Das WWW als Benutzerschnittstelle und Basisdienst zur Applikationsintegration für Workflow-Management-Systeme

Christoph Bußler, Petra Heinl, Stefan Jablonski, Hans Schuster, Katrin Stein

Universität Erlangen-Nürnberg
Lehrstuhl für Datenbanksysteme
Martensstr. 3, 91058 Erlangen

Zusammenfassung

Workflow-Management-Systeme steuern die Bearbeitung von Geschäftsprozessen durch Mitglieder eines Unternehmens. Dabei werden den Benutzern unterschiedliche Werkzeuge zur Verfügung gestellt: erstens Werkzeuge, die den Zugriff auf das Workflow-Management-System ermöglichen und somit Benutzerschnittstellen realisieren, und zweitens Werkzeuge zur Verrichtung der inhaltlichen Arbeit, bei denen es sich um in bezug auf das Workflow-Management-System externe Applikationen handelt. Diese Werkzeuge sollten auf allen Plattformen verfügbar sein, an denen Benutzer eines Workflow-Management-Systems potentiell arbeiten können. Ein Ansatz, um die sich aus der Heterogenität der Hardware ergebenden Probleme zu lösen, ist die Verwendung des World-Wide-Webs als Basisdienst für die Realisierung von Benutzerschnittstellen und die Integration oder Implementierung von externen Applikationen. Dabei kann man von der großen Verbreitung und der Plattformunabhängigkeit des World-Wide-Webs Gebrauch machen.

1 Einleitung

Workflow-Management-Systeme (WFMS) befassen sich mit dem Entwurf, der Ausführung, Verwaltung und Überwachung von Workflows. Ein Workflow ist die formale Beschreibung eines Geschäftsprozesses (z.B. Reisekostenabrechnung, Kreditbearbeitung). Er definiert eine Menge von Aufgaben, die von Mitgliedern eines Unternehmens bearbeitet werden. Zu diesem Zweck werden diesen Benutzern unterschiedliche Werkzeuge zur Verfügung gestellt. Die erste Klasse umfaßt alle Werkzeuge, die den Zugriff auf das WFMS ermöglichen und somit Benutzerschnittstellen des WFMS realisieren. Hierunter fällt beispielsweise die Arbeitsliste, mit deren Hilfe ein Benutzer zum einen die von ihm zu bearbeitenden Workflows angeboten bekommt und zum anderen die Durchführung der Workflows im Rahmen seiner Kompetenzen steuern kann. In diesem Zusammenhang sind auch Administrationswerkzeuge zu nennen, die zur Verwaltung und Überwachung des WFMS dienen. Die zweite Klasse umfaßt alle Werkzeuge, die der Benutzer zur Verrichtung der inhaltlichen Arbeit benötigt, d.h. Applikationen wie beispielsweise ein Tabellenkalkulations- oder ein Textverarbeitungsprogramm. Hierbei handelt es sich um in bezug auf das WFMS externe Applikationen, da diese nicht Teil des WFMS sind, sondern von diesem verwendet werden.

Beide Arten von Werkzeugen müssen auf dem Rechner, an dem ein Benutzer arbeitet, zur Verfügung stehen (nicht notwendigerweise dort installiert). Da es viele unterschiedliche Typen von Rechnern gibt, sollten Benutzerschnittstellen und Applikationen auf allen Plattformen verfügbar sein, an denen Benutzer eines WFMS potentiell arbeiten können. Voraussetzung dafür ist, daß sie entweder auf den

Plattformen selbst ausführbar oder entfernt aufrufbar sind. Im ersten Fall muß das Werkzeug oder die Applikation, sofern möglich, portiert werden. Im zweiten Fall, der vor allem Applikationen betrifft, ist das Problem der Integration zu lösen [SJHB96]. Die beste Lösung wäre, daß Werkzeuge und Applikationen auf einem Basisdienst implementiert werden, der auf allen Plattformen verfügbar ist, so daß sie nur einmal auf diesem Basisdienst implementiert werden und nicht integriert bzw. auf andere Plattformen portiert werden müssen.

Ziel dieses Beitrags ist es zu zeigen, wie Benutzerschnittstellen und die Integration bzw. Implementierung von Applikationen unter Verwendung der Konstruktionsmittel des World-Wide-Webs (WWW) realisiert werden können. Es wird gezeigt, daß das WWW ein geeigneter Basisdienst ist, mit dessen Hilfe von der Heterogenität der Hardware abstrahiert werden kann. WWW-Browser wie beispielsweise Netscape [Net] zeichnen sich dadurch aus, daß sie auf einer Vielzahl von Plattformen verfügbar sind. Webseiten werden nur einmal in der Sprache HTML (HyperText Markup Language) [HTML] erstellt und sind dann in allen WWW-Browsern darstellbar. Desweitern können Webseiten durch das Common Gateway Interface (CGI) [Gund96] [CGI] oder die architekturunabhängige Sprache Java [Flan96] [Java] dynamisch gestaltet werden.

Abschnitt 2 beschreibt Schnittstellen und Werkzeuge eines WFMS. Abschnitt 3 diskutiert, wie mit Hilfe des WWW Benutzerschnittstellen eines WFMS realisiert werden können. Dort wird zwischen aktiven und passiven Schnittstellen unterschieden, um besondere Charakteristika des WWW herauszuarbeiten. Abschnitt 4 zeigt eine mögliche Form der Applikationsintegration mit Hilfe des WWW auf. Abschnitt 5 faßt die Ergebnisse zusammen.

2 Werkzeuge und Schnittstellen eines WFMS

In Abschnitt 2.1 werden zunächst die für diesen Aufsatz relevanten Schnittstellen des Referenzmodells der Workflow Management Coalition vorgestellt. Abschnitt 2.2 beschreibt die Realisierung dieser Schnittstellen in kommerziellen WFMS. Abschnitt 2.3 gibt einen Überblick über die Architektur des WFMS $\mathcal{MOBILE}$ und erläutert die Vorteile dieser Architektur hinsichtlich der Realisierung von Benutzerschnittstellen und Applikationsintegration.

2.1 Das Referenzmodell der Workflow Management Coalition

Das Referenzmodell der Workflow Management Coalition für WFMS weist unterschiedliche Schnittstellen zwischen einer Workflow-Engine und ihren Clients aus [WfMC]:

- *Administration and Monitoring Tools*. Hierunter fallen Werkzeuge zur Administration eines WFMS.
- *Workflow Client Applications*. Hierunter wird im allgemeinen eine Arbeitsliste verstanden, mit deren Hilfe einem Benutzer die ihm zugeordneten Workflows angezeigt werden. Sie wird auch für das Anstoßen der Bearbeitung von Aufgaben mittels Applikationsprogrammen verwendet.
- *Invoked Applications*. Während des Ablaufs einer Workflowinstanz werden Applikationsprogramme aufgerufen, mit denen Benutzer ihre Aufgaben erfüllen. Diese Applikationsprogramme werden mit Hilfe dieser Schnittstelle in ein Workflow-Management-System integriert.

Die beiden ersten Schnittstellen lassen sich als Benutzerschnittstellen klassifizieren, die idealerweise eine graphische Oberfläche bereitstellen. Die dritte Schnittstellen ist ein klassisches API (application programming interface), mit Hilfe dessen (entfernte) Applikationsprogramme aufgerufen werden. Die weiteren Schnittstellen des Referenzmodells (Process Definition Tools und WF-Engines) sind für diesen Beitrag nicht relevant.

2.2 Realisierung der Schnittstellen in kommerziellen WFMS

In diesem Abschnitt werden die kommerziellen WFMS FlowMark von IBM [IBM95a] [IBM95b] und COSA von Software-Ley [COSA94a] [COSA94b] hinsichtlich der in Abschnitt 2.1 aufgeführten Schnittstellen untersucht.

Bei FlowMark handelt es sich um ein Client/Server-System. Mit dem FlowMark-Server können beliebig viele Buildtime- und Runtime-Clients verbunden sein. Ein Benutzer, der mit dem WFMS arbeiten möchte, muß zunächst einen Runtime-Client starten. Runtime-Clients sind für die Betriebssysteme OS/2, AIX und MS-Windows verfügbar. Dem Benutzer wird im Runtime-Client eine Arbeitsliste zur Verfügung gestellt und verschiedene Werkzeuge zur Administration, wie zum Beispiel ein Prozeßmonitor zum Anzeigen des Fortschritts von Prozessen. Diese Werkzeuge sind dementsprechend nur auf den Plattformen, auf denen der Runtime-Client zur Verfügung steht, verfügbar.

FlowMark bietet die Möglichkeit, eine Vielzahl unterschiedlicher Applikationsprogramme zu integrieren. Die Ausführung von OS/2, MS-DOS, MS-Windows und AIX-Programmen sowie von Shell-Skripten wird unterstützt. Zusätzlich wird die Möglichkeit geboten, Applikationen auf einem Host zu starten. Obwohl diese Liste sehr umfangreich ist, ist sie nicht vollständig. So können z.B. Programme, die via RPC (Remote Procedure Call) oder CORBA (Common Object Request Broker Architecture) aufgerufen werden, nicht direkt integriert werden.

Bei COSA handelt es sich ebenso um ein Client/Server-System. Beliebig viele Benutzerstationen können zum COSA-Server verbunden werden. Die Benutzerstation ist auf den PC-Betriebssystemen OS/2, MS-DOS und MS-Windows sowie auf Unix und VMS verfügbar. Wichtigster Bestandteil einer Benutzerstation ist die COSA-Memobox, die eine Arbeitsliste realisiert. Dem Administrator stehen Module für administrative Belange zur Verfügung, z.B. zur Zugriffsrechteverwaltung. Es können diejenigen Applikationsprogramme integriert werden, die über einen Betriebssystemaufruf gestartet werden. Auch hier sind z.B. Programme, die via RPC augerufen werden, nicht integrierbar.

Beide Systeme bieten Arbeitsliste und Administrationswerkzeuge auf einer festen Menge von Betriebssystemen an. Das Problem der Applikationsintegration ist für eine eingeschränkte Menge von Applikationsprogrammen gelöst.

2.3 Die Realisierung der Schnittstellen im WFMS *MOBILE*

Die Architektur eines WFMS wird in drei Phasen, ausgehend vom Implementierungsmodell über die Implementierungsarchitektur zur Implementierung, schrittweise konkretisiert [JaBu96]. Das Implementierungsmodell legt die grundlegende Systemstruktur fest, d.h. die grundlegenden Module und ihre Beziehungen. Die Implementierungsarchitektur verfeinert dieses Modell; sie legt u.a. fest, wie die Module in Prozessen gruppiert werden und miteinander kommunizieren. Diese beiden Schritte bestimmen die logische Architektur, die für jede konkrete Installation des WFMS gleich ist, wohingegen die tatsächliche Implementierung (physische Architektur) sich z.B. mit der Prozeßanzahl und ihrer Verteilung beschäftigt und somit von Unternehmen zu Unternehmen verschieden ist. Der Realisierung des WFMS *MOBILE* [JaBu96] liegt diese Vorgehensweise zugrunde.

Das Implementierungsmodell des WFMS *MOBILE* (Abbildung 1) legt fest, daß es ein Modul gibt, das für die Verwaltung und Integration von externen Applikationen zuständig ist. Ein weiteres ist für die Notifikation des Benutzers verantwortlich. Dieses Modul hält stets die aktuelle Information über die Zuweisung von Workflows zu Benutzern. Diese beiden und weitere Module, die für diesen Aufsatz jedoch nicht relevant sind, werden durch ein Kernmodul integriert (gestrichelte Pfeile in Abbildung 1). Der Benutzer ist über seine Arbeitsliste mit dem WFMS verbunden, wobei die Arbeitsliste über die Module Notifikation und Applikation mit dem WFMS verbunden ist. Das Modul Notifikation dient dabei dazu, die Arbeitsliste auf den aktuellen Zustand zu bringen. Das Modul Applikation besitzt die Information darüber, welche externen Applikationen einem Workflow zugeordnet sind. Diese werden dem Benutzer dann zur Verrichtung seiner Aufgabe zur Verfügung gestellt.

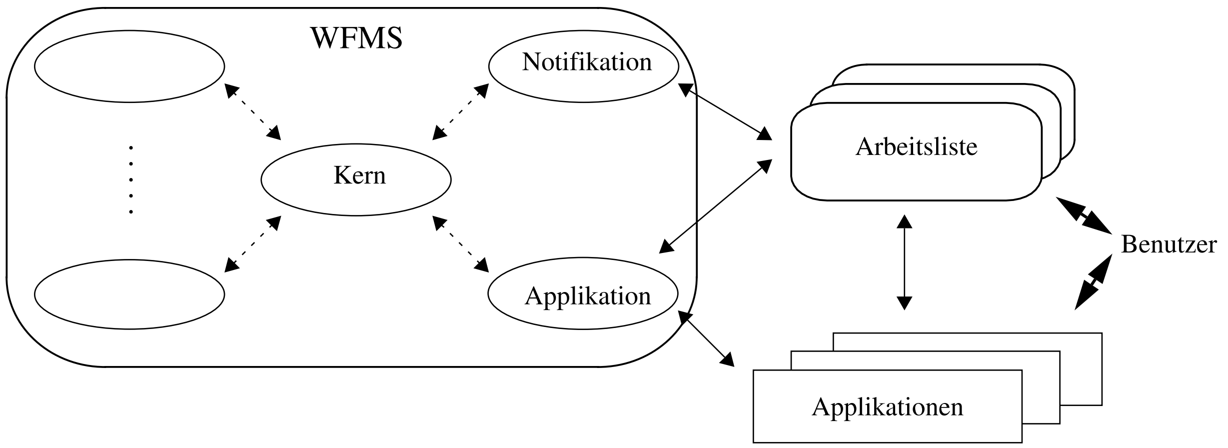

Abb. 1: Implementierungsmodell des WFMS *MOBILE*

An dieser Stelle ist noch nichts über die Realisierung der Module und der Arbeitsliste ausgesagt, und auch die Art der Kommunikation zwischen diesen ist noch völlig offen. Erst durch die weiteren Verfeinerungen werden konkrete Entscheidungen getroffen. Ein enormer Vorteil des Implementierungsmodells von *MOBILE* liegt darin, daß bei der Umsetzung in die Implementierungsarchitektur und die Implementierung aufgrund der Modularität des Modells alle weiteren Module von der Entscheidung, wie die Arbeitsliste realisiert wird und wie die Kommunikation dieser mit den beiden Modulen Notifikation und Applikation stattfindet, unbeeinflußt bleiben. Ebenso hat die Art der Kommunikation zwischen der Arbeitsliste und den externen Applikationen keinen Einfluß auf den Rest des WFMS. Eventuelle Änderungen am System werden hierdurch auf wohldefinierte Stellen begrenzt.

Die Arbeitsliste kann durch plattformspezifische Programme realisiert werden, z.B. unter MS-Windows. In diesem Fall treten alle Heterogenitätsprobleme auf, die bereits in Abschnitt 1 erwähnt wurden. Zum einen muß die Arbeitsliste portiert, zum anderen die Kommunikation zwischen heterogenen Rechensystemen geregelt werden. Im Falle der Applikationsintegration treten die gleichen Probleme auf. Ein Ansatz zur Lösung des Applikationsintegrationsproblems, der sich mit der Kommunikation zwischen heterogenen Systemen beschäftigt, wird in [SJHB96] vorgestellt. Ein völlig anderer Ansatz besteht darin, von der Heterogenität der Systemstruktur zu abstrahieren und homogene Werkzeuge einzusetzen. Hier bildet das WWW einen idealen Basisdienst. So kann die Arbeitsliste beispielsweise in einem WWW-Browser realisiert werden und über ein CGI-Skript mit dem WFMS kommunizieren (Abschnitt 3.1) oder mit Hilfe eines Java-Programms über einen ORB (Object Request Broker) (Abschnitt 3.2). Zusätzlich kann Java zur Applikationsintegration verwendet werden (Abschnitt 4).

3 WWW-Browser als Benutzerschnittstelle in WFMS

Generell können innerhalb von WFMS zwei Arten von Benutzerschnittstellen unterschieden werden. Die erste Klasse zeichnet sich dadurch aus, daß eine Kommunikation zwischen Schnittstelle und WFMS ausschließlich auf Befehl des Benutzers stattfindet. Das Werkzeug, das die Schnittstelle realisiert, tritt gegenüber dem WFMS ausschließlich als Client in Erscheinung. Jegliche Kommunikation geht von dem Werkzeug auf Befehl des Benutzers aus. Die Schnittstelle ist insofern passiv als sie ohne Einwirkung des Benutzers keinerlei Operationen ausführt. Benutzerschnittstellen der zweiten Kategorie können gegenüber dem WFMS auch als Server auftreten. Die Kommunikation kann in diesem Fall auch vom WFMS ausgehen. Die Schnittstelle ist insofern aktiv als sie auch ohne Zutun des Benutzers Befehle ausführen kann. Im weiteren wird nur kurz von passiven und aktiven Benutzerschnittstellen gesprochen.

In die Klasse der passiven Benutzerschnittstellen gehören beispielsweise Administrationswerkzeuge, mit deren Hilfe Befehle abgesetzt werden und die daraufhin den Erfolg oder Mißerfolg des abgesetzten Befehls bzw. errechnete Ergebnisse anzeigen. Sie sind Clients, die mit den Servern eines WFMS ausschließlich auf Befehl des Benutzers kommunizieren.

Aktive Benutzerschnittstellen sind z.B. Arbeitslisten. Auch dort können Befehle abgesetzt werden, beispielsweise zur Bearbeitung eines Workflows. In diesem Fall agieren sie als Clients des WFMS. Jedoch müssen sie von Zeit zu Zeit aktualisiert werden, um den Sachbearbeitern veränderte Zustände anzuzeigen, z.B. neu zugewiesene Workflows. In diesem Fall kommuniziert das WFMS mit der Arbeitsliste, um diesem die Veränderung mitzuteilen. In diesem Moment ist eine Arbeitsliste ein Server und das aufrufende WFMS ein Client. Die Rollen Client bzw. Server sind vertauscht.

Aufgrund der weiten Verbreitung des WWW bzw. der großen installierten Anzahl von WWW-Browsern liegt die Idee nahe, diese innerhalb eines WFMS zu benutzen, um passive und aktive Benutzerschnittstellen zu implementieren. Ein gewichtiger Vorteil ist, daß in diesem Fall die Heterogenität der Hardware, auf der die Browser laufen, außer Acht gelassen werden kann, da mit der Portierung eines Browsers automatisch die Benutzerschnittstelle des WFMS portiert wird. Natürlich ist es in diesem Fall notwendig, daß ein WFMS einen WWW-Server bereitstellt, der Anfragen von WWW-Browsern beantworten kann. Technisch gesehen ist dies jedoch kein Problem (sondern nur Implementierungsaufwand), so daß wir diesen Punkt im folgenden nicht weiter diskutieren.

3.1 Implementierung von passiven Benutzerschnittstellen mit WWW-Browsern

Wie schon angedeutet, ist ein WWW-Browser, der eine passive Benutzerschnittstelle realisiert, ein Client eines WFMS, genauer dessen WWW-Servers (Abbildung 2). Eine Anfrage an das WFMS ist im WWW-Browser als Link oder als auszufüllendes Formular dargestellt. Die Anfrage wird auf

Befehl des Benutzers, d.h. durch Folgen des Links bzw. Drücken des “Submit”-Buttons nach Ausfüllen des Formulars, abgesetzt (1). Vor allem durch Formulare kann dem Benutzer die Möglichkeit geboten werden, eine Vielzahl unterschiedlicher Anfragen zu spezifizieren. Zur Kommunikation zwischen dem WWW-Server und dem WFMS eignet sich das Common Gateway Interface (CGI), dessen Ziel die Vereinheitlichung der Kommunikation zwischen WWW-Server und externen Programmen ist [Gund96], da das WFMS sich für den WWW-Server als externes Programm darstellt. Soll die Anfrage von einem CGI-Skript oder -Programm bearbeitet werden, so muß entweder die URL (Uniform Resource Locator) des Links mit einem CGI-Skript oder das Formular mit der URL eines CGI-Skripts assoziiert sein. Der WWW-Server erkennt, daß es sich bei der angeforderten URL um ein CGI-Skript handelt. In diesem Fall gibt er nicht den Dateiinhalt zurück, sondern stößt die Ausführung des CGI-Skripts an (2), das die Anfrage anschließend bearbeitet. Der WWW-Server muß den Formularinhalt an das CGI-Skript übergeben, wofür es die zwei Möglichkeiten “Put” und “Post” gibt, die hier jedoch nicht vertieft werden sollen. Das CGI-Skript kommuniziert mit dem WFMS (3),(4) über dessen spezifische Schnittstelle. Hierzu muß es zunächst die vom WWW-Server erhaltene Information, d.h. den Formularinhalt, für die Anfrage aufbereiten und anschließend die Antwort des WFMS für die Darstellung im Browser aufbereiten. Das Ergebnis der Anfrage wird an den WWW-Server zurückgegeben (5), an den WWW-Browser in Form eines HTML-Dokuments gesendet (6) und vom Browser dargestellt.

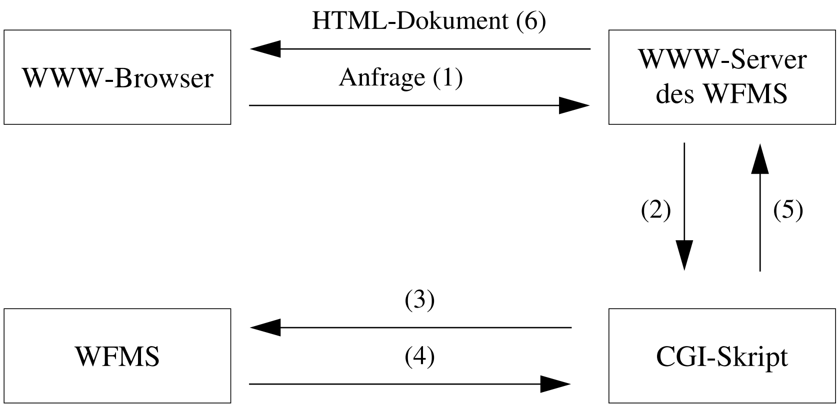

Abb. 2: Realisierung von passiven Benutzerschnittstellen

Auf diese Weise können alle Anfragen an ein WFMS, für die eine passive Benutzerschnittstelle ausreicht, im WWW-Browser bereitgestellt werden. Der Benutzer kann somit an einem beliebigen Rechner, auf dem ihm ein WWW-Browser zur Verfügung steht, eine passive Benutzerschnittstelle des WFMS aufrufen, d.h. die entsprechende Webseite laden.

3.2 Implementierung von aktiven Benutzerschnittstellen mit WWW-Browsern

Eine Arbeitsliste ist prinzipiell als rein passive Benutzerschnittstelle realisierbar. Wann immer der Benutzer einen “Refresh”-Button betätigt, wird sie im WWW-Browser neu aufgebaut, d. h. aktualisiert. Dabei werden die gleichen Schritte durchlaufen wie in Abschnitt 3.1 beschrieben. Änderungen, die zwischen den Aktualisierungen stattfinden, sind damit nicht zum Zeitpunkt der Änderung sichtbar, sondern erst auf Anfrage. Das kann zu unerwarteten Situationen führen, wenn z. B. ein Workflow, der noch auf der Arbeitsliste steht, nicht mehr ausführbar ist. Wäre die Arbeitsliste aktualisiert worden, so wäre der Workflow nicht mehr auf ihr vorhanden gewesen. Möchte der Benutzer Fehlermeldungen dieser Art vermeiden, so muß er vor jedem Befehl, den er auf der Arbeitsliste ausführt, eine Aktualisierung durchführen, was im Falle, daß keine Änderungen vorliegen, unnötigen Ressourcenverbrauch bedeutet.

Die weiteren Möglichkeiten, die CGI zur Gestaltung dynamischer Webseiten bietet, "Client Pull" und "Server Push", sind für die Implementierung aktiver Benutzerschnittstellen nicht geeignet. Client Pull bewirkt ein periodisches "Refresh" der aktuellen Webseite. In spezifizierten Zeitabständen wird eine neue HTTP-Verbindung aufgebaut und das Dokument erneut angefordert. Diese Lösung hat gegenüber der obigen den Vorteil, daß der Benutzer nicht selbst das "Refresh" anfordern muß. Sie hat jedoch den entscheidenden Nachteil, daß, hohe Aktualität vorausgesetzt, die Zeitintervalle sehr klein gewählt werden müssen und diese Lösung somit einen noch höheren Ressourcenverbrauch bedeutet als ein Benutzer-Refresh. Auch die Methode Server Push ist nicht geeignet. Die Information wird in diesem Fall vom Server ausgehend an den Browser gesendet. (Dabei wird die Verbindung aufrecht erhalten.) Dies geschieht jedoch ebenfalls periodisch und hat somit die gleichen Nachteile.

Durch Verwendung von Java kann die passive Arbeitsliste zu einer aktiven umgestaltet werden. Die prinzipielle Idee ist, daß im WWW-Browser auf der die Arbeitsliste realisierenden Webseite ein Java-Programm im Hintergrund läuft und mittels eines dedizierten Sockets eine aktive Verbindung zum WFMS unterhält. Diese Verbindung wird vom WFMS benutzt, um bei einer Inhaltsänderung der Arbeitsliste dem Java-Programm eine entsprechende Nachricht zu übermitteln. Das Java-Programm agiert in diesem Fall als Server. Es bewirkt anschließend eine Aktualisierung der Arbeitsliste und vermittelt so dem Benutzer deren neuesten Stand. In diesem Fall würde das Java-Programm ohne Zutun des Benutzers eine Anfrage an den WWW-Server starten. Diese kann auf die gleiche Art bearbeitet werden wie in Abschnitt 3.1 beschrieben. Der Vorteil dieser Methode liegt darin, daß eine Aktualisierung der Arbeitsliste ausschließlich dann stattfindet, wenn eine Änderung des Inhalts eingetreten ist.

Eine weitere Verbesserung des Protokolls zwischen WFMS und Arbeitsliste kann erreicht werden, wenn nicht nur die Existenz der Änderung vom WFMS an die Arbeitsliste weitergegeben wird, sondern auch die Daten, die sich geändert haben. In diesem Fall muß sich die Arbeitsliste nicht selbst an das WFMS bzw. dessen WWW-Server wenden, um die Änderungen zu holen, da sie vom WFMS mitgeliefert werden (dies umfaßt sowohl Löschungen als auch Neueinträge und Änderungen existierender Einträge).

Weitaus komfortabler läßt sich die Kommunikation zwischen der Java-Applikation in der Arbeitsliste und dem WFMS auf Basis von CORBA [OMG92] gestalten. Java-ORB (Object Request Broker) ermöglichen Client/Server-Anwendungen im Internet. Das erste Java-ORB-Produkt war Black Widow von der Firma Post Modern Computing [BW] [Visi] [EmFV96]. Es unterstützt das Internet-Inter-ORB-Protokoll (IIOP). Dieses Protokoll ermöglicht die Interoperabilität zwischen allen CORBA 2.0-kompatiblen ORB. Die Kommunikation zwischen beliebigen CORBA-Objekten, die in einer beliebigen Sprache implementiert sind, ist möglich, sofern diese eine entsprechende Schnittstelle anbieten und deren ORB das IIOP befolgt. Das IIOP ist Teil der UNO-Spezifikation (Universal Networked Objects) der OMG [OMG94]. Diese definiert ferner ein IOR-Format (Interoperable Object Reference), das das Referenzieren von Objekten in "fremden" ORB ermöglicht ("normale" Objektreferenzen sind ORB-spezifisch). Wie die Kommunikation zwischen Objekten in verschiedenen ORB auf der Basis von UNO, IIOP und IOR genau erfolgt, soll an dieser Stelle nicht vertieft werden, siehe z.B. [Bran95].

Durch die Verwendung von ORB muß nicht auf der Ebene von Sockets programmiert werden, sondern nach dem objekt-orientierten CORBA-Modell, d. h. mit Methodenaufrufen. Voraussetzung hierfür ist die Bereitsstellung von CORBA-Objekten durch WFMS und Arbeitsliste (Abbildung 3). Das zur Arbeitsliste gehörige Objekt (AL-Objekt in Abbildung 3) stellt an seiner Schnittstelle die Methode “update” zur Aktualisierung der Arbeitsliste zur Verfügung. Es agiert als Server-Objekt. Die Implementierung des AL-Objekts ist in dem Fall, daß die Arbeitsliste durch einen WWW-Browser realisiert wird, ein Java-Programm. Dieses wird von einem Java-ORB verwaltet. Das WFMS agiert als Client. Es muß Methodenaufrufe via CORBA absetzen können. In Abbildung 3 ist ein Notifikationsobjekt als Teil des WFMS eingezeichnet, das diese Aufgabe übernimmt. Das Notifikationsobjekt kann z.B. in C++ implementiert sein. In diesem Fall gehört es zu einem C++-ORB. Das Client-Objekt benötigt die IOR (Internet Object Reference) des AL-Objekts, um dessen Methoden aufrufen zu können.

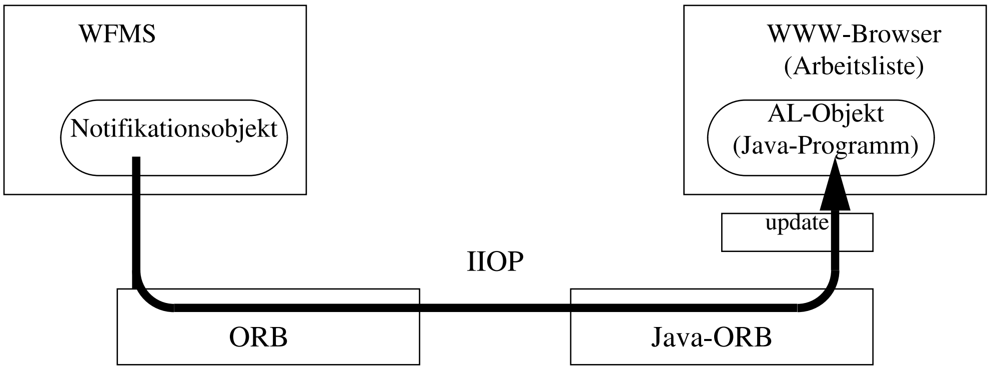

Abb. 3: Realisierung einer aktiven Arbeitsliste mit Hilfe von Java und CORBA

Java zeichnet sich vor allem dadurch aus, daß die Sprache architektur-unabhängig, portabel und interpretiert ist. Das bedeutet, daß ein Java-Applet oder -Programm nur einmal erstellt werden muß und auf jeder Plattform, auf der dem Benutzer ein Java-fähiger WWW-Browser zur Verfügung steht, ausführbar ist. Die Techniken des WWW sind nicht nur zur Implementierung von Werkzeugen, sondern auch für die Integration von Applikationen in ein WFMS anwendbar. Insbesondere die Verwendung von Java hat eine ganze Reihe von gewichtigen Vorteilen, die im nächsten Kapitel erläutert werden.

4 Java im WWW als Basisdienst zur Applikationsintegration in WFMS

Ein großes Problem des Workflow-Managements ist die Integration von Applikationsprogrammen [JaBu96] [SJHB96]. Das beinhaltet zum einen die technische Integration in ein WFMS und zum anderen die Versorgung einer Applikation mit Daten und das Entgegennehmen von Rückgabewerten. Sind Applikationen zudem nur auf bestimmten Plattformen vorhanden, müssen sie entfernt gestartet werden. Wird in interaktiven Programmen ein Windowsystem benutzt, das nicht auf der Maschine des Aufrufers installiert ist, so sind diese Programme nicht uneingeschränkt in einem Workflow verwendbar, da sie nicht an allen Maschinen, an denen Benutzer arbeiten, verfügbar sind.

4.1 WWW als graphisches Basissystem für interaktive Applikationen

Interaktive Applikation sind, wie bereits kurz erläutert wurde, unter Umständen nicht auf allen Plattformen ausführbar bzw. von diesen aus aufrufbar. Unter der Voraussetzung, daß solch eine interaktive Applikation ein API anbietet, das dem GUI (Graphical User Interface) entspricht, kann auch hier das WWW als Basisdienst eingesetzt werden, um die Applikation auf allen Plattformen verfügbar zu machen. Die Grundidee ist, die graphische Darstellung einer interaktiven Applikation durch Formulare im WWW-Browser zu implementieren. Sobald ein Benutzer ein Formular ausgefüllt hat, sendet er es durch Drücken des "Submit"-Buttons an den entsprechenden WWW-Server, der ein CGI-Skript ausführt, das mit der Applikation kommuniziert. Dieser Ablauf wurde ausführlich in Abschnitt 3.1 beschrieben; in Abbildung 2 muß lediglich das WFMS durch eine Applikation ersetzt werden. Auf diese Weise ist die graphische Schnittstelle auf einer großen Anzahl von Plattformen vorhanden und bedarf auch keiner maschinenspezifischen Anpassung. Allerdings muß für jede mögliche Eingabemaske der interaktiven Applikation ein Formular und ein zugehöriges CGI-Skript zur Kommunikation erstellt werden. Der entscheidende Vorteil gegenüber einer anderen möglichen Lösung ist jedoch wiederum, daß dies nur einmal geschehen muß und die interaktive Applikation dann auf allen Plattformen verfügbar ist, auf denen ein WWW-Browser ausführbar ist. Zudem sind graphische Schnittstellen in einem WWW-Browser einfacher zu gestalten als mit eventuell mitgelieferten Forms-Paketen eines WFMS. COSA [COSA94b] stellt z.B. mehrere hundert Funktionen zur Programmierung von Formularen bereit. Mit Java können graphische Darstellungen funktional dahingehend erweitert werden, daß ein Teil der Funktionalität vom Java-Programm übernommen wird, beispielsweise Bereichsabprüfungen schon durchgeführt werden können, bevor der Inhalt des Formulars an die entsprechende Applikation gesendet wird.

4.2 Java im WWW als ideale Applikationsintegrationsplattform für WFMS

Wenn nur die graphische Schnittstelle im WWW-Browser realisiert ist, muß zusätzlich im Netz das zugehörige Applikationsprogramm ausgeführt werden, das die Eingaben der Schnittstelle bearbeitet. Dieses Programm muß auf dem Betriebssystem ausgeführt werden, für das es erstellt wurde. Die Probleme der Heterogenität bleiben auf dieser Ebene bestehen, denn die Eingaben müssen zum Applikationsprogramm transferiert werden. Werden nun Applikationsprogramme komplett mittels Java programmiert, im weiteren als Java-Applikationen bezeichnet, so ist die gesamte Abarbeitungslogik zusätzlich zu der graphischen Oberfläche im WWW-Browser integriert. Damit lassen sich die kompletten Java-Applikationen automatisch in einen WWW-Browser laden, wenn ein Workflow, d. h. ein Anwender, der diesen bearbeitet, diese benötigt. Die Heterogenitätsprobleme entfallen auf diese Weise. Wird eine neue Version einer Java-Applikation erstellt, so wird diese nur einmal eingespielt. Benutzer können durch erneutes Laden der zugehörigen Webseite auf die neue Version zugreifen. Das Konfigurationsmanagement vereinfacht sich dadurch erheblich.

Erzeugen Benutzer mit Hilfe von Java-Applikationen Daten, die dauerhaft gespeichert werden sollen, so müssen diese vom Java-Programm so abgelegt werden, daß der Benutzer zu einem späteren Zeitpunkt mit Hilfe dieses oder eines anderen Java-Programms wieder auf sie zugreifen kann. Java bietet die Möglichkeit, Dateien zu lesen und zu schreiben. Diese befinden sich dann unter Verwaltung des Betriebssystems auf der lokalen Maschine. Diese Lösung ist hinreichend, sofern der Benutzer nicht von anderen Rechnern aus auf die Daten zugreifen will. Java erlaubt ferner den Zugriff auf URL. Ein Java-Programm kann entweder direkt aus einer URL lesen oder sich zu einer URL verbinden und über diese Verbindung lesen und schreiben. Ist der URL ein CGI-Skript zugeordnet, so kann über dieses z.B. in eine Datenbank geschrieben werden. Die Daten sind mit Hilfe eines entsprechenden CGI-Skripts von überall aus lesbar und können in einem Java-Programm verarbeitet werden.

Das folgende Beispiel soll die Idee erläutern. Wir nehmen an, daß es ein Tabellenkalkulationsprogramm gibt, das vollständig in Java programmiert ist. Startet nun ein Benutzer einen Workflow zur Erstellung einer Tabellenkalkulation, so wird überprüft, ob die zugehörige Java-Applikation bereits geladen ist. Ist sie das nicht, so wird sie automatisch über das Netz geholt und gestartet. Das Java-Programm erstellt auf Basis der vom Benutzer eingegebenen Daten eine Tabellenkalkulation. Auf Wunsch des Benutzers können die Daten über eine URL-Verbindung zu einem CGI-Skript transferiert werden, das diese in einer Datenbank speichert. In einem weiteren Workflowschritt kann der Benutzer ggf. wieder auf diese zugreifen. Hat der Benutzer die Bearbeitung abgeschlossen, wird die Java-Applikation beendet.

5 Zusammenfassung

Es wurde gezeigt, daß das WWW geeignete Konstruktionsmittel bereitstellt, um sowohl Benutzerschnittstellen von WFMS zu realisieren als auch die Applikationsintegration durchzuführen. Passive Benutzerschnittstellen können durch Anfragen an einen WWW-Server realisiert werden, denen ein CGI-Skript zugeordnet ist, das mit dem WFMS kommuniziert. Mit Hilfe von Java können aktive Benutzerschnittstellen realisiert werden. Webseiten können ebenso zur Realisierung der graphischen Benutzeroberfläche von interaktiven Applikationen eingesetzt werden, wobei hier das gleiche Vorgehen zugrunde liegt wie bei passiven Benutzerschnittstellen. Sehr vielversprechend ist die komplette Implementierung von Applikationen in der Sprache Java.

6 Literatur

Bran95 Brando, T.J.: Interoperability and the CORBA Specification, MITRE Document MP 95B-58, Februar 1995. http://www.mitre.org/research/domis/reports/UNO.html

BW Black Widow. http://www.pomoco.com/

CGI Rob McCool: The Common Gateway Interface. http://hoohoo.ncsa.uiuc.edu/cgi/overview.html/

COSA94a COSA Referenz Handbuch. Software-Ley GmbH, 1994

COSA94b COSA Administrator Handbuch. Software-Ley GmbH, 1994

EmFV96 Emmerich, W., Ferrandina, F., Vogel, A.: Integration von Java mit CORBA: portable Client/Server-Applikationen im Internet. In: OBJEKTspektrum 4/96

Flan96 Flanagan, D.: Java in a Nutshell. O'Reilly & Associates, Inc., 1996

Gund96 Gundavaram, S.: CGI Programming on the World Wide Web. O'Reilly & Associates, Inc., 1996

HTML HTML Documentation/Utilities. http://www.cam.org/~intsci/dochtml1.html/

IBM95a IBM FlowMark, Modeling Workflow, Release 2.1, International Business Machines Corp., 1995

IBM95b IBM FlowMark, Managing Your Workflow, Release 2.1, International Business Machines Corp., 1995

JaBu96 Jablonski, S., Bußler, C.: Workflow Management - Modeling Concepts, Architecture and Implementation. International Thomson Computer Press, September 1996

Java Java - Programming for the Internet. http:/java.sun.com/

Net Netscape. http://home.netscape.com/

OMG92 Object Management Group: The Common Object Request Broker Architecture and Specification (CORBA). Object Management Group (OMG), Framingham, MA, 1992

OMG94 Object Management Group: Universal Networked Objects, OMG TC Document 94-9-32. Object Management Group (OMG), Framingham, MA, September 1994

SJHB96 Schuster, H., Jablonski, S., Heinl, P., Bußler, C.: A General Framework for the Execution of Heterogenous Programs in Workflow Management Systems. In: Proc. First IFCS Int. Conf. On Cooperative Information Systems (CoopIs 96), Brussels, June 1996, pp.104-113

Visi Visigenic: VisiBroker for Java. http://www.visigenic.com/BW/bwhome.html/

WfMC Workflow Management Coalition: Reference Model and Interfaces. http://www.aiai.ed.ac.uk/WfMC/IMG/WfMCref-model.gif