GIS hilft Touristen bei der Navigation – ein erster Prototyp des mobilen *Deep Map* Systems für das Heidelberger Schloß

Alexander ZIPF, Vasu CHANDRASEKHARA, Jochen HÄUSSLER und Rainer MALAKA (Heidelberg)[1]

1. Einleitung – *Deep Map* wird mobil

Im Rahmen des von der Klaus TSCHIRA-Stiftung geförderten Projekts *Deep Map* des *European Media Laboratory* (*EML*) wird auf Basis eines geographischen Informationssystems (GIS) ein elektronischer Tour-Guide entwickelt. Der Projektteil *Deep Map*/GIS wird dabei in einer Kooperation des *EML* mit dem Geographischen Institut der Universität Heidelberg durchgeführt (MEUSBURGER / ZIPF 1998). In einer Reihe weiterer Projekte, die zum Beispiel intelligente Bedienschnittstellen, die Integration verschiedenster Informationsquellen und die drahtlose Vernetzung zum Ziel haben, wird nun am *EML* und an einer Reihe kooperierender Universitäten und Forschungseinrichtungen das ambitionierte Ziel verfolgt, *Deep Map* als intelligentes mobiles Informationssystem zu realisieren, das nicht nur weiß, wo sein Benutzer ist, sondern auch mit ihm in verschiedenen Sprachen reden kann (MALAKA / ZIPF 1998).

Deep Map soll es dem Benutzer sowohl erleichtern, sich in einer ihm fremden Stadt zu orientieren, zurechtzufinden und zu bewegen, als auch durch virtuelle Datenwelten in Raum und Zeit zu reisen. In einigen dieser Forschungsarbeiten werden sehr harte informatische Probleme angegangen, die kaum in kurzer Zeit zu lösen sein werden, trotzdem sollen immer wieder Prototypen den aktuellen Stand der Forschung in einem Gesamtsystem demonstrieren.

Ein solcher erster mobiler „Tour Guide“ mit zusätzlicher Sprachsteuerung für die tatsächliche Navigation in der realen Stadt wurde im Sommer 1999 auf einer internationalen Präsentation der Presse vorgestellt (STUTTGARTER ZEITUNG 1999). In einer Live-Übertragung konnten die eingeladenen Journalisten und Gäste an diesem Tag beobachten, wie ein amerikanischer Tourist sich von dem Prototypen durch das Heidelberger Schloß führen ließ.

Im folgenden werden die Komponenten dieses ersten Prototypen vorgestellt. Dabei sollen insbesondere die Kernbereiche des Gesamtsystems, die die geographischen Informationen verwalten, und deren Einbindung in das Gesamtsystem beschrieben werden.

2. Deep Map GIS

Der Kern des gesamten *Deep Map* Systems ist ein geographisches Informationssystem (GIS) in Kombination mit einer Datenbank. Zusammen sind beide die Grundlage für das Touristeninformationssystem für Heidelberg. Die Datenbank und die GIS-Funktionalitäten wurden in der Kooperation mit dem Geographischen Institut der Universität Heidelberg realisiert. Diplomanden und Doktoranden des Lehrstuhls Prof. MEUSBURGER wirken bei der Konzeption und

[1] Die Autoren danken der Klaus TSCHIRA-Stiftung für die Förderung der Projekte, der Firma ESRI++ Geoinformatik GmbH für die Bereitstellung von Software, der Firma Xybernaut für die Bereitstellung mobiler Hardware und dem Vermessungsamt der Stadt Heidelberg für die Überlassung von ALK-Daten. Teile der hier vorgestellten Arbeiten wurden im Rahmen der BMBF Leitprojekte *SmartKom* und *EMBASSI* gefördert (Fördernummern 01IL904D2 und 01IL905C).

Implementierung der für das Funktionieren des Prototypen erforderlichen Funktionalität mit. Als Plattform für das GIS werden *ESRI ArcView* und in den neueren Varianten die *SDE* für *Oracle* verwendet. Die geographischen Daten wurden von der Stadt Heidelberg zur Verfügung gestellt. Die inhaltlichen Daten zu Geschichte, touristischen Sehenswürdigkeiten und Personen wurden innerhalb des Projekts auch in Zusammenarbeit mit dem Geographischen Institut recherchiert. Die inhaltliche und digitale Aufbereitung stadtgeographischer und stadtgeschichtlicher Informationen als Texte und Bilder sollen auch das wissenschaftliche Wissen über die jeweiligen Sachverhalte korrekt wiedergeben (vgl. auch den Bericht von R. WEINMANN in diesem HGG-Journal). Damit ist die Datenbank auch für Historiker und Fachleute der Historischen Geographie ein interesantes Medium, das erstmals diese Informationen digital aufbereitet.

3. *Wearable* GIS

Die Entwicklung mobiler GI-Systeme ist die konsequente Fortführung des von STROBL (1997) im Rahmen der AGIT skizzierten Entwicklungspfads geographischer Informationssysteme (GIS) von Spezialrechnern über *Workstations*, PCs und dem *Web* hin zu bald breiter verfügbaren *wearable* Computern. Dieser Trend zum mobilen GIS für Fußgänger findet sich in verschiedenen Visionen wieder (COORS / JASNOCH 1999a). Für das hier vorgestellte mobile *Deep Map* System diente ein „mobile assistant IV“ der Firma Xybernaut als Experimentierplattform. In Abb. 1 sieht man das verwendete Testsystem. Bei diesem Gerät handelt es sich um einen sogenannten „wearable“ Computer, die am Körper getragen werden, so daß beide Hände frei bleiben. Die Rechner selbst werden mit einem Gürtel umgeschnallt und reichen an die Leistungsfähigkeit handelsüblicher PCs heran. Wie oben diskutiert, können die *Deep Map* Agenten auch in diesem Szenario über Rechnernetze verteilt sein und über drahtlose Kommunikationsmechanismen Informationen austauschen.

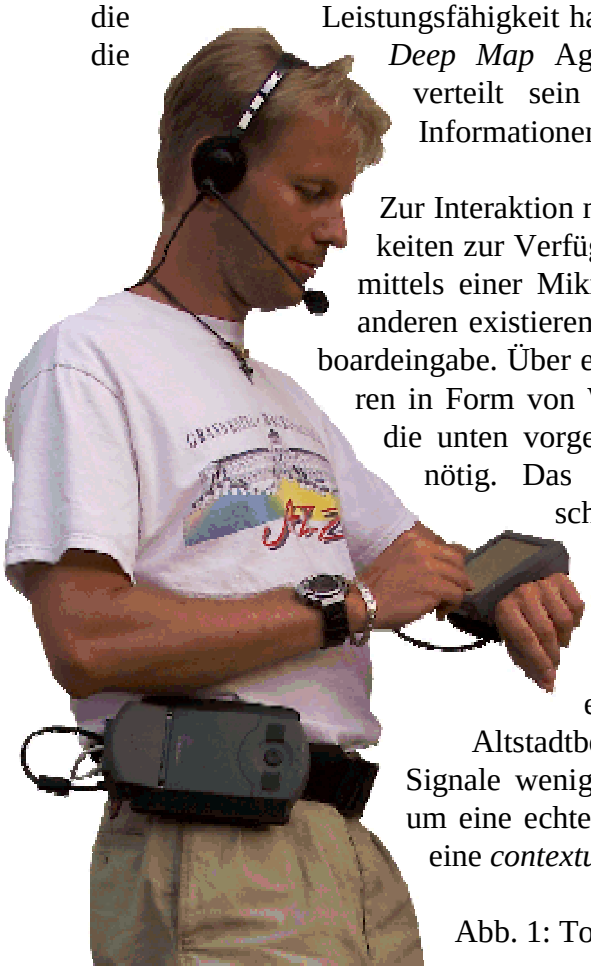

Abb. 1: Tourist mit mobilem *Deep Map* Prototypen

Zur Interaktion mit dem mobilen System stehen verschiedene Möglichkeiten zur Verfügung: Zum einen kann das Gerät über Sprachsteuerung mittels einer Mikrophon/Kopfhörer-Kombination bedient werden. Zum anderen existieren verschiedene Varianten für Displays und Maus-/Keyboardeingabe. Über eine Sprachsteuerung könnte man die berechneten Touren in Form von Weganweisungen ausgeben (PORZEL et al. 1999). Für die unten vorgestellten Visualisierungsmöglichkeiten ist ein Display nötig. Das farbige LCD-Display ermöglicht akzeptable Bildschirmauflösungen. Eine weitere Voraussetzung für den mobilen Einsatz eines solchen Systems neben der Vernetzung und den Bedienschnittstellen sind Komponenten zur Lokalisation des Touristen. Dazu können Satellitennavigationssysteme über GPS eingesetzt werden. Jedoch sind diese gerade im Altstadtbereich wegen Abschattungen und Reflexionen der Signale wenig zuverlässig. Hier sind weitere Entwicklungen nötig, um eine echte *position awareness* als eine der Voraussetzungen für eine *contextual awareness* (DIAS et. al. 1998) zu realisieren.

4. Das Gesamtsystem im Überblick

In diesem Bericht wird die Grobkonzeption des tragbaren *Deep Map* Systems vorgestellt und vor allem auf die damit verbundenen wissenschaftlichen Fragestellungen aus Sicht der Geoinformatik eingegangen werden.

Der Kern des *Deep Map* Gesamtsystems besteht aus einer multimedialen historischen Datenbank und einem geographischen Informationssystem (MEUSBURGER / ZIPF 1998). Um diesen Kern stellen eine Reihe von Komponenten weitere Funktionalitäten zur Verfügung. Beispielsweise werden in Zusammenarbeit[2] mit anderen Institutionen und Universitäten weitere Subsysteme entwickelt, die z. B. zum Sprachverstehen, der Generierung von Sprachausgabe oder zur Anbindung externer Dienste wie Reservierungssysteme für Hotels oder Veranstaltungen dienen (COORS / JASNOCH 1999b). Zu den hier vorgestellten Funktionalitäten und Komponenten von *Deep Map* GIS gehören Tourenplanung, Datenbank (mit Abfrageschnittstelle), Sichtbarkeitsanalysen, räumliche Abfragen, Geodatenverwaltung und Kartenvisualisierung.

Das Gesamtsystem ist komponentenweise aufgebaut, und die Kommunikation zwischen den einzelnen Komponenten wird über eine am *EML* (im Projekt *Integrated Map*; vgl. CHANDRASEKHARA / COORS 1999) entwickelte *JAVA*-basierte Agenten-Plattform (*JAM-Frame – Java-Agent-Management-Framework*) bewerkstelligt. Die einzelnen Komponenten sind als weitgehend unabhängige Agenten realisiert und können so leicht ausgetauscht werden. Besonders bietet dieser Entwurf durch die Verwendung der Agentenkommunikationssprache *ACL* der *FIPA* (FOUNDATION FOR PHYSICAL AGENTS 1997) eine Kommunikation der einzelnen Module auf einer höheren Abstraktionsebene als bisherige Systeme. Dies erleichtert die Möglichkeit, einzelne Komponenten oder Agenten in fremden Systemen zu nutzen sowie *FIPA*-konforme (FOUNDATION FOR PHYSICAL AGENTS 1998) fremde Agenten in *Deep Map* einzusetzen.

In unserem Fall werden verschiedene Dienste eines GIS als sogenannte Software-Agenten gekapselt und somit im Gegensatz zur starren Architektur früherer Systeme neue Wege für die Entwicklung flexiblerer geographischer Informationssysteme aufgezeigt. Die Komponenten, die nicht selbst in *JAVA* vorliegen, verfügen über *Wrapper*, die die Agentenkommunikation in Funktionsaufrufe der jeweiligen Komponente umsetzt. Diese *JAVA*-Agenten kommunizieren als *Wrapper* über *Remote Procedure Calls* (*RPC*) mit dem in *Deep Map* verwendeten GIS (*ArcView* von *ESRI*). Auf diese Weise können spätere, leistungsfähigere Varianten z. B. des Tourenplanungs-Agenten, die dann z. B. über ein ausgefeilteres Benutzer- oder Kontextmodell oder spezielle Optimierungsalgorithmen verfügen, die aktuellen Versionen ersetzen.

In Abb. 2 wird eine vereinfachte schematische Sicht auf die Architektur der Komponenten des Gesamtsystems gegeben, die nur eine mögliche Repräsentation darstellt und im wesentlichen die Komponenten behandelt, die für das Gesamtsystem unter besonderer Berücksichtigung des GIS von Bedeutung sind. Diese Architektur wurde in zahlreichen Gesprächen zwischen Dr. R. MALAKA, A. ZIPF, C. KRAY, R. PORZEL, V. COORS und V. CHANDRASEKHARA entwickelt und verfeinert.

Dabei wird eine Gliederung in drei Ebenen mit unterschiedlichen Hauptaufgaben dargestellt. Es handelt sich also um eine Unterteilung in Funktionszusammenhänge, so daß bei einzelnen

[2] Die Kooperationen sind auf der WWW-Seite *http://www.eml.villa-bosch.de/research/deepmap/deepmap.html* aufgeführt.

Modulen oder Agenten natürlich je nach Priorisierung oder Ausbau der Module eine objektive exakte Einteilung bestimmter Komponenten in die eine oder andere Ebene nicht wirklich möglich ist. Eine genauere Beschreibung der einzelnen Module, ihrer Funktionsweise und Interaktion folgt in den nächsten Anschnitten, so daß diese vereinfachte Sichtweise zunächst als erster Überblick dienen kann.

In Abb. 2 ist eine funktionale Gliederung der einzelnen Komponenten des Gesamtsystems in drei Schichten zu erkennen. In der Interaktionsschicht finden sich die Komponenten, die der Endbenutzer tatsächlich sieht oder hört, d. h. mit denen er direkt interagieren kann. Dabei handelt es sich im wesentlichen um graphische Benutzerschnittstellen (sowohl für das WWW oder für mobile Systeme als auch für Sprachein- oder -ausgabe) sowie die Möglichkeit, in 3D zu interagieren. In der kognitiven Schicht erfolgt eine Umsetzung der vom Benutzer durch die verschiedenen Interaktionsmöglichkeiten geäußerten Wünsche, Fragen oder Anweisungen auf eine interne, formale Sprache (nämlich *FIPA ACL*), über die sich die einzelnen Komponenten und Agenten austauschen und versuchen, die Fragen und Anforderungen des Benutzers aufzulösen und letztendlich in einen für die Basiskomponenten verständlichen Aufruf (GIS-Funktionsaufruf oder *SQL*-Anfrage) umzusetzen.

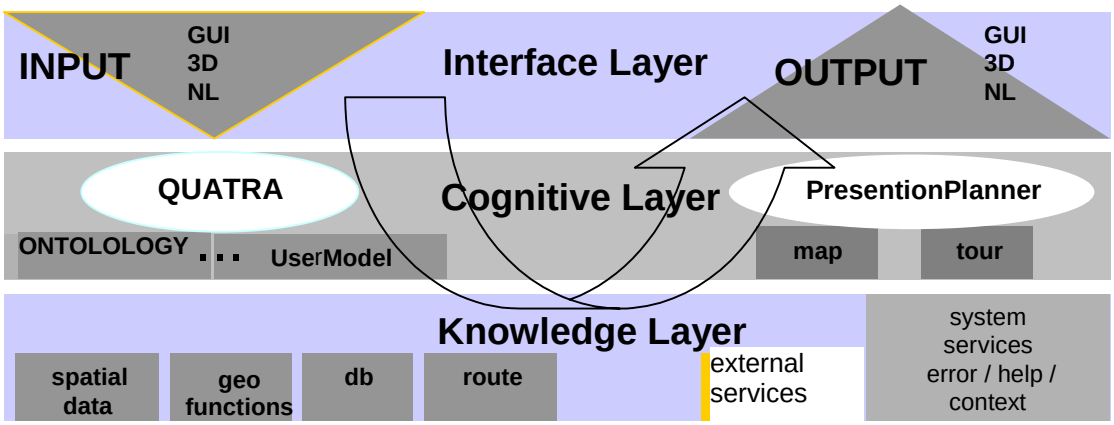

Abb. 2: Ebenensichtweise der Gesamtarchitektur (vereinfacht)

In der Wissens– oder Diensteschicht befinden sich die grundlegenden Funktionalitäten des Systems, während die oberen Schichten vorrangig dazu dienen, die Interaktion des Benutzers mit dem System zu erleichtern und über multimodale Schnittstellen zu ermöglichen. Zu den grundlegenden Diensten gehören die Funktionalitäten des GIS, der Datenbanken oder externer Dienste.

In den folgenden Abschnitten werden die wesentlichen Module der einzelnen Ebenen und ihr Zusammenspiel mit GIS und Datenbank-Funktionalität vorgestellt.

4.1 Interface Layer

In der Interaktionsschicht befinden sich die Komponenten, die mit dem Benutzer des Systems direkt kommunizieren. Es gibt zwei wesentliche Modi, in denen der Benutzer mit dem *Deep Map Tour Guide* interagieren kann. Dies ist eine graphische Bedienschnittstelle (*GUI*) für das *Pre-Trip-Planning* über das WWW (ZIPF / MALAKA 1999a), die also in einem Internet-*Browser* laufen muß, sowie Bedienschnittstellen für mobile Endgeräte bei der Interaktion mit dem mobilen *Deep Map* System vor Ort beim realen Besuch der besichtigten Stadt.

Für das mobile Szenario ist eine Unterstützung der Interaktion durch gesprochene natürliche Sprache wichtig. Hierbei zeigt sich, daß solch eine Raumkognitionskomponente auf ein breites Spektrum von GIS-Funktionen zurückgreifen muß, um eine Disambiguierung natürlichsprachlicher raumbezogener Anfragen und eine sinnvolle Generierung von Anweisungen zu ermöglichen (vgl. KRAY 1999).

Im linken Teil von Abb. 2 ist die Eingabe durch den Benutzer mit der jeweils aktuellen Komponente (also Sprache, *GUI* oder 3D-Schnittstelle) dargestellt. All diese Eingaben werden zunächst an die Kognitionsschicht und dort an das sogennante *QUATRA*-Modul (*QUery and Answer TRAnslator*) weitergeleitet, um eine eventuelle Disambiguierung durchzuführen und vor allem um eine einheitliche Funktionsweise der darunter liegenden Dienste und Datenbanken zu ermöglichen. Außerdem können die Anfragen in *QUATRA* durch die jeweiligen in *QUATRA* angesiedelten Agenten und Komponenten um Wissen aus den Datenbanken oder benutzerspezifische Aspekte über die geplante *User Model* Komponente angereichert werden.

4.2 Cognitive Layer

Wie oben angedeutet, erfolgt in der kognitiven Schicht die Umsetzung der vom Benutzer durch die verschiedenen Interaktionsmöglichkeiten geäußerten Wünsche, Fragen oder Anweisungen auf eine interne formale Sprache (nämlich die *FIPA ACL* [FOUNDATION FOR PHYSICAL AGENTS 1998]), über welche die einzelnen Komponenten und Agenten sich austauschen und versuchen, die Fragen und Anforderungen des Benutzers aufzulösen und letztendlich in einen für die Basiskomponenten verständlichen Aufruf umzusetzen. D. h. in der kognitiven Schicht befinden sich die Dienste zur Umsetzung von Anfragen von Benutzersicht in solche aus Systemsicht. Hierunter sind in unserem Zusammenhang solche Komponenten zu verstehen, die benutzersensitives Verhalten des Systems ermöglichen wie der *User Model Server*, die Dialoghistorie, die Ontologie oder ähnliche Dienste oder Komponenten, die deren Funktionen in Anspruch nehmen, um dem Benutzer einen für ihn, seine Interessen oder den aktuellen Kontext massgeschneiderten Dienst oder eine Antwort anzubieten. Die Module der Sprachverarbeitung werden in PORZEL et. al (1999) dargestellt.

Auch die GIS-Module stellen für weitere Agenten des *Deep Map* Systems Dienste zur Verfügung, um diesen Agenten die notwendigen Schlußfolgerungen zu erlauben. So benötigt das Modul *SpaCE* (*Spacial Cognition Engine*, KRAY 1999) umfangreiche Angaben über Routen und deren Segmente sowie über die Objekte in der Nähe solcher Routen, um bei der Generierung natürlichsprachlicher Weganweisungen eine vernünftige vorsprachliche Botschaft an die eigentlichen Generierer zu senden. Daher sind für den mobilen *Deep Map* Prototypen, wie später erläutert, mehrere GIS Funktionalitäten implementiert, die über räumliche Suchen, Netzwerkanalysen, Datenbankabfragen bis hin zu Sichtbarkeitsanalysen und der Übergabe von speziell selektierten Geometrien aus dem GIS an anfragende Module reichen.

4.3 Service Layer

In der Wissens– oder Diensteschicht befinden sich die grundlegenden Funktionalitäten des Systems, die die eigentlichen Anfragen des Benutzers befriedigen, während die oberen Schichten im wesentlichen dazu dienen, die Interaktion des Benutzers mit dem System zu erleichtern und über multimodale Schnittstellen zu ermöglichen. Zu diesen grundlegenden Diensten gehören die Funktionalitäten des GIS, der Datenbanken oder externer Dienste, die in das System auf dieser Ebene einbezogen werden, um auch für die Interaktion mit solchen ex-

ternen Diensten den gesamten Überbau zur Verfügung zu haben. Der Benutzer muß gar nicht merken, daß er sich jetzt mit einem Dienst unterhält bzw. dessen Dienste beansprucht, der ursprünglich nicht exklusiv für *Deep Map* entwickelt wurde. Hierzu zählen z. B: *Online*-Hotelreservierungsdienste, die in einem weiteren Projekt (*Integrated Map*) integriert werden.

Die wesentlichen GIS-Module werden im folgenden kurz skizziert.

4.3.1 Datenbank

Die historische Datenbank verwaltet die Einträge zu räumlichen Objekten (Gebäuden, Kirchen, Plätzen, Straßen etc.) und den damit über verschiedenste (auch temporalen) Beziehungen verknüpften Personen sowie den mit diesen Personen und Orten verbundenen Ereignissen (die auch abstrakte Events darstellen können) und stellt über Abfragen unterschiedliche Sichten auf diese Informationen dar. Es wurde in der Datenbank bewußt eine relativ grobe Einteilung dieser Hauptentitäten (wie räumliche Objekte, Personen, *Events* und multimediale Dokumente) getroffen und diese nur bedingt weiter in Tabellen untergliedert, da feinere Unterteilungen je nach Änderung der Anforderungen an die Datenbank relativ dynamisch sein können und daher statt über eine Schemaänderung in der Datenbank mittels *XML* in einem großen *XML*-Textfeld geschehen. Es hat sich gezeigt, daß die Abbildung von stadtgeschichtlichen, kunsthistorischen oder geographischen Ereignissen und Prozessen auf einem relationalen Datenbankschema relativ schwierig zu realisieren ist, da die meisten Inhalte untereinander verknüpft sein können und oft die subjektive Einschätzung des Modellierers und sein Verständnis der modelliertenWelt in die Modelle eingehen. Die gesamte Tabellenzahl der Datenbank ist mit über 120 dennoch relativ hoch, da noch eine komplette Verwaltung für die die thematischen Phänomene, Orte, Ereignisse und Personen beschreibende Literatur, eine Literaturdatenbank, integriert ist, die auch noch zahlreiche Datentypen wie Bilder (von Karten, Photos bis Gemälde), Videos, Musik und VR-Modelle unterstützt. Fast alle Entitäten sind miteinander verknüpft, und es werden zahlreiche Meta- und Verwaltungsinformationen (wie Bearbeiterhistorie, Bearbeitungsstand des Dokuments, Benutzerrechte, Aufnahmezeitpunkt, Schlagworte etc.) mitverwaltet. Die gesamte Eingabe steht über diverse Menü-Oberflächen und Pflegetools den entsprechenden Fachwissenschaftlern (Geographen, Kunsthistorikern, Historikern) zur Verfügung (HÄUSSLER 1999).

Die Wahl von *XML* für zukünftige weitere Verfeinerungen der Entitäten ist darin begründet, daß das Ändern der *DTD* (*Dokument Type Definition*) eines *XML*-Dokuments einfacher ist als die Änderung eines Datenbankschemas, besonders weil immer mehr Werkzeuge zur Verfügung stehen, die das Verarbeiten (z. B. Parsen) von *XML*-Dokumenten automatisiert erlauben. Außerdem sind *XML*-Dokumente relativ einfach im *Web* zu präsentieren. Über *Style Sheets* kann das Aussehen des jeweiligen Dokuments für die entsprechende Plattform (*Web*, *PDA* etc.) gesteuert werden.

4.3.2 Geodatenserver

Ein Geodatenserver ermöglicht die konsistente und performante Verwaltung von Geodaten in Datenbanken und stellt dieser Geodaten über standardisierte Schnittstellen zur Verfügung (im Rahmen von *Deep Map* wird derzeit auch an einer Implementierung der *OpenGIS-Simple Feature* Spezifikation für *CORBA* gearbeitet, damit Geometrieelemente direkt über *CORBA*, d. h. auch mit eigenen in *Java* realisierten Klienten möglich ist und nicht nur über die von den kommerziellen Herstellern unterstützte *SQL-API* bzw. eine *C-SPI*, wie diese derzeit bei der

Spatial Data Engine (*SDE*) von *ESRI* üblich ist). Derzeit sind räumliche Suchen und Abfragen wie die nach allen Objekten eines bestimmten Typs (z. B. Kirchen) in einem bestimmten Gebiet über eine Schnittstelle zu *ArcView* realisiert, jedoch wird in der nächsten Version des *Deep Map* Systems die komplette Geometrieverwaltung über den Geodatenserver erfolgen.

4.3.3 Tourenplanung und –vorschläge

Grundlegende Tourenplanungs- und Wegverfolgungfunktionalität wird schon in dieser Ebene zur Verfügung gestellt, höherwertige Dienste wie personalisierte Tourenvorschläge finden sich in der kognitiven Schicht des Systems und wurden z. B. auf der AGIT vorgestellt (ZIPF / MALAKA 1999a; 1999b).

4.3.4 Kartenvisualisierung

Ebenfalls einen serverseitigen Basisdienst stellt sowohl der *Mapserver* als auch der für den mobilen Prototypen realisierten *MapAgent* dar, der eine *RenderEngine* beinhaltet, die Karten als Rastergraphiken aus Vektorinformationen serverseitig erzeugt. Daneben besteht aber auch die Möglichkeit der Übergabe der vektoriellen Geometrierohdaten an den Klienten und die dortige Darstellung derselben als Karte (*client*-seitiges *Rendern*). Letztere Variante wurde in *Deep Map* für die in *Java* entwickelte Übersichtskarte realisiert und gehört auch zur Funktionalität des *MapAgent* des mobilen *Deep Map* Systems.

4.3.5 VR-Verwaltung und -Visualisierung

In der historisch-geographischen *Deep Map* Datenbank können zudem VR-Modelle für einzelne Gebäude mit all den *Features* verwaltet werden, d. h.: verschlagwortet, verknüpft mit Personen, Ereignissen und Dokumenten aller Art sowie zeitlich eingeordnet. Damit läßt sich einfach die Funktionalität realisieren, daß bei Klick auf eine entsprechende Sehenswürdigkeit auf der 2D-Karte das zugehörige VR-Modell geladen wird. Dies ist natürlich nur eine erste einfache Version des „Virtual Heidelberg“, aber für ein WWW-fähiges Touristeninformationssystem, das noch heutigen Bandbreitenbeschränkungen und ungewisser Rechenleistung auf Klientenseite unterworfen ist, ein vernünftiger Kompromiß.

5. Die GIS-Agenten des mobilen Prototyps im Detail

In den folgenden Abschnitten wird die oben skizzierte Grobarchitektur anhand der Schnittstellen der GIS-Agenten für die mobile *Deep Map* Variante konkretisiert. Die GIS Funktionalität wird in vier Softwarekomponenten bzw. Agenten bereitgestellt:

- GeoSpatial (2D Geometrie-Handling) (*geospatialAgent*)
- Visualisierung (2D Kartenausgabe) (*mapAgent*)
- Routenplanung (Netzwerkanalysen, Erreichbarkeit, ...) (*routeAgent*)
- Datenbank (Abfrage von DB-Inhalten) (*infoAgent*)

Eine weitere Unterteilung in noch kleinere Komponenten ist möglich (*Visibility*, 3D-Analyse, Volumen, Höhenprofil, 3D-*Topology* etc.). Abb. 3 zeigt diese Architektur, wobei das *User-Model* noch nicht in das System eingegliedert wurde. Ebenso ist die objektorientierte 4D-Datenbank noch nicht eingebunden, sondern nur die oben erwähnte historisch-geographische Datenbank auf relationaler Basis.

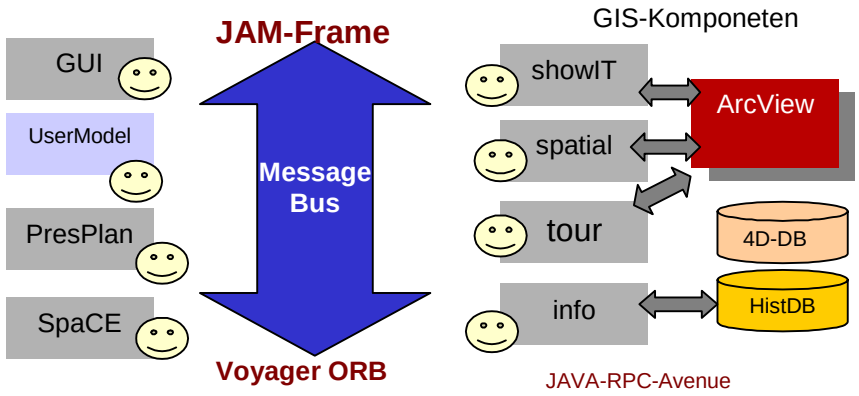

Abb. 3: Übersicht technische Architektur der GIS-Komponenten

Bei der graphischen Benutzeroberfläche (*GUI*) handelt es sich um einen Spezialfall, da zwei Versionen vorliegen – einmal in der Variante für das WWW (ZIPF / MALAKA 1999a) und einmal für den mobilen Prototypen. In Zukunft werden beide Varianten zu einer zusammengeführt werden, die sich an ihre jeweilige Ablaufumgebung adaptiert. Dies wird dann aber in einem eigenen Forschungsbereich in einem zusätzlichen Teilprojekt durchgeführt werden. Die Begründung für zwei Varianten liegt u. a. darin, daß im *Web* ein Sprach*interface* nicht erforderlich ist und die Projekte zur Sprachverarbeitung zu dem Zeitpunkt, als die ersten *Web*-Prototypen vorlagen, noch gar nicht begonnen hatten.

5.1 Info-Agent

Der Datenbank-Agent ist für die Suche nach multimedialer Sach-Informationen zu einem Objekt zuständig. Diese Objekte können z. B. Gebäude, Sehenswürdigkeiten oder andere räumliche Objekte, aber auch Personen, historische Ereignisse oder andere Informationseinheiten aus der historischen Datenbank sein. Es kann die Existenz eines entsprechenden Objekts angefragt werden sowie die Art und die Typen der zugehörigen Daten. Sind diese bekannt, können zu einem Objekt (z. B. der Peterskirche) die entsprechenden Informationen typspezifisch (z. B. nur die Texte, nur die Bilder, Videos oder Kombinationen daraus) angefordert werden.

Da teilweise auf verschiedenen Rechnern mehrere Instanzen der Datenbank laufen werden (z. B. eine mobile Variante und mindestens eine stationäre Serverinstallation), die in der Regel unterschiedlich gut erreichbar oder unterschiedlich gut gefüllt sind, muß der Datenbank-Agent entweder mehrere Anfragen an die verschiedenen Datenbanken senden oder aber entscheiden, welche er für die Bearbeitung der aktuellen Anfrage auswählt.

- *get-id* – Sucht im GIS und der Datenbank nach einem Objekt, welches den bezeichneten Namen besitzt. Optional kann der gewünschte Objekttyp mit angegeben werden. Wenn mehrere mögliche Treffer gefunden werden, da der Name nicht eindeutig aufgelöst werden konnte, werden alle Treffer in der Antwort zurückgegeben.
- *get-available-info* – Sucht im GIS und der Datenbank nach Informationen zu dem spezifizierten Objekt und gibt eine Liste der verfügbaren Informationstypen zurück.
- *is-info-available* – Diese Anfrage findet heraus, ob im GIS oder der Datenbank Informationen des angegebenen Informationstyps zu dem referenzierten Objekt zur Verfügung stehen.
- *get-info* – Fragt die Informationen eines bestimmten Typs vom GIS oder der Datenbank an. Als Antwort wird entweder die Information selbst als *InfoContent* zurückgegeben oder aber ein Verweis auf die Quelle der Information mittels einer *URL*.

5.2 Geospatial-Agent

Der Geospatial-Agent agiert als Dienstanbieter für Anfragen, die die GIS-Basisfunktionen betreffen. Hierzu zählt die räumliche Selektion. Es werden aber auch komplexere Dienste wie Sichtbarkeitsanalysen unterstützt. Daneben verwaltet er die Grundgeometrien und kann berechtigten Agenten die Geometrie von Geo-Objekten übermitteln.

- *get-geo* – Fordert Geometrien eines (oder mehrerer) Typen spezifizierter Objekte an.
- *get-objects-in-region* – Mit dieser Anfrage wird das GIS aufgefordert, nach Objekten innerhalb einer angegebenen Region zu suchen.
- *are-objects-in-region* – Hiermit wird beim GIS angefragt, ob Objekte mit den gegebenen Namen oder Objekt-IDs in der angegebenen Region vorhanden sind.
- *get-visible-objects* – Mit dieser Anfrage wird das GIS aufgefordert, diejenigen Objekte zu bestimmen, die von einer gegebenen Position aus sichtbar sind. Optional können die genauen *View*-Parameter, d. h. die Blickrichtung, der Sichtbereichswinkel und die maximale Sehtiefe sowie Inklination, spezifiziert werden.
- *are-objects-visible* – Bei dieser Anfrage wird eine Liste von Objekten oder räumlichen Einheiten an das GIS übergeben und das GIS aufgefordert, die Sichtbarkeitsgrade von der gegebenen Stelle zurückzuliefern.
- *compute-distance* – Berechnet die Entfernung zwischen zwei Objekten.

5.3 Tourenplanungs-Agent

Im Modul GIS-*Routing* (Tourenplanungs-Agent) werden verschiedene Netzwerkanalysen durchgeführt, die sowohl für die Wegfindung und Navigation als auch für die Generierung individueller Routen verwendet werden. Gleichzeitig findet die Verwaltung der einzelnen Routenabschnitte (benutzerdefiniert, topologisch und geometrisch) statt.

- *compute-route* – Beauftragt den Tourenplanungs-Agenten, eine Route zu errechnen, die alle spezifizierten Objekte einbezieht. Eine Option bestimmt, ob die Reihenfolge der Objekte auf der Route so eingehalten werden soll.
- *get-route-element* – Fordert einen Routenabschnitt einer zuvor berechneten Route an.
- *get-route* – Die Geometrie der Route (oder ein entsprechender Fehler) wird zurückgegeben.

5.4 Map-Agent

Der Map-Agent hat die Aufgabe, 2D-Karten (oder später 3D-Modelle) des spezifizierten Gebiets zu erzeugen (zu *rendern*), und stellt sie als Graphik zur Verfügung. Hauptabnehmer ist der *Presentation Planer*, der diese Graphiken dann anzeigt.

Dabei werden zwei Varianten implementiert. In der ersten Variante wird von dem angeforderten Kartenausschnitt mit ausgewählten hervorgehobenen Gebäuden oder Routen(abschnitten) die Karte gezeichnet und dann als Rastergraphik (*JPG*) exportiert und auf diesem Wege zur Verfügung gestellt, in der zweiten Variante werden die Karten als Vektorgraphik direkt in *Java* erzeugt und auf dem entsprechenden *GUI*-Element angezeigt. Hierzu ist eine Referenz auf die ensprechende *GUI* notwendig, während in der ersten Variante der *Presentation-Planner* die Graphik nach Fertigstellung durch den Map-Agent per angegebener *URL* findet und abgeholt werden kann und der *Presentation-Planner* dann immer noch entscheiden

kann, ob, wann, wo und wie diese Graphik angezeigt wird. Allerdings hat dies den Nachteil, daß Rastergraphiken a) in der Regel optisch nicht so gut aussehen, b) eine relativ hohe, dabei aber relativ konstante Übertragungslast beanspruchen sowie c) geringere Dynamik und Interaktivität möglich ist.

- Der *Request compute-map* erzeugt eine Karte mit den angeforderten Parametern. Mittels der Option *area-of-interest* wird das anzuzeigende Gebiet eingegrenzt. Beispielsweise kann eine früher berechnete Route über ihre ID angegben werden, und der Map-Agent muß dann einen sinnvollen Kartenauschnitt wählen, um die Route anzuzeigen. Neben dem Kartenausschnitt können diejenigen Objekte in der Karte spezifiziert werden, die als *Points-of-Interest* gelten und damit explizit in die Karte aufgenommen werden sollen und sich von dem vom Map-Agenten selbst zu wählenden Kartenhintergrund abheben sollen. Mittels der Angaben unter *highlight* werden diejenigen Objekte (ebenfalls von Gebäuden bis hin zu früher berechneten Routenabschnitten) angegeben, die besonders im Fokus, d. h. deutlich hervorgehoben sein sollen. Wie dies zu geschehen hat, ist nicht definiert, sondern obliegt der eigenen Logik des Map-Agent. Beispielsweise könnte dieser beim *UserModel Server* nachfragen, welche Eigenschaften, Präferenzen und Interessen der aktuelle Benutzer hat und den Karteninhalt entsprechend aufbereiten.

Abb. 4: Beispiel einer von *Deep Map* erzeugten Karte mit aktuellem Tourenabschnitt im Heidelberger Schloß

6. Diskussion und Ausblick

Das hier vorgestellte System wurde im Bereich des Heidelberger Schlosses demonstriert und vorgestellt. Es sind inzwischen Datenbank-Inhalte für einen weit größeren Bereich der Heidelberger Altstadt verfügbar. Zum Test und zur Demonstration des Systems wurde es jedoch auf den Teilbereich des Schlosses eingeschränkt. In Zusammenarbeit mit dem Heidelberger Forschungslabor von *NEC* (*NEC Europe C&C Research Labs*) und der *International University* in Bruchsal wurde für diesen Zweck eine drahtlose Infrastruktur aufgebaut, die dem mobilen Benutzer drahtlosen Zugriff auf verteilte Daten ermöglicht. Die sehr zeitaufwendige Attribuierung des Straßennetzwerkes für die Routenplanung wurde für einige Stadtteile

durchgeführt, eine Ausdehnung auf die Gesamtstadt steht jedoch noch aus, da hier zum Teil Neukartierungen vorgenommen werden mußten.

Die Wahl, *ArcView* GIS als Plattform für den mobilen Prototypen zu verwenden, fiel vor allem wegen der Verfügbarkeit des System, der Möglichkeit, mittels der objektorientierten Makrosprache *AVENUE* von *ArcView* GIS relativ schnell angepaßte GIS-Funktionen programmieren zu können und wegen des verfügbaren *know-how* am Geographischen Institut und dem *EML* mit diesem Produkt. Da es sich aber letztendlich derzeit doch um ein *Desktop*-System handelt und weniger um eine *Server*-Plattform, die im Hintergrund nur von anderen Komponenten gesteuert wird, werden momentan weitere Produkte (z. B. *MapObjects*, *SDE* und ein *PreRelease* von *ArcIMS-3.0*) evaluiert und besonders verstärkt Eigenentwicklungen in *Java* realisiert.

Insgesamt kann erwartet werden, daß in den technischen Bereichen die derzeitige Innovationsgeschwindigkeit zu kontinuierlich verbesserten Hardwarekomponenten führen wird. So werden unter den Stichworten *disappearing computer* und *wearables* laufend kleinere und leistungsfähigere Rechner sowie Komponenten zur Vernetzung entwickelt. Kritisch wird die Frage der Nutzbarkeit dieser Geräte sein. Deshalb sind intelligente Dienste wie die hier vorgestellten notwendig, um die Möglichkeiten dieser mobilen und überall verfügbaren Rechner für ein breites Anwenderspektrum nutzbar zu machen.

Die vorgestellten Komponenten von *Deep Map* stellen einen Schritt in diese Richtung dar. Sie stellen Informations-Dienste bereit, die sich an die Benutzer anpassen sollen und im mobilen Einsatz vor Ort für Touristen als Fußgänger bei der Navigation in unbekannter Umgebung Hilfestellungen leisten.

Literatur

CHANDRASEKHARA, V. / COORS, V. (1999): Integrated Map – Integration of heterogeneous data. *http://www.villa-bosch.de/eml/english/research/deepmap/integrated/integrated.html*.

COORS, V. / JASNOCH, U. (1999a): Using Wearable GIS in Outdoor Applications. – In: Computer Graphik TOPICS 11(2). – 11-13.

COORS, V. / JASNOCH, U. (1999b): Deep Map: A Virtual Tourist Guide in Heidelberg. – In: Computer Graphik TOPICS 11(2). – 13-15.

DIAS, A., et al. (1998): Keeping Contextual Awareness. – In: Proceedings of GIS_Planet 98. – Lisbon (Portugal).

FOUNDATION FOR PHYSICAL AGENTS (FIPA) (1997): Agent Communication Language ACL / Agent management (Specification version 2.0). *http://www.fipa.org/spec/fipa97.html.*

FOUNDATION FOR PHYSICAL AGENTS (FIPA) (1998): (Specification version 1.0) (1998). *http://www.fipa.org/spec/fipa98.html.*

HÄUSSLER, J. (1999): Prototyp eines GIS-gestützten historisch-geographischen Stadtführers für das WWW. [Dipl.arb. Univ. Heidelberg].

KRAY, C (1999): SPACE. Spatial Cognition Engine. http://www.villa-bosch.de/eml/english/research/deepmap/talking/space.html

MALAKA, R. / ZIPF, A. (1998): Deep Map – a visionary scenario. *http://www.villa-bosch.de/eml/english/research/deepmap/scenario.html*. (präsentiert im *Deep Map* Workshop II.)

MEUSBURGER, P. / ZIPF, A. (1998): Auf dem Weg zu einem 4D-GIS als Grundlage für den Prototyp eines historischen Stadtinformationssystems – das Projekt *Deep Map*. – In: KARRASCH, H., et al. (Hrsg.): Globaler Wandel – Welterbe. (= HGG-Journal 13). – Heidelberg. 212-214.

PORZEL, R., et al. (1999): TALKING MAP – Natural Language Processing for Tourist information Systems. *http://www.villa-bosch.de/eml/english/research/deepmap/talking/ talking.html.*

STROBL, J. (1997): Geo-Datenbasen und Karten im WWW. – In: AGIT 97. – Salzburg.

STUTTGARTER ZEITUNG (1999): Reisen ohne Sprachbarrieren. [27. Juli 1999].

ZIPF, A. / MALAKA, R. (1999a): 3D und 4D – Deep Map – das historische Touristeninformationssystem für Heidelberg. – In: GeoBit 07/99.

ZIPF, A. / MALAKA, R. (1999b): Web-basierte Planung und animierte Visualisierung von 3D Besichtigungstouren im Rahmen des Touristeninformationssystems Deep Map. – In: AGIT 99. – Salzburg.