ANT – Active Node Technology
Technischer Überblick und Anwendungen

Bernhard Zwantschko (bzwan@iicm.edu)
und
Christian Gütl (cguetl@iicm.edu)

Institut für Informationsverarbeitung und Computergestützte neue Medien
Technische Universität Graz, Austria; Schießstattgasse 4a

Zusammenfassung

Die vorliegende Arbeit stellt in kurzen Umrissen die wesentlichsten Merkmale der Active Node Technology (ANT) vor. ANT dient der transparenten Integration verschiedenster Netzwerkkomponenten und Services durch Generieren eines virtuellen Adreßraumes. ANT inkludiert dabei ein Naming- und Directory Service, Property Support sowie asynchrone und snchrone Kommunkation. Weiters werden neueste Designaspekte durch die Implementierung dynamisch generierter Services sowie der strikten Trennung von Daten und Strukturen berücksichtigt.

Im zweiten Teil werden erste Erfahrungen mit ANT in diversen Projekten vorgestellt und ein Ausblick auf zukünftige Entwicklungen gegeben.

Keywords

Virtueller Adreßraum, Protokollunabhängigkeit, Services, Corba, ANT

1. ANT – Active Node Technology

Im Rahmen eines Projektes für die Einführung eines Medizin Telematik Zentrums (MTZ) stand das IICM vor der Aufgabe, ein strukturiertes, sicheres, multimediataugliches, verteiltes, dynamisches und hoch kooperatives Netzwerk zu schaffen. Das Netzwerk sollte es ermöglichen bestehende medizinische Dienste investitionserhaltend und einfach zu integrieren, Patientenakten, obwohl verteilt gespeichert virtuell zentral darzustellen und zu administrieren, Methoden für die Gewährleistung von Sicherheit und Vertraulichkeit transparent zu integrieren.

Mögliche Basistechnologien für die Implementierung eines solchen Netzwerkes wären unter anderem CORBA [3] oder Remote Method Invocation (RMI) [6]. Beide Technologien zeigten jedoch Nachteile (siehe Vergleich ANT-CORBA-RMI), die in der Folge zum Design einer maßgeschneiderten Middleware führten.

Active Node Technology (ANT) ist ein verteiltes Netzwerk aus aktiven Java Objekten die einen virtuellen Adreßraum aufspannen und sich durch Einfachheit, transparente Protokollintegration und Plattformunabhängigkeit auszeichnen. ANT enthält per Definition ein Naming- und Directoryservice, Property-Unterstützung und synchrone sowie asynchrone Kommunikation. Das Datenmanagement unterliegt einer strikten Trennung von Daten einerseits und Struktur sowie Protokollen andererseits. Die vorteilhafte Plattformunabhängigkeit erbt ANT von Java ebenso wie ein dynamisches Servicemanagement.

In der vorliegenden Arbeit wird eine kurze Vorstellung der Active Node Technology aufgezeigt und im Anschluß werden aktuelle und zukünftige Entwicklungen basierend auf dieser Technologie diskutiert.

2. ANT – Ein virtueller Adreßraum

Der virtuelle Adreßraum stellt die Basis von ANT dar und ist am ehesten vergleichbar mit einem UNIX Filesystem. Alle Objekte (ANTs) sind hierarchisch organisiert und haben eine eindeutige Adresse. Wie unter UNIX existiert nur ein eindeutiges Root Objekt.

Die Adresse der einzelnen Objekte ergibt sich aus dem Namen der Objekte im Pfad getrennt durch einen Schrägstrich (/). Die Adresse des Objektes `joe` (siehe Abb. 1) ist somit `/tmp/coll/joe`.

Die Spitze des Adreßraumes wird durch das Root Objekt und andere Runtime Objekte gebildet. Diese Objekte existieren nur im Memory und haben in der Regel keine persistente Entsprechung. Runtime Objekte haben zweierlei Aufgaben: Sie stellen temporäre Objektspeicher dar oder sie fungieren als Bindeglied zu Subräumen. Im letzteren Fall werden sie als *mounter* (nach dem UNIX Befehl `mount`) bezeichnet.

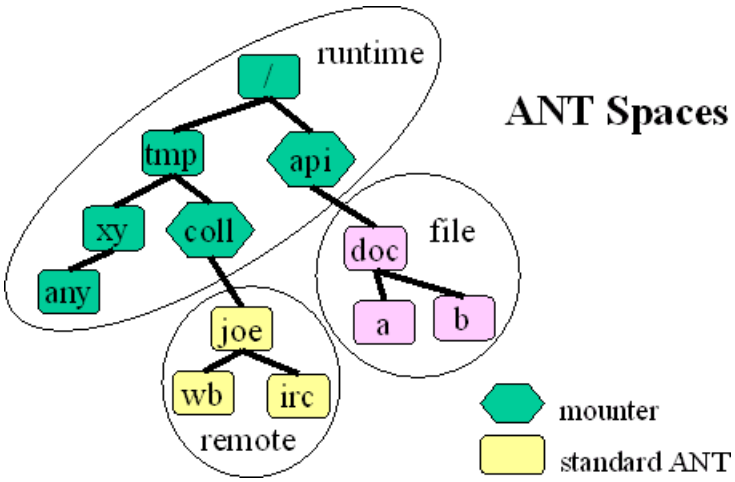

Abb. 1: ANT Space. Der dargestellte virtuelle Objektraum zeigt die temporären Runtime Objekte an der Spitze der Adreßhierarchie. Zwei *mounter* spannen Subräume auf, wobei `api` ein Verzeichnis und `coll` den Objektraum eines anderen Rechners "protokollunabhängig" erreichbar macht.

Ein *mounter* spannt ein Subraum auf, z. B. den Inhalt eines Dateisystems, eines Web Servers, den ANT Baum auf einem Remote Rechner (ANT Server) oder eine Anwendung, die sich in Form eines Servicebaumes darstellt (siehe ....). Die Hauptaufgabe des *mounters* ist dabei das Schaffen eines protokollunabhängigen und damit transparenten Zugriffes auf die Daten. So sollen Operationen wie kopieren, schreiben, etc. protokollübergreifend ohne Kenntnis des zugrunde liegenden Protokolles möglich sein.

Die Bezeichnung *virtueller* Adreßraum trägt dem Umstand Rechnung, daß viele Subräume nur virtuell existieren. So wird nicht der gesamte Inhalt eines Web Servers in Form eines ANT Baumes aufgebaut (und damit dupliziert), der *mounter* schafft lediglich die Illusion eines solchen Baumes. Für den Anwender ist dies jedoch vollkommen transparent.

Subräume und damit *mounter* können verschiedenster Art sein. Die offensichtlichste Variante ist die Anbindung persistenter Datenspeicher. Als Beispiele seien hier genannt: Dateisysteme (Protokoll *file*), Web- und News-Server (Protokolle *http*, *ftp*, *hwtp*, *news*), Datenbankanbindung über ODBC.

Ein besonderes Protkoll stellt das *DIDI*-Protokoll (DINO-DINO[1]-Protokoll) dar. Dieses Protokoll ermöglicht das Mounten von ANT Adreßräumen auf anderen Rechnern, d.h. es können sogenannte ANT Server mit eingebunden werden. Dadurch kann ein Teil des server-seitigen ANT Baumes für den Client transparent sichtbar gemacht werden. Alle für den Client sichtbaren ANTs können wie lokale ANTs verwendet werden. Die Verbindung bedient sich der Serialisation von Java Objekten und benutzt eine sichere (SSL) TCP/IP Verbindung. Der ANT-Server kann durch zusätzliche Module auch mit User Management und entsprechendem sicherem Zugriff ausgestattet werden. Damit wird gewährleistet, daß in sensiblen Bereichen - wie z.B. bei Patientendaten nur autorisierte Personen Objekte abrufen können. Der server-seitige Adreßraum wird dabei wie jeder andere ANT Adreßraum durch Mounten diverser Resourcen und Services erzeugt und kann mit Hilfe der *DIDI* Verbindung auf einfache Art und Weise administriert werden.

Mit ANT lassen sich nicht nur Ressourcen sondern auch Applikationen *mounten*. Die Anwendung wird als Hierarchie von Objekten dargestellt, wobei jedes Objekt spezifische Services anbietet. So kann beispielsweise ein Workflow Management Programm durch entsprechende Räume bzw. Folder und Workflow-Objekte dargestellt werden. Eine andere sehr nützliche Anwendung kann im Bereich der Zusammenarbeit realisiert werden. Durch ein geeignetes Protokoll kann über den Mount-Mechanismus ein Chat-System für zwei oder auch

[1] DINO, Distributed Interactive Network Objects, ist der historische Projektname der durch die Entwicklung der Active Node Technology abgelöst wurde.

mehrere Teilnehmer verwirklicht werden. Auf die Anwendung von Agent Systemen aufbauend auf den Basisfunktionen der ANTs wird im nachfolgenden noch erläutert werden.

3. ANT – Die Komponenten

Vom technologischen Standpunkt ist die Active Node Technology die Definition einer stark abstrahierten Schnittstelle. Die einzelnen Methoden lassen sich im Hinblick auf ihre Verwendung im wesentlichen in fünf logische Gruppen aufspalten. Diese werden im folgenden noch genauer erörtert.

Die Beschreibung der Schnittstelle erfolgt durch eine einzige Java-Klasse: `DAnt`. Im Gegensatz zu RMI oder CORBA geht ANT jedoch über die Beschreibung von Interfaces hinaus und bietet mit der Klasse `DAnt` die Implementierung eines wohldefinierten Zugriffsobjektes. Die Klasse `DAnt` ist `final` und stellt die einzige Verbindung zwischen diversen Anwendungen und den ANT Objekten dar. Durch dieses Design läßt sich ein eindeutiges und gleiches Verhalten aller ANT Objekte - einschließlich der Fehlerbehandlung - erreichen. Dadurch kann ein HTML Dokument im Dateisystem genauso angesprochen werden wie ein Textdokument auf einem WWW-Server oder ein email Objekt im Rahmen eines temporären Workflow-Systems.

Um Java Programmierern einen leichten Zugang zur Active Node Technology zu ermöglichen, wurde das Design der Klasse `DAnt` stark an die Java Klasse `File` angelehnt. Das Generieren eines `DAnt` Objektes erzeugt noch kein (persistentes) ANT Objekt, sondern stellt lediglich alle Möglichkeiten des Zugriffes auf dieses Objekt zur Verfügung. Wie bei der Klasse `File` kann die Existenz des ANT Objektes festgestellt werden mit `ant.exists()` oder es wird eine der vielfältigen Operationen auf dem Objekt aufgerufen. Diese sind im folgenden näher beschrieben.

Strukturoperationen

Die Active Node Technology inkludiert implizit eine Naming- und Directoryservice, da jedes ANT Objekt eine lokal eindeutige Adresse zugewiesen erhält. ANT bietet nun alle Möglichkeiten der Erzeugung, Modifikation und Auflösung von Strukturen.

Die Erzeugung von ANT Objekten erfolgt in zwei Schritten. Wie bereits oben erwähnt wird zuerst das Zugriffsobjekt generiert. Mit folgendem Code wird z. B. im Verzeichnis `doc` (siehe Abb. 1) ein weiteres Verzeichnis mit dem Namen c erzeugt:

```
DAddress address = new DAddress(“/api/doc/c“);
DAnt ant = new DAnt(address);
ant.mkdir();
```

Die erste Zeile erzeugt ein Adreßobjekt, die zweite Zeile erzeugt das entsprechende Zugriffsobjekt für das neue Verzeichnis, die letzte Zeile schließlich legt ein neues Verzeichnis an. Das Erzeugen eines neuen Dokumentes erfolgt entsprechend mit `ant.create()`.

Das Aufspannen eines neuen Unteradreßraumes wird mit dem mount Befehl durchgeführt. Durch folgende Befehle wird das ANT Objekt any (siehe Abb. 1) den Inhalt eines Hyperwave Servers (siehe auch [2])verfügbar machen:

```
DUrl url = new DUrl(“hwtp://www.iicm.edu/”);
any.mount(url);
```

Weitere Strukturoperationen die von ANT zur Verfügung gestellt werden umfassen das Löschen von Objekten inklusive rekursivem Löschen und die einfache Unterstützung von "drag and drop" durch die Operationen `move`, `copy` und `symlink` (erzeugt einen symbolischen Link) und entsprechender Abfragen wie `isMoveableTo` oder `isCopyableTo`. Die Interpretation eines symbolischen Links ist dabei von dem zugrunde liegenden Protokoll abhängig und kann z. B. eine Verknüpfung im Dateisystem oder ein wirklicher Link auf einem Hyperwave Server sein. Jedes ANT Objekt hat einen innerhalb seines Verzeichnisses eindeutigen Namen der mit `getName` abgefragt und `rename` modifiziert werden kann. Operationen wie `list`, `getChild`, `getParent` oder `getAddress` erlauben die einfache Navigation durch und die Darstellung der ANT Struktur.

Entsprechend dem Listener Konzept von Java (seit Version 1.1) generieren Strukturoperationen entsprechende Events, die von registrierten Listenern entweder beieinsprucht werden können (vetoable events) oder lediglich

entsprechend behandelt werden können. So kann z. B. ein ANT Explorer immer den jeweils aktuellen Baum darstellen, ohne der Notwendigkeit des Pollings (wie z. B. auch der Explorer und Windows).

Properties

Alle ANT Objekte unterstützen per Definition Properties, einfache Schlüssel-Werte Paare. Der Schlüssel ist dabei immer ein String Objekt. Die Modifikation von Properties kann wie die Strukturoperationen von entsprechenden Listenern wahrgenommen oder auch beeinsprucht werden.

Properties können vielfältig benutzt werden. Sie beinhalten Metainformationen zu Dokumenten wie beispielsweise den Dokumenttyp, das Datum der letzten Änderung oder z. B. den Autor. Werden dabei die Schlüssel entsprechend den Konventionen des Dublin Core [7] verwendet, können plattform- und protokollübergreifend Dokumente lokalisiert und vergleichbar gemacht werden. Eine weitere Einsatzmöglichkeit von Properties findet sich bei der Administration von Benutzerberechtigungen oder bei der Anwendung im Zusammenhang mit Hyperwave Objekten.

Properties stellen eine sehr flexible und erweiterbare Schnittstelle dar. Durch den normierten Zugriff auf die Properties erleichtert ANT wesentlich das protokoll- und plattformübergreifende Zusammenwirken unterschiedlichster Objekte. Der Zugriff auf Properties wird im wesentlichen durch die folgenden Operationen erfolgen: `getProperty`, `setPropery` und `getProperties`.

Kommunikation

Ein wesentlicher Aspekt im Design verteilter Architekturen stellt die einfache Möglichkeit der Kommunikation zwischen den einzelnen Objekten dar. Asynchrone als auch Synchrone Kommunikation wurden bereits im Design der Active Node Technology vorgesehen.

Die Operationen `syncSend` sowie `asyncSend` erlauben auf einfachste Weise die Kommunikation durch Senden und Empfangen von Nachrichten. Sowohl synchrones als auch asynchronos Messaging kann vollkommen transparent auch über Servergrenzen hinweg - d.h. unterschiedliche Java Virtual Machines - durchgeführt werden. Empfangene Nachrichten werden dabei vom entsprechenden ANT Objekt verarbeitet oder auch an ein interessiertes Datenobjekt weitervermittelt. Das Interesse an bestimmten Informationen wird dabei durch Implementierung eines Interfaces deklariert.

Die zweite Möglichkeit der Kommunikation erfolgt durch den typsicheren Aufruf von Methoden. Die von einem ANT Objekt unterstützten Methode können in einem ersten Schritt von diesen angefordert werden (`getMethods` oder `getMethod`), in einem zweiten Schritt erfolgt dann ein synchroner oder asynchroner Methodenaufruf. Die Methoden werden dabei nicht durch Introspection erzeugt, was vielfältigen Laufzeitfehlern vorbeugt und damit ein stabileres und definiertes Verhalten erzwingt.

Services

Eine der interessantesten Komponenten der Active Node Technology stellen die Services dar. Jedes ANT Objekt definiert eigene Services, spezielle - meist selbstvisualisierende - Zugriffsobjekte.

Das Verwenden von Services erfolgt wiederum in zwei Schritten. Zuerste wird vom ANT Objekt ein oder alle ServiceInfo Objekte angefordert. Diese beinhalten eine textuelle Beschreibung des Services, die dem Benutzer dargestellt werden kann. Im zweiten Schritt wird dann eine Visualisierung des Services vom ServiceInfo Objekt angefordert (`getComponent`) und entsprechend dargestellt.

Ein wichtiges Service stellt der Property Editor dar, den jedes ANT Objekt zur Verfügung stellen sollte. Damit können entsprechend Metadaten oder Benutzerberechtigungen administriert werden. Weitere Anwendungen von Services wären die Administration von Servern und Datenbanken oder auch die Implementierung einer großen aber flexiblen Applikation als Servicebündel (siehe auch 5.).

Die Services sind ein Schritt vorwärts von einem Client-Server Modell zu einem verteilten, objektorientierten Modell, wo jedes Objekt je nach Kommunikationssituation sowohl Client als auch Server sein kann. Dies erlaubt ein vollkommen neues Design von Diensten, weg von monolithischen Servern hin zu verteilten und miteinander

kommunizierenden Service Providern. Als logische Weiterentwicklung von ANT werden u.a.im Kapitel 5 Implementierung und Einsatz von Agentsystemen diskutiert.

Datenmodell

Ein weiteres Designprinzip der Active Node Technology stellt die strikte Trennung von MIME-Type spezifischen Daten und protokollspezifischen Strukturen dar (siehe Abb. 2).

Die bisher beschriebenen Komponenten von ANT erlauben einen transparenten Zugriff auf Objekte, unabhängig von dem dabei verwendeteten Protokoll und unabhängig vom physischen Ort des Objektes. Diese Transparenz ist aber nur dann vollständig, wenn „Daten“ auf eine standardisierte Art und Weise dargestellt werden, und zwar als Kombination von MIME-Type und Datenobjekt.

Dem trägt das bei der Active Node Technology verwendet Konzept voll Rechnung. Es wird dabei das JavaBeans Activation Framework (JAF) von Sun Microsystems für das Handling der Daten benutzt. Hierbei können ähnlich den Services abhängig nur vom MIME Type entsprechende Viewer, Editoren, etc. für die Daten angefordert werden. Für eine ausführlichere Diskussion des JAF sei auf [5] verwiesen.

Die linke Seite in Abbildung 2 stellt den Strukturteil von ANT dar. Dieser Teil konstituiert sich aus den hierarchisch angeordneten ANT Objekten, die damit auch den virtuellen Adreßraum formen. Er ist zuständig für die oben beschriebenen Strukturoperationen genauso wie für das transparente Implementieren von Protokollen. Daten bzw. Dokumente werden von speziellen Datenobjekten (`DataHandler`) verwaltet, welche über die Methoden `getData` und `setData` gelesen bzw. geschrieben werden können (rechte Seite in Abb. 2).

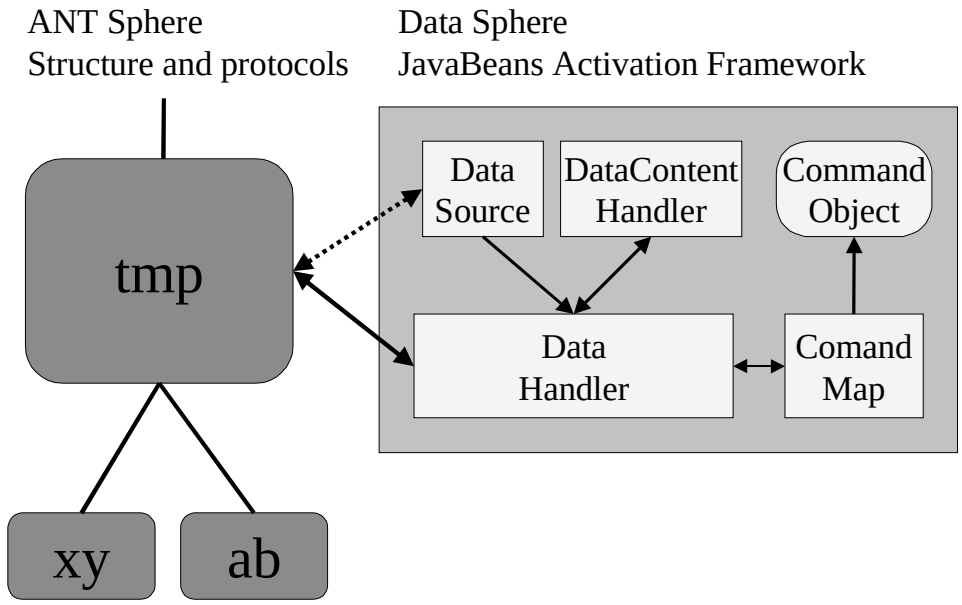

Abb. 2: Rigorose Trennung zwischen protokollspezifischer und datenspezifischer Struktur.

Die strikte Trennung von Daten und Protokollen erlaubt ein einfaches Verschieben und Kopieren von Objekten. So können multimediale Dokumente durch einfaches "drag and drop" vom Verzeichnissystem auf einen WWW Server kopiert und damit publiziert werden. Durch die Verwendung des JAF wird die zukünftige Kompatibilität sowie die Partizipation an der Entwicklung von Dokumentviewern und Editoren gewährleistet.

4. Eine Gegenüberstellung: ANT – CORBA – RMI

Die folgende Tabelle gibt eine kurzen Überblick über Unterschiede zwischen CORBA, RMI und ANT bezüglich einiger ausgewählter Aspekte.

Wesentliche Vorteile von ANT gegenüber anderen Technolgien ergeben sich durch die Integration eines Naming- und Directoryservices aber vor allem durch die einfache und dynamische Erweiterbarkeit des Systems. So können neue Services ebenso wie neue Protokolle zur Laufzeit hinzugefügt werden. Das erlaubt eine investitionserhaltende und kostengünstige Integration bestehender Protokolle und Dienste.

CORBA	RMI	ANT
Eine Protokolldefinition	Eine Vereinbarung für *"Remote Method Invocation"* im Netzwerk	Eine Protokoll- und Strukturintegration
Definiert Request von Objekten in Netzwerken, definiert aber keine Struktur	Definiert ausschließlich "Method calls" zwischen 2 Partnern, definiert aber keine Struktur	Definiert Object Request, RMI, Struktur und Strukturoperationen protokollunabhängig im Netzwerk
Definiert Stubs und Skeletons für Kommunikation	Stubs und Skeletons implizit	Stubs und Skeletons implizit
Unterstützt nicht das Senden von komplexen Objekten im Netz	Senden von komplexen Objekten durch Serialization	Senden von komplexen Objekten durch Serialization
Statisch generierte Language Mappings durch IDL Compiler	Keine statischen Mappings notwendig (Java)	Keine statischen Mappings notwendig (Java)
Service Provider müssen sich bei ORB "anmelden"	Services müssen wie bei CORBA bekanntgegeben werden ("anmelden")	Keine Anmeldung von Services, diese werden dynamisch gefunden
Keine Objektversionskontrolle	Objektversionskontrolle implizit	Objektversionskontrolle implizit

Tab. 1: Gegenüberstellung von Corba, RMI und ANT.

5. Die ANT im praktischen Einsatz - Erste Erfahrungen

Explorer – Das Basiswerkzeug

Aufgrund des Designs von ANT, hierarchische Struktur sowie Objektzugriff über Services und Commands (= Services der Daten), folgt als logisches Administrationswerkzeug die Verwendung eines Explorers. Im folgenden wird beschrieben, wie die einzelnen Komponenten von ANT durch den Explorer dem Benutzer verfügbar gemacht werden.

Die Hauptaufgabe des Explorers stellt die Visualisierung des virtuellen Adreßraumes und die Navigation durch denselben. Die Strukturoperationen wie move/copy/link werden wie gewohnt durch einfache Drag and Drop Operationen ermöglicht. Auch das Clipboard kann effektiv zum Einsatz gebracht werden um Objekte mit cut/copy/paste zu bearbeiten.

Weiters stellt der Explorer sowohl ein Menü als auch ein entsprechendes Kontextmenü zur Verfügung. Sowohl Menü als auch Konntextmenü wechseln dynamisch, je nach selektiertem Objekt, ihre Zusammensetzung. Der Explorer erlaubt nicht nur das Löschen, Umbenennen sowie Erzeugen von Objekten sondern auch das Mounten neuer Subräume. Dadurch kann ein persönlicher Desktop eingerichtet werden, wo die meist benutzten Datenquellen wie Dateisystem, diverse Webserver und Netzwerkrechner ebenso wie spezifische Dienste in den Preferences gespeichert und damit bereits beim Start des Explorers verfügbar sind.

Am interessantesten ist aber die Möglichkeit, die von einem ANT Objekt angebotenen Services und Commands durch einfachste Interaktionen zu benutzen. Services und Commands erscheinen als Liste (eventuell sogar strukturiert) im Kontextmenü sowie im Menü. Durch Klicken auf den entsprechende Menüpunkt wird das Service oder Command visualisiert und damit für den Benutzer verfügbar. Solche Services oder Commands können ein Property Editor, ein Document Viewer oder Editor oder auch spezielle Services wie Workflow Management Services oder ein Shopping-Agent sein.

Die folgende Abbildung 3 zeigt einen Explorer (als Applet) innerhalb eines Web Browsers der die Navigation auf einem Hyperwave Information Server unterstützt. Durch einen einfachen Doppelklick auf ein Objekt wird dieses im rechten Fenster (einem HTML Frame) dargestellt. Weiters kann mit dem Explorer die

Serveradministration durchgeführt werden sowie einzelne oder mehrere Dokumente von und zum Server verschoben werden.

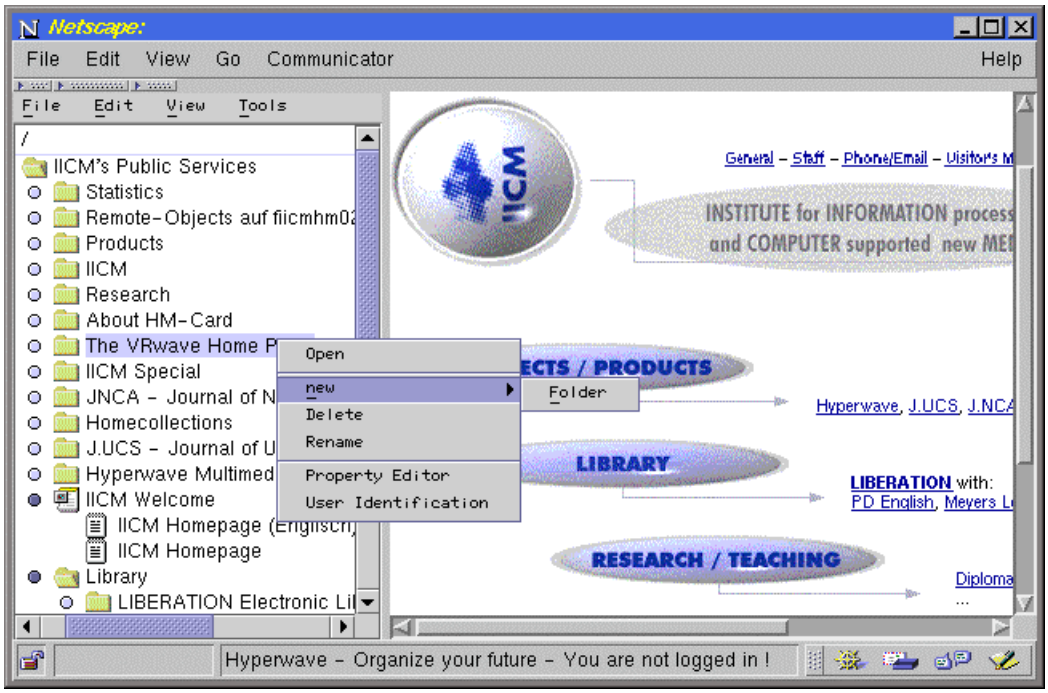

Abb. 3: ANT Explorer mit geöffnetem Kontextmenü als Applet in einem Web Browser

MTZ – Medizin Telematik Zentrum

Als ausschlaggebend und richtungsweisend für die Gestaltung der ANT Technologie kann nachfolgend kurz beschriebenes medizin-technisches Projekt angesehen werden. Die Rahmenbedingungen für diese Anwendung waren gegeben durch die Notwendigkeit der Integration von verteilten, unterschiedlichen, medizinischen Systemen, die praktisch als Insellösungen designt im implementiert worden sind. Weiters sollten verteilte Patientenakten virtuell zentral zugänglich gemacht werden, wobei hier vor allem die Aspekte der Sicherheit und Vertraulichkeit der Informationen von enormer Sensibiltät sind. Unter Patientenakten sind nicht nur medizinische Daten (inklusive Real-Time Daten wie EKGs) zu verstehen, sondern auch der gesamte Bereich der Administration.

Durch das modulare und verteilte Konzept der Active Node Technology ist es leicht möglich, einzelne Dienste und Services zu integrieren. Diese Integration erfolgt durch einfaches *Mounten* des Services, wie es oben bereits beschrieben wurde. Es genügt eine einzelne Java Klasse zu implementieren, die als Gateway zwischen dem (eventuell bereits bestehenden) Subsystem und dem MTZ dient. Sicherheit und Vertraulichkeit wird durch entsprechende, serverseitige Module wie dynamisches Usermanagement und Zugriffsprotokollierung transparent in das System integriert und steht damit allen Services zur Verfügung. Als Basisservice für die Verwaltung der Patientenakten wurde ein Hyperwave Information Server mit transparenter Integration von Real Time Datenbanken eingesetzt.

Chronos – Controlling Tool

Ein weiteres Projekt, das die Vorteile von ANT verwendet ist Chronos. Es handelt sich dabei um ein Controlling Tool, mit dem Projektpläne erzeugt (bzw. importiert) werden können, das eine online Zeiterfassung sowie Abweichungserfassung ermöglicht und drittens Projektberichte/Mitarbeiterberichte sowie Abweichungsanalysen liefert.

Chronos verwendet ANT und den ANT Explorer um auf einfache Art und Weise strukturierte Daten zu verwalten (Projekte, Mitarbeiter in diversen Abteilungen) und stellt die verschieden Funktionen in Form von Services zur Verfügung. Die Speicherung der Daten erfolgt unter Verwendung eines ANT Subraumes. Dadurch kann Chronos sowohl im Verzeichnissystem, auf Datenbanken ebenso wie auf Intra/Internet Servern erfolgen.

Die aktuelle Implementierung verwendet einen Hyperwave Information Server als Datenspeicher und benutzt dessen Usermanagement transparent für eine entsprechende Zugriffsverwaltung. Chronos ist damit ein voll netzwerktaugliches Controlling Werkzeug.

Chronos stellt ein Beispiel für die Verwendung von ANT zur Implementierung einer Applikation dar. Chronos als Programm besteht selbst aus strukturierten ANT Objekten, die eigenständig ihren Platz in der Gesamtstruktur managen und dem Benutzer ihre speziellen Services anbieten. Durch dieses Design kann Chronos jederzeit und auf einfach erweitert oder modifiziert werden.

Sounddatenbank

Der Hyperwave Information Server (siehe auch [2]) eignet sich insbesondere für die Verwaltung von großen, unstrukturierten Datenmengen. Götzinger [9] zeigt in seiner Arbeit die Möglichkeit einer Implementierung einer Audiodatenbank [10] unter Verwendung eines Hyperwave Servers auf. Diese Anwendung verwendet vordefinierte Metadatensets zur Beschreibung der entsprechenden Audiodaten, ein zusätzliches Textobjekt zur Bereitstellung des Inhaltes und zur Volltextsuche, sowie Audiodaten in verschiedenen Formaten. Zum komfortablen Editieren bestehender Audio-Datensets und zum Einfügen neuer eignet sich die gebotene ANT Funktionalität. Durch einfache geringfügige Erweiterung von Services des oben vorgestellten Explorers kann ohne großen Aufwand die zusätzlich benötigte Funktionalität erreicht werden. Der vorhandene Property Editor wurde dabei um die zusätzlich verwendeten Metadatensets erweitert. Der gemeinsame Upload von Soundfiles verschiedener Formate wird ebenfalls als ein zusätzliches Application Service möglich. Ein spezieller Hyperwave Mounter (Protokoll hwtp) stellt die notwendige Verbindung zur Serverapplikation dar. Diese Client-Applikation stellt ein Beispiel für den Einsatz der Servicekonzeption dar (siehe auch 3.).

6. Konzepte geplanter Einsatzgebiete

Die gewonnen Erfahrungen der in Abschnitt 5 gezeigten Applikationen haben gezeigt, daß sich mit der ANT Technologie in weiten Bereichen Anwendungsmöglichkeiten erschließen. Nachfolgende Konzepte sollen geplante Anwendungen, die im Rahmen der Forschungstätigkeit am IICM umgesetzt werden, kurz erläutern.

Agents
In [12] findet man u.a. folgende Beschreibungsmerkmale über Agents: diese können statisch oder mobil sein, diese können für sich autonom sein oder miteinander interagieren, diese können verwendet werden als Interface Agents, als Information Agents, etc. Betrachtet man aus diesem Gesichtspunkt die in Abschnitt 2 und 3 dargestellte Funktionsweise der ANTs, so läßt sich der virtuelle Adreßraum nutzen, um virtuelle Agent-Räume zu erstellen. Durch die Unterstützung der Serialisation kann es den Agents erlaubt werden, sich in den virtuellen Räumen zu bewegen und aufzuhalten. Das beschriebene Kommunikations- und Listenerkonzept erlaubt es, eine stabile Interaktion der Agents untereinander zu ermöglichen. Aus diesen Überlegungen entstand am IICM ein Konzept zur Schaffung einer Agent-Middleware, die auf der Technologie der ANTs aufbaut. Unter Berücksichtigung der von der FIPA [13] empfohlenen Standards, soll eine einfach verwendbare und erweiterbare Basistechnologie vor allem für den Forschungsbereich zur Verfügung gestellt werden.

Wissensauffindung und -verarbeitung
In unserer Informationsgesellschaft werden rasch verfügbare, präzise und relevante Information in allen Bereichen der Gesellschaft immer wichtiger. Dabei ist die Informatik gefordert, Wissen (= nachgefragte, relevante Information) zur Verfügung zu stellen [11]. Den größten Wissensspeicher der Welt stellt wohl das Internet dar, welches durch rasche Wachstumsraten gekennzeichnet ist. Um ein möglichst aktuelles und umfangreiches Abbild dieser Informationen zu erreichen, ist es sinnvoll, ein verteiltes Suchsystem [11] aufzubauen. Im idealsten Falle wird zu jedem lokalen Informationssystem, z.B. Web Server, ein ANT basierendes Agentsystem angeschlossen, die Informationen über entsprechende Mounter mit dem ANT-Agent-System verbunden und die Interaktion erfolgt über die Dino-Dino-Protokollebene. Spezielle Agents könnten nun nahe den jeweiligen Informationsquellen Wissen über Informationen aufbauen. Durch Schaffung einer hierarchischen Struktur können mehrere Informationsquellen zu einem Informationspool zusammengefaßt werden, und diese können wiederum zu weiteren Informationsclustern verknüpft werden. Diese Hierarchien sind sowohl geographisch als auch inhaltlich aufzubauen.

Im einfachsten Falle können zu bestimmten Themen Virtuelle Informationsräume aufgebaut werden, wo der Benutzer durch einfache Navigation die nachgefragten Informationen auffindet. Eine komplexere und dynamischere Variante ist durch intelligente Suchagents implementierbar, denen Suchanfragen gestellt werden. Die Agents können mittels des ANT Systems miteinander kommunizieren oder bei Bedarf ihren Aufenthaltsort wechseln. Entsprechend gestaltete PDA (personal digital agents) können die Usergewohnheiten beobachten und daraus lernen und so bevorzugte virtuelle Agenträume aufsuchen und den Benutzer bei der Suche unterstützen.

7. Danksagung

Wir danken allen Mitarbeitern am IICM, insbesondere der Dino Gruppe, die wesentliche und wertvolle Vorarbeit geleistet haben. Weiters danken wir den Projektpartners des MTZ Projektes für das konstruktive Feedback.

8. Literaturverzeichnis

[1] Gosling et.al 96.
James Gosling: "The Java(TM) Language Specificaton",
The Java Series, Addison-Wesley, (1996) #

[2] Maurer 96
Maurer H. ed.: "HyperWave - The Next Generation Web Solution",
Addison-Wesley, (1996)

[3] OMG.
OMG.: "Corba Specificaton",
http://www.omg.com , (1998)

[4] Schmaranz 96.
Schmaranz K.: "Professional Electronic Publishing in Hyper-G - The Next Generation Publishing Solution on the Web",
Proceedings WebNet 96, San Francisco (1996)

[5] Sun Microsystems.
Bart Calder, Bill Shannon.: "JavaBeans Activation Framework Specification",
http://www.sun.com , (1997)

[6] Sun Microsystems.
Sun Microsystems: "Remote Method Invokation Specification",
http://java.sun.com , (1997)

[7] Dublin Core
Dublin Core Metadata
http://purl.oclc.org/metadata/dublin_core (1997)

[8] Healtheon
http://www.healtheon.com

[9] Götzinger 98
"Testimplementation auf dem Hyperwave Information Server", Diplomarbeit. IICM: TU-Graz; Graz (1998) und http://www2.iicm.edu/cguetl/education/thesis/wgoetz98

[10] Hyperwave Audio Database
http://www2.iicm.edu/demo/hyperwave/sounddb

[11] Gütl et.al 98
"Future Information Harvesting and Processing on the Web", European Telematics. Advancing the Information Society; Barcelona (1998) and http://www.iicm.edu/cguetl/papers/fihap

[12] Jennings 98
"Agent Technology: Fountations, Applications, and Markets", Springer Verlag, Heidelberg; S. 29 ff (1998)

[13] FIPA - Foundation for Intelligent Physical Agents
http://drogo.cselt.stet.it/fipa