Orchestrierung von OGC Web Diensten im Katastrophenmanagement

am Beispiel eines Emergency Route Service auf Basis der OpenLS Spezifikation

A. Weiser, P. Neis, A. Zipf
Geoinformatik und Vermessung, Fachhochschule Mainz

Zusammenfassung
Flexibilität und Wiederverwertbarkeit sind wesentliche Ziele bei der Entwicklung komplexer Anwendungen auf Basis von OGC Web Services (OWS). Im Projekt OK-GIS wird die Eignung von Web Service Orchestration (WSO) als Lösungsansatz am Beispiel eines exemplarischen Teilszenarios einer Evakuierung evaluiert. Die durchzuführenden Aktionen werden auf eine Dienstekette von OGC Basisdiensten gemappt. Wichtigster Dienst ist hierbei ein Emergency Route Service (ERS), der als erster basierend auf der OpenLS Spezifikation implementiert wurde.

Summary
Flexibility and reusability are major goals when developing complex applications based on OGC Web Services (OWS). Within the Project OK-GIS we evaluate the suitability of the Web Service Orchestration (WSO) technology as possible solution with the example of a partial evacuation scenario. Here, the actions to be performed get mapped on a service chain of basic OWS. The most important service hereby is the Emergency Route Service (ERS) which is the first open source ERS implementation based on the OpenLS specification.

1 Katastrophenmanagement und GIS? – OK-GIS!

Simulation/Vorhersage als auch Management von Katastrophen sind seit langem etablierte Anwendungsgebiete der Geoinformatik - leider in jüngerer Zeit mit verschärfter Brisanz. Interoperabilität ist bekanntlich gerade beim Katastrophenmanagement von besonderer Bedeutung, da Katastrophen sich i.d.R. weder um administrative noch um Systemgrenzen kümmern. Die Nutzung offener Standards von OGC, ISO, W3C etc. folgt daraus zwingend. Allerdings bleibt zu untersuchen, inwiefern die verfügbaren Standards dem anspruchsvollen Katastrophenszenario genügen, welche *best practices* ableitbar sind oder welche Erweiterungen der Standards nötig sind. Aktuell arbeiten nationale und internationale Projekte an dieser Thematik (z.B. Ochestra: http://www.eu-orchestra.org, Oasis: http://www.oasis-fp6.org). In dem BMBF Verbundprojekt „OK-GIS – Offenes Katastrophenmanagement mit freiem GIS“ (http://www.ok-gis.de) liegt ein weiterer Schwerpunkt auf der Realisierung freier Softwarekomponenten, die diese Standards implementieren und als Open Source zur Verfügung stellen. In OK-GIS arbeiten die Fachhochschulen Mainz, Oldenburg/Ostfriesland/Wilhelmshaven und Osnabrück, sowie mehrere Firmenpartner (LatLon Bonn, Intevation Osnabrück, Leiner&Wolff Heidelberg) sowie Anwender an unterschiedlichen Teilen des Projektes. Ziele umfassen

a.) Entwicklung und Umsetzung eines offenen Gesamtkonzepts und
b.) Entwicklung von freien Software-Komponenten

für die Verwaltung, Nutzung, Visualisierung und mobile Erfassung von Geodaten im Katastrophenmanagement. Sowohl Gesamtarchitektur als auch realisierte Komponenten sollen möglichst generisch sein, damit sie für verschiedene Katastrophenarten geeignet sind. Durch eine enge Kooperation mit potentiellen Anwendern (z.B. Feuerwehr und Stadtwerke Osnabrück) werden die Konzepte an ausgewählten Szenarien evaluiert. Ein exemplarisches Szenario betrifft die Entschärfung von Bomben. Dies war ursprünglich für WWII-Bombenfindlinge gedacht, ist aber nach den jüngeren Ereignissen (etwa bei Terroranschlägen nutzbar). Konkret geht es u.a. um die Unterstützung der Evakuierung der betroffenen Bevölkerung. Ein hierbei notwendiger Teilaspekt ist das Routing von Einsatzkräften und anderen Personen. Gefahrengebiete sollen dabei natürlich umgangen werden. Hierzu wird die Realisierung eines Emergency Route Service (ERS) auf Basis der OGC OpenLS Route Service (OpenLS RS) Specification vorgestellt (Abschnitt 2).

Ein wesentliches Ziel von OK-GIS ist die Schaffung eines möglichst flexiblen Baukastens für unterschiedliche Szenarien und Anforderungen im Katastrophenfall. Es soll versucht werden, von der

Notwendigkeit der direkten Implementierung eines konkreten Szenarios im Code nach Möglichkeit ein Stück wegzukommen, hin zu einer stärkeren Konfigurierbarkeit beim Aufbau eines solchen Systems. Technisch soll diese Freiheit durch die Verwendung von profilabhängigen Konfigurationsvarianten und der Aggregation höherwertiger Funktionalität durch Nutzung feingranularer Basisdienste erreicht werden. Letztere sollen entsprechend den Erfordernissen des jeweiligen Profils zu spezifischen Workflows kombiniert werden. Es wird untersucht, inwiefern es möglich ist für OGC Web Services (OWS) hierzu die Technik der Web Service Orchestration (WSO) (Peltz, 2003) einzusetzen. Hierfür werden in OK-GIS Profile für unterschiedliche Anwendungsfälle, Rollen und Clients etc. entwickelt. Durch WSO soll es mittels XML-basierter Sprachen wie BPEL (Business Process Execution Language) (Chen et al. 2006) prinzipiell möglich sein, Workflows als Service-Chains abzubilden. Diese werden dann in einer entsprechenden Engine ausgeführt. Ähnliche Ansätze diskutieren Kiehle et al. (2006) und Lemmens et al. (2006). Die Praktikabilität dieses Konzeptes soll überprüft werden.

Zunächst werden die neu implementierten Komponenten - nämlich der OpenLS RS und der Emergency Route Service (ERS) vorgestellt (Abschnitt 2). Es folgt eine kurze Einführung in Web Service Orchestration (WSO) und unsere Erfahrungen mit der Orchestrierung von OWS (Abschnitt 3). Schließlich wird der mögliche Einsatz von ERS und OpenLS RS in Zusammenhang mit WSO an einem beispielhaften Workflow im Szenario Bombenentschärfung diskutiert (Abschnitt 4).

2. Ein Emergency Route Service auf Basis der OpenLS Spezifikation

GDI-Frameworks wie *deegree* (Fitzke et al. 2004) bieten schon eine große Zahl an OWS zum Aufbau von Geodateninfrastrukturen (Bernard et al. 2004) an. Vor kurzem kam z.B. auch der *Web Processing Service* (WPS) hinzu (vgl. Heier und Kiehle 2006). Zudem gibt es diverse Aktivitäten im Bereich OGC *Sensor Web* (Botts et al. 2006). Letzteres ist für Katastrophenmanagement von besonderer Bedeutung und wird auch im Projekt OK-GIS bearbeitet (Claussen, in Arbeit). Die Notwendigkeit Routenplanungsfunktionalität für Einsatzkräfte und Bürger zur Verfügung zu stellen ist offensichtlich. Dies wurde durch die in OK-GIS durchgeführten Szenario- und Anforderungsanalysen (Biermann et al. 2005) explizit bestätigt.

Für Routenplanung gab es bisher keine Standard-konforme freie Implementierung. Aber was heisst hier Standard-konform? Ein Routenplanungsdienst taucht in den üblichen Übersichten der OGC Spezifikationen meist nicht auf. Dennoch gibt es hierfür eine relevante Spezifikation. Diese „versteckt“ sich unter den im Rahmen der Open Location Services (OpenLS) Initiative definierten Diensten. Die Bedeutung von Interoperabilität auch für LBS diskutiert Zipf (2001) - auch am Beispiel der OpenLS Initiative. Da die Dienste wie normale OWS als Web Services definiert wurden, sind sie auch in einer „normalen“ Web-GDI nutzbar. Es bestehen allerdings Einschränkungen gegenüber normalen OWS: so verfügen die Services nicht über den sonst obligatorischen *getCapabilities*-Request. Hier besteht Harmonisierungsbedarf. Allerdings bieten die Dienste eine Reihe interessanter Funktionen (s.u.).
Im Katastrophenfall dient die Routenplanung vor allem dazu Einsatzkräfte schnell zum Einsatzort zu leiten. Aber sie ist z.B. auch bei Straßensperrungen nutzbar.

2.1 OpenLS - die OpenGIS Location Services

OpenLS ist die Abkürzung für *Open Location Services* oder *OpenGIS Location Services*. Diese OGC-Initiative erarbeitet seit 2000 freie Spezifikationen (Schnittstellen und Protokolle), um für LBS relevante Dienste zu standardisieren. Das *OpenLS Service-Framework* besteht aus fünf sogenannten *Core Services* (OpenLS 2000):

- Part 1: *Directory Service* – Ortsbezogene “Gelbe Seiten” - Suche (Online-Branchenverzeichnis)
- Part 2: *Gateway Service* – Schnittstelle zu Positionierungsdiensten auf Basis von Mobilfunknetzen (gemäß Standard von LIF (Location Interoperabiltiy Forum)).
- Part 3: *Location Utility Service* – Geocoding & Reverse Geocoding etc.
- Part 4: *Presentation Service* – Karten mit eingezeichneten Routen, POIs etc.
- Part 5: *Route Service* - Routenplanung auf Netzwerkgraphen nach diversen Kriterien

Das Service-Framework bildet die OpenLS Plattform zusammen mit einer Reihe von per XML Schema spezifizierten Datentypen, den sog. ADT’s (Abstract Data Types), welche das *OpenLS Information Model definieren.* Die Plattform wird als *GeoMobility Server* (GMS) bezeichnet. Weitere Details finden sich in Zipf (2004).

2.2 Umsetzung von ERS und Route Service in OK-GIS

Im Folgenden werden der umgesetzte OpenLS RS und der darauf aufbauende ERS vorgestellt. Da im realisierten OpenLS RS fast alle OpenLS Core Services mit Ausnahme des *Gateway Service* intern umgesetzt sind und verwendet werden, kann die Standard-konforme Fertigstellung der übrigen OpenLS *Core Services* nun leicht erfolgen. Aktuell liegt eine sehr umfangreiche Implementierung (Neis 2006) des OpenLS RS vor. Zusätzlich wurde ein ERS und ein entsprechender Web-Client realisiert.

2.2.1 OpenLS Route Service (RS)

Der OpenLS RS ermöglicht die Routenplanung auf einem das Strassennetz repräsentierenden Graphen. Als Routingalgorithmus wurde der bekannte Dijkstra-Algorithmus (Dijkstra 1959) umgesetzt. Die OpenLS Spezifikation definiert, wie der RS anzusprechen ist, welche Optionen er bietet und wie die Antworten von ihm aufgebaut sein müssen. Bei der Routenplanung können neben den üblichen Start und Zielorten eine Vielzahl möglicher Kriterien Eingang finden (Zeit, Distanz, Bewegungstyp). Selbst Start und Ziel lassen sich in einer Fülle verschiedener Varianten angeben (z.B. über Adressen, Koordinaten, POIs, Geometrien etc.). Bei Adressen sind u.a. strukturierte und unstrukturierte Adressangaben möglich. Um Tippfehler bei Adressen oder POIs zu bereinigen, kommt der Levenshtein Algorithmus (Ringlstetter 2003) zum Einsatz. Dieser berechnet jeweils einen Ähnlichkeitswert (auch als *Distanz* bezeichnet, hier aber nicht geometrisch gemeint) zwischen der angegebenen Adresse oder POI und einem Referenzeintrag in der Datenbank. Der Eintrag mit der geringsten Distanz wird verwendet. Ebenso umfangreich sind die Ausgabevarianten. Diese gliedern sich in vier Bereiche:

1. *RouteSummary* - grundsätzliche Informationen, wie Gesamtstrecke und Gesamtzeit der Route.
2. *RouteGeometry* - Geometrie der ermittelten Route (Linie mit allen Wegpunkten)
3. *RouteInstruction* - ”Schritt für Schritt“ Fahr- oder Gehanweisungen der ermittelten Route. Dies wurde in einfacher Form in mehreren Sprachen implementiert.
4. *RouteMaps* - Karten, in denen die ermittelte Route dargestellt wird. Es können mehrere *RouteMaps* gleichzeitig angefragt werden, z.B. eine Übersichtskarte oder detaillierte Start- und Zielpunktansicht der Route.

Der implementierte RS unterstützt alle verpflichtenden und den allergrößten Teil der optionalen Parameter der OpenLS RS Spezifikation. Auch die Berücksichtigung von Einbahnstrassen wurde realisiert.
Die Spezifikation bringt den Vorteil mit sich, dass in einer Routenanfrage auch eine so genannte *AvoidList* angegeben werden kann. In dieser können Gebiete oder Straßen stehen, durch welche die Route nicht verlaufen soll. Wird in einem Request eine *AvoidList* angegeben, ermittelt der RS die optimale Route zwischen Start- und Ziel unter Berücksichtigung der Ausschlussgebiete durch Umfahrung dieser Gebiete. Diese Funktion nutzt der ERS. Wenn ein als Fläche definiertes Gebiet nicht durchfahren werden soll, erfolgt eine Verschneidung zwischen einem (oder mehreren) Polygon(en) und dem Straßennetz (Liniengraph), um die nicht befahrbaren Straßen zu ermitteln.

2.2.2 Emergency Route Service (ERS)

Ein ERS ist ein spezieller RS, welcher bei der Routenermittlung „automatisch“ aktuelle Gefahrengebiete und gesperrte Straßen berücksichtigt und umfährt. Anfragen an den ERS erfolgen genauso wie an den OpenLS RS. Der ERS fügt aktuelle Gefahrengebiete und gesperrte Straßen in den ursprünglichen Request in eine neue oder vorhandene *AvoidList* ein. Der komplette Request wird anschließend an den normalen RS weitergeleitet. Dieser ermittelt die Route unter Berücksichtigung der gesperrten Bereiche und gibt das gewünschte Ergebnis zurück. Ein einfacher ERS wird bereits von Keßler et al. (2003) vorgestellt. Allerdings wurde hier kein Standard berücksichtigt und lediglich Start und Ziel als Parameter unterstützt.

Das Einfügen der aktuellen Gefahrengebiete und gesperrten Strassen lässt sich auf unterschiedliche Weise umsetzen. In Abbildung 2 wird hierzu ein WFS angesprochen, während bei Keßler et al. (2003) dafür ein spezieller Dienst definiert wird. Im vorliegenden Prototyp wird zunächst direkt auf eine Geodatenbank zugegriffen. Es ist zudem denkbar dieses mittels *Web Service Orchestration* umzusetzen (s.u.). Das folgende XML-Fragment zeigt ein vereinfachtes Beispiel für einen *RouteRequest* mit automatisch eingefügten Gefahrengebieten und gesperrten Straßen, wie er vom ERS erstellt wurde und an den OpenLS RS gesendet wird:

```
<xls:XLS ... >
  <xls:RequestHeader/>
  <xls:Request methodName="RouteRequest" requestID="1" version="1.1">
    <xls:DetermineRouteRequest>
      <xls:RoutePlan>
        <xls:RoutePreference>Fastest</xls:RoutePreference>
        <xls:WayPointList> ... </xls:WayPointList>
        <xls:AvoidList>
          <xls:AOI>
            <gml:Polygon>
              <gml:exterior>
                <gml:LinearRing xsi:type="gml:LinearRingType">
                  <gml:pos>3434774.787 5794688.517</gml:pos>
                  <gml:pos>3434932.208 5794547.765</gml:pos>
                  <gml:pos>3434678.483 5794249.592</gml:pos>
                  <gml:pos>3434506.247 5794384.788</gml:pos>
                  <gml:pos>3434774.787 5794688.517</gml:pos>
                </gml:LinearRing>
              </gml:exterior>
            </gml:Polygon>
          </xls:AOI>
          <xls:Address xsi:type="xls:AddressType" countryCode="DE-NI">
            <xls:StreetAddress>
              <xls:Building xsi:type="xls:BuildingLocatorType" number="50"/>
              <xls:Street officialName="Am Stollenbach"/>
            </xls:StreetAddress>
            <xls:Place type="Municipality">Osnabrück</xls:Place>
            <xls:PostalCode>49074</xls:PostalCode>
          </xls:Address>
        </xls:AvoidList>
      </xls:RoutePlan>
    </xls:DetermineRouteRequest>
  </xls:Request>
</xls:XLS>
```

Listing 1: Beispiel „RS *RouteRequest*“

2.2.3 Graphische Ausgabe von Gefahrenrouten

Mit Hilfe der Angabe des *RouteMapRequest*-Elementes und mindestens einem *RouteMapOutput*-Element können bei der Routenanfrage Karten zur berechneten Route angefordert werden. Durch die Attribute des *RouteMapOutput*-Elements kann eine Karte den Anforderungen angepasst werden. Der *RouteMapRequest* entspricht dabei nicht dem WMS GetMap-Request, sondern bietet spezielle Möglichkeiten, um Elemente wie Routen oder POIs auf eine Basiskarte zu visualisieren. In unserer Umsetzung wurde dies intern unter Nutzung von *SLD User Styles* realisiert, die an SLD-WMS gesendet werden. Vorteile der Nutzung von SLD (Styled Layer Dscriptor) und wie diese leicht erzeugt werden können beschreiben Weiser und Zipf (2006).

Das *RouteGeometryRequest*-Element verfügt u.a. über das optionale *scale*-Attribut. Dieses definiert einen Maßstab. Für diesen kann ein Wert berechnet werden, der in unserem Fall für die Generalisierung der Route mittels des Douglas-Peucker-Algorithmus verwendet wird.

Nr.	Dauer	Fahranweisung	Distanz [KM]
1	PT0S	Sie starten auf: Römereschstr.	0
2	PT27S	Fahren Sie gerade aus auf: Römereschstr. für 0.4 KM - ca <1 Minute(n)	0.4
3	PT12S	Fahren Sie links auf: An Der Netter Heide für 0.2 KM - ca <1 Minute(n)	0.2
4	PT11S	Fahren Sie rechts auf: Am Schellenkamp für 0.2 KM - ca <1 Minute(n)	0.2
5	PT27S	Fahren Sie links auf: Bramscher Str. 002-159 für 0.4 KM - ca <1 Minute(n)	0.4
6	PT48S	Fahren Sie rechts auf: Oldenburger Landstr. für 0.7 KM - ca <1 Minute(n)	0.7
7	PT0S	Sie haben Ihr Ziel erreicht!	0

Abb. 1: ERS in Aktion – Screenshot des Prototypen

In Abbildung 1 sieht man das Ergebnis als *RouteMap* mit einigen weiteren Angaben aus der *RouteSummary*, sowie einfache Fahranweisungen. Diese können in mehreren Sprachen ausgegeben werden. Hierbei werden z.Zt. u.a. Richtungsanweisungen wie „halb rechts“, „halb links unterstützt.

2.3 Erweiterungsmöglichkeiten von RS und ERS

Bei der Implementierung des RS wurde darauf geachtet, dass verschiedene Teilkomponenten wie der Graph oder der Routing-Algorithmus erweiterbar oder austauschbar sind. Der Grund liegt in geplanten Ergänzungen und Weiterentwicklung des Route Service im Projekt OK-GIS. So sollen weitere spezifische Eigenschaften der Routenplanung für Einsatzkräfte berücksichtigt werden. Hierzu werden in weiteren Arbeiten beim Projektpartner FH Osnabrück Komponenten realisiert, die dann mit den hier vorgestellten Arbeiten integriert werden. Der realisierte OpenLS RS wird auch in einer weiteren Arbeit zum Einsatz kommen (Bock, in Arbeit). In einem Projekt zur interoperablen Modellierung und Visualisierung von 3D Stadtmodellen am Beispiel von Heidelberg (gefördert durch die Klaus-Tschira Stiftung, KTS) wird u.a. eine 3D-Webanwendung realisiert. Diese ermöglicht die OGC-konforme Planung und Visualisierung virtueller Besichtigungsrouten durch das 3D-Stadtmodell unter Nutzung des OpenLS Route Service und des Web3D Service (Quadt und Kolbe 2005). Das Szenario ist damit ähnlich wie bei Schilling und Zipf (2002). Im Gegensatz zur damaligen Lösung wird hier aber besonders auf Interoperabilität durch Nutzung offener OGC-Standards Wert gelegt. Fischer et al. (2006) stellen zudem einen mobilen Klienten für den W3D Service vor.

3. Ochestrierung von OGC Web Services mit BPEL.

Ziel von OK-GIS ist eine modulare Architektur, um flexibel auf die unterschiedlichen Erfordernisse im Katastrophenmanagement reagieren zu können. So sollen komplexere Funktionen und Workflows einer bestimmten Granularität so weit wie möglich mittels geschickter Aggregation standardisierter Basisdienste realisiert werden. Diese Funktionen sollen dann selbst wieder als Web Service zur Verfügung stehen. Wir nennen diese aus Basis-OWS zusammengesetzten Dienste zunächst „*virtuelle Dienste*“. Die zur

Aggregation notwendige Middleware kann wie üblich mit herkömmlichen Programmiersprachen für einen bestimmten Anwendungsfall entwickelt werden. Dadurch können alle Vorteile objektorientierter Sprache genutzt werden. Der Preis ist, dass jede Änderung innerhalb der Programmierplattform vorgenommen und kompiliert werden muss.
Das Versprechen der Alternative WSO ist es, die Verkettung von Diensten in vorgegebenen Mustern einfacher und flexibler realisieren zu können - v.a. durch Konfiguration über so genannte *Orchestration scripts* mittels Sprachen wie BPEL (Business Process Execution Language). Die Definition der BPEL Dokumente kann z.T. graphisch mittels einer GUI geschehen. BPEL benötigt WSDL (Web Service Description Language) und X-Path. WSDL ist eine Spezifikation zur Beschreibung einer Schnittstelle eines Web-Dienstes. Sie stellt das Bindeglied der OE (Orchestration Engine) zu den verwendeten Diensten dar. In diesen sind Aufrufbedingungen, Abhängigkeiten sowie Alternativen enthalten. Ihr Aufbau gliedert sich in einen abstrakten und einen konkreten Teil, wobei der abstrakte Teil die Funktionalität des Dienstes und der konkrete Teil die tatsächliche Implementierung als Dienst beschreibt. Die abstrakte Definition von Ein- und Ausgangsnachrichten (Messages) wiederum erfordert die Definition dieser Nachrichten als XML-Schema, damit in der OE eine entsprechende Variable als Container zur Verfügung gestellt werden kann. Diese Zusammenhänge stellen sehr feine Rädchen im Getriebe der WSO-Engine dar und bergen deshalb ein nicht zu unterschätzendes Fehlerpotential.
In der Praxis entstanden die meisten Probleme bei der Definition der WSDL und speziell bei der Definition von Service-Operationen und ihren Nachrichten, insbesondere da die Interpretation der Definitionen oftmals, bedingt durch die Implementierung, leicht unterschiedlich stattfindet. So lässt z.B. die weiter unten vorgestellte OracleBPEL PM für die Bestimmung der Service Response-Message keine Definition von XML Schema-Basistypen zu, sondern erfordert die Definition eines eigenen komplexen Ausgabetyps. Die tatsächliche Verbindung des BPEL-Skripts zur WSDL-Definition wird über sog. *Partnerlinks* vollzogen. In diesen wird die Rolle des Dienstes innerhalb der Dienstearchitektur festgelegt (Juric 2004).

3.1 WSO-Tests in OK-GIS

OWS stellen in OK-GIS die Basis für weitergehende Funktionen dar. Leider unterliegen OWS hierbei gewissen Einschränkungen: Ein bekanntes Problem ist die fehlende SOAP–Unterstützung (Simple Object Access Protocol) von OWS (vgl. OGC 03-014). Die Dienste kommunizieren stattdessen i.d.R. über das http-Protokoll. Die WSO-Engines nutzen aber üblicherweise das SOAP-Protokoll. Somit ist zu klären, ob eine Orchestrierung von OGC-Webdiensten überhaupt möglich ist. Hierzu wurden erste Tests durchgeführt. Diese umfassen *GetCapabilities*-Requests für WMS, WFS und WCS, sowie *GetMap* für den WMS und *GetCoverage* für den WCS. Bei den letzten beiden Requests sollte die Frage geklärt werden, ob die Orchestrierung für die speziellen http-GET Requests mit der OGC-typischen, kommaseparierten KVP-Codierung durchführbar ist (laut OGC besteht eine Inkompatibilität zwischen WSDL und den OGC http-GET KVP Codierungen (KVP = Key Value Pairs) (vgl. OGC 04-060r1).

Als Orchestration Engines wurden zwei Programme getestet. Beide verwenden BPEL4WS 1.1 (Business Process Execution Language for Web Services), eine für Web Services spezialisierte Untermenge von BPEL. Beide Programme bieten neben der eigentlichen Engine auch eine GUI zur graphischen Definition von BPEL Skripts. Die erste Engine ist ActiveBPEL 2.0 von der Active Endpoints Inc. mit dem ActiveBPEL Designer als GUI. Die zweite Engine ist der Oracle BPEL Process Manager 10.1.2 (nachfolgend OracleBPEL PM) mit der GUI BPEL Designer. Beide Engines sind Freeware.

3.1.1 Test mit der ActiveBPEL Engine

Die Nutzung und Konfiguration der ActiveBPEL-Engine ist eher spartanisch. Die im ActiveBPEL Designer erstellten BPEL-Projekte und vorab erstellten WSDL-Dateien müssen manuell in der Engine eingebunden werden. Jedoch lässt ein anderes Kriterium ActiveBPEL für die Nutzung in OK-GIS ausscheiden: leider wird das http-Protokoll nur indirekt unterstützt. Wenn unter ActiveBPEL ein sogenannter *http-binded service* benutzt werden soll, muss ein *custom invoke handler* in Java implementiert werden. Dieser stellt im Prinzip lediglich die Schnittstellenbeschreibung zu dem genutzten Dienst dar. Diese Beschreibung „ersetzt“ damit jedoch die ursprünglich dafür vorgesehene WSDL. Da weiterhin Code kompiliert werden muss, entfällt der Vorteil einer skriptbasierten Prozessdefinition - zumindest für die Verwendung von *http-binded services* für OWS.

3.1.2 Test mit dem OracleBPEL PM

Die Unterstützung von http-basierenden Diensten ist im OracleBPEL PM deutlich besser. Hier genügt die WSDL als Schnittstellenbeschreibung für die beteiligten Dienste. Der BPEL Designer bietet Unterstützung bei der Integration der WSDL in den BPEL-Prozess, in der die Informationen für Prozessvariablen und sogenannte *Partnerlinks* der WSDL entnommen werden. Entsprechend gut funktioniert die Orchestrierung. Zukünftig sollen auch komplexere Workflows getestet werden. Hierzu ist es aber notwendig, für alle beteiligten Dienste WSDL-Beschreibungen zu definieren. Für BPEL müssen XPath-Ausdrücke zur Adressierung der Daten in komplexen Service-Responses erstellt werden. Da die WSDL nicht seitens des OGC vorliegt, ist dies bei der erstmaligen Definition zeitaufwändig und fehleranfällig. In Abschnitt 4 wird ein möglicher komplexerer Workflow konzeptionell vorgestellt und diskutiert. So kann untersucht werden, ob der hier noch ausprogrammierte ERS auch durch ein entsprechendes BPEL-Dokument ersetzt werden kann.

3.1.3 BPEL-basierte Prozessierung von Binärdateien (Karten/Coverages)

Die Weiterleitung von Binärdateien ist mit der derzeitigen Version des Oracle BPEL PM nur für Dateien des XML Schema-Typs *base64* möglich. Damit sind die Rückgabetypen von WMS *GetMap*- oder WCS *GetCoverage*-Responses als „normal“ (hexadezimal) codierte Bitmaps inkompatibel zur derzeitigen Version des Oracle BPEL PM. Laut Angaben von Oracle soll erst ab Version 10.1.3 Binärdateien unterstützt werden. Somit können derzeit die oben erwähnten http-GET KVP-Tests nicht durchgeführt werden. Allerdings stellt sich auch die Frage nach dem Sinn große Binärdateien durch die WSO-Engine durchzuleiten.

4. Orchestration am Beispiel eines Emergency Routing bei einem Bombenfund

Im Folgenden wird diskutiert, wie ein Evakuierungsszenario unter Nutzung des ERS und Service Orchestration realisiert werden kann. Abb. 2 zeigt den Workflow des Szenarios einer Evakuierung bei einem „Bombenfund“ (vereinfacht nach Biermann et al. 2005). Aus diesem Szenario greifen wir uns den Teilaspekt der Erzeugung einer Notfallroute zwischen Evakuierungsgebiet und dem Evakuierungszentrum unter Berücksichtigung des Gefahrengebietes heraus. Als vorbereitende Maßnahmen müssen zudem das Evakuierungsgebiet und das nächstgelegene Evakuierungszentrum (mit ausreichender Kapazität) bestimmt werden.

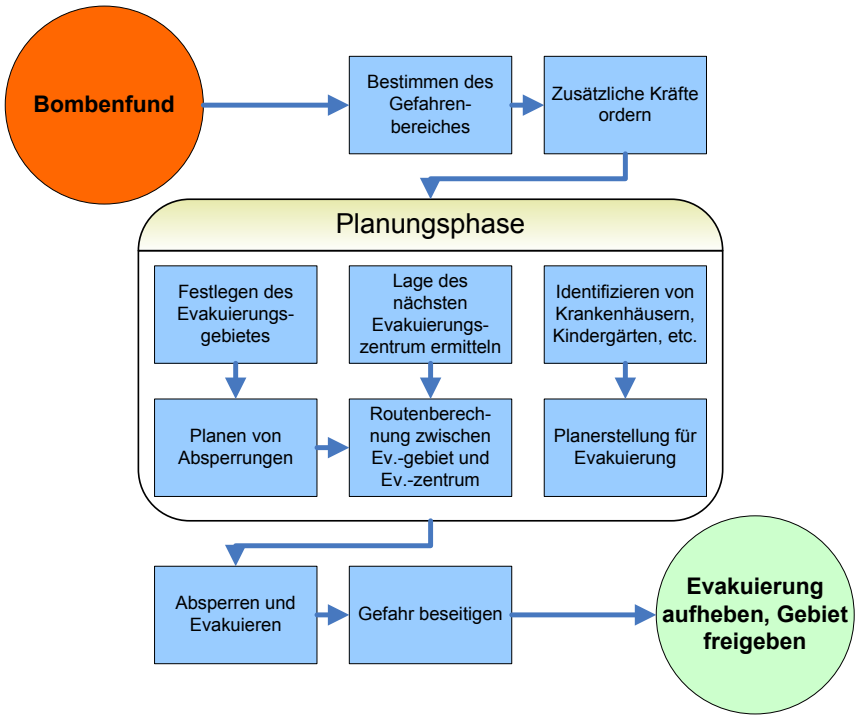

Abbildung 2: Ablaufplan für Evakuierung bei Bombenfund (vereinfacht nach Biermann et al. 2005).

Diese Teilaspekte sollen als atomare, wiederverwendbare Bestandteile des Szenarios per WSO als Dienst gekapselt werden. Abbildung. 3 zeigt einen ersten Entwurf wie dieser Teilaspekt des Szenarios auf die OWS Basisdienste unter Verwendung einer WSO Engine abgebildet werden könnte.

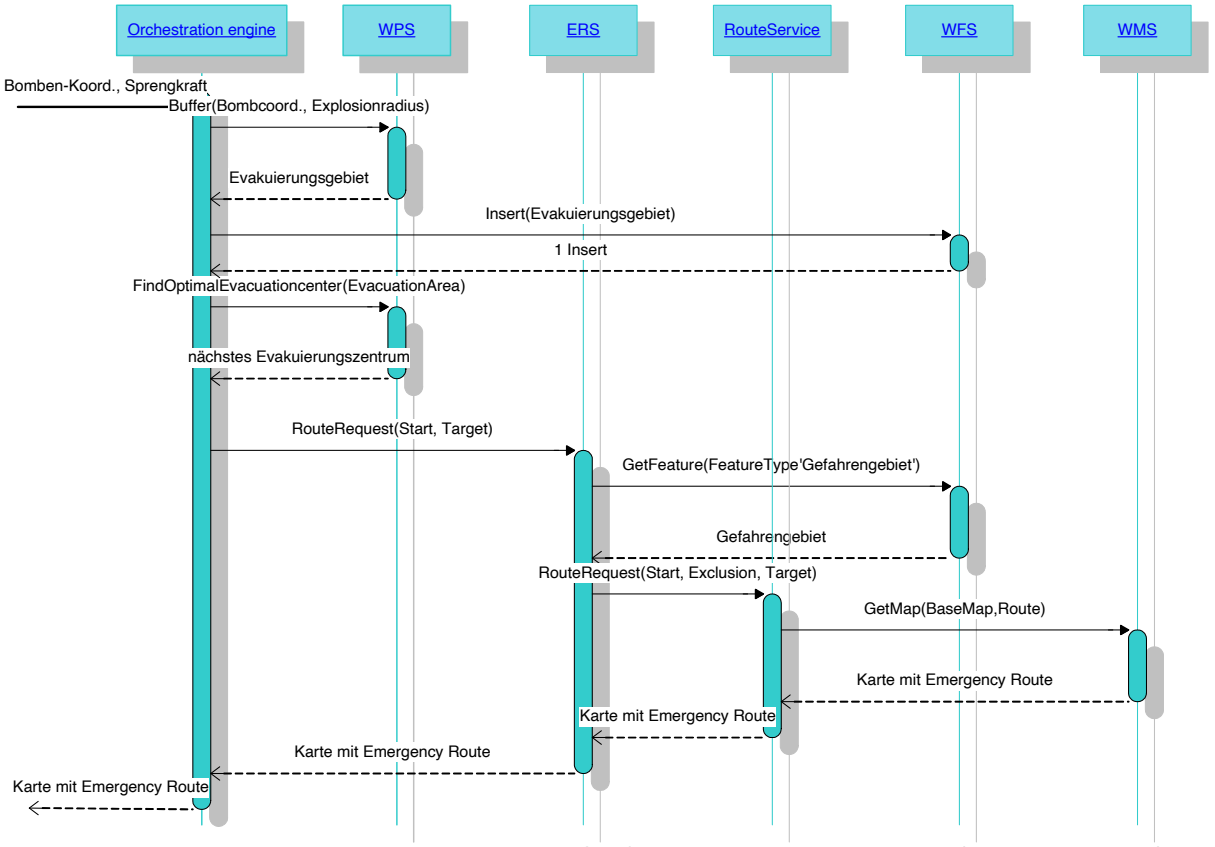

Abb. 3: UML Interaktionsdiagramm der beteiligten OWS des Evakuierungszenarios (stark vereinfacht)

Die Orchestration Engine (OE) integriert hier einen WPS, einen WFS und den oben vorgestellten ERS. Dieser verwendet intern einen OpenLS RS, der wiederum u.a. einen SLD-WMS nutzt. Die OE fungiert als eigenständiger, die Prozesskette kapselnder Webdienst und erhält vom Nutzer lediglich die Koordinaten der Bombe, sowie ihre Sprengkraft als Eingabe. Aus Sprengkraft und Punkt muss nun ein Evakuierungsgebiet berechnet werden. Dies kann auf unterschiedliche Weise umgesetzt werden. Eine sehr einfache ist, dass die Sprengkraft mittels einer (für das Bombenszenario) OE-internen XML-Lookuptable auf einen zu evakuierenden Kreisradius gemappt wird. Mit den Bombenkoordinaten und dem Radius kann der beteiligte WPS das Evakuierungsgebiet über eine einfache Bufferbildung (vgl. Heier und Kiehle 2006) ermitteln. Hierbei sind auch komplexere Varianten denkbar: So könnte der WPS auch noch die von der Evakuierung betroffene Bevölkerung berechnen. Hierzu ist eine Verschneidung mit auf Flächenbasis vorliegenden Bevölkerungsdaten notwendig. Zusätzlich wird eine Aggregationsfunktion benötigt, die die Bevölkerungszahlen im betroffenen Gebiet aufsummiert. Eine derartige Funktionalität (*Spatial Join* mit nachfolgender Aggregation) wurde von Stollberg (2006) für ein exemplarisches Waldbrandszenario (vgl. Friis-Christensen et al. 2006) als WPS im Rahmen einer Diplomarbeit der FH Mainz in Kooperation mit dem EU-Projekt „Ochestra“ am JRC in Ispra umgesetzt. Diese lässt sich auf das hier behandelte Szenario anpassen.
Im nächsten Schritt wird von der OE ein Prozess „*FindOptimalEvacuationCenter*“ des WPS aufgerufen, der für das übergebene Evakuierungsgebiet (GML – Geography Markup Language) das nächste (bzw. nächste ausreichend dimensionierte) Evakuierungszentrum berechnet. Auch hier soll aus Gründen der Übersichtlichkeit nicht auf die Realisierungsoptionen eingegangen werden. Mit den ermittelten Geometrien der Gefahrenzonen und des Evakuierungszentrums wird der *RouteRequest* an den ERS gestellt. Der holt sich vom WFS mittels *GetFeature* alle aktuellen Gefahren- und Sperrgebiete ab. Diese werden neben Start und Ziel als *AvoidAreas* in den folgenden Request an den RouteService eingefügt. Der RS

berechnet die Route inklusive *RouteGeometry*. Diese wird in einem *GetMap*-Request des RouteService als *UserLayer-SLD* zusammen mit den Parametern für eine Basiskarte übergeben. Als Ergebnis erhält der aufrufende Klient die Karte mit eingezeichneter Notfallroute sowie weitere relevante Informationen zur Route.

5. Fazit und Ausblick

Locker verbundene Dienste-Infrastrukturen bieten neue Möglichkeiten für die Geoinformatik. WSO mit BPEL4WS ist hier ein Schritt in die richtige Richtung zur Realisierung einer flexiblen *Service Oriented Architecture* (SOA). Den Bedürfnissen der wachsenden Nutzergemeinde soll mit einer neuen, stark erweiterten Spezifikation (WSBEPL 2.0) Rechnung getragen werden. In dem zukünftigen Standard sind Funktionalitäten von der BPEL4WS 1.1 Spezifikation abgeleitet. Diese sind nach aktuellem Stand jedoch strukturell inkompatibel zur Vorgängerversion. Derzeit ist die Orchestrierung von nicht-SOAP basierten Webdiensten in der Praxis schwierig, da die Engines i.d.R. SOAP-Kompatibilität der Dienste voraussetzen. SOAP ist als Protokoll „state of the art" bei modernen Webanwendungen. Dem trägt auch OGC mit Untersuchungen zur Verwendung von SOAP in OWS Rechnung (OGC 04-060r1 und OGC 03-014). Dennoch bietet das Protokoll nicht nur Vorteile (Wolter 2001), weshalb eine WSO Engine alle möglichen „*bindings*" bieten sollte - inklusive http-Get/Post.
WSO erübrigt es also für „neue" Funktionalitäten, die aber mittels aggregierter OWS realisiert werden können, jeweils extra einen speziellen Dienst zu definieren. Die teilweise zu beobachtende Tendenz den neuen WPS als „black box" für derartige Aggregationen zu nutzen, erscheint nur bedingt im Sinne der Erfinder. Denn hierdurch geht Transparenz verloren und die Flexibilität und Austauschbarkeit der einzelnen Komponenten einer OWS Dienstekette leidet.
OWS-basierte Diensteketten wie die oben dargestellte, lassen sich prinzipiell mit WSO orchestieren. Hierdurch können komplexe *Service Chains* als ein *virtueller Dienst* im System zur Verfügung gestellt werden. Auf diese Weise kann ein Baukasten von Diensten für aufwändigere Funktionalitäten erstellt werden. Dabei wird der Workflow v.a. per Konfiguration realisiert. Programmierung in einer konventionellen Programmiersprache ist hierfür nicht mehr notwendig. So können neue Anforderungen sehr dynamisch berücksichtig werden. Die dabei zur Zeit praktisch auftretenden Probleme wurden dargelegt, weitere Untersuchungen zu Performance, Sicherheitsaspekten etc. stehen noch aus.

Danksagung:
Wir danken allen Partnern im Projekt OK-GIS sehr herzlich für die fruchtbaren Diskussionen und die gute Zusammenarbeit. OK-GIS ist gefördert vom BMBF im Programm FH3. Förderkennzeichen 1704C05.

Literatur:

Bernard, L.; Fitzke, J. und Wagner, R. (eds) (2004): Geodateninfrastrukturen. Wichmann Hüthig. Heidelberg

Biermann, J., Gervens, T., Henke, S. (2005): Analyse der Nutzungszenarien und Aktivitäten der Feuerwehr . Internes Projektdokument. Projekt OK-GIS.

Bock, N. (in Arbeit): Web-Anwendung zur Visualisierung von 3D Stadtmodellen auf Basis der OGC W3DS und OpenLS Spezifikation. Diplomarbeit FH Mainz. (Arbeitstitel).

Botts, M., A. Robin, J. Davidson, I. Simonis (2006): OpenGIS® Sensor Web Enablement Architecture Document. V. 1.0. OGC Ref. Nr. 06-021r1. Discussion Paper.

Claussen, K. (in Arbeit): Umsetzung von OGC Sensor Web Standards im Katastrophenmanagement. Masterarbeit. FH Mainz. (Arbeitstitel).

Chen, L., Wassermann, B., Emmerich, W., Foster, H (2006): Web Service Orchestration with BPEL. London Software Systems; Dept. of Computer Science, University College London http://sse.cs.ucl.ac.uk/omii-bpel/publications/tut15-emmerich.pdf

Dijkstra, E. W.: A note on two problems in connexion with graphs. In: Numerische Mathematik. 1 (1959), S. 269271. 1959.

Fischer, M., Basanow, Zipf (2006): Mainz Mobile 3D - A PDA based client for the OGC Web 3D Service and corresponding server. International Workshop on 3D Geoinformation 2006 (3DGeoInfo'06). Kuala Lumpur. Malaysia.

Fitzke, J; Greve, K; Müller, M. and A. Poth (2004): Building SDIs with Free Software - the deegree Project. In: Proceedings of GSDI- 7, Bangalore, India. Online: http://gsdidocs.org/gsdiconf/GSDI-7/papers/TStgJF.pdf

FRIDA: Freie Vektor-Geodaten Osnabrück. Online unter: http://frida.intevation.org

Friis-Christensen A., L. Bernard, I. Kanellopoulos, J. Nogueras-Iso, S. Peedell, S. Schade, C. Thorne (2006): Building service oriented applications on top of a spatial data infrastructure – a forest fire assessment example. AGILE 2006.
Heier, C.; Kiehle, C. (2006): Automatisierte Liegenschaftsauskunft mittels OGC Web Processing Service. GIS 2006, 7: 12-16.
Kiehle, C.; Greve, K. & Heier, C. (2006): Standardized Geoprocessing – Taking Spatial Data Infrastructures one step further. Proceedings of the 9th AGILE International Conference on Geographic Information Science. Visegrád, Hungary.
Keßler, C., S. Schade, A. Starke, S. Tegtmeyer & A. Walkowski (2003): ariadne – GI-Dienste für Notfall-Management-Systeme. In: Bernard, L., A. Sliwinski & K. Senkler [Hrsg.](2003): Geodaten- und Geodienste-Infrastrukturen – von der Forschung zur praktischen Anwendung. Münster: 297-309. (= IfGIprints, Bd. 18)
Juric, M., Mathew, B., Sarang, P. (2004). Business Process Execution Language for Web Services. Hrsg. Packt publishing Ltd., Birmingham, 2006
Lemmens, R. C. Granell , A. Wytzisk, R. de By, M. Gould , P. van Oosterom (2006): Semantic and syntactic service descriptions at work in geo-service chaining. AGILE 2006.
Neis, P. (2006): Routenplaner für einen Emergency Route Service auf Basis der OpenLS Spezifikation. Diplomarbeit. FH Mainz.
OASIS. http://www.oasis-open.org/apps/org/workgroup/wsbpel/
OPENLS: OGC Open Location Services Version 1.1 http://www.opengeospatial.org/functional/?page=ols.
Open Geospatial Consortium Inc. OWS2 Common Architecture. Hrsg. OGC. RefNum. OGC 04-060r1; Vers. 1.0.0; Status: OGC Discussion Paper.
Open Geospatial Consortium Inc. OWS 1.2 SOAP Experiment Report. Hrsg. OGC. RefNum. OGC 03-014; Vers. 0.8; Status: OGC Discussion Paper.
OK-GIS Projektseite: Offenes Katastrophenmanagement mit freiem GIS. http://www.ok-gis.de
Peltz, C.. Web services orchestration and choreography. IEEE Computer, 2003. http://devresource.hp.com/drc/technical_white_papers/WSOrch/WSOrchestration.pdf
Ringlstetter, C. (2003): OCR-Korrektur und Bestimmung von Levenshtein-Gewichten. Magisterarbeit im Studiengang Computerlinguistik. Ludwig-Maximilians-Universität, München.
Quadt und Kolbe (2005): Web 3D Service Version 0.3.0 OGC Ref. Nr. 05-019. Discussion Paper.
Schilling, A. u. Zipf, A. (2002): Dynamische Generierung von VR-Stadtmodellen aus 2D- und 3D-Geodaten für Tourenanimationen In: GIS - Geo-Informations-Systeme. Zeitschrift für raumbezogene Information und Entscheidungen. 06/2002. 24-30.
Stollberg, B. (2006): Geoprocessing in Spatial Data Infrastructures - Design and Implementation of a Service for Aggregating Spatial Data. Diplomarbeit. FH Mainz.
Weiser, A., Zipf, A. (2006): Ein graphischer Editor zur automatischen Generierung von OGC Styled Layer Descriptor (SLD) Dateien für das Web-Mapping: GeoVis 2006 - Visualisierung und Erschließung von Geodaten. DGFK Kommission für Geoinformation und Visualisierung. Geoforschungszentrum GFZ Potsdam.
Wolter, R. (2001): Simply SOAP. Microsoft Corporation. http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnexxml/html/xml10152001.asp
Zipf, A. (2004): Mobile Anwendungen auf Basis von Geodateninfrastrukturen - von LBS zu UbiGIS. In: Bernard, L.; Fitzke, J. und Wagner, R. (eds): Geodateninfrastrukturen. Wichmann Verlag. Heidelberg.
Zipf, A. (2001): Interoperable GIS-Infrastruktur für Location-Based Services (LBS) - M-Commerce und GIS im Spannungsfeld zwischen Standardisierung und Forschung. In: GIS Geo-Informations-Systeme. Zeitschrift für raumbezogene Information und Entscheidungen. 09/2001. 37-43.