceValue>nexus:http://nemesis.informatik.uni-stuttgart.de:8080/
            spase/seife||0x7ec28ccd5e1b11dd975d001731c341e7/0xec03abea0a7111dfba9f9235d
            7ef9531</awql:referenceValue>
          </awql:equal>
        </awql:or>
      </awql:restriction>
    </awql:awql>
  </soapenv:Body>
</soapenv:Envelope>
```

Listing 4.1: AWQL-Query mit NOL-Filter

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:awql="http://www.nexus.uni-stuttgart.de/2.0/AWQL"
  xmlns:nsas="http://www.nexus.uni-stuttgart.de/1.0/NSAS"
  xmlns:nsat="http://www.nexus.uni-stuttgart.de/1.0/NSAT"
  xmlns:nscs="http://www.nexus.uni-stuttgart.de/1.0/NSCS"
  xmlns:gml="http://www.opengis.net/gml"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soapenv:Header/>
  <soapenv:Body>
    <awql:awql xmlns:sfcs="http://www.nexus.uni-stuttgart.de/SmartFactoryExtendedClassSchema">
      <awql:ecs>http://www.nexus.uni-stuttgart.de/SmartFactoryExtendedClassSchema</awql:ecs>
      <awql:geoDataFormat>WKT</awql:geoDataFormat>
      <awql:restriction>
        <awql:and>
          <awql:within>
            <awql:target>nsas:pos.nsas:value</awql:target>
            <awql:referenceValue>
              <nsat:WKT srscode="4326">POLYGON((48.755057047364936 9.093503952026367,
                                                48.755057047364936 9.126462936401367,
                                                48.73499316019377 9.126462936401367,
                                                48.73499316019377 9.093503952026367,
                                                48.755057047364936
                                                    9.093503952026367))</nsat:WKT>
            </awql:referenceValue>
          </awql:within>
          <awql:or>
            <awql:equal>
              <awql:target>nsas:type.nsas:value</awql:target>
              <awql:referenceValue>nscs:Restaurant</awql:referenceValue>
            </awql:equal>
            <awql:equal>
              <awql:target>nsas:type.nsas:value</awql:target>
              <awql:referenceValue>sfcs:MobileFactoryObject</awql:referenceValue>
            </awql:equal>
          </awql:or>
        </awql:and>
      </awql:restriction>
    </awql:awql>
  </soapenv:Body>
</soapenv:Envelope>
```

Listing 4.2: AWQL-Query mit Typen- und Orts-Filter

5 Die mobile Anwendung: Task App

Zur Realisierung des Anwendungskonzepts sind im Rahmen dieser Diplomarbeit zwei Anwendungen entstanden. Eine mobile Android-Anwendung (App) und ein Web-Client, der über ein stationäres Terminal abrufbar sein soll (siehe Kapitel 6). Die App sowie die Webanwendung bieten eine Worklist für den Benutzer und eine Darstellung der Tasks auf einer Karte. Auf der Karte werden zusätzlich die zu den Tasks gehörenden Objekte (Werkzeuge, Ressourcen) angezeigt. Dieses Kapitel enthält eine knapp gehaltene Einführung in die Entwicklung von Android-Anwendungen und eine detaillierte Beschreibung der Task App.

5.1 Android SDK

Die Wahl des mobilen Endgeräts ist ein sehr wichtiges Kriterium für mobile Anwendungen, da die Benutzerakzeptanz stark von der Bedienbarkeit des Geräts abhängt. Ein Fabrikarbeiter wird mobile Endgeräte mit großen Tasten bevorzugen, die auch mit Handschuhen bedienbar sind. Das Design eines mobilen Endgeräts sprengt leider den Rahmen dieser Diplomarbeit und daher ist die Wahl auf das G1 vom Mobilfunkhersteller HTC gefallen, das sowohl über einen großen Touch-Screen als auch über eine Tastatur verfügt.

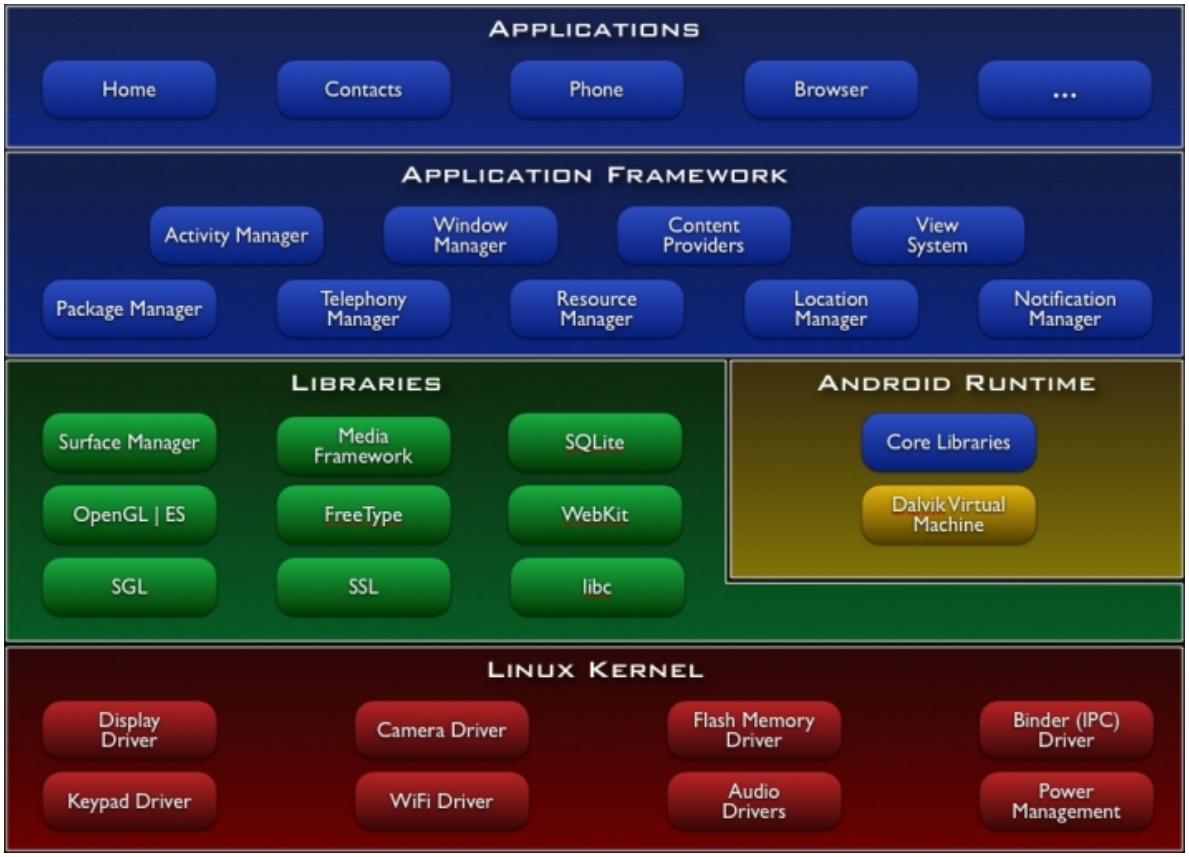

Abbildung 5.1: Architektur des Android Betriebssystems [Inca]

Das G1 ist mit Android ausgestattet. Android ist ein Software-Stack für mobile Endgeräte, das ein Betriebssystem, Middleware und Schlüsselapplikationen umfasst [Inca]. Die rote Schicht (Abbildung 5.1) repräsentiert die Dienste, die vom Linux-Systemkern bereitgestellt werden. Da der Linux-Systemkern unter der GNU General Public License (GPL) bzw. der GNU Lesser General Public License (LGPL) veröffentlicht ist, müssen alle Modifikationen dieser Software bzw. daraus abgeleitete Software unter den gleichen Lizenzbedingungen veröffentlicht werden. Die grüne Schicht beinhaltet somit ausschließlich Open-Source-Bibliotheken. Zu diesen Bibliotheken gehört u. A. SQLite, welche laut [Teco8] die de-facto-Standard-Datenbank für maschinennahe Softwareprodukte ist, da sie mit weniger als 500 Kilobytes Speicher auskommt und trotzdem fast alle Funktionen des SQL-92 Standards unterstützt.

In der grünen Schicht des Architekur-Schichtenmodells befindet sich außerdem die Dalvik-Laufzeitumgebung. Diese unterscheidet sich von der Java-Runtime insofern, dass jede Anwendung anstatt in einer gemeinsamen Java Virtual Machine (JVM) in jeweils eigenen Instanzen einer Dalvik Virtual Machine gestartet wird. Die Dalvik Virtual Machine liest keinen Java-Bytecode, sondern verwendet eigenen Bytecode, der „dex“ genannt wird. Die Entwicklung geschieht dennoch in Java. Man verwendet lediglich ein Werkzeug namens *dx*,

	Dalvik	JVM
Architektur	Register-basiert	Stack-basiert
OS-Unterstützung	Android	Verschiedene
Ausführbare Dateien	APK	JAR
Konstanten-Sichtbarkeit	Applikation	Klasse

Tabelle 5.1: Unterschiede von Dalvik gegenüber JVMs [Sch]

um die Java-kompilierten Klassen in dex-Bytecode umzuwandeln. Dabei werden mehrere Klassen in einer .dex-Datei zusammengefasst, weshalb die Sichtbarkeit von Konstanten dann über Klassengrenzen hinweg gilt. Die Unterschiede von Dalvik gegenüber gewöhnlichen JVM sind in Tabelle 5.1 zusammengefasst.

Die blaue Schicht umfasst zum einen die Dienste, die Google als Service-Prozesse bezeichnet. Diese Dienste sind unsichtbar für den Benutzer des Mobilfunkgeräts. Entwickler können diese Dienste allerdings über einen Message Bus verwenden, um so z. B. mittels Telephony Manager ein Mobilfunkgespräch zu initiieren. Zum anderen beinhaltet die blaue Schicht die Anwendungen, die Interaktion mit dem Benutzer erfordern. Alle Android-Anwendungen haben einen speziellen Aufbau, der sich stark von anderen mobilen Plattformen wie z. B. JavaME unterscheidet. Android-Anwendungen werden in Komponenten aufgeteilt, die einzeln auch für andere Anwendungen nutzbar gemacht werden können. Zu diesen Komponenten gehören unter anderem folgende Typen:

- **Activities** repräsentieren je eine visuelle Schnittstelle, mit der der Benutzer interagieren kann. Zu einer Activity gehört je ein Screen. Zusätzlich können aber auch Popupfenster verwendet werden.
- **Services** haben keine visuellen Schnittstellen, sondern erledigen vielmehr Aufgaben im Hintergrund. Ein gutes Beispiel hierfür ist der Media Player, der Songs von einer Playlist abspielt. Der Benutzer kann den Player über diverse Activities steuern und die Playlist konfigurieren. Sobald der Benutzer die Ansicht für den Player schließt, erwartet er trotzdem, dass der Song im Hintergrund weiterläuft.
- **Content Providers** sind Komponenten, die Daten kapseln und den Anwendungen verfügbar machen.

5.2 Anforderungen

Der Benutzer möchte mit der Task App seine kontextbezogenen Arbeitsaufgaben unterwegs verwalten. Er muss dazu die Informationen der Tasks abrufen können. Die Information über die Position der zu erledigenden Aufgabe sowie der benötigten Betriebsmittel muss in einer Karte veranschaulicht werden. Der Benutzer soll Tasks annehmen, diese dann aktualisieren und letztlich die Tasks beenden können. Für komplizierte Aufgaben, wie z. B. die Delegation

oder Überwachung der Tasks, ist die Task App nicht gedacht. Dafür gibt es bereits die Oracle BPEL-Arbeitsliste, zumal diese Aufgaben nicht vom Benutzer verlangen, dass er sich zu dem Ort der Tasks hinbewegt, und somit kein Bedarf für eine mobile Anwendung bei den genannten Funktionen besteht.

Bevor der Benutzer Daten vom WfMS empfangen kann, muss er sich zunächst mit seinem Benutzernamen und Passwort anmelden.

5.3 Entwurf

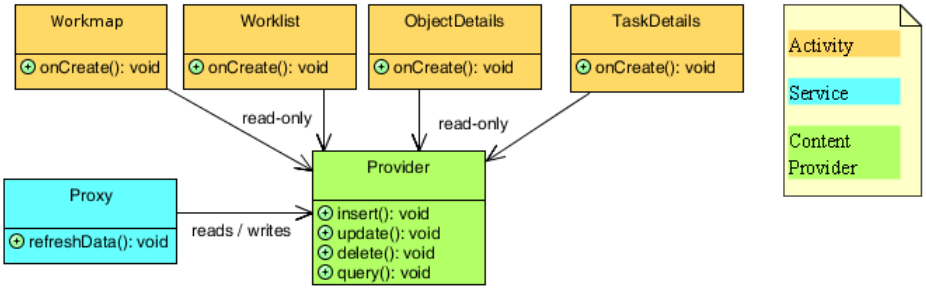

Abbildung 5.2: Activities, Services und ContentProvider

Abbildung 5.2 zeigt den groben Aufbau des mobilen Clients. Es gibt vier Activities, die für je einen Screen zuständig sind. Der Service *Proxy* ist für die Synchronisierung der Daten mit dem Workflow-System sowie dem Umgebungsmodell zuständig. Die Daten werden in einer SQLite-Datenbank gespeichert, welche über einen Content Provider zugänglich gemacht sind.

5.4 Datenmodell

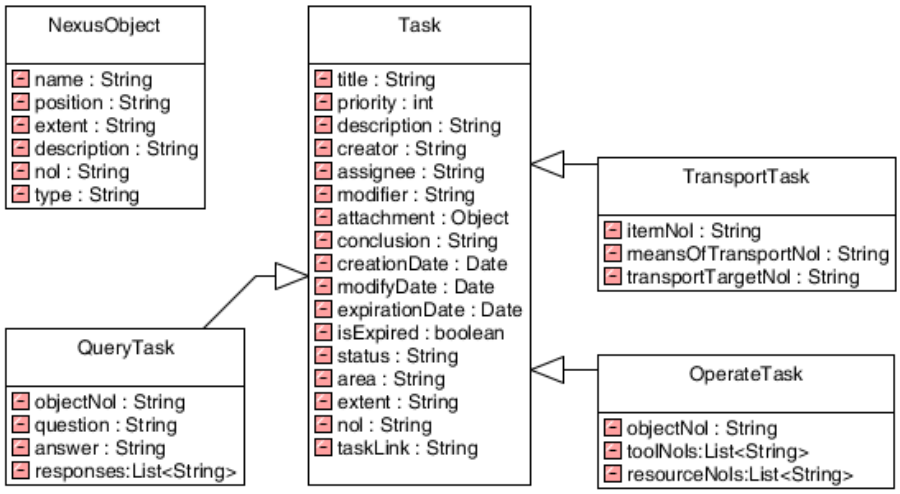

Abbildung 5.3: Datenmodell

Das Datenmodell ist in Abbildung 5.3 als UML-Klassendiagramm dargestellt. Die speziellen Tasks leiten sich von der Vaterklasse *Task* ab. Da alle Instanzen der Tasks und der generellen Nexus-Objekte auch im Umgebungsmodell vertreten sind und dort mit einem NOL versehen sind, geschieht die Identifizierung aller Datenobjekte in der Task App über den NOL. *Gibt es im Nexus-Modell Tasks, die nicht mit dem Datenmodell des WfMS konsistent sind, werden diese Tasks verworfen und dem Benutzer nicht angezeigt*.

Die Daten werden auf dem Android-Mobilfunkgerät in einer SQLite-Datenbank gespeichert (Rich Client, vgl. Abschnitt 2.3.1). Wenn der Benutzer die Anwendung schließt und erneut startet, sich aber gerade in einem Funkloch befindet, kann er zumindest die bereits empfangenen Daten einsehen. Die Datenbank erhöht somit die Toleranz gegen Netzunterbrechungen. Das Schema der relationalen Datenbank ist in Abbildung 5.4 skizziert. Da relationale Datenbanken keine Vererbung kennen, gibt es eine Tabelle *tasks* für alle Eigenschaften, die die Vaterklasse *Task* besitzt. Für die Attribute der Kindklassen gibt es jeweils eine eigene Tabelle (*operateTasks*, *queryTasks* und *transportTasks*). Da die NOLs und die Task-Identifier vom Oracle-System aus langen Zeichenketten bestehen, sind diese für die SQLite-Datenbank als Primärschlüssel weniger geeignet. Daher gibt es zu jeder Tabelle den Integer-Primärschlüssel *_id*.

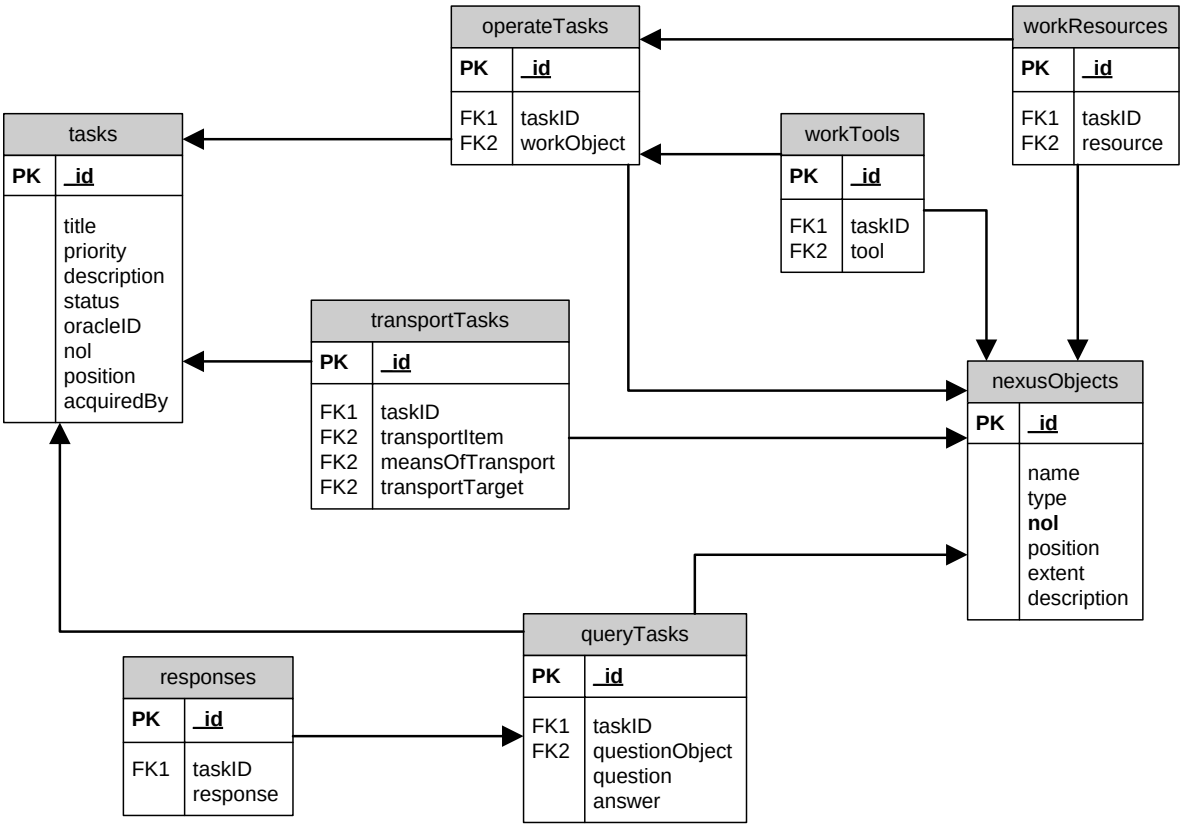

Abbildung 5.4: Relationales Datenbankschema der Task App

5.5 Icons

Icons helfen dem Benutzer beim Bedienen einer Anwendung. Optimalerweise drückt das Piktogramm im Icon schon den Zweck der Funktion aus, welche beim Drücken des Icons ausgelöst wird. Bisher sind nicht alle Objekt-Typen der Smart Factory durch Icons repräsentiert. Eine Erweiterung in dieser Richtung ist aber ohne Weiteres durch wenige Änderungen im Code zu bewerkstelligen.

Die Objekte in der Task App sind in drei Gruppen aufgeteilt. Zur einen Gruppe gehören alle Nexus-Objekte, die vom Nexus-Typ *sfcs:Task* abgeleitet sind. Das sind im Einzelnen die Query Tasks, Operate Tasks sowie Transport Tasks. Die Icons für Tasks sind viereckig und je nach Priorität der Tasks nach dem Ampelsystem eingefärbt (Tabelle 5.2). Da die Prioritätsskala von 0 bis 5 geht, sind den Werten folgende Farben zugeteilt: 0-1 grün, 2-4 gelb, 5 rot. Die nächste Gruppe umfasst alle anderen Nexus-Objekte (z.B. Objekte vom Typ *nscs:Building*). Die Icons für generelle Nexus-Objekte sind kreisförmig und blau eingefärbt (Tabelle 5.3). In der Task App gibt es zusätzlich noch ein Icon für die Position des Benutzers. Der Benutzer ist aber nicht im Nexus Location Server gespeichert, da die Position von Mitarbeitern nicht

Icon	Bedeutung	Nexus-Typ
	Operate Task mit niedriger Priorität	sfcs:ActionTask
	Query Task	sfcs:QuestionTask
	Transport Task	sfcs:TransportTask
	Task mit mittler Priorität	sfcs:TransportTask
	Task mit hoher Priorität	sfcs:TransportTask

Tabelle 5.2: Task-App-Icons für Tasks

überwacht werden darf. Letzlich gibt es noch Icons, die als Buttons verwendet werden, um spezielle Aktionen auszuführen (Tabelle 5.4).

5.6 Activities

Im Folgenden werden die Activities beschrieben, also die Komponenten für die Benutzerinteraktion. Zur Veranschaulichung sind jeweils auch Screenshots zu den Activities abgebildet, die mit dem Android-Emulator des Android SDKs aufgenommen wurden.

Icon	Bedeutung	Nexus-Typ
	Mobiles Fabrik-Objekt	sfcs:MobileFactoryObject
	Werkzeug	sfcs:Tool
	Ersatzteil	sfcs:Part
	Fräßmaschine	sfcs:Millingmachine
	Gebäude	nscs:Building
	Universitätsgebäude	nscs:UniversityBuilding
	Raum	nscs:Room
	Geschäft	nscs:Shop
	Restaurant	nscs:Restaurant

Tabelle 5.3: Task-App-Icons für generelle Nexus-Objekte

Icon	Bedeutung
	Benutzer
	Task annehmen
	Task beenden
	Änderungen speichern
	Details anzeigen
	In der Karte anzeigen

Tabelle 5.4: Zusätzliche Icons in der Task App

5.6.1 Worklist

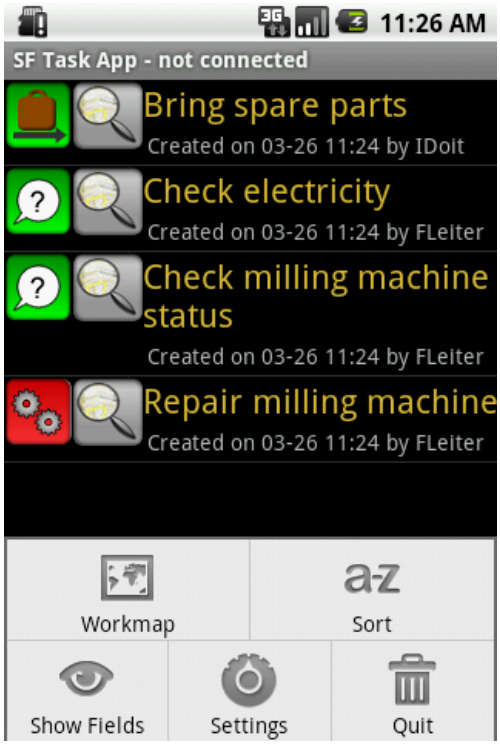

Abbildung 5.5: Screenshot der Activity Worklist mit eingeblendetem Menü

Die Worklist ist das Herzstück der Anwendung (Abbildung 5.5). Hier erhält der Benutzer einen Überblick über alle Tasks, die er zu erledigen hat. Die Details der Tasks lassen sich für mehr Übersichtlichkeit stufenweise ausblenden. Die Sortierung der Liste kann optional nach Titel, Priorität und Erstelldatum eingestellt werden. Außerdem kann der Benutzer über einen Popup-Dialog die Einstellungen (Benutzername, Passwort, Server-URLs) vornehmen (Abbildung 5.6). Ein einfacher Klick auf einen Eintrag wechselt zur Activity TaskDetails. Über das Menü und über die Lupensymbole bei den Listen-Einträgen gelangt der Benutzer zur Kartenansicht. In der Titelleiste wird die Netzwerkverfügbarkeit angezeigt.

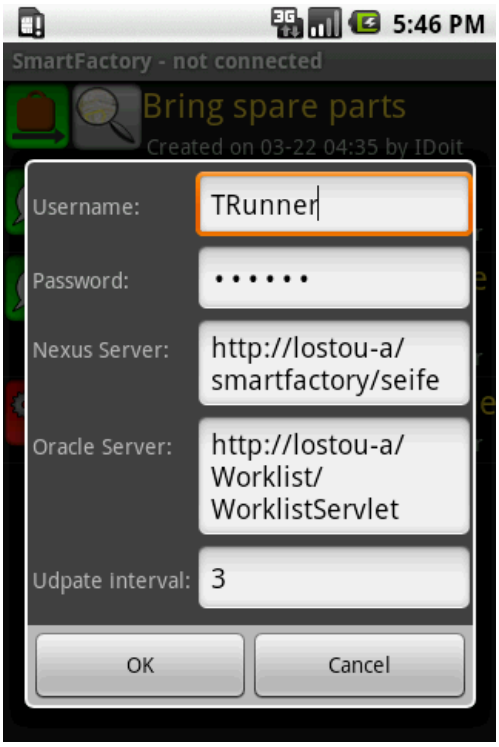

Abbildung 5.6: Screenshot der Popup-Dialogs für die Einstellungen

Die Worklist aktualisiert sich in regelmäßigen Intervallen, die der Benutzer definieren kann. Es handelt sich demnach um eine Mischung aus Pull- und Push-Worklist (vgl. Abschnitt 2.1.3). Die Worklist zeigt nur offene Tasks, die der Staff Query des angemeldeten Benutzers genügen. Bereits abgeschlossene Tasks oder Tasks, die nicht für den Benutzer bestimmt sind, werden demnach nicht angezeigt.

5.6.2 Workmap

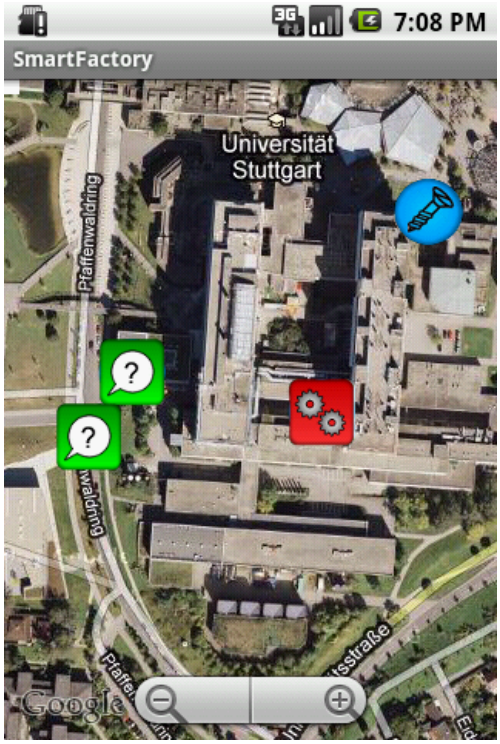

Abbildung 5.7: Screenshot der Activity Workmap

Die Workmap zeigt dem Benutzer die Tasks in ihrem Kontext an. Für die Kartendarstellung wurde die kostenfreie Google-Maps-API für Android verwendet. Marker zeigen die Positionen von Tasks sowie die generellen Nexus-Objekte. Die Karte bietet sowohl Satellitenbilder (Abbildung 5.7) als auch Kartenbilder. Über einen Filter können Tasks, Objekte oder die eigene Position ausgeblendet werden (Abbildung 5.8). Die eigene Position wird mittels GPS ermittelt. Falls nicht genügend GPS-Satelliten empfangen werden oder der GPS-Sensor des Mobilfunkgeräts ausgeschaltet ist, wird die Position des Benutzers anhand der Cell-ID der Mobilfunkzelle berechnet. Der Benutzer kann die Tasks und die Objekte in der Karte anklicken. Dann erscheint ein Fenster, in dem der Name des Objekts angezeigt wird. Wenn der Benutzer zusätzlich in das Fenster klickt, werden die Details zum ausgewählten Objekt angezeigt. Somit muss der Benutzer seine Auswahl bestätigen, um zu den Details zu gelangen, und unbeabsichtigtes Anklicken, z. B. beim Verschieben der Karte, bleibt ohne Folgen.

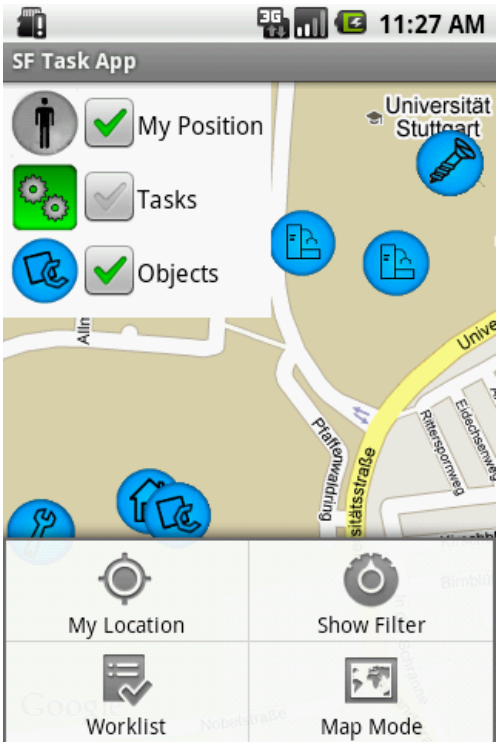

Abbildung 5.8: Screenshot der Activity Workmap mit eingeblendetem Filter und Menü

Die Positionsbestimmung des Benutzers ließe sich auch über das UbiSense-Positionierungssystem ermitteln. Diese Funktion ist bisher nicht in der Task App implementiert.

5.6.3 TaskDetails

Die Detailansicht für Tasks zeigt alle allgemeinen sowie die spezifischen Attribute für die Tasks an. In der ersten Zeile steht der Titel des Tasks zusammen mit dem task-typischen Icon. Die Färbung in grün, gelb und rot spiegelt die Priorität wieder (vgl. Abschnitt 5.5). Mit dem ersten Menüeintrag *Workmap* wird der Task auf der Karte angezeigt. Mit dem Menüpunkt *Claim* kann der Benutzer den Task annehmen. Der Task verschwindet dann von der Worklist anderer Benutzer, sobald deren Worklist sich aktualisiert. Der Menüeintrag *Complete* ist erst verfügbar, wenn der Benutzer den Task angenommen hat. Mit Klick auf *Complete* wird der Task beendet und verschwindet automatisch von der Worklist des Benutzers.

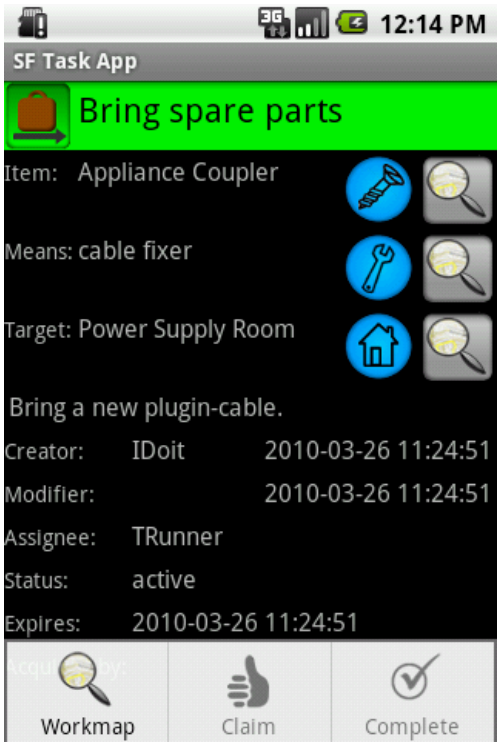

Abbildung 5.9: Screenshot der Activity TaskDetails bei einem Transport Task mit eingeblendetem Menü

Ein Transport Task ist mit drei Nexus-Objekten verbunden. Dabei ist das Transport Item der zu transportierende Gegenstand, Means of Transport bezeichnet das Mittel, mit dem der Benutzer den Gegenstand transportieren kann, und Transport Target gibt das Ziel an, wohin der Gegenstand zu transportieren ist. In Abbildung 5.9 ist die Detailansicht eines Transport Tasks zu sehen. Der Benutzer kann Transport Item, Means of Transport und Transport Target anklicken, um die Details zu den Objekten anzuzeigen, oder auf die Lupensymbole klicken, um die Objekte auf der Karte anzuzeigen.

Abbildung 5.10: Screenshot der Activity TaskDetails bei einem Operate Task

Der Operate Task bezieht sich immer auf ein Work Object, an dem die Arbeit zu verrichten ist. Zusätzlich zu einem Operate Task gibt es eine unbestimmte Anzahl von Werkzeugen und Ersatzteilen (Abbildung 5.10). Der Benutzer kann sowohl Work Object als auch die Liste der Werkzeuge und Ersatzteile anklicken, um Details zu einem Objekt anzuzeigen.

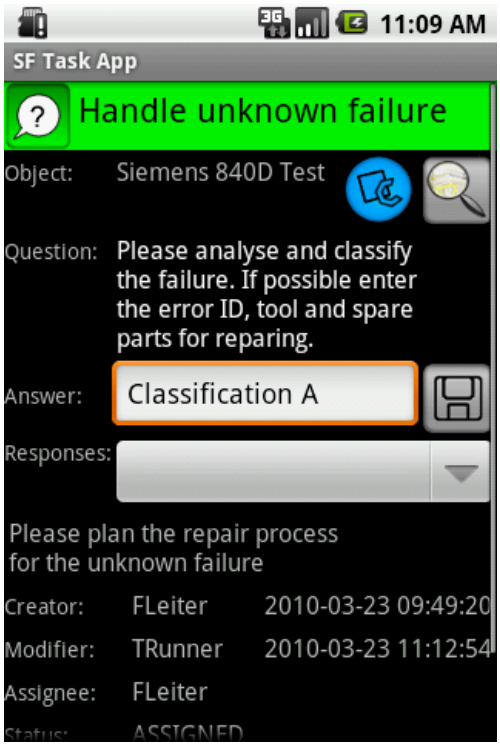

Abbildung 5.11: Screenshot der Activity TaskDetails bei einem Query Task

In Abbildung 5.11 sieht man die Detailansicht zu einem Query Task. Dem Benutzer wird hier angezeigt, auf welches Objekt sich die Frage bezieht. Zu einem Query Task können Antwortmöglichkeiten vordefiniert werden. Dann kann der Benutzer die richtige Antwort über eine Drop-Down-Box auswählen. Wenn keine Antworten vordefiniert sind, kann der Benutzer einen beliebigen Text in das Textfeld eintragen. Die Antwort wird beim Beenden des Tasks automatisch im Payload des Tasks im Oracle Business Process Manager gespeichert. Der Benutzer kann aber auch schon vor dem Beenden des Tasks die eingegebene Antwort mit dem Server synchronisieren, indem er das Diskettensymbol zum Speichern der Antwort anklickt. Die Antwort wird wegen der Nachvollziehbarkeit im Payload des Oracle Tasks gespeichert und nicht im Nexus-Umgebungsmodell, da der Task nach dem Beenden aus dem Umgebungsmodell gelöscht wird.

5.6.4 ObjectDetails

Abbildung 5.12: Screenshot der Activity ObjectDetails

Die Detailansicht für generelle Objekte aus dem Umgebungsmodell zeigt in der ersten Zeile den Namen des Objekts zusammen mit dem typ-spezifischen Icon. Zuätzlich wird eine Beschreibung des Objekts angezeigt und der Benutzer hat die Möglichkeit, mit einem Klick auf das Kartensymbol direkt zur Kartenansicht des Objekts zu springen.

5.7 Service: Proxy

Die Klasse *Proxy* ist der einzige Service in der Task App und für die Datensynchronisierung mit dem WfMS und dem Umgebungsmodell zuständig (siehe Abschnitt 4.3). Das Besondere an diesem Service ist, dass dieser von den Activities nicht nur gestartet und beendet werden kann, sondern die Activities mit *Proxy* auch interagieren können. Das Klassendiagram in Abbildung 5.13 veranschaulicht die Abhängigkeiten von *Proxy*. Im Listing 5.1 sieht man, wie man den Service *Proxy* startet. Beim Aufruf von *bindService()* muss man einen Callback mitgeben, über den man eine Instanz eines Binders erhält. Mit dem Binder lassen sich Methoden des Services aufrufen. Mit dem ersten Parameter des Typs *Intent* spezifiziert die Activity, welchen Service sie nutzen möchte. Der zweite Parameter ist der Callback. Der

dritte Parameter gibt an, ob der Service neu erzeugt werden soll, wenn dieser noch nicht mit *startService()* erzeugt wurde.

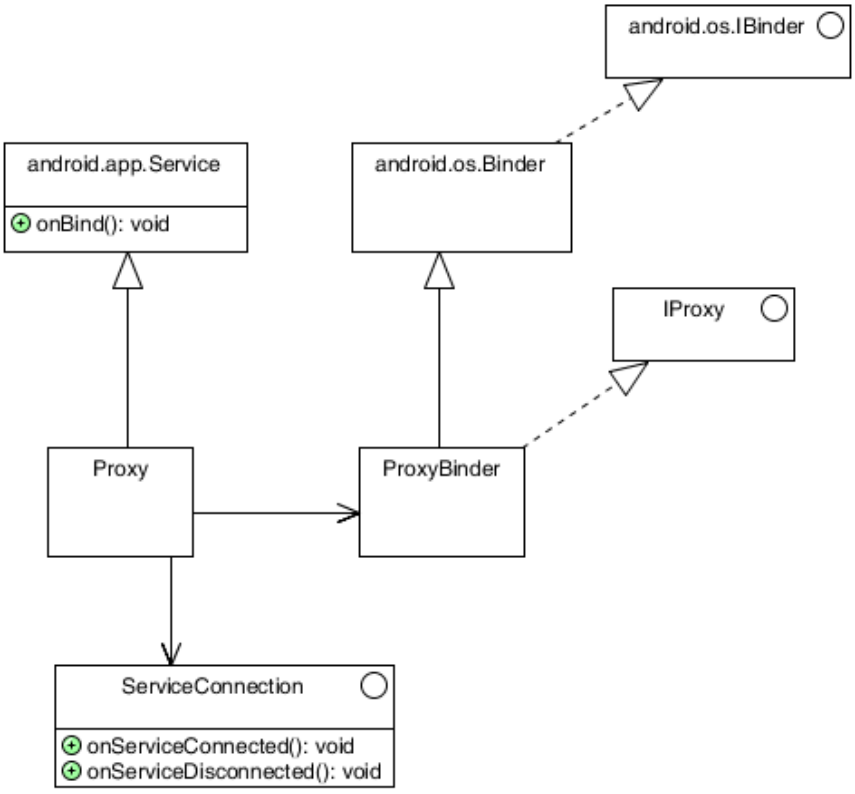

Abbildung 5.13: Serivce Proxy und seine Abhängigkeiten

```
IProxy proxy = null;
ServiceConnection scProxy = new ServiceConnection() {

	@Override
	public void onServiceConnected(ComponentName name, IBinder service) {
		proxy = (IProxy) service;

		// Methode von IProxy aufrufen
		proxy.addConnectionStatusListener(Worklist.this);
	}

	@Override
	public void onServiceDisconnected(ComponentName name) {

	}
```

```
};
bindService(new Intent(this, Proxy.class), scProxy, Context.BIND_AUTO_CREATE);
```

Listing 5.1: Aufruf von Proxy

5.8 Content Provider

Der Content Provider namens *Provider* verwaltet die SQLite-Datenbank (siehe Abschnitt 5.4). Ein Content Provider kann über den Content Resolver aufgerufen werden. Der Content Resolver ist eine Member-Variable von Activities und Services und bietet die Methoden *query(), update(), insert()* sowie *delete()*. Listing 5.2 zeigt einen beispielhaften Query-Aufruf auf die Tabelle der Tasks. Rückgabewert der Funktion ist ein Objekt des Typs *Cursor,* mit dem man über die Reihen der Ergebnistabelle iterieren kann. Der erste Parameter *uri* gibt den Tabellennamen an.

```
String uri = "content://de.unistuttgart.provider.smartfactory/tasks";
static final Uri uri = Uri.parse(Task.uri);
ContentResolver cr = this.getContentResolver();
Cursor c = cr.query(uri, null, null, null, null);
if (c.moveToFirst()) { // hole ersten Eintrag
       Task task;
       int taskID = c.getInt(Provider.Operate.ID_COLUMN);
       int type = c.getInt(Provider.Task.TYPE_COLUMN);
       switch (type) {
              case Task.OPERATE_TASK_TYPE:
                     // Extract Operate Task specific attributes
                     // ...
                     break;
              case Task.QUERY_TASK_TYPE:
                     // Extract Query Task specific attributes
                     // ...
                     break;
              case Task.TRANSPORT_TASK_TYPE:
                     // Extract Transport Task specific attributes
                     // ...
                     break;
              default:
                     throw new IllegalStateException("Unexpected type " + type);
       }
       // Extract general task details
       task.setTitle(c.getString(Provider.Task.TITLE_COLUMN));
       task.setCreator(c.getString(Provider.Task.CREATOR_COLUMN));
       task.setOwner(c.getString(Provider.Task.OWNER_COLUMN));
       task.setPriority(c.getInt(Provider.Task.PRIORITY_COLUMN));
       // ...
}
c.close();
```

Listing 5.2: Aufruf des Content Providers

6 Der Web-Client

6.1 Google Web Toolkit

Der Web-Client wurde mit dem Google Web Toolkit (GWT) entwickelt, welches die Entwicklung von komplexen Browser-Anwendungen stark vereinfacht. Die komplette Implementierung kann über Java realisiert werden und GWT erzeugt aus dem kompilierten Java-Code einen JavaScript-Code, der für alle gängigen Browser kompatibel ist. So muss sich der Entwickler nicht mit den verschiedenen Eigenheiten der Browser beschäftigen. Außerdem ist reiner Java-Code leichter wartbar als HTML gemischt mit JavaScript. Für den WebClient wurde insbesondere SmartGWT verwendet. Das ist eine Erweiterung von GWT, die mehr Widgets (visuelle Bedienelemente) und Funktionen bietet.

Da der cross-kompilierte JavaScript-Code letztlich im Browser ausgeführt wird, sind spezielle Schnittstellen für die Server-Kommunikation notwendig. GWT nutzt hierfür das sog. GWT RPC Framework (siehe Abb. 6.1). Der Entwickler definiert die asynchronen Schnittstellen und implementiert den serverseitigen Code in einem Servlet. GWT erzeugt daraus eine Proxy-Klasse, deren Code im Browser ausgeführt werden kann.

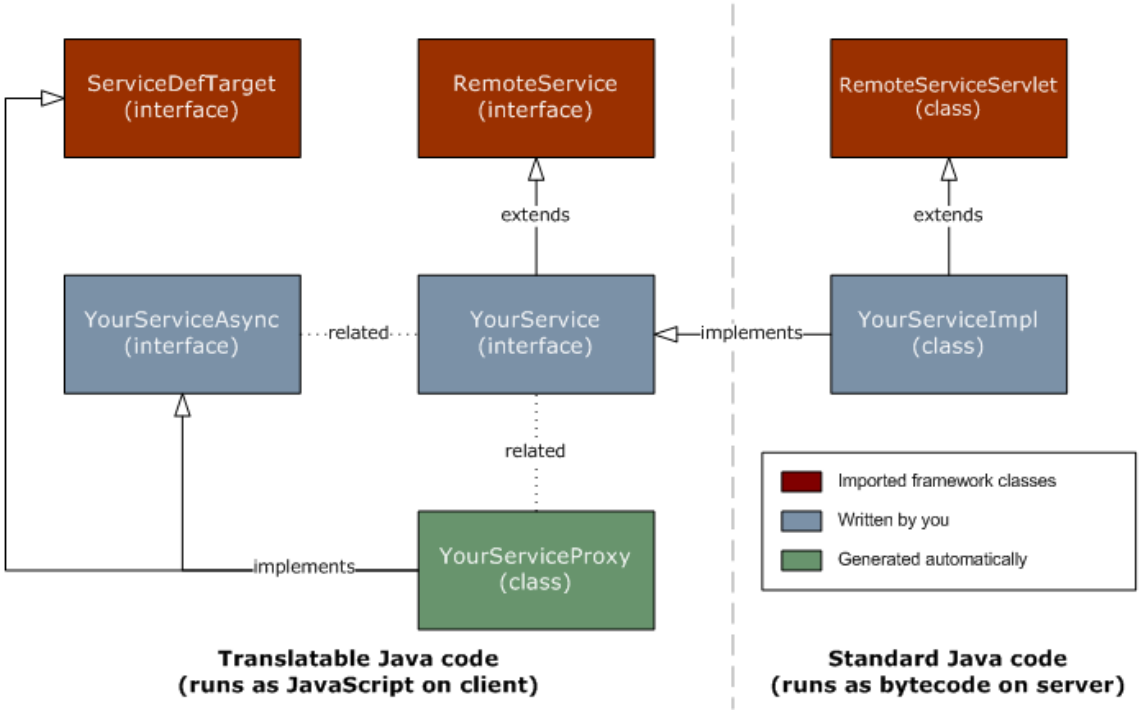

Abbildung 6.1: Die GWT-RPC-Architektur [Incc]

6.2 Entwurf

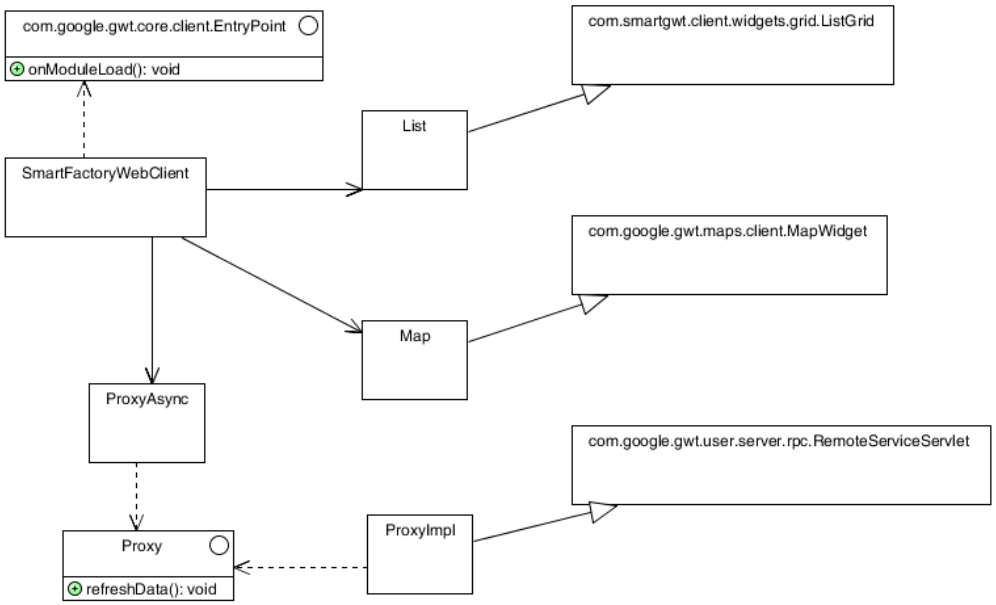

Abbildung 6.2: Grobentwurf des Web-Clients

In Abbildung 6.2 sieht man den stark vereinfachten Aufbau der Web-Anwendung. Beim Aufruf der Web-Anwendung wird als erstes die Methode *onModuleLoad* aufgerufen. Diese initialisiert unter anderem auch das Widget für die Karte (*Map*) und die Liste der Nexus-Objekte namens *List*. Für die Aktualisierung der Daten wird ein asynchroner Service erzeugt, welcher mit dem RPC-Servlet kommuniziert.

6.3 Benutzeroberfläche

Die Benutzeroberfläche ist in der Abbildung 6.3 dargestellt. Das Fenster ist in Liste, Karte und XML-Ansicht unterteilt. Zusätzlich ist am unteren Fensterrand eine Statusleiste angebracht, die den Benutzer über laufende Ladevorgänge informiert. Mit dem Button *Settings* können Benutzername, Passwort und die URLs zum Oracle-Server sowie dem Umgebungsmodell eingestellt werden. Liste und Karte werden in konfigurierbaren Zeitabständen oder mit Klick auf den Refresh-Button aktualisiert. Klickt der Benutzer in der Liste auf den Info-Button, wird ein neues Fenster im Browser geöffnet, das die Taskdetails in der BPEL-Arbeitsliste von Oracle anzeigt (siehe Abb. 6.4). Da nicht alle Attribute der verschiedenen Objekte in der

Tabelle angezeigt werden können, gibt es unterhalb der Liste noch ein zusätzliches Fenster, das den vom Location Server empfangenen XML-Code für das Nexus-Objekt anzeigt. Der Web Client ruft nicht nur Tasks ab, sondern alle Nexus-Objekte eines bestimmten Typs (vgl. Abschnitt 4.3). Deswegen gibt es den Filter-Button, mit dem der Benutzer angeben kann, welche Nexus-Objekt-Typen er vom Location Server anfordern möchte.

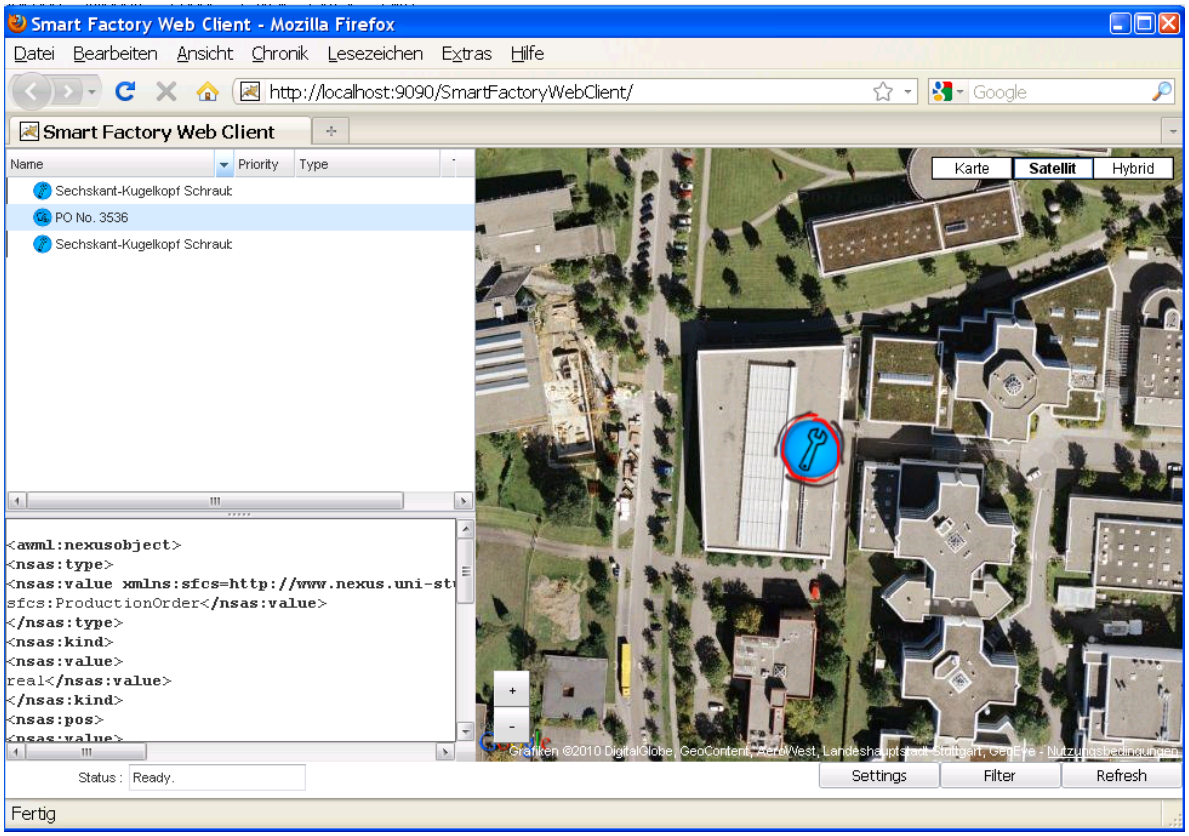

Abbildung 6.3: Die Benutzeroberfläche des Web-Clients

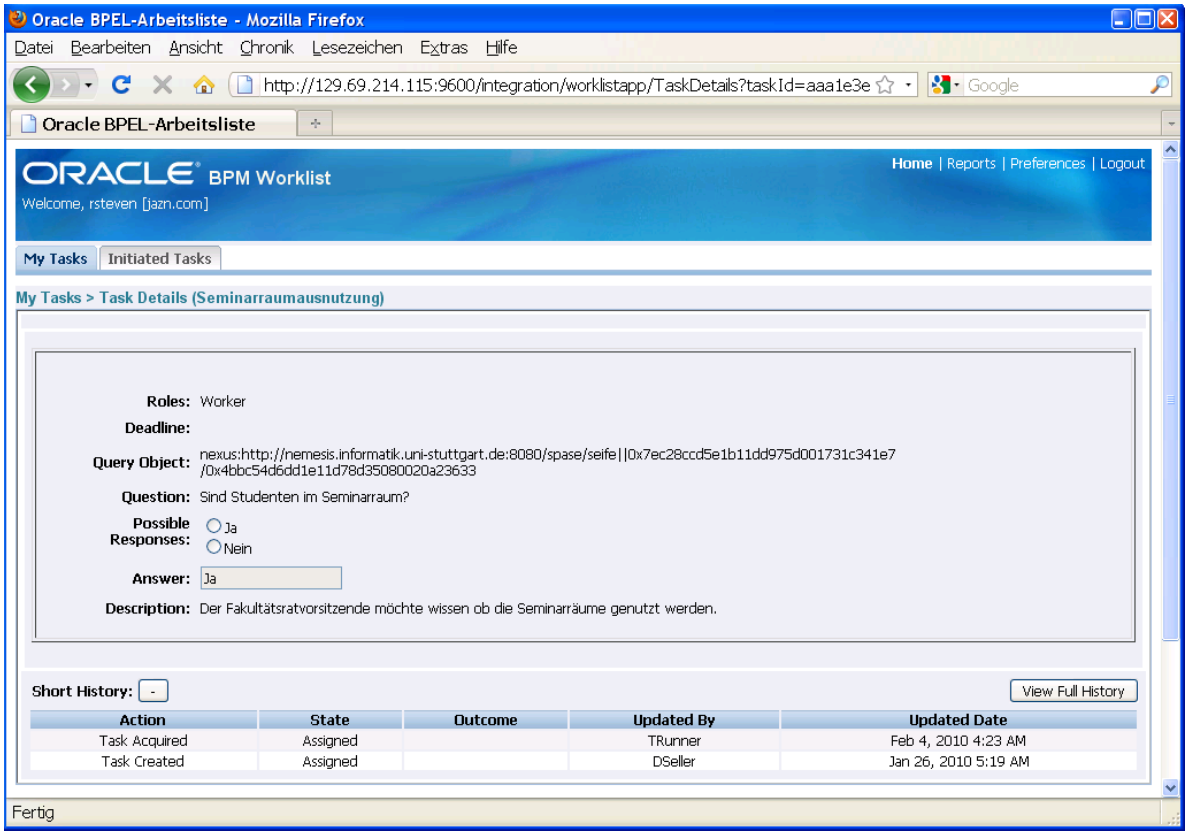

Abbildung 6.4: Ansicht der Details zu einem Query Task

7 Deployment

Dieses Kapitel beschreibt, welche Installationen und Konfigurationen erforderlich sind, damit die Task App sowie der Web Client funktionieren.

7.1 Installation

Tabelle 7.1 zeigt, welche Komponenten installiert sind und mit welcher URL sich diese aufrufen lassen. Dazu gehören der Location Server, die BPEL Prozesse, die auf dem Oracle Process Manager Einsatz finden, und das Servlet bzw. der WebService für die Datenintegration (siehe 4.3).

Komponente	URL
Location Server	`http://nemesis.informatik.uni-stuttgart.de:8080/smartfactory/seife`
Oracle BPEL-Arbeitsliste	`http://lostou-a.informatik.uni-stuttgart.de:9600/integration/worklistapp/Login`
Operate Task	`http://lostou-a.informatik.uni-stuttgart.de:9600/orabpel/default/OperateTask/1.0/OperateTask?wsdl`
Transport Task	`http://lostou-a.informatik.uni-stuttgart.de:9600/orabpel/default/TransportTask/1.0/TransportTask?wsdl`
Query Task	`http://lostou-a.informatik.uni-stuttgart.de:9600/orabpel/default/QueryTask/1.0/QueryTask?wsdl`
Störungsprozess	`http://lostou-a.informatik.uni-stuttgart.de:9600/orabpel/default/StoerungsBehebungsProzess/1.0/StoerungsBehebungsProzess?wsdl`
WorklistServlet	`http://lostou-a.informatik.uni-stuttgart.de:9600/Worklist/WorklistServlet`
WebService zum Beenden von Tasks	`http://lostou-a.informatik.uni-stuttgart.de:9600/WorklistWS/WorklistWSSoapHttpPort?wsdl`

Tabelle 7.1: Installierte Komponenten und zugehörige URLs

Personenname	Benutzername	Rolle	Manager
Hans Buyer	HBuyer	Customer	
Michael Manage	MManage	Backoffice	
Daniel Seller	DSeller	Backoffice	Michael Manage
Franz Leiter	FLeiter	ProductionManager	Michael Manage
Ignaz Doit	IDoit	Worker	Franz Leiter
Thorsten Runner	TRunner	Worker	Franz Leiter
Thomas Nobreak	TNobreak	Worker	Franz Leiter
Josef Heiler	JHeiler	Service	Michael Manage

Tabelle 7.2: Konfiguration der Mitarbeier im WfMS [Lae09]

7.2 Konfiguration

Die Mitarbeiter müssen im Oracle WfMS konfiguriert werden. Da das Anwendungsszenario dieser Arbeit das gleiche wie bei [Lae09] ist, konnten die Mitarbeitereinstellungen exakt übernommen werden. Die Konfiguration ist in Tabelle 7.2 zu sehen. Die Tasks des Störungsprozesses werden standardmäßig der Gruppe Worker zugewiesen und der Benutzer FLeiter wird als Verantwortlicher und Ersteller des Tasks angegeben.

7.3 Installation der Task App

Zum Kompilieren der Task App benötigt man das Android-SDK. Android-Anwendungen werden damit in ein Android-Package (.apk) übersetzt und verpackt und lassen sich dann auf das mobile Endgerät übertragen. Die Anwendung wird beim Öffnen der apk-Datei automatisch installiert. Wichtig für den Erfolg der Installation ist die Manifest-Datei im Android-Package. In dieser Datei werden alle Activities, Services und Content Provider aufgelistet, damit Android diese beim Ausführen finden kann. Zusätzlich wird in der Manifest-Datei festgelegt, welche Rechte die Anwendung besitzt. Für die Task App sind die Rechte gesetzt, die den Zugriff auf das Netzwerk und die Positionierungssensoren erlauben. Damit die Google Maps Karte einwandfrei funktioniert, muss man einen Developer Key beantragen [Incd]. Den erzeugten Developer Key muss man bei der Task App in *res/layout/map.xml* eintragen.

7.4 Installation des Web Clients

Der Web Client ist aktuell unter `http://lostou-a.informatik.uni-stuttgart.de:8080/SmartFactoryWebClient/` vorinstalliert. Mit dem Ant-Build-Skript im Entwicklungsordner lässt sich die Web-Anwendung aber bequem auf beliebige Application Server installieren.

	HTC G1	Motorola Milestone	Windows Server
Prozessorgeschwindigkeit	528 MHz	550 MHz	2,50 GHz
Arbeitsspeicher (RAM)	192 MB	256 MB	18 GB
Android Version	1.6	2.0.1	
Datenverbindung	Wi-Fi 802.11b/g	Wi-Fi 802.11b/g	Ethernet 802.3

Tabelle 7.3: Geräte im Performance-Test

Zum Kompilieren benötigt man das Google Web Toolkit (GWT). Der Ort der GWT-Installation muss in der Datei *build.properties* spezifiziert werden, bevor das Ant-Skript ausgeführt werden kann. Damit auch hier die Google Maps Karte funktioniert, muss man auch für GWT-Anwendungen unter [Incb] einen Google Maps Key registrieren.

7.5 Performance-Test

Um Aussagen über die Performance der Task App machen zu können, wurde die App mit zwei verschiedenen Android-Geräten getestet. Das ältere Modell G1 vom Mobilfunkhersteller HTC ist seit Anfang 2009 erhältlich und hat ein Android-Betriebssystem in der Version 1.6. Das neuere Modell Milestone ist Ende 2009 erschienen und kommt mit der neueren 2.0-Version des Android-Betriebssystems daher. Wichtig für den Performance-Test sind vor allem die technischen Daten über die Prozessorleistung und den Arbeitspeicher (siehe Tabelle 7.3). Das Servlet für den Web Client wurde als Referenz in den Test mit einbezogen. Es läuft auf einem Tomcat-Application-Server, der auf einem Windows Server installiert ist. Die mobilen Geräte waren beim Test über das Wireless-LAN mit den Servern verbunden. Der Tomcat-Server war hingegen auf dem gleichen Rechner wie der Oracle-Server installiert und konnte den Nexus-Server über das lokale Netzwerk ohne Funkverbindung erreichen.

Getestet wurde die Dauer der Datensynchronisierung (siehe Tabelle 7.4), welche aus vier Schritten besteht: Die Task App fordert zuerst die Tasks vom Workflow-System an, um anschließend die Kontextdaten und die mit den Tasks verbundenen Objekte vom Nexus-System abzufragen. Nachdem die XML-Antwort der Server mit einem SAXParser in das Datenmodell der Task App umgewandelt ist, werden die Daten in der SQLite-Datenbank gespeichert (vgl. Abschnitt 4.3). Der Web Client empfängt nur Daten vom Nexus-System, da für die Anzeige der Tasks die BPEL-Arbeitsliste von Oracle verwendet wird. Die Zeit für den vollständigen Datenabruf steigt in etwa linear zu der Anzahl der Tasks. Das Motorola-Handy ist bei allen Tests um ca. ein Drittel schneller als das G1.

Die Zeiten für die einzelnen Phasen sind in Abbildung 7.1 und 7.2 zu erkennen. Es ist offensichtlich, dass die Anfrage an den Oracle-Server die meiste Zeit in Anspruch nimmt. Das Aktualisieren der Datenbank geht beim Milestone auch bei 100 Tasks noch zügig. Das G1 wird deutlich langsamer, je mehr Einträge in der Datenbank aktualisiert werden müssen.

Anzahl Tasks	1	10	50	100
HTC G1	1,629 s	8,349 s	37,815 s	82,271 s
Motorola Milestone	1,45 s	5,099 s	28,576 s	53,492 s
Windows Server	0,14 s	0,216 s	0,435 s	0,847 s

Tabelle 7.4: Zeitmessungen bei der Datensynchronsierung

Die Verarbeitungsschritte des SAXParsers und die Anfrage an den Nexus-Server fallen kaum ins Gewicht.

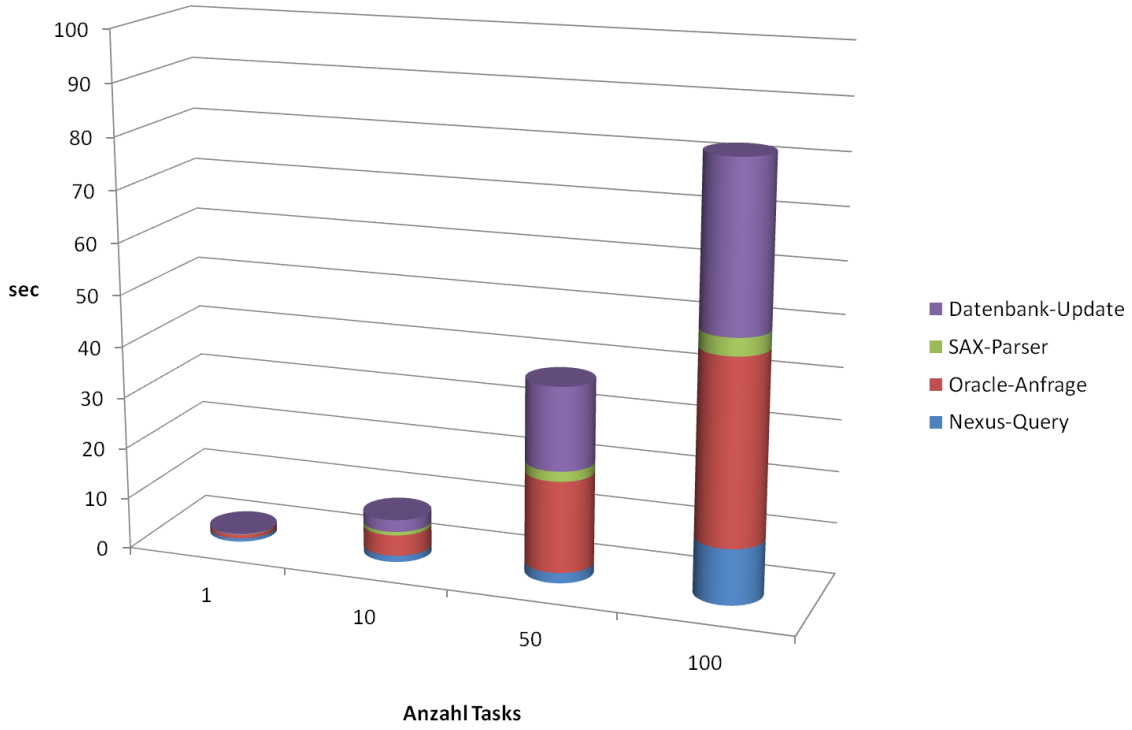

Abbildung 7.1: Datensynchronisierung beim HTC G1

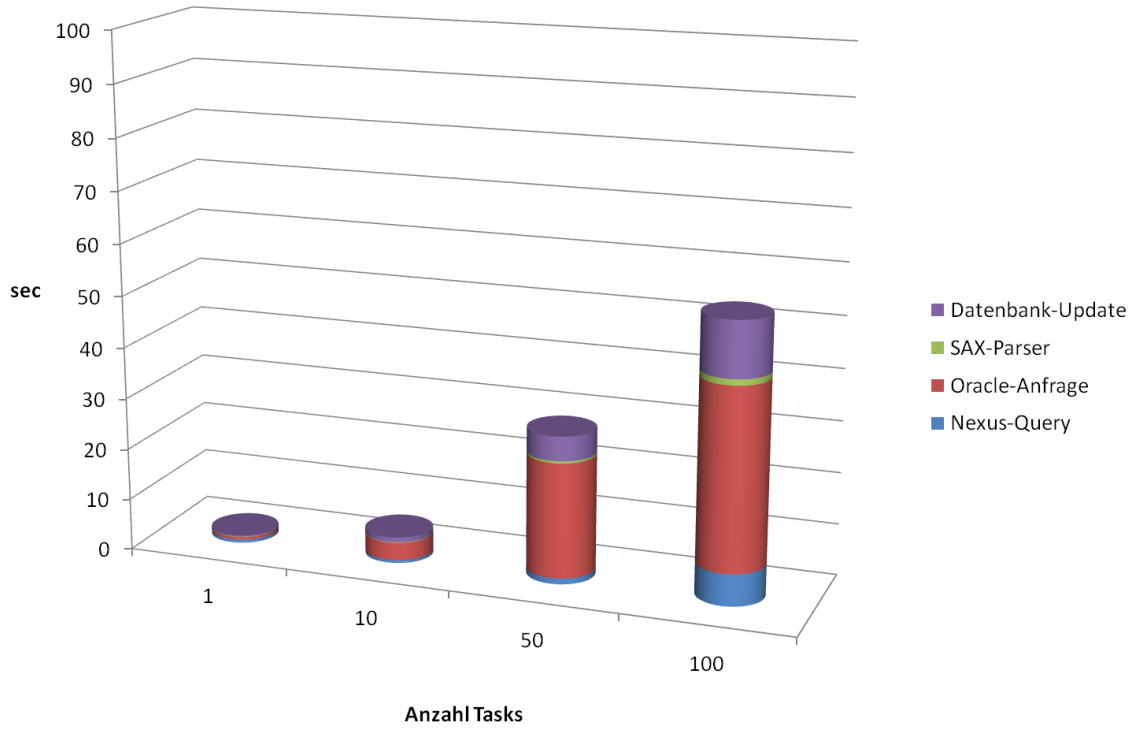

Abbildung 7.2: Datensynchronisierung beim Motorola Milestone

8 Zusammenfassung und Ausblick

Abschließend werden in diesem Kapitel die Ergebnisse mit den Zielen dieser Arbeit verglichen. Es werden Möglichkeiten zur Verbesserung aufgezeigt und ein Blick in die Zukunft soll dazu anregen, das Konzept der kontextbezogenen Arbeitsaufgabenverwaltung – mit Blick auf aktuelle IT-Trends – weiterzudenken.

8.1 Ergebnisse und ausstehende Verbesserungen

Ziel dieser Diplomarbeit war es, eine flexible, mobile Integrationslösung für die kontextbezogene Arbeitsaufgabenverwaltung zu konzipieren und prototypisch zu implementieren. Das Ergebnis sind gleich zwei Anwendungen. Zum einen die mobile App und zum anderen der Web Client. Die Notwendigkeit für zwei Lösungen begründet sich dadurch, dass je nach Verwendungskontext die Task App für den mobilen Einsatz bzw. ein Web Client für die Übersicht den Anforderungen eher entspricht. Der Schwerpunkt der Arbeit lag in der Entwickwicklung der Task App. Die Verwendung von Android als Entwicklungsplattform für die Task App hat sich in vielerlei Hinsicht ausgezahlt. Erstens wird der modulare Aufbau der Anwendung schon von Seiten der API stark vorgegeben. Die Anwendung ist dadurch aufgeteilt in Activities, Services und Provider und somit gut wartbar. Zweitens sorgt die integrierte SQLite-Datenbank dafür, dass der Benutzer auf Daten zugreifen kann, auch wenn er sich gerade in einem Funkloch befindet. Drittens ist die Verwendung der Google Maps API in Android sehr komfortabel. Viertens ist ein GPS-Sensor in den Android-Mobilfunkgeräten integriert. Damit kann die Position des Benutzers auf der Karte eingezeichnet werden.

Die Context Tasks aus [Lae09] haben sich als zuverlässig und wohl durchdacht erwiesen und konnten mit nur wenigen Anpassungen im Störungsprozess angewandt werden. Offene Fragen von [Lae09] konnten in dieser Arbeit aus Zeitmangel nicht gelöst werden. Dazu gehört die Koordinierung von mehreren Mitarbeitern, die an einer Aufgabe gleichzeitig beteiligt sind, und die Navigationsunterstützung durch einen Routenplaner. Die Google Maps API für Android unterstützt leider keine Routenplanung. Anders verhält es sich mit der Google Maps API für das Google Web Toolkit. Hier gibt es Funktionen zum Anzeigen von Routen in der Karte und in Textform. Da die Grafiken der Google Karten zweidimensional sind, wird die Visualisierung von unterschiedlichen Stockwerken der Gebäude durch diese Lösung nicht unterstützt. Eine mögliche Erweiterung in dieser Richtung könnte mittels Overlays erzielt werden. Je nachdem, in welchem Stockwerk sich der Benutzer befindet, ändern sich die eingezeichneten Overlays. Dies erfordert aber eine Positionsbestimmung, die auch in Gebäuden funktioniert (GPS funktioniert nicht in Gebäuden).

8.2 Ausblick

Aktuelle Trends deuten darauf hin, dass mobile Anwendungen immer mehr Informationen des Benutzerkontexts verwenden. Bei Augmented Reality werden Anwendungen mit den Informationen aus der realen Welt angereichert. Es gibt bereits erste Prototypen, die dem Benutzer ein digital angereichertes Bild seiner Umgebung zeigen (z. B. Wikitude und Layar). Ein Bild wird mit der Kamera des Mobilfunkgeräts aufgenommen und je nachdem, in welche Richtung der Benutzer das Gerät hält, erscheint zu dem Bildausschnitt die passende Information. Reale Gegenstände werden z. B. farblich markiert oder digitale Objekte werden zusätzlich in das Bild eingefügt. Diese Art von Interaktion wird durch Lagesensoren im Mobilfunkgerät ermöglicht. Wenn man dieses Prinzip auf den Störungsprozess bezieht, könnte man sich vorstellen, dass z. B. defekte Maschinen auf dem Display der Handy-Kamera rot eingefärbt erscheinen.

Gerade im Fabrikumfeld kann es für den Mitarbeiter wichtig sein, dass er beide Hände frei hat. Kompakte Computer können für Spezialanwendungen in die Kleidung integriert werden. Hier sind unterschiedlichste Speziallösungen denkbar. Den Beweis für den Mehrwert von in Kleidung integrierter Hard- und Software erbringt die automatische Dokumentation als Teil des Störungsprozesses. Ein Handschuh erkennt mittels RFID, welches Werkzeug bzw. welches Ersatzteil der Mitarbeiter in der Hand hält. RFID (Radio-frequency identification) ist die Technologie für die Barcodes von morgen. Da RFID-Chips mehr Informationen speichern können als die Strichcodes, die man aus Supermärkten kennt, ist es technisch möglich, beliebig viele Gegenstände mit einem solchen RFID-Chip zu versehen und mit einer eindeutigen Nummer zu kennzeichnen. Das Auslesen der RFID-Chips geschieht anders als beim konventionellen Strichcode nicht visuell sondern über elektromagnetische Wellen. RFID-Chips unterscheiden sich u. A. in der Art der Beschreibbarkeit und der verwendeten Funktechnologie. So lassen sich günstigere Exemplare nur einmal beschreiben und nur aus nächster Nähe scannen. Bei anderen Modellen kann man hingegen beliebig oft Daten überschreiben. RFID-Chips finden bereits regen Einsatz im Handel und in der Logistik, sind aber noch zu teuer, um an jedem einzelnen Joghurt-Becher angebracht zu werden. Die automatische Erfassung der Werkzeuge beim Störungsprozess zeigt, welches Potential in RFID liegt, um auf einfache Weise digitale Informationen in der realen Welt unterzubringen (Pervasive Computing).

Literaturverzeichnis

[Flao4] Flaig, Tobias: *Gegenüberstellung Nexus-Plattform - OGC Web Services Spezifikation*, Universtität Stuttgart, Diplomarbeit, 2004

[GBH+05] Grossmann, Matthias ; Bauer, Martin ; Honle, Nicola ; Kappeler, Uwe-Philipp ; Nicklas, Daniela ; Schwarz, Thomas: Efficiently Managing Context Information for Large-Scale Scenarios. In: *PERCOM '05: Proceedings of the Third IEEE International Conference on Pervasive Computing and Communications*. Washington, DC, USA : IEEE Computer Society, 2005, S. 331–340

[gml07] *OpenGIS Geography Markup Language (GML) Encoding Standard 3.2.1*. `http://portal.opengeospatial.org/files/?artifact_id=20509`. Version: 2007

[Hau09] Hauser, Stefan: *Analyse und Implementierung von Context Integration Processes in verschiedenen Anwendungsgebieten*, Universität Stuttgart, Diplomarbeit, Juni 2009. – 123 S.

[HS] Haustein, Stefan ; Seigel, James: *kSOAP 2*. `http://ksoap2.sourceforge.net/`

[Hub08] Huber, Eduard: *SmartGPS - Lokationsmodell für PerFlows*, Universtität Stuttgart, Diplomarbeit, 2008

[Inca] Inc., Google: *Android Developers*. `http://developer.android.com/`

[Incb] Inc., Google: *Anmelden für Google Maps-API*. `http://code.google.com/intl/de/apis/maps/signup.html`

[Incc] Inc., Google: *RPC Plumbing Diagram*. `http://code.google.com/intl/de/webtoolkit/doc/latest/DevGuideServerCommunication.html#DevGuidePlumbingDiagram`

[Incd] Inc., Google: *Sign Up for the Android Maps API*. `http://code.google.com/intl/de/android/maps-api-signup.html`

[JGH+06] Jandt, Silke ; Gutscher, Andreas ; Heesen, Jessica ; Müller, Jürgen ; Rossnagel, Alexander: *Datenschutzfragen mobiler kontextbezogener Systeme*. `http://ebooks.ub.uni-muenchen.de/6875/`. Version: 2006

[Lae09] Laengerer, Carsten: *Entwicklung von Konzepten fuer eine kontextbezogene Arbeitsaufgaben-Verwaltung fuer BPEL*, Universtitaet Stuttgart, Diplomarbeit, 2009

[LCW08] Lucke, D. ; Constantinescu, C. ; Westkämper, E.: Smart factory - a step towards the next generation of manufacturing. In: *Manufacturing Systems and Technologies for the New Frontier*, 2008

[Lim] Limited, Research In M.: *BlackBerry Enterprise Server for Microsoft Exchange: Feature and Technical Overview*. http://www.blackberry.com/knowledgecenterpublic/livelink.exe/Feature_and_Technical_Overview.pdf?func=doc.Fetch&nodeId=1442748&docTitle=Feature+and+Technical+Overview

[LR00] Leymann, Frank ; Roller, Dieter: *Production Workflow - Concepts and Techniques*. PTR Prentice Hall, 2000

[LW07] Lucke, Dominik ; Wieland, Matthias: Umfassendes Kontextdatenmodell der Smart Factory als Basis für kontextbezogene Workflow-Anwendungen. In: Roth, Jörg (Hrsg.) ; Küpper, Axel (Hrsg.) ; Linnhoff-Popien, Claudia (Hrsg.): *4. GI/ITG KuVS Fachgespräch Ortsbezogene Anwendungen und Dienste"*, Dr. Hut-Verlag, September 2007, S. 47–51

[Mot03] Motzer, Steffen: *Query-basierte Abfrage des Lokationsdienstes in Nexus*. 2003. – Studienarbeit Nr. 1884

[OAS07] OASIS: *BPEL 2.0 Specification*. http://docs.oasis-open.org/wsbpel/2.0/wsbpel-v2.0.pdf. Version: 2007

[Sch] Schönefeld, Marc: *Reconstructing Dalvik applications*. http://2009.confidence.org.pl/materialy/prezentacje/marc_schoenefeld_reconstructing_confidence_2009.pdf

[SMa] Sun Microsystems, Inc.: *JSR-000118 Mobile Information Device Profile 2.0*. http://jcp.org/aboutJava/communityprocess/final/jsr118/

[SMb] Sun Microsystems, Inc.: *JSR-000139 Connected Limited Device Configuration 1.1*. http://jcp.org/aboutJava/communityprocess/final/jsr139/

[Tec08] Technologies, Spectrum D.: *A Spectrum White Paper: Thoughts on Google Android*. http://www.spectrumdt.com/documents/SDTAndroidTechnicalWhitePaper_000.pdf. Version: Februar 2008

[UHW+09] Urbanski, Stephan ; Huber, Eduard ; Wieland, Matthias ; Leymann, Frank ; Nicklas, Daniela: PerFlows for the computers of the 21st century. In: *Pervasive Computing and Communications, IEEE International Conference on* 0 (2009), S. 1–6

[WK10] Werner Kurzlechner, Thomas P.: *Die 10 strategischen IT-Trends 2010*. http://www.cio.de/strategien/methoden/2212689/index1.html. Version: Januar 2010

[WKNL07] Wieland, Matthias ; Kopp, Oliver ; Nicklas, Daniela ; Leymann, Frank: Towards Context-Aware Workflows. In: Pernici, Barbara (Hrsg.) ; Gulla, Jon A. (Hrsg.): *CAiSEÂ´07 Proceedings of the Workshops and Doctoral Consortium Vol.2, Trondheim, Norway, June 11-15th, 2007*, Tapir Acasemic Press, June 2007

[WLL+09] Wieland, Matthias ; Langerer, Carsten ; Leymann, Frank ; Siemoneit, Oliver ; Hubig, Christoph: Methods for Conserving Privacy in Workflow Controlled Smart Environments. In: *Mobile Ubiquitous Computing, Systems, Services and Technologies, International Conference on* 0 (2009), S. 16–21

Alle URLs wurden zuletzt am 29.03.2010 geprüft.

Erklärung

Hiermit versichere ich, diese Arbeit selbständig verfasst und nur die angegebenen Quellen benutzt zu haben.

(Michael Schäfer)