Hochschule Karlsruhe
Technik und Wirtschaft
UNIVERSITY OF APPLIED SCIENCES

LUBW

Bachelorthesis

Im Studiengang Kartographie und Geomatik

Evaluierung und Integration von Geoprocessing Services in das Räumliche Informations- und Planungssystem (RIPS) der Landesanstalt für Umwelt, Messungen und Naturschutz (LUBW)

von

Jens Riebel

März 2012

Aufgabenblatt für die Bachelorarbeit

von

Jens Riebel

an der

HOCHSCHULE KARLSRUHE – TECHNIK UND WIRTSCHAFT

Fakultät für Geomatik – Studiengang Geomatik

in Zusammenarbeit mit der

LUBW – Landesanstalt für Umwelt, Messungen und Naturschutz Baden-Württemberg im Informationstechnischen Zentrum der LUBW in Karlsruhe

---

Thema: **Evaluierung und Integration von Geoprocessing Services in das Räumliche Informations- und Planungssystem (RIPS) der Landesanstalt für Umwelt, Messungen und Naturschutz (LUBW)**

Ausgangssituation

Im Rahmen des Umweltinformationssystems Baden-Württemberg (UIS-BW) wird zur Erfassung und Auswertung vorhandener Geodaten das Räumliche Informations- und Planungssystem (RIPS) eingesetzt. Betreut vom Informationstechnischen Zentrum der Landesanstalt für Umwelt, Messungen und Naturschutz (LUBW) werden neben dem Geodatenmanagement auch GIS-Anwendungen und standardisierte Webdienste entwickelt. Letztere basieren auf einem eigenen Webservice Framework das in .NET entwickelt wurde. Für GIS-Funktionen wird der Esri ArcGIS Server eingesetzt. Durch Schnittstellen in den dezentralen Fachanwendungen können sowohl Fachanwender der verschiedenen Dienststellen des Landes als auch die Öffentlichkeit die RIPS-Services und Daten nutzen.
Zurzeit greifen die im RIPS eingesetzten Werkzeuge mittels SOAP oder WPS Protokoll unter Einsatz des ArcGIS Web Application Developer Framework (ADF) auf den ArcGIS Server zu.
Die Version 10.1 des ArcGIS Server wird die letzte Version sein, welche ADF unterstützt, auch dabei ist die Unterstützung bereits stark eingeschränkt. In Zukunft soll die Integration von Daten in Webanwendungen durch den Einsatz der „ArcGIS Web Mapping“- APIs für Javascript, Flex oder Silverlight erfolgen. Somit besteht an der LUBW der Bedarf, die vorhandene Architektur, welche den Zugriff über das ArcGIS Web ADF zur Verfügung stellt, auf die Technologie „Webservice über REST“

umzustellen. Die bisher im Web ADF implementierte Business Logik soll künftig durch sogenannte *geoprocessing services* ersetzt werden. Dabei kann eine Toolbox oder eine Karte mit speziellem Tool Layer direkt am ArcGIS Server veröffentlicht werden.

Aufgaben und Ziele

Im Zuge der Bachelorarbeit soll anhand von Beispielanwendungen diese Umstellung untersucht und vorbereitet werden. Hierfür soll der an der LUBW entstandene Prototyp zur dreidimensionalen Darstellung ausgewählter Themen verschiedener Fachkomponenten verfügbar gemacht werden. Als ein wichtiger Anwendungsfall soll eine bereits als Desktop-Lösung realisierte 3D-Darstellung in einer Fachanwendung für die „Grundwasser-Datenbank (GWDB)“ prototypisch im Web umgesetzt werden. Dabei wird ein geplantes Gebäude in eine topographische Karte einmodelliert und mit einer errechneten Ebene für den dort vorhandenen Grundwasserflurabstand verschnitten. Ziel ist dabei die Beantwortung der Frage, ob die Kellersohle des geplanten Gebäudes vom Grundwasser beeinflusst wird und daher z.B. abdichtende Baumaßnahmen erforderlich sind. Da bereits die Darstellung von vorhandenen oder geplanten Umweltobjekten in einem 3D-Raum wesentliche Informationen für die Bewertung der Umwelt liefern, ist der oben geschilderte Anwendungsfall so modular aufzubauen, dass auch die jeweiligen Einzelfunktionen separat verwendet werden können. Aus aktuellen Anforderungen der Landesregierung im Bereich erneuerbarer Energien stellt sich als weitere Aufgabe, die Planung einer Windenergieanlage (WEA) auf ihre Einsehbarkeit von unterschiedlichen Standpunkten aus und auf die optische Veränderung des Landschaftsbildes hin zu bewerten. Dabei ist es erforderlich, nach einer maßstabsgerechten Modellierung einer WEA – unter Verwendung von Google SketchUp – unterschiedliche Umweltfachthemen wie z.B. Biotope, Schutzgebiete etc. gemeinsam mit der WEA im 3D-Raum zu positionieren. Beide Anwendungsfälle sollen exemplarisch unter Verwendung der REST-Technologie von ESRI in die bestehende RIPS-Infrastruktur der LUBW integriert werden.

Das Ergebnis wird mit einer schriftlichen Ausarbeitung dokumentiert und im Rahmen eines Kolloquiums präsentiert.

Leitung der Bachelorarbeit:	Prof. Dr. Peter Freckmann
Zweiter Prüfer:	Dipl.-Agrarbiologe Manfred Müller
Bearbeitungszeit:	3 Monate
Tag der Ausgabe:	30.12.2011
Tag der Abgabe:	30.03.2012
Anschrift des Kandidaten:	Jens Riebel
	Bahnhofstr. 201
	75059 Zaisenhausen
	Tel.: 07258-327220
	E-Mail: jens.riebel@googlemail.com

Datum Leiter der Bachelorarbeit Zweiter Prüfer

Erklärung

Hiermit versichere ich, dass ich diese Bachelorthesis selbstständig verfasst und keine anderen als die von mir angegebenen Quellen und Hilfsmittel benutzt habe.

Karlsruhe, den

Jens Riebel

Danksagung

An dieser Stelle möchte ich mich bei Herrn Prof. Dr. Peter Freckmann und Herrn Biologiedirektor Manfred Müller für die Betreuung dieser Arbeit bedanken.
Bedanken möchte ich mich auch bei allen Mitarbeitern des Referats 53.2 der Landesanstalt für Umwelt, Messungen und Naturschutz für die Unterstützung in allen fachlichen und technischen Fragen, welche bei der Erstellung dieser Arbeit auftraten.
Besonderer Dank gilt meiner Familie, die mir das Studium ermöglicht und mich immer unterstützt hat.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

ADF	Application Development Framework
ALK	Automatisierte Liegenschaftskarte
API	Application Programming Interface
DGM	Digitales Geländemodell
FME	Feature Manipulation Engine
JSON	JavaScript Object Notation
OGC	Open Geospatial Consortium
PDF	Portable Document Format
REST	Representational State Transfer
RIA	Rich Internet Application
RIPS	Räumliches Informations- und Planungssystem
SOAP	Simple Object Access Protocol
SOE	Server Object Extensions
TIN	Triangulated Irregular Network
UDO	Umwelt-Daten und -Karten Online
UIS	Umweltinformationssystem
URL	Uniform Resource Locator
WCS	Web Coverage Service
WFS	Web Feature Service
WMS	Web Map Service
WPS	Web Processing Service
XAML	Extensible Application Markup Language

1 Einleitung

1.1 Vorwort und Zielsetzung

Im Zuge des Aufbaus von einheitlichen Geodateninfrastrukturen auf europäischer (INSPIRE), bundes- (GDI-DE) und landesweiter (GDI-BW) Ebene existiert innerhalb der Umweltverwaltung Baden-Württemberg mit dem Umweltinformationssystem (UIS) ein zentrales System, um Umweltdaten zur Verfügung zu stellen. Bestandteil dieses UIS ist auch das an der Landesanstalt für Umwelt, Messungen und Naturschutz(LUBW) entwickelte Räumliche Informations- und Planungssystem(RIPS). Hauptaufgabe des RIPS ist die Bereitstellung von Geobasisdaten und Geofachdaten, sowie die Entwicklung und Integration von Diensten in die Fachanwendungen der LUBW. Im Rahmen von RIPS werden diese Dienste in einem Framework bereitgestellt. Die Fachanwendungen können über verschiedene Schnittstellen auf diese zentralen Dienste zugreifen. Vorteil eines solchen Aufbaus ist, dass Funktionalitäten zentral zur Verfügung gestellt werden können, unabhängig von den eingesetzten Anwendungen.
Werkzeuge aus dem ArcGIS Umfeld werden zurzeit über ein Web ADF eingebunden. In kommenden Versionen der ArcGIS Software wird ESRI jedoch die Unterstützung für diese Lösung einstellen. Daher besteht eines der Ziele dieser Arbeit darin, zu untersuchen, wie in Zukunft die Integration von ArcGIS Funktionalitäten in das Rips Umfeld sichergestellt werden kann. Darüber hinaus sollen Möglichkeiten untersucht werden, dreidimensionale Darstellungen innerhalb von Rips umzusetzen.
Im GIS Bereich wird die dreidimensionale Visualisierung von räumlichen Daten in Zukunft eine größere Rolle spielen. Zurzeit wird im Rips Umfeld für die 3D Darstellung die Software GeoPro3D der Firma Disy verwendet. Einsatzbereich ist hierbei die Visualisierung von Grundwasserdaten und die Untersuchung von Grundwasserflurabständen. In Zukunft soll diese Software nicht mehr zum Einsatz kommen. Im Rahmen einer Bachelor-Thesis wurde an der LUBW nach geeigneten Möglichkeiten gesucht, GeoPro3D abzulösen. Es zeigte sich, dass der Einsatz von 3D PDF Dokumenten einen geeigneten Ersatz darstellen könnte. Ein Vorteil des 3DPDF Formats ist darüber hinaus, dass beliebige Themen dreidimensional dargestellt werden können. Ziel dieser Arbeit ist daher auch das Erstellen und Einbinden einer 3DPDF Anwendung in das bestehende Rips Framework. Zu den zu visualisierenden Sachverhalten gehört hierbei insbesondere auch die Darstellung von Windenergieanlagen im Gelände, eine aktuellen Anforderung der Landesregierung im Bereich erneuerbare Energien. Ziel hierbei ist es, die Einsehbarkeit und die eventuelle Änderungen des Landschaftsbildes durch den Bau von Windenergieanlagen zu überprüfen.

1.3 Das Umweltinformationssystem Baden-Württemberg

Politik und Verwaltung sind auf möglichst umfassende und aktuelle Informationen über den Zustand der Umwelt angewiesen [UIS_2006, S.9]. Ziel des Umweltinformationssystems Baden-Württemberg ist die Zusammenfassung und Verfügbarmachung der von vielen Stellen erfassten und verwalteten Umweltdaten [UM_3].
Von besonderer Bedeutung ist hierbei der Einsatz von einheitlichen Standards und Formaten. Darüber hinaus sind für eine sachgerechte Nutzung in der Regel Zusatzinformationen notwendig [ebd.]. Aufgaben und Ziele des UIS BW sind unter anderem:

- Planung und Verwaltungsvollzug
- Umweltbeobachtung und Monitoring

- Investitionsschutz und Weiterentwicklung
- Notfall und Vorsorgefall
- Informationsbereitstellung für Politik, Verwaltung

[vgl. UIS_2006, S. 10]
Da der Themenbereich Umwelt in der Landesverwaltung Baden-Württemberg nicht ausschließlich von einem Ressort bearbeitet wird, verfolgt das UIS einen ressortübergreifenden Ansatz. Dies hat zur Folge, dass das UIS BW weitgehend unabhängig von der aktuellen Organisation der Umweltverwaltung ist [ebd., S.12].
Zu den Nutzern des UIS gehören in erster Linie die Dienststellen der Landesverwaltung sowie des kommunalen Bereichs in Baden-Württemberg. Zunehmend wichtiger wird auch die Information der interessierten Öffentlichkeit [ebd., S.20].

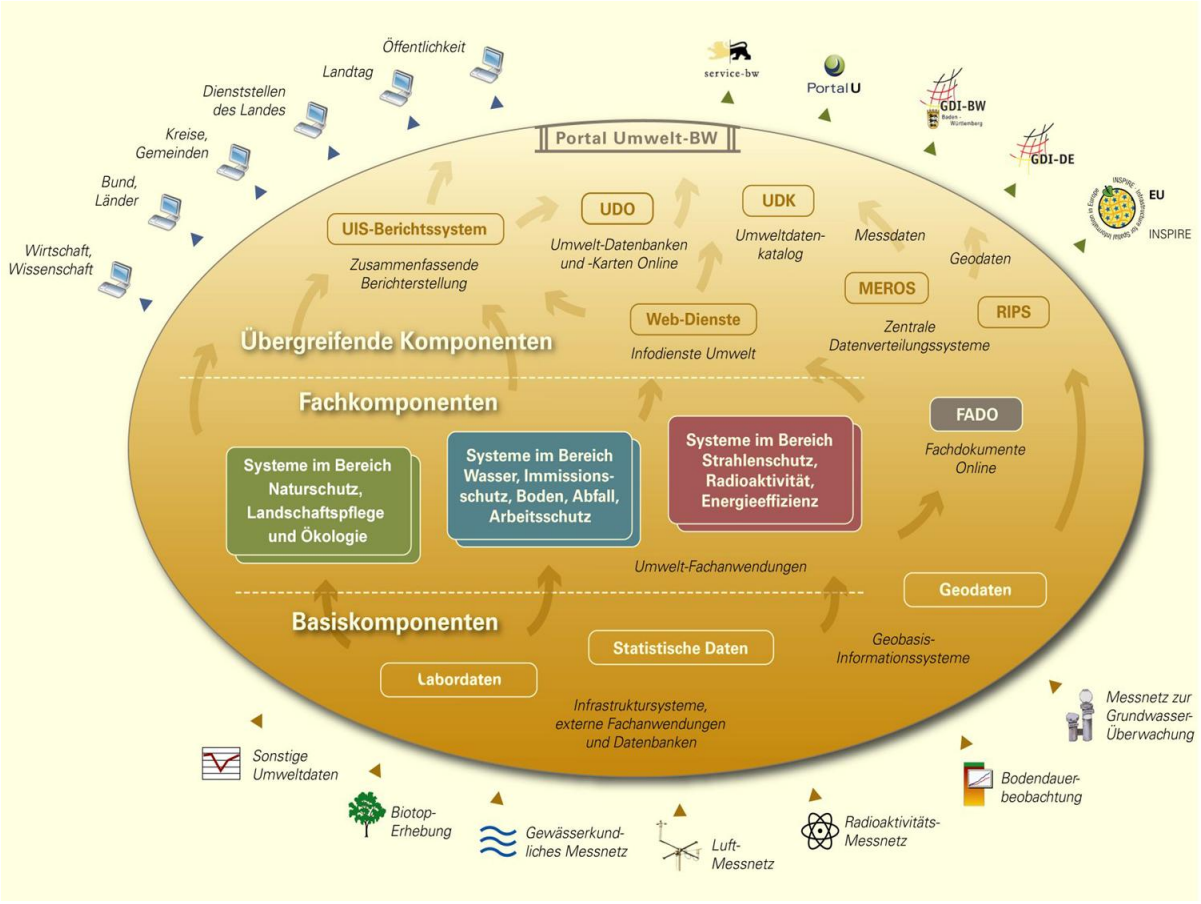

Abbildung 1: Das UIS Baden-Württemberg, aus: [UM_4,S. 4]

Das UIS BW besteht aus zahlreichen, nach Aufgaben differenzierten Komponenten. Es lassen sich drei Hauptkategorien unterscheiden:

- Basiskomponenten
- Fachkomponenten
- Übergreifende Komponenten

Zu den Basiskomponenten zählen alle Komponenten, die nicht speziell der Bearbeitung von Umweltinformationen dienen, welche jedoch im Rahmen des UIS BW Verwendung finden [UM_2]. Die Fachkomponenten für die einzelnen Umweltbereiche bilden als fachspezifische Systeme den Hauptbestandteil des UIS BW. Übergreifende Komponenten schließlich dienen der Zusammenführung und fachübergreifenden Darstellung der verschiedenen Umweltdaten. Zu den übergreifenden Komponenten zählen etwa das UIS-Berichtssystem oder das Räumliche Informations- und Planungssystem (RIPS) [ebd.].

1.4 Das Räumliche Informations- und Planungssystem

Das Räumliche Informations- und Planungssystem (RIPS) ist ein wesentlicher Bestandteil des UIS BW. Schwerpunkt ist der Aufbau einer anwenderfreundlichen und robusten Geodateninfrastruktur für das UIS [RIPS_1]. Im Wesentlichen werden im Vorhaben des RIPS drei Zwecke verfolgt:

- Breitstellung der Geobasisdaten für die Fachkomponenten und das Berichtssystem des UIS
- Organisation und Betrieb eines Datenpools mit Geobasis- und Geofachdaten
- Entwicklung von Geodatenbanken und Diensten für die Fachanwendungen

[vgl. RIPS_2006, S. 9]

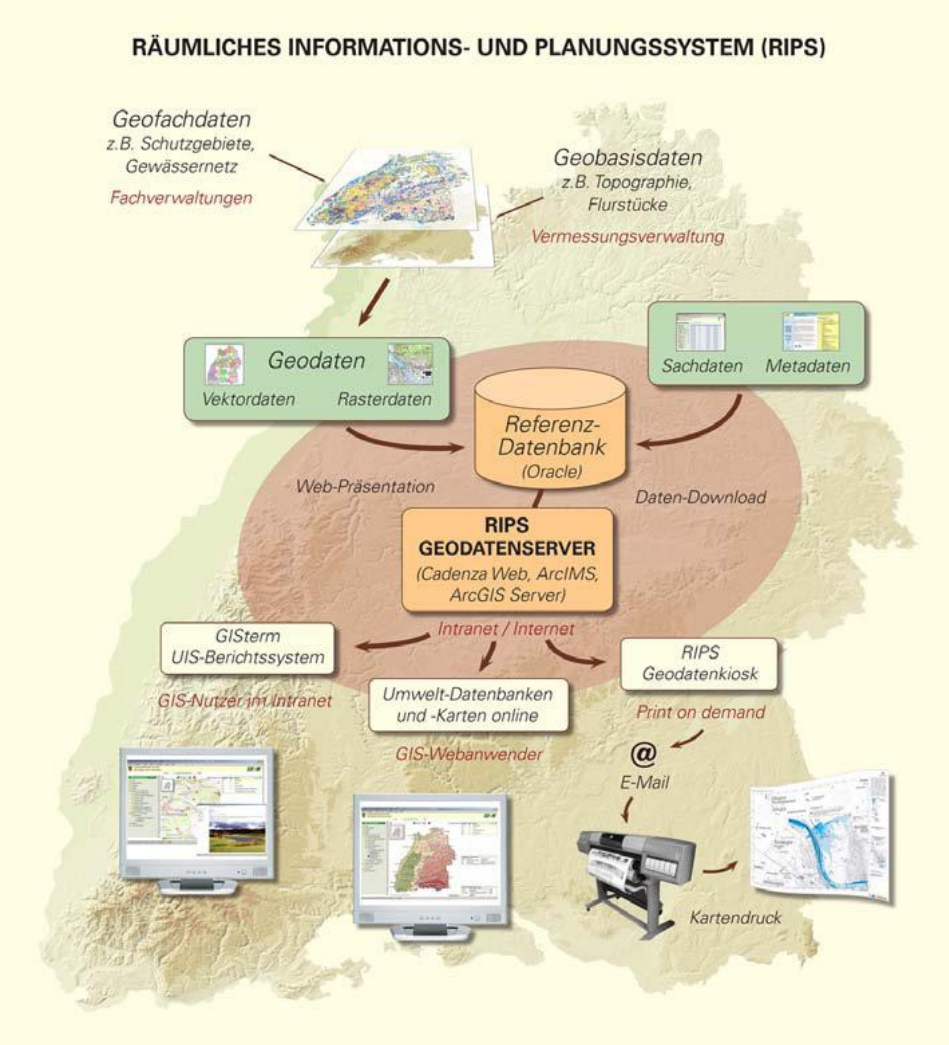

Abbildung 2: Räumliches Informations- und Planungssystem, aus: [UM_5, S. 1]

Wichtigste Komponente ist die themenübergreifende UIS-Datenbank, die aus allen umweltrelevanten Fachsystemen des Landes gefüllt wird. Neben der Umweltverwaltung und anderen Dienststellen profitieren auch interessierte Bürger von den in der UIS-Datenbank zur Verfügung gestellten Informationen [UM_1]. Zu den weiteren Komponenten des RIPS gehören öffentliche Kartendienste, welche beispielsweise genutzt werden, um Bürger über die jeweilige Umweltsituation am Wohnort zu informieren. Darüber hinaus werden die öffentlichen Kartendienste auch bei Planungsprozessen oder der Entscheidung bei Infrastrukturmaßnahmen eingesetzt.
Über den öffentlich zugänglichen Bereich hinaus werden im Rahmen des RIPS auch spezielle Dienste für die verschiedenen Verwaltungseinrichtungen des Landes angeboten [ebd.].
Schließlich spielt RIPS auch eine wichtige Rolle im Rahmen des Aufbaus einheitlicher Geodateninfrastrukturen (INSPIRE, GDI-DE, GDI-BW) sowie beim Geodatenmanagement und der Bereitstellung von Geobasis- und Geofachdaten [ebd.].

2 Technische Grundlagen

2.1 3D PDF

Das Softwareformat PDF (*Portable Document Format*) bietet seit der Reader Version 8.1 (Dateiversion 1.7) die Möglichkeit, 3D Objekte darzustellen. Hierbei lassen sich alle gängigen CAD Formate in eine PDF Datei einbinden [ADOBE_1]. Vorteile des PDF Formats liegen unter anderem in seiner weiten Verbreitung sowie der kostenlosen Anzeigemöglichkeit. Lediglich für die Erstellung eines 3D PDF wird lizenzpflichtige Software benötigt. Das an der LUBW eingesetzte Softwarepaket Feature Manipulation Engine (FME) bietet jedoch bereits die Möglichkeit, aus verschiedenen Eingabeformaten PDF Dateien zu erzeugen. Somit würden bei einer Umsetzung als zentrale, serverseitige Lösung keine weiteren Lizenzierungskosten anfallen. Insbesondere die Unterstützung aller relevanten ESRI Dateiformate, wie Shape-Dateien, ist ein Vorteil von FME. Im Falle der GWDB Anwendung können beispielsweise die Grundwassermesspunkte als Shape-Datei ausgelesen werden. Diese Datei kann nun direkt mit FME in ein 3D PDF Dokument integriert werden. FME bietet darüber hinaus die Möglichkeit, aus vorhandenen Rasterdaten ein TIN (*Triangulated Irregular Network*, Unregelmäßiges Dreiecksnetz) zu berechnen. Dieses kann mit einer beliebigen Oberfläche versehen und dann als dreidimensionales Oberflächenmodell in einem 3D PDF dargestellt werden. Auch bei der Visualisierung von Windenergieanlagen könnten beispielsweise Google SketchUp Dateien, aber auch CAD Dateien etc. direkt mittels FME in 3D PDF Dokumente integriert werden.

2.2 REST

Bei REST (*Representational State Transfer*) handelt es sich um eine Softwarearchitektur für verteilte Hypermedia-Systeme, wie das World Wide Web [WIKI_1]. Es wurde als ein architektonisches Modell für den Ablauf der Kommunikation im Internet entwickelt [Fielding_2000, S. 106]. Fielding definiert sechs Beschränkungen, welchen eine REST-konforme Kommunikation unterliegen sollte:

- Client-Server
- Zustandslos
- Cache-fähig
- Einheitliche Schnittstelle
- Geschichtetes System
- Code on Demand

[vgl. Fielding_2000, S. 77ff]

Unter der Client-Server Beschränkung wir die strikte Trennung von Benutzeroberfläche und Datenhaltung verstanden. Dies hat zur Folge, dass sowohl Client als auch Server unabhängig voneinander weiterentwickelt werden können, so lange die einheitliche Schnittstelle unverändert bleibt [ebd., S. 78].
Als weitere Beschränkung wird die Zustandslosigkeit gefordert. Jede vom Client an den Server gestellte Anfrage muss Informationen darüber enthalten, wie sie verarbeitet werden soll. Der Zustand der aktuellen Sitzung befindet sich vollständig auf Clientseite, der Server arbeitet Anfragen immer vollständig ab. Diese Beschränkung dient insbesondere der Sichtbarkeit für

Wartungsanfragen, der Zuverlässigkeit bei Netzwerkausfällen, sowie der Skalierbarkeit der Server [ebd., S. 78f].
Die nächste Beschränkung definiert den Einsatz von Zwischenspeicherung(Caching) von Daten. Daten aus einer Antwort des Servers können als cache-fähig beschrieben werden. Stellt der Client dann zu einem späteren Zeitpunkt die gleiche Anfrage an den Server, so kann dieser die Antwort aus dem Cache abrufen. Vorteil hierbei ist eine verbesserte Effizienz sowie potentiell kürzere Wartezeiten. Nachteil ist eine eventuelle Verwendung von mittlerweile veralteten Daten im Cache, bei Abruf zu einem späteren Zeitpunkt [ebd., S. 79f].
Das zentrale Unterscheidungsmerkmal der REST Schnittstelle ist die Beschränkung auf eine einheitliche Schnittstelle zwischen den einzelnen Komponenten. Im Rahmen von REST sind Ressourcen die eigentlichen, am Server vorhandenen, Ziele von Clientanfragen. Repräsentationen hingegen sind die vom Server tatsächlich übertragenen Elemente [ebd., S. 87f].
Hauptmerkmale der Beschränkung sind die eindeutige Identifizierung von Ressourcen, die Manipulation dieser Ressourcen ausschließlich über die entsprechende Repräsentation mittels URI, selbstbeschreibende Nachrichten zwischen Client und Server sowie die ausschließliche Verwendung von Hypermedia zur Kommunikation und Zustandsänderung von Client und Server [WIKI_1].Ein geschichtetes System im Sinne von REST erhöht die Skalierbarkeit des Systems und kann dem Lastenausgleich auf Serverseite dienen. Jedes Element einer Schicht kann nur seine eigenen Anfragen sehen. Trotzdem kann der Client weiterhin über die ihm bekannte Schnittstelle kommunizieren. Nachteile eines verteilten Systems können langsamere Zugriffszeiten sein, was wiederrum durch Caching ausgeglichen werden kann[Fielding_2000, S. 83].
Code on Demand schließlich bedeutet, dass der Client vom Server ausführbaren Code herunterladen und ausführen kann. Dies kann zu einer Vereinfachung der benötigten Fähigkeiten des Klienten führen. Diese letzte Beschränkung ist allerdings optional [ebd., S. 84].

2.3 REST API des ArcGIS Server

Esri bietet für den Zugriff auf mittels ArcGIS Server veröffentlichte GIS Dienste unter anderem eine REST API (Application Programming Interface) an. Alle am ArcGIS Server exponierten Ressourcen und Operationen jedes veröffentlichten GIS Dienstes sind über hierarchisch geordnete Endpunkte in Form von URLs (Uniform Resource Locators) erreichbar [ESRI_7]. Beim Arbeiten mit der REST API wird in der Regel von einem bekannten Endpunkt ausgegangen. Dieser repräsentiert den Server Katalog, welcher alle angebotenen Dienste auflistet [ebd.]. Bei einer Standardinstallation ist der Katalog erreichbar über die URL: http://<host>/arcgis/rest. Das Wurzelverzeichnis der angebotenen Dienste befindet entsprechend unter http://<host>/arcgis/rest/services [ebd.].

Abbildung 3: REST Schnittstelle in einem Webbrowser

Über dieses Dienste-Verzeichnis („*Services Directory*“) kann der Inhalt des ArcGIS Servers durchsucht und für die Entwicklung benötigte Informationen abgerufen werden [ESRI_11]. Beim erstmaligen Aufruf werden alle im Wurzelverzeichnis hinterlegten Dienste sowie eventuell vorhandene Unterverzeichnisse angezeigt. Sind weitere Informationen über einen speziellen Service erwünscht, so können diese durch einfaches Anklicken der entsprechenden Service-URL abgerufen werden [ebd.].

Beim Betrachten eines Kartendienstes besteht hier auch die Möglichkeit, eine Vorschau durchzuführen um die richtige Anzeige zu gewährleisten. Bei einem Geoprocessing Dienst wird eine Liste der vom Dienst verwendeten Variablen angezeigt.

Die Service-URL stellt darüber hinaus den von den ESRI Entwicklungs-APIs (für Javascript, Flex, Silverlight) benötigten Endpunkt dar. Über die URL kann der entsprechende Karten-, Geoprocessing- oder andere Dienst angesprochen werden [ebd.].

ESRI unterscheidet bei den Servicearten nach Ressourcen und Operationen, obwohl es sich bei Beiden im Sinne der REST Definition um Repräsentationen handelt. Operationen sind immer mit Diensten verknüpft, jedoch individuell abrufbar, beispielsweise die „*ExportMap*“ Operation eines Kartendienstes. Ressourcen verfügen über keine assoziierten Operationen, beispielsweise Netzwerk- oder Geodatendienste [ESRI_7].
Zu den unterstützen Ausgabeformaten der REST Schnittstelle gehören neben HTML und JSON beispielsweise auch Image oder das ESRI Format .lyr [ebd.].

2.4 ArcGIS Server

Mit der Serverversion von ArcGIS stellt die Firma ESRI eine Möglichkeit zur Verfügung, um auf GIS Dienste über das Internet zuzugreifen. Im Vorfeld beispielsweise mit der Desktoplösung ArcGIS erstellte Karten, Werkzeuge etc. können mittels ArcGIS Server einer großen Zahl an Nutzern zugänglich gemacht werden [ESRI_6]. Der Zugriff auf die vom Server bereitgestellten Dienste kann über REST, SOAP oder OGC-konforme (WMS, WFS, WCS) Schnittstellen erfolgen. Es können unter anderem Dienste für Karten, Geoprocessing, Geocoding, Geodaten oder auch Netzwerkanalyse angeboten werden [ebd.]. Über die REST-Schnittstelle besteht somit die Möglichkeit, Geoprocessing mit Hilfe des ArcGIS Server durchzuführen. Ein veröffentlichter Geoverarbeitungsdienst kann über REST eine Reihe von Anwendungen („*tasks*“) anbieten. Tasks sind zum einen Elemente in veröffentlichten Werkzeugboxen („*toolboxes*“), wie Modelle oder Skripte, zum anderen können sie auch Kartendokumente mit Werkzeug-Layern darstellen [ESRI_1]. Hierbei ist der Zugriff auf die veröffentlichten Dienste auf verschiedene Arten möglich. Sie können beispielsweise direkt in einer Desktopanwendung wie ArcGIS Desktop oder Explorer eingebunden werden. Weiterhin bestehen Zugriffsmöglichkeiten auch über die verschiedenen von ESRI angebotenen Web-APIs, für Javascript, Flex oder Silverlight [ebd.].
Ein Geoprocessing-Dienst im ArcGIS Server Umfeld basiert auf der Veröffentlichung von Geoverarbeitungs-Werkzeugen oder Karten mit integrierten Werkzeugebenen. Die Werkzeuge werden mit Hilfe der Desktoplösung von ArcGIS erstellt. Hierbei kann sowohl der integrierte Model Builder, Python Skripte oder eine Kombination aus Beidem zu Einsatz kommen. So veröffentlichte Werkzeuge stehen dann jedem Nutzer mit Netzwerkzugriff auf den Server zur Verfügung [ebd.].

2.4.1 Vorbereitung

Für den Desktopeinsatz erstellte Werkzeuge müssen unter Umständen an die Serverumgebung angepasst werden, bevor eine Veröffentlichung erfolgen kann. Insbesondere für Zwischenschritte benötigte Daten sowie Ergebnisdaten sollten an den dafür vorgesehenen Stellen gespeichert werden. Für temporär benötigte Daten sollte die Speicherung im sogenannten „*Scratch Workspace*“ oder der dortigen Geodatenbank erfolgen. Wird der Dienst mittels Model Builder erstellt, sollten temporäre Daten „*managed*“ gesetzt werden. Damit werden die Daten bei Ausführung am Server automatisch in die vorgesehenen temporären Ordner gespeichert [ESRI_1]. Beim Verwenden von Python kann der Workspace aus der Variablen „*env.scratchWorkspace*“ abgerufen und anschließend verwendet werden.
Da nicht alle Datentypen, wie Feature Klassen, vom ArcGIS Server unterstützt werden, sollten auch Ein- und Ausgabedatentypen überprüft werden [ebd.]. Ausgabedaten werden in einem speziellen Ausgabeordner abgelegt (standardmäßig: „*arcgisoutput**Werkzeugname*\“), sofern kein spezieller Ausgabeordner bei der Werkzeugerstellung angegeben wurde.

2.4.2 Veröffentlichung der Dienste

Die Veröffentlichung der entsprechend vorbereiteten Dienste kann auf drei Arten erfolgen [ESRI_1]:

- Als reiner Geoprocessing Service
- Als Geoprocessing Service mit verbundenem Quellkartendokument
- Als Geoprocessing Service mit verbundenem Ergebniskartendienst

Reine Geoprocessing Dienste geben die Ausgabedaten direkt an den anfragenden Klienten zurück. Ist ein Quellkartendokument mit dem Geoprocessing Service verbunden so hat dieser zusätzlich die Möglichkeit, auf sämtliche darin enthaltenen Daten zuzugreifen. Auch hier erfolgt die Rückgabe der Ergebnisse direkt an den Klienten. Ist ein Ergebniskartendienst mit dem geoprocessing service verbunden, so können Ausgaben auch direkt an diesen Kartendienst geschickt und dort weiterverarbeitet werden [ebd.].

Für die Veröffentlichung werden Administratorenrechte auf den ArcGIS Server benötigt. Sind diese vorhanden, kann die Veröffentlichung unter Verwendung von ArcCatalog zum einen durch Rechtsklick auf das entsprechende Werkzeug und Auswahl von „*Publish to ArcGIS Server*“ erfolgen. Zum anderen kann die Veröffentlichung durch die Auswahl der Funktion „*Add New Service*“, durch Rechtsklick auf den gewünschten Server, durchgeführt werden. Dies ermöglicht ein sofortiges Konfigurieren des Dienstes, noch vor der Veröffentlichung. Andernfalls muss der Dienst erst angehalten werden um die Konfiguration zu ändern.

2.4.3 Konfiguration des veröffentlichten Dienstes

Ein veröffentlichter Dienst kann sowohl synchron als auch asynchron ausgeführt werden. Eine Synchrone Ausführung bedeutet hier, dass der Client wartet, bis der Server den Dienst ausgeführt hat, da keine Daten auf dem Server gespeichert werden. In dieser Zeit kann der Client keine weiteren Arbeiten durchführen. Daher sollte eine synchrone Ausführung nur für Dienste verwendet werden, die relativ schnell abgearbeitet werden können. Bei der asynchronen Ausführung kann der Client andere Arbeiten ausführen, während er auf das Ergebnis des Servers wartet. In diesem Fall speichert der Server die Ergebnisse unter Verwendung einer sogenannten „*Job ID*“ ab. Die Daten müssen dann durch expliziten Aufruf des Klienten abgerufen werden. Die asynchrone Ausführung ist minimal langsamer als die synchrone, daher sollte diese Funktion für Dienste erst gewählt werden, bei denen die Ausführung mehr als wenige Sekunden in Anspruch nimmt.
Soll ein reiner Geoprocessing Service ohne angehängtes Kartendokument veröffentlicht werden, ist die Option „*A toolbox*“ zu wählen, andernfalls „*A map*“, mit Eingabe der entsprechenden Pfade. Sollen andere als die standardmäßig eingestellten Verzeichnisse verwendet werden, so können diese im Folgenden definiert werden. Zu beachten ist hierbei, dass der ArcGIS Server über Zugriffsrechte auf die entsprechenden Verzeichnisse verfügt.
Sollen Einstellungen nachträglich geändert werden, ist darauf zu achten, dass der Dienst zuerst angehalten werden muss. Dann können Änderungen mittels ArcCatalog oder auch dem ArcGIS Server Manager erfolgen.

2.5 ESRI Web APIs

Wie bereits kurz erläutert, bietet ESRI für den Zugriff auf mittels ArcGIS Server veröffentlichte Dienste verschiedene Web-APIs (Application Programming Interfaces) an [ESRI_12]. Damit besteht die Möglichkeit, Geoprocessing Dienste leicht in die meisten bestehenden Webarchitekturen einzubinden. Neben Unterstützung für Javascript werden auch Schnittstellen für Adobe Flex sowie Microsoft Silverlight angeboten. Neben diesen Web-APIs bestand bisher die Möglichkeit, mittels Web Application Development Frameworks (Web-ADF) für Java und .NET zu arbeiten. Ab der Version 10.1

wird ESRI jedoch die Unterstützung für die Web-ADFs einstellen. Daher wird von ESRI empfohlen, verstärkt die angebotenen Web-APIs zu nutzen [ESRI_13].
Alle APIs haben gemein, dass sie relativ einfache Möglichkeiten bieten, um auf veröffentlichte Dienste im ESRI-Umfeld zuzugreifen. Ein Einsatz der APIs für Javascript oder Flex an der LUBW würde unter Umständen einen hohen Einarbeitungsaufwand bedeuten. Um die bisher an der LUBW im RIPS-Umfeld eingesetzte .NET Web-ADF Lösung zu ersetzen bietet sich, auf Grund des vorhandenen Wissens in der .NET Entwicklung, der Einsatz der Silverlight-API an.

2.6 Microsoft Silverlight und die Silverlight API

Microsoft Silverlight wurde ursprünglich entwickelt, um als Alternative zu Adobe Flash zu fungieren. Die erste Version, noch auf Basis von Javascript, erschien 2007, die neueste Version 5 erschien am 9. Dezember 2011 [MS_2]. Silverlight bietet, insbesondere für .NET Entwickler, vielfältige Möglichkeiten, Rich Internet Applications (RIA) zu entwickeln. Bei RIAs handelt es sich um Webanwendungen, die Desktopanwendungen vergleichbare Funktionalitäten bieten. Die Ausführung einer RIA erfolgt in der Regel im jeweiligen Webbrowser, ohne die Installation von clientseitiger Software nötig zu machen.
Silverlight ist browser- und plattformunabhängig [MS_3], wobei jedoch die Unterstützung beispielsweise für den Opera-Browser eingeschränkt ist [MS_1]. Die Implementierung für Linux Systeme wird unter der Bezeichnung Moonlight entwickelt. Neben der Entwicklung von Webanwendungen können mit Silverlight auch sogenannte „*out-of-browser*“ Desktopanwendungen entwickelt werden [MS_3], welche als lokale Anwendung ohne Internetzugriff ausführbar sind. ESRI bietet seine Silverlight-API seit 2009 und der Version 2 von Silverlight an [Haddad_2009]. Mit Hilfe der API können ArcGIS Server Dienste, wie Karten, Geoprocessing-Werkzeuge oder Routingdienste in Webanwendungen integriert werden. Gerade im Bereich der Geoverarbeitung existiert somit die Möglichkeit, mit Hilfe von Modellen oder Skripten erstellte Werkzeuge, nach deren Einbindung in einen ArcGIS Server, vielen Webanwendungen zur Verfügung zu stellen. Ein Vorteil dieser Architektur ist die Tatsache, dass einem Silverlight Entwickler, gerade bei komplexen Abläufen, die genaue Umsetzung innerhalb eines Werkzeuges nicht bekannt sein muss. Vielmehr reicht es für die Implementierung als Web Geoprocessing Dienst aus, die entsprechenden Ein- und Ausgabeparameter zu kennen. Dadurch böte sich bei einem Einsatz von Silverlight die Möglichkeit, die Entwicklung und Implementierung von neuen Geoprocessing Diensten zu beschleunigen. ESRI stellt im Rahmen der Silverlight API verschiedene Ressourcen zur Verfügung, neben den Silverlight Kernelementen auch Karten, Tasks (Aufgaben), Grafiken sowie ein spezielles Toolkit um die Anwendungsentwicklung zu beschleunigen [ESRI_2].
Karten können zum einen von einem ArcGIS Server stammen, zum anderen auch Microsoft Bing Karten sein. Unterstützt werden verschiedene Projektionen, sowie dynamische und gecachete Kartendienste. Durch den Einsatz von Grafikelementen besteht die Möglichkeit, dem Benutzer direkt graphische Eingaben zu ermöglichen, welche bei Bedarf gespeichert oder weiterverarbeitet werden können, beispielsweise in einem Geoprocessing Werkzeug. Unter Tasks versteht ESRI unter anderem Abfragen auf Datenbanken, oder auch Kartenlayer. Ebenso werden Geoprocessing Services, Routing Services oder auch einfache geometrische Operationen, welche ESRI in einem sogenannten Geometry Service zusammenfasst, darunter verstanden [ebd.]. Beim sogenannten Toolkit handelt es sich um eine von ESRI entwickelte Sammlung häufig benötigter Werkzeuge, wie Editierfunktionen oder die Navigation in einer Bildschirmkarte. Das Toolkit kann bei Bedarf installiert werden und steht anschließend in der Silverlight Anwendung zur Verfügung.

2.7 Anbindung der Silverlight API über REST

Die Anbindung der von einem ArcGIS Server bereitgestellten Dienste erfolgt über die beschriebene REST API. Dies gilt sowohl für Kartendienste als auch für die sogenannten Tasks, unter die auch das Geoprocessing fällt. Zum Einbinden in die Anwendung wird lediglich die REST URL des gewünschten Dienstes benötigt [ESRI_4]. Über das Services Verzeichnis des REST Dienstes (in der Regel: http://<host>/rest/services) können alle benötigten Informationen abgerufen werden. Zum Einbinden eines Kartendienstes beispielsweise wird eine URL in der Form: http://<host>/ArcGIS/rest/services/_dienst_/MapServer verwendet. Soll ein bestimmter Layer innerhalb der Karte verwendet werden, so lässt sich dieser über seinen Index im Kartendokument ansprechen (…/MapServer/0).

Geoprocessing Services werden analog eingebunden:
http://<host>/ArcGIS/rest/services/_dienst_/GPServer/
Zu beachten hierbei ist, dass das zu verwendende Werkzeug angegeben wird („*…/GPServer/Werkzeug*“).
Sind die REST URL der gewünschten Dienste bekannt, kann die eigentliche Implementation in Silverlight erfolgen. Im Folgenden soll der allgemeine Ablauf am Beispiel eines Geoprocessing Dienstes erläutert werden.

2.8 Einbinden eines Geoprocessing Service über REST

Die Verwendung aller Dienste erfolgt in Silverlight nach dem gleichen Schema, trotz teilweise unterschiedlicher Funktionalität [ESRI_3]. Nachfolgend wird ein einfacher Geoprocessing Dienst über die REST Schnittstelle in eine Silverlight Applikation eingebunden. Zunächst muss ein Task Objekt für den Geoprocessing Dienst erstellt werden. In C# geschieht dies durch folgenden Aufruf:

```
geoprocessorTask = new
Geoprocessor("http://<host>/ArcGIS/rest/services/_dienst_/GPServer/Werkzeug");
```

Es muss lediglich dem Konstruktor der Klasse die REST URL des zu verwendenden Dienstes übergeben werden, um auf diesen zugreifen zu können. Als nächstes müssen die benötigten Parameter an das so instanziierte Objekt übergeben werden. Im Falle eines Geoprocessing Dienstes handelt es sich um ein Geoprocessing Listenelement:

```
List<GPParameter> parameters = new List<GPParameter>();
```

Das so erzeugte Objekt muss nun gefüllt werden, in diesem Falle mit zwei Geometrien:

```
parameters.Add(new GPFeatureRecordSetLayer("Input_Rad", args.Geometry));
parameters.Add(new GPFeatureRecordSetLayer("Input_Kreis", Features[0].Geometry));
```

Damit kann der Geoprocessing Service prinzipiell bereits ausgeführt werden. Bei Geoprocessing Tasks tritt hier der oben erwähnte Sonderfall auf, das die Aufgaben sowohl synchron als auch asynchron vom Server bearbeitet werden können. Daher gibt es bei einem Geoprocessing Objekt auch zwei Möglichkeiten des Aufrufs:

```
geoprocessorTask.ExecuteAsync(parameters);
geoprocessorTask.SubmitJobAsync(parameters);
```

Hierbei bewirkt “*ExecuteAsync*” (trotz des Namens der Funktion) eine synchrone Ausführung, “*SubmitJobAsync*” eine asynchrone. Um die Ergebnisse des Dienstes abfragen zu können muss nun

lediglich eine entsprechende Ereignisbehandlung implementiert werden, entweder für synchrone oder asynchrone Dienste. Um zu überprüfen, ob der Dienst korrekt ausgeführt wird sollte auch eine Fehlerbehandlung verwendet werden:

```
geoprocessorTask.JobCompleted += GeoprocessorTask_JobCompleted;
geoprocessorTask.ExecuteCompleted += GeoprocessorTask_ExecuteCompleted;
geoprocessorTask.Failed += GeoprocessorTask_Failed;
```

In den Eventhandlern können die Ergebnisse nun verarbeitet werden, beispielsweise am Bildschirm angezeigt oder an einen weiteren Geoprocessing Service weitergeleitet werden:

```
private void GeoprocessorTask_ExecuteCompleted(object sender,
GPExecuteCompleteEventArgs e)
        {
            // Weiterverarbeitung der Ergebnisse
        }
private void GeoprocessorTask_JobCompleted(object sender, JobInfoEventArgs e)
        {
            // Weiterverarbeitung der Ergebnisse
        }
private void GeoprocessorTask_Failed(object sender, TaskFailedEventArgs e)
        {
            lbMsg.Content = e.Error.Message;
        }
```

Sollen andere Dienste mittels der Silverlight API konsumiert werden, so erfolgt die Umsetzung auf ähnliche Weise. Durch Weitergabe der REST URL an den jeweiligen Konstruktor, Übergabe der eventuell benötigten Parameter, Aufrufen der entsprechenden Funktionen und Weiterverarbeitung der Ergebnisse.
Der Vorteil des Zugriffs mittels einer REST URL auf ArcGIS Server Dienste besteht darin, dass diese innerhalb einer Anwendung beliebig miteinander kombiniert werden können, so lange die URL bekannt ist. So können beispielsweise Kartendaten, welche mit einem Kartendienst bereitgestellt werden, als Eingabeparameter für einen Geoprocessing Dienst verwendet werden. Dessen Ausgabe wiederrum könnte als Parameter für einen Routing Dienst Verwendung finden, etc. Genauso ist es möglich, mehrere verschiedene Dienste innerhalb derselben Anwendung parallel dem Nutzer zugänglich zu machen. Beispielsweise das Überlagern von Karteninhalten, welche auf verschiedenen Servern bereitgestellt werden.

2.9 WPS

Die Anbindung von Fachanwendungen im Rips Umfeld erfolgt unter anderem über WPS (Web Processing Service). Bei WPS handelt es sich um einen vom OGC (Open Geospatial Consortium) definierten Standard zum Veröffentlichen von, und den Zugriff auf Geoverarbeitungsdienste. Die aktuell gültige Version 1.0.0 datiert vom Mai 2007. Die Schnittstellenspezifikation von WPS verfügt über Mechanismen, die benötigten georeferenzierten Daten zu identifizieren, die Berechnungen durchzuführen, sowie das Ergebnis dem Klienten zur Verfügung zu stellen [WPS_1, S. xiii]. Darüber hinaus wird es dem Dienstanbieter ermöglicht, dem Klienten einen über das Internet erreichbaren Prozess anzubieten, ohne dass der Klient über die im Hintergrund ablaufenden Operationen informiert sein muss [ebd.]. Dadurch wird gewährleistet, dass Klient und Prozess über eine standardisierte Schnittstelle miteinander kommunizieren können.
Die WPS Schnittstelle definiert drei Operationen, die ein konformer Webdienst anbieten muss:

- GetCapabilities
- DescribeProcess
- Execute

[ebd., S. 4]

Mit der „*GetCapabilities*“ Operation kann der Klient Metadaten über die angebotenen Dienste des WPS konformen Servers abfragen. Als Antwort wird in der Spezifikation ein XML Dokument gefordert, welches Metadaten über den Server sowie über alle von diesem angebotenen Dienste enthält [ebd., S.11]. Wie alle WPS Anfragen besteht auch die „*GetCapabilities*“ Anfrage aus Schlüssel-Wert Paaren, welche mittels HTTP GET an den Server übermittelt werden. Verpflichtend für „*GetCapabilities*“ sind lediglich zwei Parameter, „*service*“ und „*Request*“. Der „*service*“ Parameter wird bei einer WPS Frage immer auf den Wert „*WPS*“ gesetzt, der „*Request*“ ist in diesem Fall „*GetCapabilities*“. Somit weißt eine konforme Anfrage folgende Form auf:

```
http://www.rips.de/rips?service=WPS&Request=GetCapabilities
```

Die „*DescribeProcess*“ Operation ermöglicht das Abfragen der vollständigen Informationen über einen bestimmten Prozess, inklusive den Ein- und Ausgabeparametern [ebd., S. 16]. Ergänzend zu den bei der „*GetCapabilities*“ Operation benötigten Parametern müssen weitere angegeben werden. Neben „*service*“ und „*Request*“ werden „*version*“ und „*Identifier*“ benötigt. Damit kann der Server die Anfrage einem oder mehreren bestimmten Diensten zuordnen und die gewünschten Informationen zur Verfügung stellen. Eine WPS konforme „*DescribeProcess*“ Anfrage lautet:

```
http://www.rips.de/rips?service=WPS&Request=DescribeProcess&version=1.0.0&Identifier=Clip
```

Die Antwort auf die „*DescribeProcess*“ Anfrage muss ebenso in einer XML codierten Datei erfolgen. Diese Datei enthält neben den auch bei der „*GetCapabilities*“ Anfrage zurückgelieferten Werten wie Identifikator, Titel und Kurzbeschreibung auch Informationen über die Ein- und Ausgabeparameter. Beide Arten von Parametern können innerhalb der WPS Spezifikation aus drei verschiedenen Arten von Daten bestehen. Sie können vom Typ „*ComplexData*“ sein, beispielsweise in GML (Geographic Markup Language) codierte Geometrien. Bei „*LiteralData*“ handelt es sich um einfache Datentypen wie Zahlen oder Buchstabenfolgen. Der dritte Datentyp, „*BoundingBoxData*“, definiert sich als die eine Gebietsumrandung festlegende „*Bounding Box*“, welche in bestimmten, unterstützten Koordinatensystemen vorliegt.

Die „*Execute*“ Operation schließlich dient dem eigentlichen Ausführen eines bestimmten Geoprocessing Dienstes auf dem Server. Zum Ausführen müssen die Eingabeparameter in der definierten Form übergeben werden, entweder direkt in der Anfrage oder unter Angabe eines über das Internet erreichbaren Pfades. Werden die Ausgaben auf dem Server gespeichert und dort vorgehalten, so wird als Antwort eine auf die entsprechenden Daten verweisende URL übergeben. Alternativ können einzelne Datensätze auch direkt in einem XML Dokument zurückgegeben werden [ebd., S.30]. Die Parameter für den „*Execute*“ Aufruf sind dieselben wie beim „*DescribeProcess*“ Aufruf. Ergänzt werden sie durch die Angabe der Eingabedaten („*DataInputs*“):

```
http://www.rips.de/rips?service=WPS&Request=Execute&version=1.0.0&Identifier=Clip&DataInput
s=inputs
```

3 Beispielhafte Umsetzung eines Geoprocessing Service

Im Folgenden soll die beispielhafte Umsetzung eines Geoprocessing Service unter Einsatz der beschriebenen Technologien aufgezeigt werden.
Im Rahmen einer Webanwendung soll es Anwendern ermöglicht werden, Radwege für den von Ihnen betreuten Landkreis zu digitalisieren. Beim Erfassen der Radwege sollen die Benutzer jedoch nur Wege innerhalb ihres Kreisgebietes erfassen können. Daher ist es erforderlich, eventuell über die Kreisgrenze hinausgehende Erfassungen abzuschneiden. Als eine Möglichkeit, dieses Problem zu lösen, bietet sich der Einsatz eines Geoprocessing Service an. In einem ArcGIS Modell könnte die vom Benutzer digitalisierte Linie mit der in der Datenbank hinterlegten Kreisgrenze verschnitten werden. Dadurch wäre die Digitalisierung ausschließlich innerhalb des Kreisgebietes sichergestellt. Die Umsetzung soll in Microsoft Silverlight erfolgen, um das Ergebnis später einfach in die übrige, sich in der Entwicklung befindliche, Anwendung integrieren zu können.
Die Anforderungen an die Anwendung sind somit:

- Möglichkeit, Radwege über eine Weboberfläche zu erfassen
- Auswahlmöglichkeit und Anzeige für entsprechende Landkreise
- Verschneiden der erfassten Radwege mit den Kreisgrenzen
- Rückgabe der verschnittenen Linie

3.1 Erstellung des Modells in ArcGIS

Ein Modell im Rahmen von ArcGIS stellt eine grafische interaktive Modellierungsumgebung für Prozessketten dar [ESRI_9]. Innerhalb eines Modells können sowohl in ArcGIS integrierte Werkzeuge als auch eigene, in Python erstellte, Skripte verwendet werden. Es ist auch möglich, Modelle innerhalb anderer Modelle zu verwenden. Die einzelnen Bausteine können in „Flow-Chart Manier“ abgebildet und miteinander verknüpft werden [ebd.].
Für die zu erstellende Anwendung wird lediglich ein relativ einfaches Modell, bestehend aus zwei Eingabeparametern sowie einem Werkzeug, benötigt. Zu beachten ist hierbei, dass bei einer Verwendung des Modells mit einem ArcGIS Server nicht alle in ArcGIS vorhandenen Datentypen verwendet werden können. Bei Bedarf müssen geeignete Ersatztypen gewählt werden (vgl. [ESRI_1]).
Als ersten Schritt wird zunächst eine neue Toolbox (Werkzeugkiste) erstellt. Anschließend wird in dieser Toolbox ein Modell erstellt. Nun kann unter Verwendung des „Model Builder“ der gewünschte Workflow abgebildet werden. In diesem Falle sollen zwei Eingabedatensätze miteinander verschnitten werden, und zwar die linienhaft vorliegenden Radwege mit den flächenhaft vorliegenden Kreisen. Geeignet, die gewünschte Operation durchzuführen, ist das Clip-Werkzeug in ArcGIS. Als „Input Feature“ werden die Radwege ausgewählt, als „Clip Feature“ die Kreise. Abbildung 4 zeigt das fertige Modell.

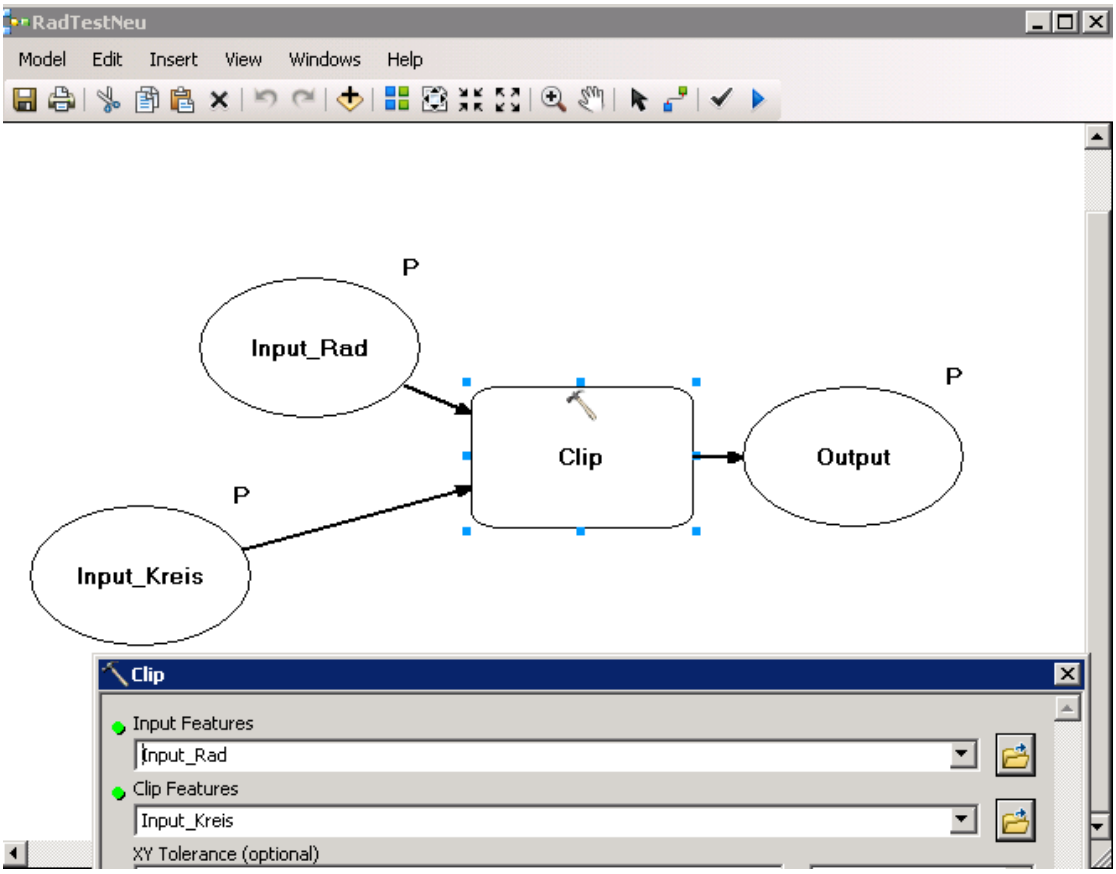

Abbildung 4: Modell der Radwegeanwendung

Ein großer Vorteil beim Einsatz des Model Builder stellt seine Übersichtlichkeit dar, wie sich aus der Abbildung gut erkennen lässt. Auch, und gerade, bei komplexeren Modellen kann durch die graphische Repräsentation der auszuführenden Geoverarbeitungsschritte eine bessere Übersicht und einfachere Kontrolle gewährleistet werden.

Ist das Modell erstellt, müssen nun, wie bereits erwähnt, die Eingabeparameter für den Einsatz mittels ArcGIS Server angepasst werden. Beim Einsatz in einer reinen Desktopumgebung stellen beide Eingabeparameter sogenannte „Feature Layer“ dar. Diese sind jedoch nur bedingt geeignet für den Servereinsatz (vgl. [ESRI_1]). Daher müssen beide Eingaben im Parameterdialog des Modells (Abb. 5) als „Feature Set“ gesetzt werden.

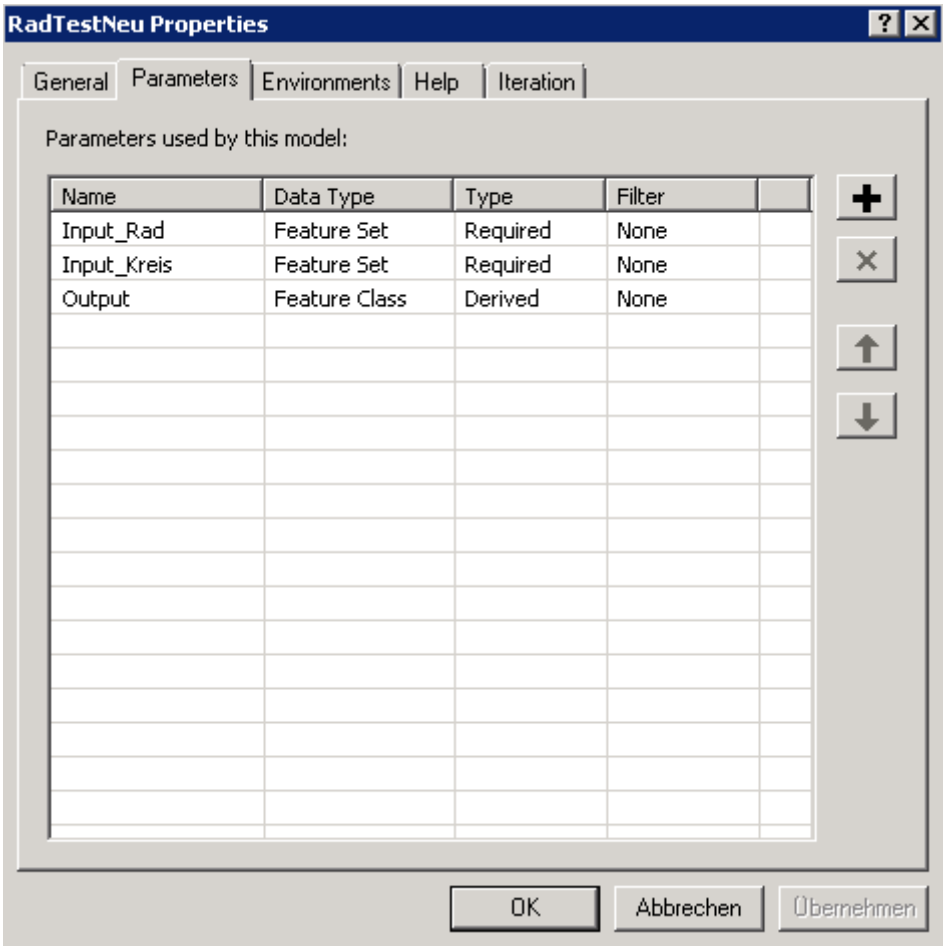

Abbildung 5: Parameterdialog

Der Einsatz von Feature Layern wäre ebenfalls möglich, würde jedoch bedeuten, dass die Eingabe der Parameter lediglich als String-Datentyp erfolgen könnte. In diesem String könnten nur Pfade zu vom Server erreichbare (lokale) Objekte hinterlegt werden. Da bei dieser Anwendung die Eingabe jedoch durch den Benutzer über eine Weboberfläche erfolgen soll, können Feature Layer in diesem Fall nicht verwendet werden. Das fertige Werkzeug kann nun veröffentlicht werden.

3.2 Veröffentlichen des Modells mit ArcGIS Server

Ist der gewünschte Geoprocessing Workflow in einem Modell oder Skript abgebildet, kann die entsprechende Toolbox am ArcGIS Server veröffentlicht werden. Wie bei der Modellerstellung beschrieben, sollten gültige Datentypen verwendet werden. Ist dies der Fall, so erfolgt die eigentliche Veröffentlichung wahlweise mit Hilfe von ArcCatalog oder direkt aus ArcMap heraus. In beiden Fällen kann durch einfaches Rechtsklicken die Veröffentlichung durchgeführt werden. Im sich öffnenden Dialog (Abb. 6) muss lediglich der Server, auf dem die Toolbox veröffentlicht werden soll, ausgewählt werden.

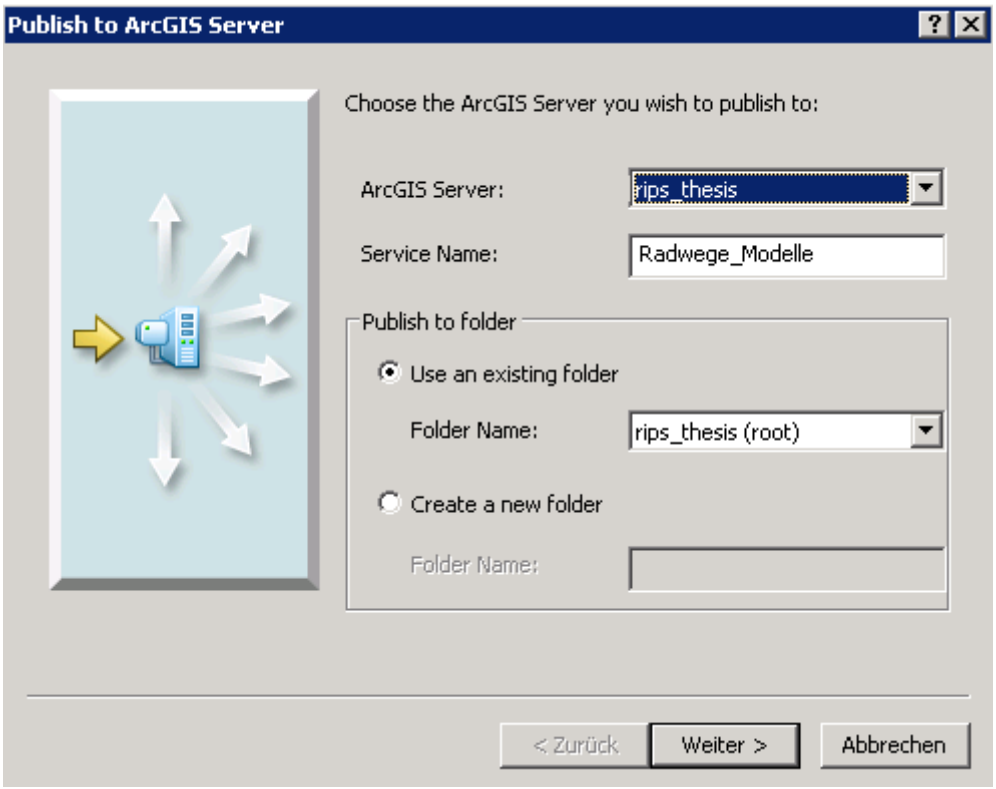

Abbildung 6: Veröffentlichen des Models

Ist das Modell veröffentlicht, können nun weitere Einstellungen vorgenommen werden. Dies lässt sich am Besten im ArcCatalog durchführen. Auch der Einsatz des ArcGIS Server Manager ist möglich. Wird der gewünschte Server ausgewählt, werden alle auf diesem Server laufenden Dienste angezeigt. Durch Rechtsklick (oder Auswahl im ArcGIS Server Manager) erhält man Zugriff auf die Konfigurationsoptionen. Zu beachten ist hierbei, dass der Dienst angehalten werden muss, um Änderungen vornehmen zu können. Die wichtigsten Einstellungsmöglichkeiten finden sich im Reiter „parameters“ (Abb. 7).

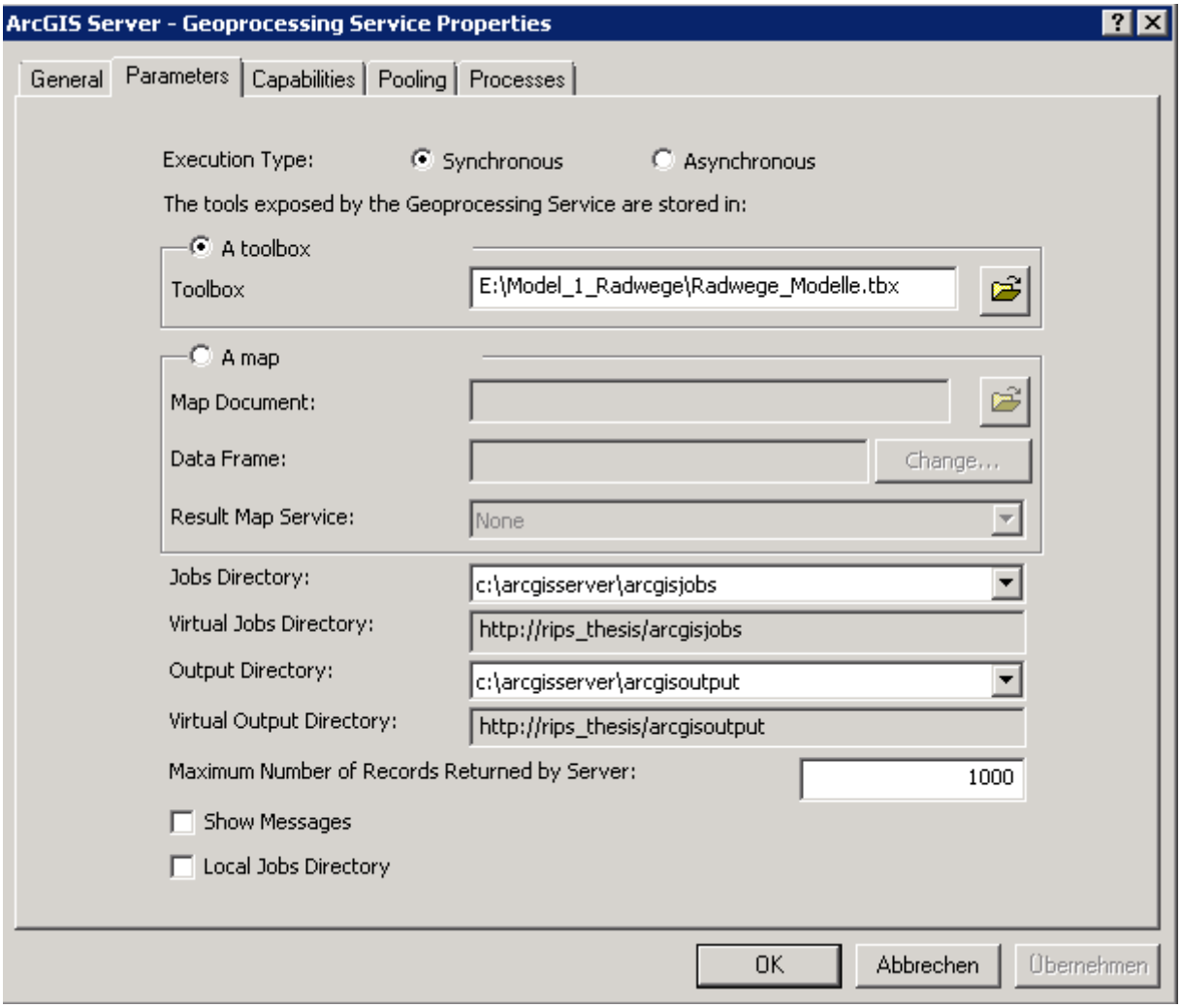

Abbildung 7: Eigenschaften des Geoprocessing Dienstes - Parameter

Die Ausführungsart („*Execution Type*“) ist die erste Wahl, die getroffen werden muss. Im Rahmen von Geoprocessing Services verwendet ESRI die Bezeichnungen Synchron und Asynchron (vgl. 2.5.3). Bei synchroner Ausführung wartet der Klient, bis er eine Antwort vom Server erhält. Es werden keine Daten auf dem Server zwischengespeichert. Bei Asynchroner Ausführung wartet der Klient nicht auf den Server. Er kann, während der Server mit der Berechnung beschäftigt ist, andere Aktionen ausführen. Der Server speichert die Ergebnisdaten lokal ab, sobald der Geoprocessing Dienst seine Arbeit beendet hat. Der Klient muss nun explizit beim Server nachfragen um die Ergebnisse zu erhalten. ESRI empfiehlt, ab einer Bearbeitungsdauer von etwa drei Sekunden einen asynchronen Dienst zu verwenden. Bei kürzeren Laufzeiten kann ein synchroner Dienst verwendet werden (vgl. [ESRI_1]). Im Falle der Radwegeanwendung wird das Modell relativ zügig ausgeführt, im Zeitrahmen von unter einer Sekunde. Daher bietet sich hier der Einsatz eines synchronen Dienstes an. Die zweite, zu treffende Einstellung definiert, ob es sich beim veröffentlichten Dienst um in einer Toolbox hinterlegte Werkzeuge handelt, oder in ein Kartendokument eingebundene Werkzeuge(vgl. 2.5.3). In diesem Fall wird das Werkzeug mit Hilfe einer Toolbox veröffentlicht, daher ist die Option „*a toolbox*“ zu wählen. Darunter befinden sich im Parameterdialog Eingabemöglichkeiten für die Auftrags- und Ausgabeverzeichnisse („*Jobs*“ und „*Output*“). Standardmäßig sind hier die bei der Installation des ArcGIS Server gewählten Verzeichnisse voreingestellt. Das „*Jobs*“-Verzeichnis ist der Ort, an dem der ArcGIS Server die Zwischen- und Endergebnisse eines asynchronen Geoprocessing Dienstes hinterlegt. Jeder Auftrag wird mit einer einzigartigen ID versehen. Unter dieser ID legt der Server

automatisch die benötigten Verzeichnisse zum Speichern der Daten an. Der Client kann dann nach Beendigung des Dienstes, unter Verwendung der ID, auf die Ergebnisse zugreifen. Beim Ausgabeverzeichnis handelt es sich um den Ort, an dem temporär vom Server benötigte Daten gespeichert werden. Bei Geoprocessing Services wird zwingend ein solches Verzeichnis benötigt [ESRI_8]. Weitere Einstellungsmöglichkeiten existieren. So kann die maximale Anzahl der zurückgegebenen Ergebnisse festgelegt werden. Dies ist insbesondere interessant, wenn die vom Server verwendete Bandbreite kontrolliert werden soll. Die Option „*show messages*“ kann verwendet werden, um Benutzern die vom Server erzeugten Nachrichten zur Laufzeit anzuzeigen. Da diese Nachrichten teilweise Pfade und Verweise auf Daten enthalten, können, wenn diese Informationen nicht anzeigt werden sollen, die Nachrichten ausgeblendet werden [ebd.]. Im „*capabilities*“ Reiter (Abb. 8) können die vom jeweiligen Dienst anzubietenden Möglichkeiten ausgewählt werden. Im Falle eines Geoprocessing Service besteht hier lediglich die Auswahlmöglichkeit: „*Geoprocessing*“. Bei einem Kartendienst beispielsweise könnte hier auch die KML Schnittstelle aktiviert werden, etc.

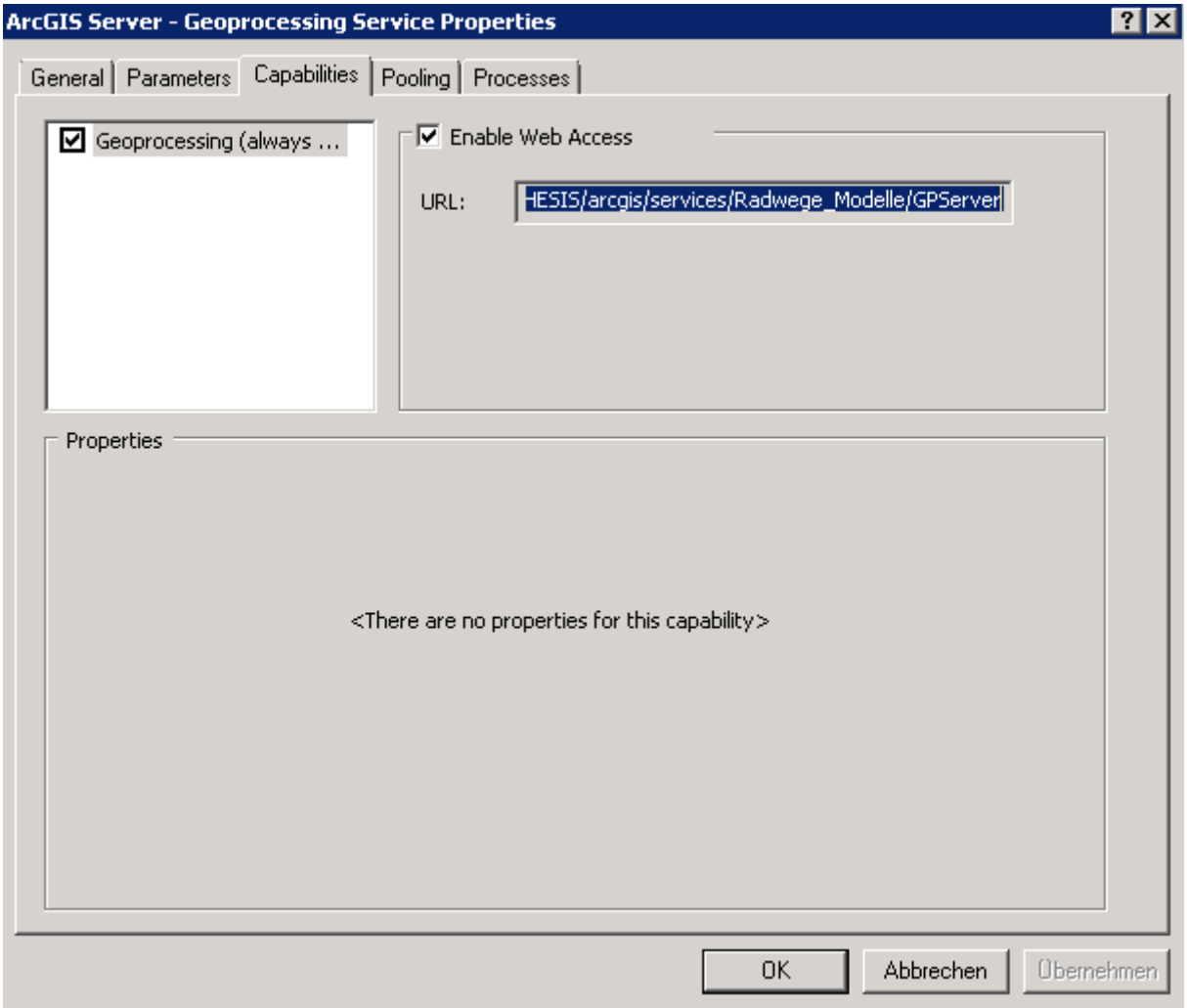

Abbildung 8: Eigenschaften des Geoprocessing Dienstes - Capabilities

Im „*pooling*“ Reiter kann eingestellt werden, ob der Dienst „*pooled*“ oder „*unpooled*“ betrieben werden soll. „*Pooled*“ bedeutet hier, dass der Dienst vielen Benutzern zur Verfügung gestellt werden soll, welche jeweils eine zufällige Verbindung zum Dienst erhalten. Vorteil hierbei ist eine höhere Geschwindigkeit bei der Ausführung des Dienstes. Ein „*unpooled*“ angebotener Dienst ermöglicht das

Editieren von Daten am Server. Er ist jedoch langsamer, insbesondere da für jeden anfragenden Klienten eine spezielle Verbindung offengehalten wird, über die er auf den Dienst zugreifen kann [ESRI_10]. Wird ein „*unpooled*“ –Dienst verwendet, so kann hier auch die Anzahl der minimal und maximal anzubietenden Verbindungen eingestellt werden.
Im Rahmen dieser Anwendung wird die empfohlene Einstellung, „*pooled*“, verwendet. Im „*processes*“ Reiter schließlich besteht die Möglichkeit, Dienste isoliert oder geteilt anzubieten. Im Falle einer isolierten Ausführung wird jede Instanz des Dienstes in einem isolierten Thread ausgeführt. Die Möglichkeit Threads zwischen verschiedenen Instanzen zu teilen besteht auch, jedoch kommt es hier bei einem Threadausfall zum Ausfall aller damit verbundenen Instanzen des Dienstes. ESRI empfiehlt die isolierte Ausführung [ebd.]. Letztlich kann hier auch ein sogenanntes „*Recycling*“ eingestellt werden. Dadurch werden im gewählten Intervall die Instanzen eines „*pooled*“ ausgeführten Dienstes neu erstellt. Dadurch wird die fortlaufende Benutzbarkeit des Dienstes sichergestellt [ebd.]. Beim Radwegedienst werden hier eine hohe Isolation, sowie ein Recycling alle 24 Stunden ausgewählt.

3.3 Erstellung einer Silverlight Anwendung

Ist der Geoprocessing Service nun erstellt und veröffentlicht steht er über die REST Schnittstelle allen anfragenden Applikationen zur Verfügung. Wie bereits erwähnt soll die beispielhafte Umsetzung einer solchen Anbindung unter Verwendung von Microsoft Silverlight erfolgen. Als Entwicklungsumgebung kommt Microsoft Visual Studio 2010 zum Einsatz. Um einen einfachen Zugriff auf die veröffentlichten Dienste zu erhalten, muss zunächst die ESRI API für Silverlight installiert werden. Diese Schnittstelle bietet eine Vielzahl an Klassen, Methoden und Funktionen um ArcGIS Server Funktionalitäten in Silverlight integrieren zu können. Nach der Installation kann in Visual Studio ein neues Silverlight Projekt erstellt werden (Abb. 9). In einer Silverlight Anwendung wird getrennt nach Design-Teil (beschrieben mit XAML, „*Extensible Application Markup Language*“) und Code-Teil, worin der eigentliche Programmcode, in der gewählten Programmiersprache, ausgeführt wird.

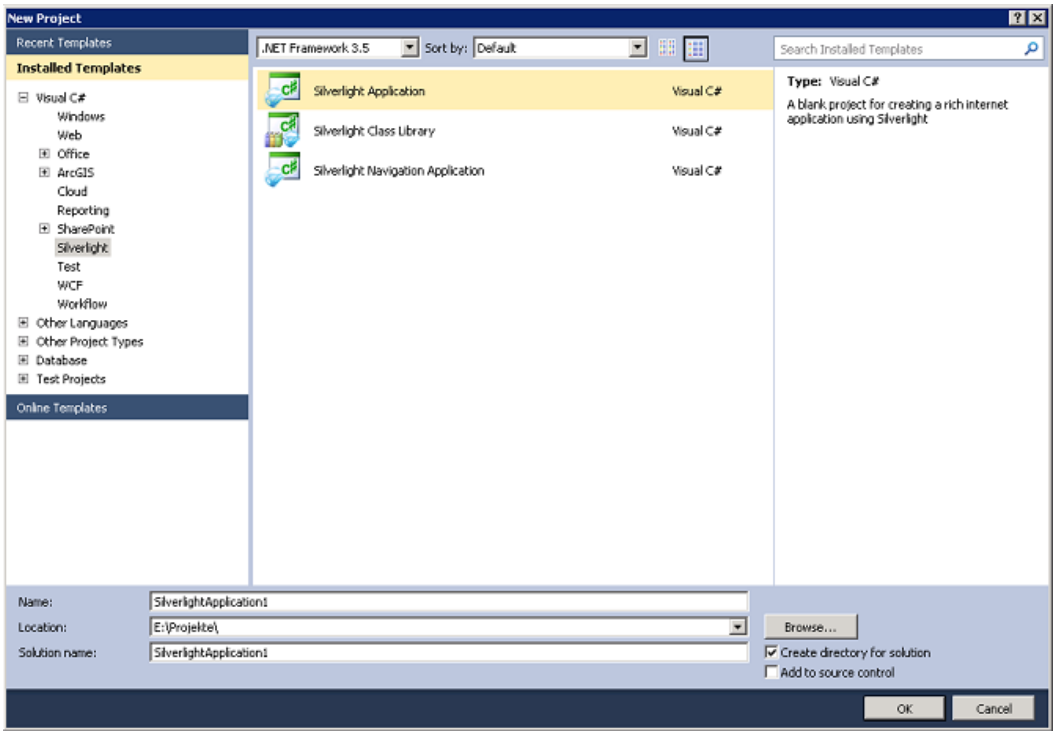

Abbildung 9: Neue Silverlight Anwendung

3.3.1 Design der Oberfläche

Im XAML-Teil der Anwendung muss nun zunächst ein Namensraumverweis auf das ArcGIS Silverlight Schema eingefügt werden:

```
xmlns:esri="http://schemas.esri.com/arcgis/client/2009"
```

Nun ist es möglich, ein Kartensteuerelement zur Anwendung hinzuzufügen:

```
<esri:Map x:Name="MyMap"> </esri:Map>
```

Innerhalb des Kartenelements können nun beliebige Layer-Elemente hinzugefügt werden. Hierbei sollte darauf geachtet werden, dass die Reihenfolge beim Hinzufügen der späteren Anzeigereihenfolge entspricht. Das heißt, das letzte Element wird als oberste Ebene in der Karte angezeigt. Der RIPS Kartendienst ist zweigeteilt und stellt daher eigentlich zwei getrennte Map Services dar, daher müssen nun beide Teile eingebunden werden:

```
<esri:ArcGISTiledMapServiceLayer ID="Hintergrund"
Url="http://193.197.158.219/ArcGIS/rest/services/rips_hintergrund_exploded/MapServer"
/>
<esri:ArcGISTiledMapServiceLayer ID="ALK"
Url="http://193.197.158.219/ArcGIS/rest/services/rips_alk_exploded/MapServer" />
```

Die Einbindung erfolgt einfach als *„ArcGISTiledMapServiceLayer“* mit Angabe der URL, unter der der gewünschte Kartendienst erreichbar ist.
Neben den als Hintergrundkarten fungierenden RIPS Diensten müssen auch zwei sogenannte Grafikebenen (Graphics Layer) eingefügt werden, in der später die Anzeige der Kreise sowie die Digitalisierung der Radwege am Bildschirm erfolgen kann. Da diese Layer als Letzte hinzugefügt werden, wird sichergestellt, dass die Karten nicht die eingezeichneten Radwege überlagern:

```
<esri:GraphicsLayer ID="MyKreisLayer" />
<esri:GraphicsLayer ID="MyGraphicsLayer" />
```

Weitere Elemente, welche für diese Anwendung benötigt werden, müssen nun ebenfalls in das Seitenlayout eingebunden werden. In diesem Fall handelt es sich lediglich um ein *„ComboBox“* Element sowie einen Button. In der *„ComboBox“* soll später mittels Drop-Down Auswahl der Kreis des Bearbeiters gewählt werden können. Der Button soll nach dem Anklicken den Editiermodus starten. Die *„ComboBox“* wird wie folgt eingefügt:

```
<ComboBox Height="23" Name="cbKreise" Width="180"
SelectionChanged="cbKreise_SelectionChanged" HorizontalAlignment="Left"
Margin="0,30,0,246" />
```

Dem Element wird neben Informationen über Positionierung und Layout auch eine Ereignisbehandlung für *“SelectionChanged”* mitgegeben. Dadurch kann später im Code Teil eine vom Benutzer getätigte Auswahl behandelt werden.
Der Button zum Starten des Eingabemodus soll als klickbares Bild erstellt werden. Hierzu wird der Button in ein *„esri:toolbar“* Element eingefügt:

```
<esri:Toolbar x:Name="Toolbar" MaxItemHeight="60" MaxItemWidth="60"
HorizontalAlignment="Left" Width="285"
ToolbarItemClicked="MyToolbar_ToolbarItemClicked" Margin="0,60,0,170" Height="70">
```

```
        <esri:Toolbar.Items>
                <esri:ToolbarItem Text="BB einzeichnen">
                    <esri:ToolbarItem.Content>
<Image Source="/SA_Radwege;component/Images/Icon_Rectangle.png"
Stretch="UniformToFill" Margin="5" />
                    </esri:ToolbarItem.Content>
                </esri:ToolbarItem>
        </esri:Toolbar.Items>
</esri:Toolbar>
```

Für den Button wird somit ein Bild („*Icon_Rectangle.png*“) geladen. Über die Ereignisbehandlung der Werkzeugleiste („*My_Toolbar_ToolbarItemClicked*“) kann im Code Teil dann das Ereignis des Klicks auf den Button behandelt werden und so der Editiermodus gestartet werden.
Aus dem „*esri:*“ Namensraum müssen nun noch die benötigten Symbole und ein Editor Element hinzugefügt werden:

```
<esri:SimpleLineSymbol x:Key="DrawLineSymbol" Color="Green" Width="4" />
<esri:SimpleLineSymbol x:Key="DefaultLineSymbol" Color="Red" Width="4" />
<esri:SimpleFillSymbol x:Key="DefaultFillSymbol" Fill="#33FF0000" BorderBrush="Red"
BorderThickness="2" />
<esri:Editor x:Key="MyEditor" Map="{Binding ElementName=MyMap}"
LayerIDs="MyGraphicsLayer" />
```

Es werden mehrere verschiedene Symbole benötigt. Es wird festgelegt, dass die zum Zeichnen verwendete Linie grün dargestellt werden soll („*DrawLineSymbol*“). Diese Symbologie wird auch für die fertig verschnittenen Linien verwendet. Nachdem der Benutzer seine Eingaben beendet hat, werden die an den Geoprocessing Dienst weitergeleiteten Linien für die Dauer der Bearbeitung rot dargestellt. Ist die Verschneidung abgeschlossen, erscheinen die Radwege wieder grün. Die Polygone der Kreise werden in anderen Farben dargestellt. Dafür wird ein Füllelement definiert („*DefaultFillSymbol*“). Die Umrandungsfarbe („*BorderBrush*“) wird als rot festgelegt, die Füllfarbe („*Fill*“) als ein hellerer Rotton (#33FF0000).
Das Editor-Element („*MyEditor*“) wird an das Kartenelement („*MyMap*“) gebunden. Dadurch ist es später möglich, den Editiermodus innerhalb des Kartenelements auszuführen. Somit sind alle benötigten Designelemente in XAML definiert. Die für die Umsetzung erforderliche Logik kann nun im Code Teil, dem sogenannten „*Code Behind*“, implementiert werden.

3.3.2 Code Behind

Beim Code Behind handelt es sich um den Code, der bei der Kompilierung der XAML Seite mit dieser verbunden wird [MS_1]. Hierin kann Programmlogik in einer .NET Sprache hinterlegt werden. Im Falle der Beispielanwendung erfolgt dies in C#.
Im Konstruktor der Seitenklasse („*MainPage*“) werden zunächst die benötigten Objekte für Geoprozessor und Suchanfrage definiert. Beide Objekte werden unter Angabe der entsprechenden URL der REST Schnittstelle initialisiert:

```
geoprocessorTask = new
Geoprocessor("http://rips_thesis/ArcGIS/rest/services/Radwege_Modelle/GPServer/RadTest
Neu");
QueryTask qT = new
QueryTask("http://rips_thesis/ArcGIS/rest/services/Kreise_Map_Service/MapServer/0");
```

Als nächstes muss ein “*Draw*”-Objekt (Zeichnen-Objekt) erstellt werden. Dem Objekt werden die im

XAML-Teil definierten Zeichenvorschriften zugewiesen. Ebenso wird hier die Ereignisbehandlung für das Beenden des Zeichenvorgangs definiert („*DrawComplete*“):

```
MyDrawObject = new Draw(MyMap)
            {
                LineSymbol = LayoutRoot.Resources["DrawLineSymbol"] as LineSymbol,
                FillSymbol = LayoutRoot.Resources["DrawFillSymbol"] as FillSymbol
            };
MyDrawObject.DrawComplete += MyDrawObject_DrawComplete;
```

Das Abfrageobjekt (“*QueryTask*”) wird nun erweitert. Beim Laden der Seite soll die ComboBox bereits mit den zur Verfügung stehenden Kreisen gefüllt werden, daher wird dem Objekt ein Event Handler mitgegeben sowie die Abfrage ausgeführt:

```
qT.ExecuteCompleted += QueryTask_ExecuteCompleted;
qT.Failed += QueryTask_Failed;
Query query = new Query();
// Specify fields to return from initial query
query.OutFields.AddRange(new string[] { "LANGNAME" });
// This query will just populate the drop-down, so no need to return geometry
query.ReturnGeometry = false;
// Return all features
query.Where = "1=1";
qT.ExecuteAsync(query, "initial");
```

Es wird das Feld “LANGNAME” aus der Datenbank abgefragt und die Abfrage ausgeführt („*ExecuteAsync*“). Mit diesem wird dann beim Laden der Seite die ComboBox gefüllt.
Beim ersten Ausführen wird auch der User State „*initial*“ übergeben. Mit diesem wird in der Ereignisbehandlung dann sichergestellt, dass kein Zeichenvorgang erfolgt, so lange noch kein Kreis ausgewählt wurde. Das Anzeigen auf der Karte soll erst erfolgen, wenn ein Kreis ausgewählt wurde.
Der Rest des benötigten Codes besteht nun ausschließlich aus Event Handlern für die Ereignisse, welche zur Laufzeit eintreten können.
Für das Abfrageobjekt wird die Ereignisbehandlung für das erfolgreiche Ausführen der Abfrage definiert:

```
private void QueryTask_ExecuteCompleted(object sender,
ESRI.ArcGIS.Client.Tasks.QueryEventArgs args)
        {
            …
        }
```

Hierin wird zunächst geprüft, ob das Ereignis das erste Mal auftritt (der „UserState“ ist „initial“). In diesem Fall soll lediglich die ComboBox gefüllt werden:

```
FeatureSet featureSet = args.FeatureSet;
// If initial query to populate states combobox
if ((args.UserState as string) == "initial")
{
   // Just show on initial load
   cbKreise.Items.Add("Kreis...");

   foreach (Graphic graphic in args.FeatureSet.Features)
          {
              cbKreise.Items.Add(graphic.Attributes["LANGNAME"].ToString());
          }
   cbKreise.SelectedIndex = 0;
```

```
   return;
}
```

Das „*featureSet*“-Objekt bildet die aus der Abfrage zurückgelieferten Daten ab. In der „*If*“-Schleife wird geprüft, ob es sich um die erste Abfrage an die Datenbank handelt. In diesem Falle wird dann nichts in die Karte eingezeichnet, lediglich die Auswahlbox soll gefüllt werden. Aus der Abfrage wird das Feld „LANGNAME“ in die ComboBox („*cbKreise*“) geschrieben. Der erste Eintrag der ComboBox wird als „Kreis…“ festgelegt. Ist der User State „*initial*“, wird nun aus der Methode gesprungen („*return*“).
Handelt es sich bei der Abfrage nicht um die erste Abfrage, welche beim Laden der Seite erfolgt, so ist der User State nicht „*initial*“. Dadurch wird diese erste „*If*“-Bedingung übersprungen. Stattdessen werden folgende Befehle ausgeführt:

```
// Remove the first entry if "Select..."
if (cbKreise.Items[0].ToString().Contains("Kreis..."))
    cbKreise.Items.RemoveAt(0);
```

Hiermit wird das beim ersten Füllen angezeigte „Kreis…“ aus der ComboBox entfernt. Die Grafikebene „*MyKreisLayer*“, in der die Kreise eingezeichnet werden, wird anschließend gelöscht:

```
GraphicsLayer graphicsLayer = MyMap.Layers["MyKreisLayer"] as GraphicsLayer;
             graphicsLayer.ClearGraphics();
```

Danach wird überprüft, ob die Datenbankabfrage nur ein Element enthält. In diesem Fall wird dieses Element (ein ausgewählter Kreis) in einer temporären Variable gespeichert, da es später noch für den Geoprozessor benötigt wird:

```
// If 1 Kreis selected: Save Data for GP
if (featureSet.Features.Count == 1)
   {
    selKreis = featureSet;

   }
```

In einer letzten „*If*“-Bedingung wird nun, für den Fall, dass ein gültiges Element aus der Datenbankabfrage zurückgeliefert wurde, dieses Element eingezeichnet:

```
if (featureSet != null && featureSet.Features.Count > 0)
            {
                // Show selected feature attributes in DataGrid
                Graphic selectedFeature = featureSet.Features[0];
                //Get Current Kreis as GPFeature
                // Hightlight selected feature
                selectedFeature.Symbol = LayoutRoot.Resources["DefaultFillSymbol"] as
ESRI.ArcGIS.Client.Symbols.Symbol;
                graphicsLayer.Graphics.Add(selectedFeature);

                // Zoom to selected feature (define expand percentage)
                ESRI.ArcGIS.Client.Geometry.Envelope selectedFeatureExtent =
selectedFeature.Geometry.Extent;

                double expandPercentage = 30;

                double widthExpand = selectedFeatureExtent.Width * (expandPercentage /
100);
```

```
                double heightExpand = selectedFeatureExtent.Height * (expandPercentage
/ 100);

                ESRI.ArcGIS.Client.Geometry.Envelope displayExtent = new
ESRI.ArcGIS.Client.Geometry.Envelope(
                selectedFeatureExtent.XMin - (widthExpand / 2),
                selectedFeatureExtent.YMin - (heightExpand / 2),
                selectedFeatureExtent.XMax + (widthExpand / 2),
                selectedFeatureExtent.YMax + (heightExpand / 2));

                MyMap.ZoomTo(displayExtent);
             }
```

Neben dem Einzeichnen soll auch die Karte auf den gewählten Kreis zoomen und dieser den Ausschnitt füllen. Dies geschieht durch die *„ZoomTo“*-Methode des *„MyMap“*-Objekts. Diesem wird die anzuzeigende Ausdehnung (*„displayExtent“*) übergeben.
Als nächstes Ereignis muss die Auswahl eines anderen Kreises durch den Benutzer behandelt werden. Dies geschieht über den Event Handler *„cbKreise_SelectionChanged“*:

```
private void cbKreise_SelectionChanged(object sender, SelectionChangedEventArgs e)
        {
            if (cbKreise.SelectedItem.ToString().Contains("Kreis..."))
                return;
            QueryTask queryTask = new
QueryTask("http://rips_thesis/ArcGIS/rest/services/Kreise_Map_Service/MapServer/0");
            queryTask.ExecuteCompleted += QueryTask_ExecuteCompleted;
            queryTask.Failed += QueryTask_Failed;

            ESRI.ArcGIS.Client.Tasks.Query query = new
ESRI.ArcGIS.Client.Tasks.Query();
            query.ReturnGeometry = true;
            query.Text = cbKreise.SelectedItem.ToString();
            query.OutSpatialReference = MyMap.SpatialReference;
            query.OutFields.Add("*");
            queryTask.ExecuteAsync(query);

        }
```

Ändert der Benutzer seine Auswahl eines Kreises, so wird eine Abfrage auf die Datenbank durchgeführt. In dieser Abfrage muss nun, neben dem Namen des Kreises auch dessen Geometrie abgefragt werden, um ihn in der Karte einzeichnen zu können. Dies wird in der Abfrage durch Setzen von *„ReturnGeometry“* auf wahr (*„true“*) erreicht. Die Ergebnisse sollen den gleichen Raumbezug wie die aktuelle Karte aufweisen (*„query.OutSpatialReference = MyMap.Spatialreference“*). Abschießend wird die Abfrage ausgeführt (*„ExecuteAsync“*). Dadurch wird wiederrum das Ereignis *„QueryTask_ExecuteCompleted“* ausgeführt, diesmal mit einem einzelnen Kreisobjekt sowie dessen Geometrie. Dadurch kann der Kreis dann eingezeichnet, und der Kartenausschnitt entsprechend gezoomt werden.
Als weiteres Ereignis ist ein Klick des Benutzers auf den *„Zeichnen“* Button zu behandeln. Dieser soll den Editiermodus starten:

```
private void MyToolbar_ToolbarItemClicked(object sender,
ESRI.ArcGIS.Client.Toolkit.SelectedToolbarItemArgs e)
        {
            lbMsg.Content = "";
            MyDrawObject.DrawMode = DrawMode.Polyline;
            _activeSymbol = LayoutRoot.Resources["DefaultLineSymbol"] as Symbol;
```

```
            MyDrawObject.IsEnabled = (MyDrawObject.DrawMode != DrawMode.None);
        }
```

Das im Konstruktor definierte Zeichnen-Objekt („*MyDrawObjekt*“) wird somit beim Klick auf den Button aktiviert. Es sollen Polylinien eingezeichnet werden. Die Symbologie wird als das definierte „*DefaultLineSymbol*“ festgelegt.
Hat der Benutzer nun die Eingabe beendet, so wird durch einen Doppelklick die Ereignisbehandlung für „*DrawComplete*“ des Zeichnen-Objekts ausgeführt. Zunächst wird hierin das eingezeichnete Objekt der Anzeige hinzugefügt, und zwar der „*MyGraphicsLayer*“ Ebene:

```
private void MyDrawObject_DrawComplete(object sender, ESRI.ArcGIS.Client.DrawEventArgs
args)
        {
            GraphicsLayer graphicsLayer = MyMap.Layers["MyGraphicsLayer"] as
GraphicsLayer;
            ESRI.ArcGIS.Client.Graphic graphic = new ESRI.ArcGIS.Client.Graphic()
            {
                Geometry = args.Geometry,
                Symbol = _activeSymbol,
            };
            graphicsLayer.Graphics.Add(graphic);
```

Neben dem Einzeichnen soll die erstellte Geometrie auch an den Geoprozessor übergeben werden, damit sie mit dem gewählten Kreis verschnitten werden kann. Da jeder Geoprozessor seine Variablen in einer „GPParameter“ Liste übergeben bekommen muss, wird diese zunächst gefüllt:

```
List<GPParameter> parameters = new List<GPParameter>();
            parameters.Add(new GPFeatureRecordSetLayer("Input_Rad", args.Geometry));
            parameters.Add(new GPFeatureRecordSetLayer("Input_Kreis",
selKreis.Features[0].Geometry));
```

Da der Geoprocessing Service des Servers zwei Eingaben erwartet, müssen auch Beide in der Liste übergeben werden. Der „*Input_Rad*“ ist hier die gerade eingezeichnete Geometrie der Radwege („*args.Geometry*“). Als „*Input_Kreis*“ wird die Geometrie des aktuellen Kreises verwendet, welche vorher zwischengespeichert wurde. Diese beiden Parameter stellen genau Jene dar, die beim Erstellen des Geoprocessing Services vorher festgelegt wurden. Da der Dienst synchron ausgeführt werden soll, wird ein „*ExecuteAsync*“ Event Handler benötigt:

```
geoprocessorTask.ExecuteCompleted += GeoprocessorTask_ExecuteCompleted;
geoprocessorTask.Failed += GeoprocessorTask_Failed;
```

Falls beim Ausführen des Geoprocessing Dienstes Fehler auftreten, so werden diese im “*GeoprocessorTask_Failed*” Handler behandelt. Die Parameter sind nun definiert und dem Geoprozessor Objekt übergeben, somit kann das Objekt ausgeführt werden:

```
geoprocessorTask.ExecuteAsync(parameters);
```

Der Prozess wird nun an den ArcGIS Server übermittelt und dort ausgeführt. Da eine synchrone Ausführung gewählt wurde, wartet die Silverlight Anwendung (der Klient) nun auf die Rückmeldung des Servers. Im Falle eines Fehlers findet das Ereignis „*GeoprocessorTask_Failed*“ statt. In der Ereignisbehandlung des Fehlerfalls wird dem Benutzer die entsprechende Fehlermeldung angezeigt:

```
private void GeoprocessorTask_Failed(object sender, TaskFailedEventArgs e)
        {
```

```
            lbMsg.Content = e.Error.Message;
        }
```

Wird der Dienst erfolgreich ausgeführt so wird das Ereignis „*ExecuteCompleted*“ ausgelöst:

```
 private void GeoprocessorTask_ExecuteCompleted(object sender,
GPExecuteCompleteEventArgs e)
        {
            GraphicsLayer graphicsLayer = MyMap.Layers["MyGraphicsLayer"] as
GraphicsLayer;
            graphicsLayer.ClearGraphics();
            foreach (GPParameter gpParameter in e.Results.OutParameters)
            {
                GPFeatureRecordSetLayer gpLayer = gpParameter as
GPFeatureRecordSetLayer;
                foreach (Graphic graphic in gpLayer.FeatureSet.Features)
                {
                    graphic.Symbol = LayoutRoot.Resources["DrawLineSymbol"] as
ESRI.ArcGIS.Client.Symbols.Symbol;
                    graphicsLayer.Graphics.Add(graphic);
                }
            }
        }
```

Vom Server wird das Ergebnis des Geoprocessing Service zurückgeliefert. Der Grafiklayer für die Radwege wird geleert, anschließend die Geometrie aus dem Ergebnis in den Layer eingefügt und somit angezeigt. Der Zugriff erfolgt über die Ausgabeparameter der Ereignisargumente. Diese werden in einer „*for each*“-Schleife durchlaufen, und dem Grafikobjekt zugeordnet.

4 Umsetzung der 3D Anwendung und Integration in das bestehende RipsWeb

4.1 Bestandsaufnahme

Im Rahmen einer Bachelor Thesis (S. Hacker, 2011: *Evaluierung von Datengrundlagen und Werkzeugen zur Web-basierten 3D Visualisierung*) wurde an der LUBW ein Konzept zur Umsetzung von 3D Darstellungen entwickelt. Der Schwerpunkt lag hierbei insbesondere auf der Untersuchung von Alternativen zum bisher eingesetzten 3D Werkzeug (GeoPro3D). Es zeigte sich, dass der Einsatz von 3D PDF Dokumenten den gestellten Anforderungen genügen würde. Es wurde ein schematischer Ablauf entwickelt, mit Hilfe dessen die Umsetzung erfolgen könnte.

Im Rahmen dieses Konzepts soll der Zugriff der Fachanwendungen auf die PDF Erstellung über die bestehende WPS Schnittstelle des RipsWeb der LUBW erfolgen. Über die Schnittstelle können die Anwendungen auf eine serverseitig vorliegende Anwendung zugreifen, welche die eigentliche Konvertierung der Eingangsdaten in ein 3D PDF vornimmt. Aufgabe dieser Anwendung ist das Ansprechen der FME Umgebung, die Übermittlung der von FME benötigten Daten, sowie das Speichern und zur Verfügung stellen des fertigen 3D Dokuments. Als Rückgabe an die anfragende Anwendung soll eine URL mit einem Verweis auf das erzeugte PDF gesendet werden.

4.2 Konzept

Ausgehend von den vorhandenen Grundlagen soll die Umsetzung der 3D PDF Anwendung unter bestmöglicher Ausnutzung bereits vorhandener Infrastruktur erfolgen. Daher werden die benötigten Karten-, Bild- und DGM-Daten mit Hilfe des ArcGIS Server zur Verfügung gestellt. Die Abfrage kann dann über die entsprechenden REST Schnittstellen erfolgen.

Die eigentliche Programmlogik wird in die bestehende RipsWeb.dll integriert. Damit können bereits vorhandene Methoden bei Bedarf genutzt werden, beispielsweise Datenbankzugriffe oder Koordinatenumrechnungen.

Das eigentliche Erstellen des PDF soll mittels der vorhandenen FME Infrastruktur erfolgen.

Für die Integration in bestehende Anwendungen ist es schließlich auch erforderlich, eine WPS Anbindung zu erstellen, und die neuen Funktionalitäten hierrüber Anwendern zur Verfügung zu stellen.

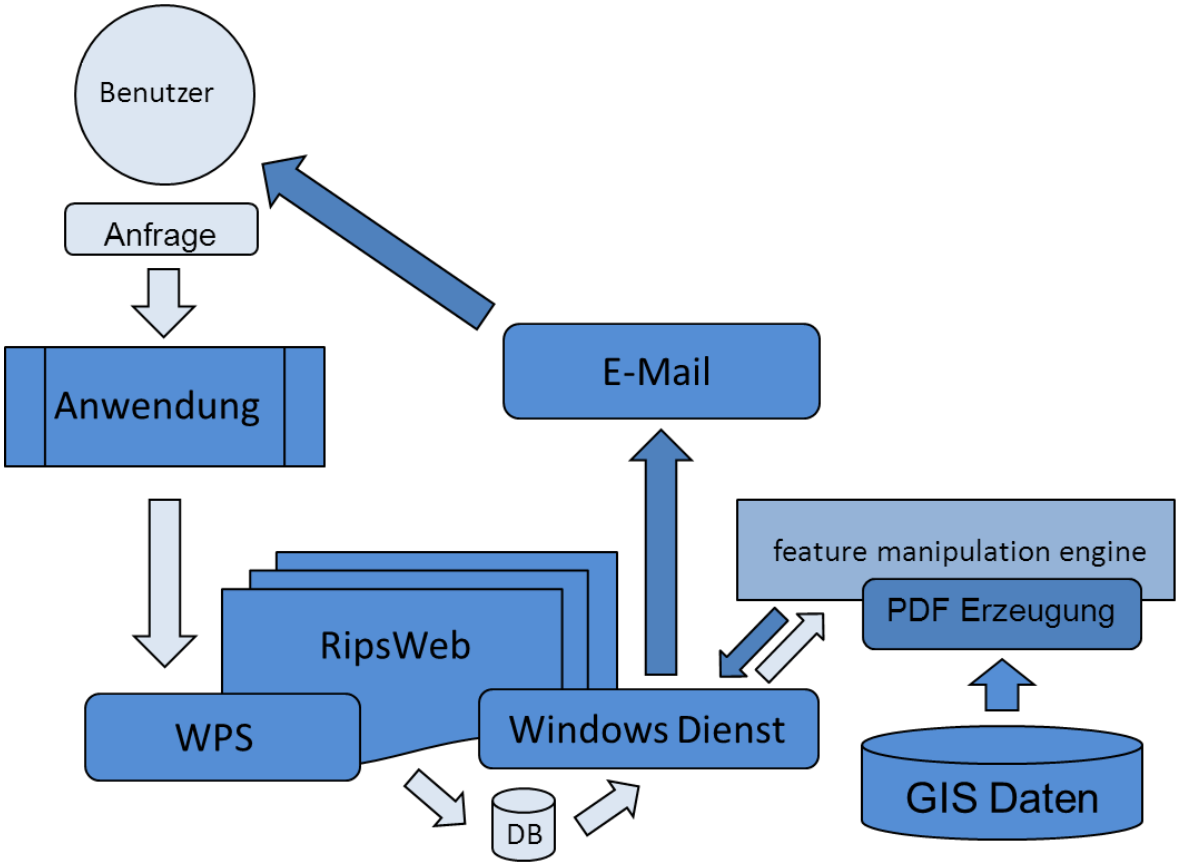

Abbildung 10: Konzept 3D Anwendung

Da an der LUBW momentan lediglich eine Einzelplatzlizenz der FME Umgebung vorhanden ist, muss dies bei der Umsetzung berücksichtigt werden. Als Lösung bietet sich an, FME über einen Kommandozeilenaufruf vom Webserver aus, welcher die eigentliche Anwendung beherbergt, zu starten. Für das Ausführen von Prozessen über solche Remoteverbindungen gibt es von Microsoft/Sysinternals bereits eine entsprechende Anwendung („psexec“), welche aus einer .Net Anwendung heraus gestartet werden kann.
Da durch diese Vorgehensweise nicht sichergestellt werden kann, dass FME bei jeder Anfrage erreichbar ist, ist es darüber hinaus erforderlich, die Verarbeitung von Anfragen asynchron durchzuführen. Hierzu wird in der vorhandenen Datenbank eine neue Tabelle angelegt, in welche die eingehenden Anfragen gespeichert werden. Über einen Windows Dienst auf dem Webserver wird dann periodisch überprüft, ob FME erreichbar, und der Auftrag somit durchführbar ist. Ist dies der Fall, wird die eigentliche PDF Erstellung mittels FME gestartet. Ist das Dokument erstellt, wird es auf dem Webserver abgelegt, und eine entsprechende Email mit einem Verweis auf das Dokument an den anfordernden Benutzer gesendet.
Nachteil dieser Vorgehensweise ist, dass der Nutzer unter Umständen einige Zeit warten muss, bis er das von ihm angeforderte PDF verwenden kann. Dies stellt jedoch momentan die einzige Möglichkeit dar, unter Verwendung nur einer FME Lizenz einen fehlerfreien Ablauf zu gewährleisten.

4.3 Erstellung der FME Workbench

Zur Erzeugung von 3D PDF Dokumenten gibt es zurzeit an der LUBW lediglich die Möglichkeit, dies mittels des Programms Feature Manipulation Engine (FME) der Firma Safe Software Inc.

durchzuführen. Vorteil des Programms ist die Möglichkeit, es über Kommandozeilenbefehle anzusteuern, d. h. die grafische Oberfläche muss nicht eingesetzt werden. Dadurch wird ein programmatischer Aufruf aus einer anderen Anwendung heraus möglich.
FME bietet die Möglichkeit, Konvertierungen zwischen einer Vielzahl an unterschiedlichen Formaten durchzuführen. Hierbei liegt der Schwerpunkt insbesondere auf dem GIS- und CAD- Bereich, woraus die meisten momentan gängigen Formate zur Verfügung stehen. Neben reinen Konvertierungen zwischen verschiedenen Formaten bietet FME auch integrierte Transformatoren, um verschiedene Datenmanipulationen durchführen zu können. So kann beispielsweise aus einem eingelesenen GeoTIFF direkt in FME ein TIN („*Triangulated Irregular Network*“) erzeugt werden.
Dieser Umstand stellt einen großen Vorteil bei der Erzeugung der benötigten Dokumente dar. Sämtliche Umwandlungen können in einer zentralen Anwendung durchgeführt werden. Ein 3D PDF besteht in der Regel aus dreidimensionalen Oberflächen und darauf angebrachten Texturen. Daher benötigt auch FME für die Erzeugung des PDF eine Oberfläche und die darauf anzubringende Textur. FME speichert vom Benutzer definierte Umwandlungen in sogenannten „*Workbenches*“. Hierin werden sämtliche Eingabedaten, Transformationen, Manipulationen, eventuelle Parameter sowie die Ausgangsdaten hinterlegt. Im Falle der 3D PDF Erzeugung werden Karten- oder Bilddaten (für die Textur) sowie ein TIN (erstellt aus dem vorliegenden DGM) benötigt. Innerhalb der FME Workbench können mit diesen Eingabedaten nun die nötigen Umwandlungen durchgeführt werden.

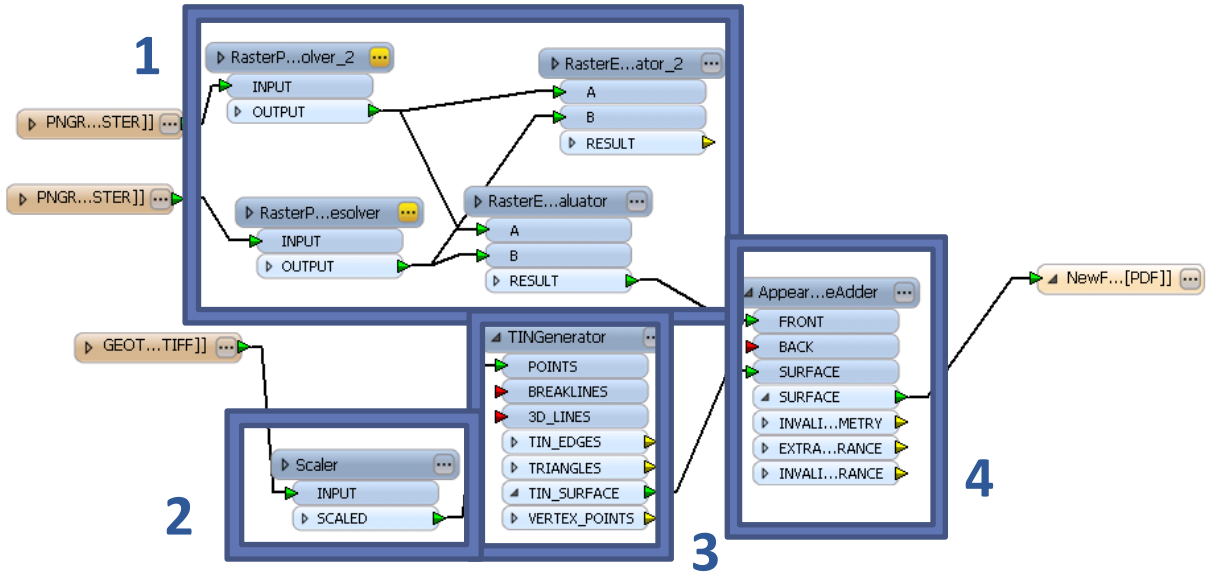

Abbildung 11: FME Workbench

Die später als Textur dienenden Bilddaten können direkt in FME miteinander kombiniert werden. Dies ist unter Umständen erforderlich, da beispielsweise der Kartenhintergrund und die Kartenbeschriftung im Rips Web als separate Bilddateien vorliegen(Abb. 11, 1.).
Aus dem als Bilddatei (GeoTIFF) vorliegenden DGM wird das TIN für das PDF erzeugt. Um die geforderte Bandbreite an Maßstäben, von ALK Karten bis zu landesweiten Darstellungen, zu ermöglichen, werden Überhöhung und Oberflächenauflösung als variable Parameter festgelegt. Das Eingangs-TIFF wird mittels des in FME integrierten „*Scaler*“-Werkzeugs um einen vom Benutzer festgelegten Wert in Z-Richtung überhöht(Abb. 11, 2.). Dieser überhöhte Datensatz wird dann als Eingabedaten des Werkzeugs „*TIN-Generator*“ verwendet. Innerhalb des „*TIN-Generator*“ wird auch die Oberflächenauflösung als variabler Parameter hinterlegt (Abb. 11, 3.).
Auf das fertige gerechnete TIN können nun mit dem „*Appearance Adder*“ Werkzeug die Bilddaten als

Textur aufgebracht werden(Abb. 11, 4.). Im letzten Schritt wird dieses texturierte TIN dann in ein 3D PDF geschrieben.

4.4 Kartendaten und DGM

Um die Umwandlung in FME durchführen zu können, werden verschiedene Ausgangsdaten benötigt. Wichtigster Datensatz ist hierbei das die 3D Darstellung ermöglichende DGM. An der LUBW liegt ein entsprechendes landesweites DGM, mit Auflösungen von bis zu 1 Meter pro Pixel, vor. Für die Anwendung ausreichend ist die Verwendung des DGM5, mit einer Auflösung von 5 Metern pro Pixel. Bei den kleinsten sinnvollen Darstellungen handelt es sich um Oberflächen im Bereich von 1 km^2, was bei einer Auflösung von 5 Metern pro Pixel einer Punktwolke von 40000 Punkten pro km^2 entspricht. Dies stellt eine ausreichende Menge dar, um mittels FME eine sinnvolle Oberflächenberechnung durchführen zu können. Ist in Zukunft eine noch höher auflösende Darstellung gewünscht, so kann dies durch den Austausch des zugrunde liegenden DGM relativ einfach erreicht werden. Neben dem DGM müssen auch die auf der Oberfläche anzuzeigenden Karten- oder Bildinhalte zugänglich gemacht werden.

Durch den Einsatz des ArcGIS Server bietet sich die Möglichkeit, beliebige Inhalte bei der PDF Erzeugung einzusetzen. Die gewünschten Daten müssen lediglich am Server veröffentlicht werden, und stehen anschließend für die Verwendung in Anwendungen zur Verfügung. Ermöglicht wird dies durch die vom ArcGIS Server angebotene REST Schnittstelle. Über diese Schnittstelle kann durch einen einfachen HTTP-Aufruf auf die Daten zugegriffen werden. Kartendaten können als sogenannter „*Map Server*“ bereitgestellt werden, Bilddaten analog über einen „*Image Server*“. Diese Daten können dann über die REST Schnittstelle angesprochen und verarbeitet werden. Im Falle der vorliegenden Anwendung werden Rasterdateien der Bild- und Kartendaten benötigt. Entsprechend wird über die REST Schnittstelle ein „*image*“ abgefragt. Dieses wird definiert durch die Angabe der geographischen Koordinaten, der verwendeten Projektion, der Bildgröße sowie des Bildformats. Mit Hilfe dieser Angaben, welche einfach über die URL in der Abfrage übergeben werden, erhält der anfragende Klient (die PDF Anwendung) als Rückgabe dann die angeforderte Bilddatei. Vorteil der REST Schnittstelle ist daher, das auf Geodaten, in diesem Fall georeferenzierte Raster- und Bilddaten, von beliebigen Klienten aus zugegriffen werden kann. Sind die Daten nun auf einem ArcGIS Server veröffentlicht, können sie als Eingabe für die FME Workbench, und damit letztlich für ein 3D PDF, verwendet werden.

4.5 Integration in das bestehende RIPS Web Framework

Um die Anbindung an die verschiedenen Fachanwendungen sicherzustellen, muss die Programmlogik in das bestehende Framework integriert werden. Hierzu sind verschiedene Arbeiten notwendig. Eine neue Klasse muss erstellt werden, welche den Code für das Ansprechen des ArcGIS Server, der FME, etc. enthält. Der Zugriff auf die Datenbank muss ermöglicht und die WPS Schnittstelle angepasst werden.

4.5.1 Erweiterung der Datenbank

Durch die asynchrone Umsetzung des Dienstes werden Änderungen an der Datenbank notwendig. Um die zu bearbeitenden Aufträge zu erfassen und zuordnen zu können wir eine neue Tabelle angelegt.

Bezeichner	Oracle Datentyp	C# Datentyp	Inhalt
AUFTRAG_ID	NUMBER	INT32	Eindeutiger Bezeichner für jeden Auftrag
X_MIN	NUMBER	DOUBLE	Minimale geographische X Koordinate
X_MAX	NUMBER	DOUBLE	Maximale geographische X Koordinate
Y_MIN	NUMBER	DOUBLE	Minimale geographische Y Koordinate
Y_MAX	NUMBER	DOUBLE	Maximale geographische Y Koordinate
SIZE_X	NUMBER	INT32	Größe in X-Richtung in Pixel des PDF
SIZE_Y	NUMBER	INT32	Größe in Y-Richtung in Pixel des PDF
EMAIL	VARCHAR2	STRING	E-Mail-Adresse des Benutzers
AUFTRAG_BEARB	NUMBER	INT32	Boolescher Wert mit bearbeitet = ja/nein
ERSTELL_DATUM	DATE	DateTime	Datum des Datenbankeintrags

Tabelle 1: Datenbankschema

In der Tabelle PS_3DPDF wird mit Hilfe des Schemas in Tabelle 1 jeder Auftrag gespeichert. Mit dem AUFTRAG_ID Wert kann jeder Auftrag eindeutig zugeordnet werden. Die Werte X_MIN, X_MAX, Y_MIN, Y_MAX definieren die Umrisse des vom 3D PDF darzustellenden Gebietes, jeweils als minimale und maximale Werte in geographischer X- und Y-Richtung. Über die beiden Werte SIZE_X und SIZE_Y kann vom Benutzer eine individuelle Größe für das PDF festgelegt werden. Im Feld EMAIL wird die E-Mail-Adresse des Benutzers hinterlegt. Die beiden Felder AUFTRAG_BEARB sowie ERSTELL_DATUM schließlich dienen der Protokollierung. Beim Anlegen eines neuen Auftrags wird das aktuelle Datum sowie die Uhrzeit im Feld ERSTELL_DATUM hinterlegt. Im Feld AUFTRAG_BEARB wird mit den numerischen Werten 1 und 2 festgehalten, ob der entsprechende Auftrag bereits ausgeführt wurde. 1 steht hierbei für einen noch nicht ausgeführten, 2 für einen erledigten Auftrag.

4.5.2 Erstellen der neuen Klassen in RipsWeb und Umsetzen der Funktionalität

Für die Integration in das bestehende RipsWeb Framework ist es notwendig, eine neue Klasse mit den erforderlichen Funktionen zu erstellen. Alle Teilprojekte des RipsWeb Framework sind in einer, „*RipsWebservices*“ genannten, .NET Solution zusammengefasst. Innerhalb dieser Solution werden in einzelnen Teilprojekten die unterschiedlichen Aufgaben umgesetzt.

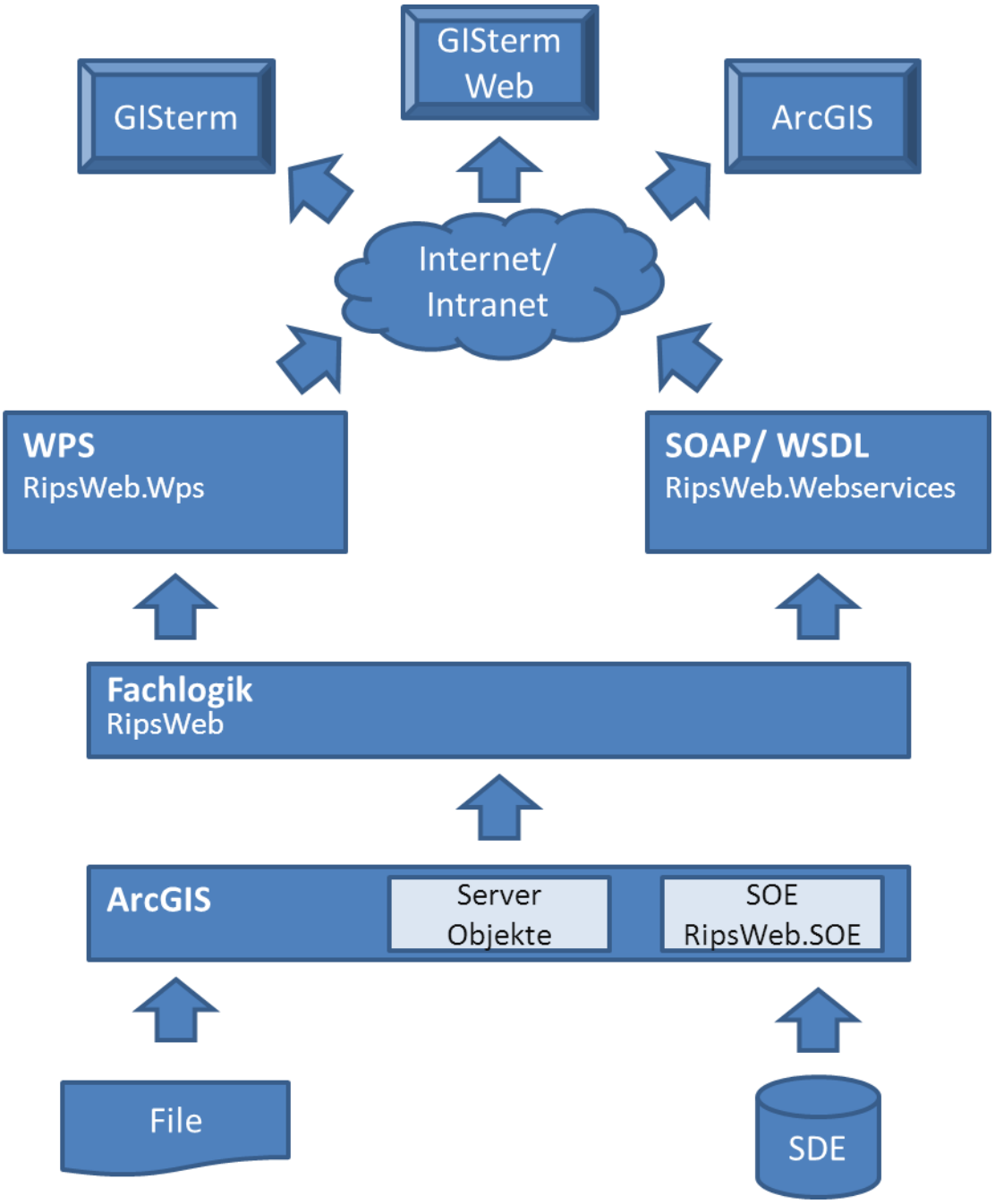

Abbildung 12: Struktur Rips Webservices, nach: [Haberer_2011, S. 4]

Im Projekt „*RipsWeb.Webservices*“ ist eine XML-Schnittstelle für Geofunktionen umgesetzt. Diese XML-Webservices basieren auf WSDL und SOAP Protokollen. Die Funktionalitäten der ArcGIS Server Object Extensions (SOE) sind im Projekt „*RipsWeb.SOE*“ implementiert. Über die SOE kann aus dem .NET Umfeld auf die Funktionen, welche ein ArcGIS Server zur Verfügung stellt, zugegriffen werden. Das Projekt RipsWeb repräsentiert die Zwischenschicht zwischen der XML- und der SOE-Schicht. In der ursprünglichen Konzeption gehörte der Verbindungsaufbau zwischen ArcGIS Server und Anwendungen zu den Hauptaufgaben der RipsWeb.dll. Mittlerweile ist das Projekt auch um Funktionen erweitert, welche keine SOE Grundlage haben und direkt in der RipsWeb.dll ausgeführt werden können. Innerhalb von RipsWeb wird somit auch die 3D PDF Funktionalität angelegt. Eine

weitere wesentliche Schicht in den RipsWeb Services stellt die WPS Schnittstelle im Projekt „*RipsWeb.WPS*“ dar. Hierüber können Klienten die Dienste der RipsWeb.dll über das WPS Protokoll ansprechen. Weitere Projekte in der Solution stellen Hilfsfunktionen zur Verfügung. So existiert beispielsweise ein „*RegisterSOE*“ Projekt, mit welchem „*RipsWeb.SOE*“ an einem ArcGIS Server registriert werden kann.
Daher wird die 3D PDF Funktionalität direkt in das Projekt RipsWeb integriert. Die Anbindung nach Außen wird anschließend innerhalb von „*RipsWeb.WPS*“, unter Einsatz von WPS, realisiert. Innerhalb des Projekts RipsWeb sind die unterschiedlichen Klassen mit den entsprechenden Funktionen in einer Ordnerstruktur abgelegt.

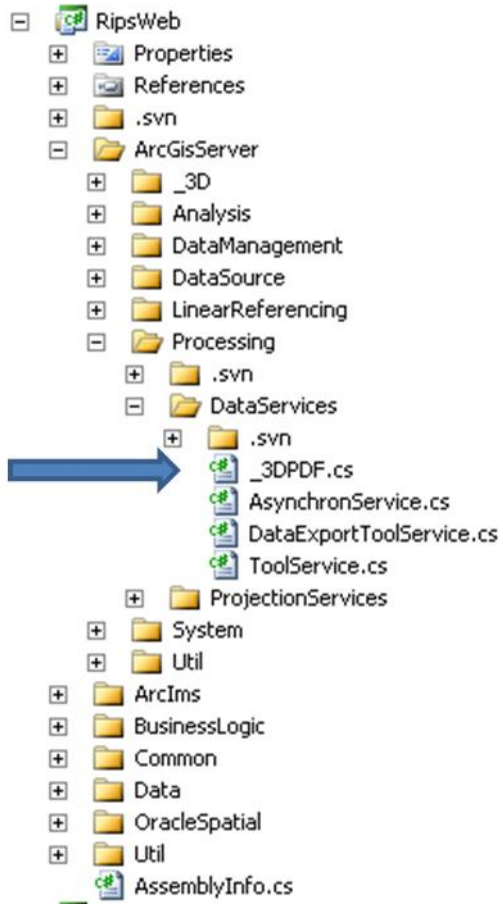

Abbildung 13: Struktur RipsWeb.dll

Die neu zu erstellende Klasse „*_3DPDF*“ wird in „*RipsWeb\ArcGISServer\Processing\DataServices*“ angelegt. Hier finden sich bereits Klassen zum Datenexport oder dem asynchronen Ausführen von Geodiensten.
Innerhalb der Klasse ist es aufgrund der gewünschten Funktionalität erforderlich, zwei Methoden anzulegen. Zum einen wird eine Methode benötigt, welche Benutzereingaben verarbeiten, und diese anschließend in der Datenbank ablegen kann. Die zweite Methode muss dann anschließend den Datensatz eines Auftrages auslesen, die benötigten Rasterdaten vom GIS Server abrufen, die FME Workbench starten, sowie nach erfolgreichem Durchführen aller Operationen eine E-Mail an der anfragenden Benutzer senden.
Bei der Methode zum Schreiben des Datenbankeintrags handelt es sich um einige wenige, relativ einfache Operationen. Da die Klasse „*_3DPDF*“ in das bestehende RipsWeb Umfeld integriert wurde, stehen auch sämtliche bereits vorhandenen Funktionen zur Verfügung. Dies erleichtert insbesondere

den Datenbankzugriff auf die Oracle Datenbank, in welcher die Aufträge abgelegt werden sollen. Als Parameter der Methode werden sämtliche Informationen, welche in der Datenbank gespeichert werden sollen, benötigt. Lediglich die automatisch gesetzten Felder AUFTRAG_ID, AUFTRAG_BEARB sowie ERSTELL_DATUM müssen nicht direkt übergeben werden. Da die Klasse später über WPS angesprochen werden soll muss die Methode auch einen Rückgabewert liefern, in diesem Fall vom Datentyp String. Somit lautet der Methodenaufruf:

```
public static string CREATE_PDF_JOB(string[] BB, int sizeX, int sizeY, string eMail)
```

Im Parameter BB (als String Array) werden die minimalen und maximalen X- und Y-Koordinaten (die „*Bounding Box*“) übergeben. Dies entspricht den Datenbankfeldern X_MIN, X_MAX, Y_MIN, Y_MAX. Die beiden Integer Werte „*sizeX*“ und „*sizeY*“ beinhalten die Pixelanzahl in X- und Y-Richtung des zu erstellenden Dokuments, entsprechend den Werten SIZE_X und SIZE_Y in der Datenbank. Die E-Mail Adresse des anfragenden Benutzers schließlich wird als String Wert „*eMail*“ übergeben, in der Datenbank als EMAIL abgelegt. Wie bereits erwähnt, werden die übrigen Felder der Tabelle automatisch gefüllt, bzw. erfordern keine expliziten Eingaben des Benutzers.
Der eigentliche Datenbankzugriff kann mittels integrierten Funktionen von RipsWeb realisiert werden:

```
RipsWeb.Data.CommonConnection pConnection =
RipsWeb.Data.ApplicationConfiguredOracleConnection.getCommonConnection("3DPDF_dbSource
ARCGISSERVER", "3DPDF_dbUserARCGISSERVER", "3DPF_dbPwdARCGISSERVER");
```

Hierzu wird ein neues Objekt der Klasse „*CommonConnection*“ erzeugt („*pConnection*“). Dabei wird mittels der vorhandenen Methode „*getCommonConnection*“ der Klasse „*ApplicationConfiguredOracleConnection*“ eine Datenbankverbindung erstellt. Die Zugriffsdaten, welche im Konstruktor benötigt werden, müssen in der „*app.config*“ Datei des ausführenden Projekts hinterlegt werden(Datenbank, Benutzer und Passwort).
Der weitere Code der Methode wird innerhalb eines „*try-catch*“ Blocks abgelegt, um eventuelle Fehler abfangen und behandeln zu können. Als erste Operation muss die erzeugte Datenbankverbindung geöffnet werden:

```
pConnection.openConnection();
```

Der an die Datenbank zu sendende SQL Code wird in einem String gespeichert:

```
string strInsert = "INSERT INTO PS_3DPDF
(AUFTRAG_ID,X_MIN,X_MAX,Y_MIN,Y_MAX,SIZE_X,SIZE_Y,EMAIL,AUFTRAG_BEARB) VALUES
(PS3DPDF_SEQ.nextval," + BB[0] + "," + BB[1] + "," + BB[2] + "," + BB[3] + "," +
                      + sizeX + "," + sizeY + ",'" + eMail + "',1)";
```

Der String wird dynamisch aufgebaut, und die an die Methode übergebenen Werte eingesetzt. Das Feld AUFTRAG_BEARB wird, da es sich um einen neuen Auftrag handelt, auf den Wert 1 gesetzt. Wenn der Auftrag später erfolgreich abgearbeitet worden ist, so wird dieser Wert auf 2 geändert. Das Datumsfeld ERSTELL_DATUM wird automatisch gefüllt. Um dies zu gewährleisten, darf innerhalb der INSERT Anweisung hier kein expliziter Wert gesetzt werden. Ist dies der Fall, so wird mittels der SQL Anweisung DEFAULT SYSDATE das Datum sowie die Uhrzeit, zu welchem der Eintrag erzeugt wurde, gespeichert.

Um die Datenbankoperation nun auszuführen, muss der String lediglich an die „*executeScalar*“ Methode des „*pConnection*“ Objekts übergeben werden:

```
pConnection.executeScalar(strInsert);
```

Da, wie beschrieben, die Methode auch über WPS angesprochen werden soll, wird noch ein Rückgabewert benötigt. Dies kann ein einfacher String sein, welcher den Benutzer über den erfolgreichen Eingang seiner Anfrage informiert:

```
string result = "Ihre Anfrage wurde entgegengenommen und wird schnellstmöglich
bearbeitet";
```

Die Methode zum Durchführen der PDF Erzeugung muss einige komplexere Operationen durchführen. Da sie später aus einem Windows Dienst heraus zu bestimmten Intervallen ausgeführt werden soll, werden kein Rückgabewert und auch keine Eingabeparameter benötigt:

```
public void EXECUTE_PDF_JOB()
```

Zunächst müssen die benötigten Variablen deklariert werden, in denen die aus der Datenbank ausgelesenen Werte gespeichert werden:

```
int id = 0;
string[] xY = new string[4];
string eMail = "";
int MapX;
int MapY;
```

Es werden dieselben Variablen wie in der Methode zur Erstellung des Eintrags benötigt. Die Abfragen an die Datenbank werden wiederrum als String Objekte erzeugt. Um den Bearbeitungszustand eines Eintrags überprüfen zu können, werden zwei Abfragen verwendet. Die erste Abfrage wählt den Eintrag, welcher die kleinste ID sowie den Bearbeitungszustand 1 (d.h., „noch nicht bearbeitet“) hat, aus:

```
string strSelectMin = "SELECT MIN(AUFTRAG_ID) FROM PS_3DPDF WHERE AUFTRAG_BEARB = 1 ";
```

Damit wird sichergestellt, dass jeder Auftrag nur einmal ausgeführt wird. Auch wird so der älteste Auftrag zuerst ausgeführt, nämlich der Auftrag mit der kleinsten ID. Um die benötigten Daten für den Auftrag auszulesen wird ein weiterer String mit dem entsprechenden SQL Befehl angelegt:

```
string strSelect = "SELECT AUFTRAG_ID,X_MIN,X_MAX,Y_MIN,Y_MAX,SIZE_X,SIZE_Y,EMAIL FROM
PS_3DPDF";
```

Fehler werden wieder innerhalb eines „*try-catch*“ Blocks abgefangen. Die Datenbankverbindung muss geöffnet werden sowie ein „*DataReader*“ Objekt mit den SQL Anweisungen der Verbindung zugewiesen werden:

```
pConnection.openConnection();
IDataReader drMin = pConnection.getDataReader(strSelectMin);
```

Aus dem „*DataReader*“ Objekt können nun die aus der Datenbank zurückgegebenen Werte ausgelesen werden:

```
drMin.Read();
```

Da im String „*strSelecMin*“ nur die niedrigste ID von Einträgen, welche noch nicht bearbeitet worden sind abgefragt wird, muss überprüft werden, ob ein solcher Eintrag überhaupt vorliegt. Es kann der Fall auftreten, dass sämtliche Aufträge bereits abgearbeitet worden sind, was zu einem Fehler in der Anwendung führen würde. Daher wird mittels folgender Anweisung die Existenz eines gültigen Eintrages überprüft und der folgende Codeblock nur ausgeführt, wenn dies der Fall ist:

```
if (!drMin.IsDBNull(0))
{
…
}
```

Dies wird durch eine „If“-Abfrage erreicht. Wenn der erste (und in diesem Fall einzige) Eintrag des Ergebnisses der Datenbankabfrage nicht leer ist, so wird der folgende Code ausgeführt. Es wird geprüft ob der Eintrag leer ist („*isDBNull*“), ist dies nicht der Fall („!“-Operator), so wird der Code ausgeführt.
Innerhalb des „If“-Blocks können nun die weiteren Operationen durchgeführt werden. Der Variable „*id*“ wird der Wert aus der Abfrage nach der minimalen ID zugewiesen:

```
id = drMin.GetInt32(0);
```

Der Abfrage-String wird um diese ID in der SQL WHERE Bedingung ergänzt:

```
strSelect += " WHERE AUFTRAG_ID=" + id;
```

In einer weiteren Abfrage werden nun die Werte des Eintrags mit der niedrigsten ID abgerufen:

```
IDataReader dr = pConnection.getDataReader(strSelect);
```

Diese werden anschließend den vorher definierten Variablen zugewiesen:

```
xY[0] = dr.GetDouble(1).ToString();
xY[1] = dr.GetDouble(2).ToString();
xY[2] = dr.GetDouble(3).ToString();
xY[3] = dr.GetDouble(4).ToString();
MapX = dr.GetInt32(5);
MapY = dr.GetInt32(6);
//Max Size for Map Service: 2000x2000:
if (MapX > 2000) MapX = 2000;
if (MapY > 2000) MapY = 2000;
eMail = dr.GetString(7);
```

Zu beachten ist hierbei, dass eine Abfrage über die REST Schnittstelle des ArcGIS Server nur Bilddateien mit einer Größe von bis zu 2000x2000 Pixeln erlaubt. Daher wird zur Sicherheit überprüft, ob dieser Wert überschritten wurde. Ist dies der Fall, so wird die Anzahl der Pixel auf den Maximalwert 2000 gesetzt.
Durch den Aufruf dreier weiterer Methoden werden die übrigen Arbeitsschritte durchgeführt:

```
getMaps(xY, MapX, MapY);
startFME(id);
sendEmail(eMail,id);
```

Abschließend wird im „*finally*“ Teil des „*try-catch*“ Blocks die Datenbank aktualisiert, der Auftrag auf bearbeitet gesetzt und die Datenbankverbindung geschlossen:

```
string strUpdate = "UPDATE PS_3DPDF SET AUFTRAG_BEARB = 2 WHERE AUFTRAG_ID = " + id;
pConnection.executeScalar(strUpdate);
pConnection.closeConnection();
```

Dadurch kann sichergestellt werden, dass ein Auftrag nur als erledigt markiert wird, wenn er auch tatsächlich erfolgreich ausgeführt worden ist.
Die drei Methoden „*getMaps()*“, „*startFME()*“ sowie „*sendEmail()*“ beinhalten die weiteren nötigen Operationen. Die Methode „*getMaps()*“ wird aufgerufen, indem die nötigen Daten zur Erzeugung der Rasterdateien über die REST Schnittstelle übergeben werden. Dies sind die Koordinaten der Bounding Box (als String Array), sowie die Pixelanzahl in X- und Y-Richtung:

```
getMaps(xY, MapX, MapY);
```

Innerhalb der Methode wird mit Hilfe der „*WebClient*“ Klasse auf die Schnittstelle der Karten- und Bild-Server zugegriffen. Die benötigte URI-Adresse wird in einer Variablen gespeichert, sowie die „*WebClient*“ Objekte deklariert:

```
//REST Uri
Uri redir;
//Web Clients:
var clientALK = new WebClient();
var clientMap = new WebClient();
var clientDGM = new WebClient();
```

Die Koordinaten der Bounding Box, welche die Abmessungen des Kartenausschnitts definiert, werden aus dem String Array („*xY*“) ausgelesen und in 4 separate String Variablen gespeichert. Hierbei wird gleichzeitig das Dezimalkomma durch einen Punkt ausgetauscht, da das Kommazeichen innerhalb der REST URL als Trennzeichen interpretiert werden würde:

```
//Bounding Box Coordinates:
string strXMin = xY[0].Replace(",", ".");
string strXMax = xY[1].Replace(",", ".");
string strYMin = xY[2].Replace(",", ".");
string strYMax = xY[3].Replace(",", ".");
```

Die Pixel Anzahl des DGM Ausschnitts wird auf ein Drittel der Kartenausschnitte festgelegt. Dies stellt eine hinreichende Anzahl an Punkten dar, um mittels FME ein TIN generieren zu können:

```
//DGM Size (1/3 Map Size):
string dgmX = (MapX / 3).ToString();
string dgmY = (MapY / 3).ToString();
```

Als letztes werden die Werte für die Größe des PDF Dokuments ausgelesen und zwei Variablen zugewiesen:

```
//Map Size: // Maximum = 2000 px
string strMapX = MapX.ToString();
string strMapY = MapY.ToString();
```

Da nun sämtliche für die Erstellung des REST Zugriffs benötigten Daten vorliegen können die entsprechenden URL erzeugt und ihren *„WebClient“*-Objekten zugewiesen werden. Die vom ArcGIS Server zurückgelieferten Rasterdateien werden in Byte Arrays gespeichert und unter Verwendung einer weiteren Methode (*„saveToDisc()“*) auf dem FME Rechner gesichert. Somit stehen sie anschließend der FME Workbench für die Durchführung der Konversion zur Verfügung:

```
//Generate URIs, Read through Web Client
//ALK
redir = new Uri(ConfigurationSettings.AppSettings["3DPDF_ALK_URI"] +
"?bbox=" + strXMin + "," + strYMin + "," + strXMax + "," + strYMax +
"&size=" + strMapX + "," + strMapY +
"&f=image&format=png");
//Save in Byte []
byte[] bALK = clientALK.DownloadData(redir);
saveToDisc(bALK, ConfigurationSettings.AppSettings["3DPDF_FME_DIR"] + "schrift.png");
```

Die URL der Kartenserver sowie des Zielverzeichnisses sind hier als Variablen definiert, welche in der *„app.config“* des entsprechenden Projekts beliebig angepasst werden können. In der Methode *„savetoDisc()“* wird lediglich das Byte Array mit der entsprechenden Rasterdatei gespeichert:

```
private void saveToDisc(byte[] file,string path)
        {
            IntPtr token = IntPtr.Zero;
            LogonUser("user",
                        "domain",
                        "password",
                        (int)LogonType.NewCredentials,
                        (int)LogonProvider.WinNT50,
                        ref token);

            using (WindowsImpersonationContext context =
WindowsIdentity.Impersonate(token))
            {
                CloseHandle(token);
                using (BinaryWriter writer =
                new BinaryWriter(File.Open(path, FileMode.Create)))
                {
                    writer.Write(file);
                }
            }
        }
```

Hierbei wird unter Einbindung von 2 DLL (`advapi32.dll` und `kernel32.dll`) eine sogenannte *„Impersonation“* durchgeführt, um auf den FME Rechner zugreifen zu können. Dadurch wird der .NET Anwendung innerhalb dieser Methode ein anderes Benutzerkonto zugewiesen, welches über Lese- und Schreibrechte auf die Verzeichnisse des FME Rechners verfügt. Dies gilt nur für die Operationen, die innerhalb des *„Impersonation“* Kontext ausgeführt werden. Anschließend wird die *„Impersonation“* beendet und die .NET Anwendung verfügt lediglich wieder über die ihr vorher gegebenen Rechte.
An eigentlichen Operationen wird innerhalb dieses Kontext lediglich ein *„BinaryWriter“* Objekt erzeugt, welches in ein Datei-Objekt die Daten aus dem übergebenen Byte Array (*„byte[] file“*) schreibt.
Somit werden mit Hilfe der beiden Methoden *„getMaps()“* und *„saveToDisc()“* die benötigten Rasterdaten erzeugt und in Dateien gespeichert, welche anschließend als Eingabedaten der FME Workbench dienen.

Das Starten der FME Anwendung erfolgt innerhalb der Methode „*startFME()*“. Als Übergabeparameter wird die ID des auszuführenden Auftrags übergeben. Dies dient jedoch lediglich der Benennung der fertigen PDF Datei, welche neben dem Datum auch die ID des Auftrags aufweist. Um vom Web Server, auf dem die Anwendung ausgeführt wird, auf einem anderen Rechner einen Prozess zu starten, wird das PSEXEC Werkzeug von Microsoft/Sysinternals benötigt. Damit ist es möglich, über eine Remote Zugriff auf einem anderen Rechner einen Prozess (hier die FME Anwendung) auszuführen. Dafür muss die Datei PSEXEC jedoch auf dem Web Server installiert sein. Um den Prozess starten zu können, müssen darüber hinaus die Zugangsdaten eines Kontos mit entsprechenden Rechten auf dem Zielrechner bekannt sein. Diese werden dann als Parameter an PSEXEC übergeben. Weitere benötigte Angaben sind der Pfad des auszuführenden Prozesses, sowie optionale Parameter für diesen. Der Aufruf von PSEXEC erfolgt mittels folgender Syntax:

```
PSEXEC.EXE \\Zielrechner –u Benutzername –p Passwort Pfad Parameter
```

Um einen externen Prozess aus einer .NET Anwendung heraus aufrufen zu können, wird die Klasse „*Process*“ aus dem „*System.Diagnostics*“ Namensraum benötigt:

```
Process FME = new Process();
```

Dem initialisierten „*Process*“ Objekt werden nun die benötigten Parameter übergeben. Dies geschieht in der „*StartInfo*“ Eigenschaft:

```
FME.StartInfo.FileName = ConfigurationSettings.AppSettings["3DPDF_PSEXEC_DIR"];
string args = @"/AcceptEULA \\ripsfme -u server -p pass" +
              @" ""c:\programme\fme\fme.exe"" " +
              @" ""d:\fme_3D_test\3dpdf_local.fmw"" ";
FME.StartInfo.Arguments = args;
FME.StartInfo.UseShellExecute = false;
```

Benötigt wird neben dem Dateinamen („*FileName*“) auch die Argumente, welche an den ausführenden Prozess übergeben werden sollen(„*Arguments*“). Somit wird der allgemeine Aufruf von PSEXEC hier umgesetzt. In der „*FileName*“ Eigenschaft wird der Pfad angegeben, in den Argumenten Zielrechner, Benutzername, Passwort, Zielprozess sowie Argumente für diesen Zielprozess.

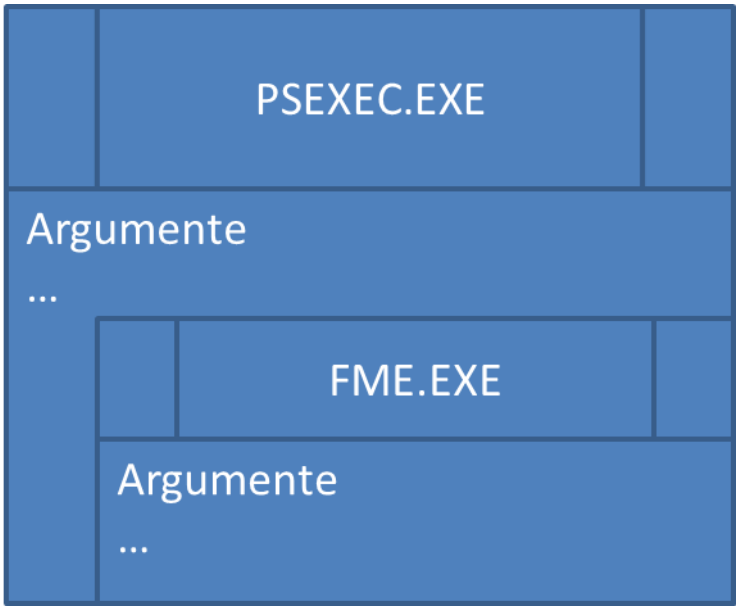

Abbildung 14: PSEXEC.EXE

Anschließend kann der Prozess gestartet werden. Es wird nun innerhalb einer „*While*“-Schleife alle 2 Sekunden geprüft, ob der Prozess beendet wurde:

```
FME.Start();
while (!FME.HasExited)
{
	FME.Refresh();
	Thread.Sleep(2000);
}
```

Sobald dies der Fall ist, wird die Schleife verlassen, und der Exit Code des Prozesses überprüft. Ein Exit Code von 0 bedeutet im Falle einer Windows Anwendung, dass diese erfolgreich ausgeführt wurde.

```
if (FME.ExitCode == 0)
{
	…
}
else
{
	//Fehlerbehandlung
}
```

Wird ein anderer Exit Code zurückgegeben, so trat bei der Ausführung ein Fehler auf, welcher im „*Else*“- Teil der Schleife behandelt werden kann. Im Falle des erfolgreichen Beendens des Prozesses wird nun, im „*If*“- Teil, das von FME erzeugte PDF Dokument in ein Byte Array eingelesen und der Prozess PSEXEC/FME beendet:

```
byte[] bArr_pdf;
string path = ConfigurationSettings.AppSettings["3DPDF_FME_DIR"] +"3D_Test.pdf";
FME.Close();
using (BinaryReader reader =
	new BinaryReader(File.Open(path, FileMode.Open)))
	{
		bArr_pdf = reader.ReadBytes((int)reader.BaseStream.Length);
	}
```

Das PDF Dokument ist nun im Byte Array *„bArr_pdf“* abgelegt. Anschließend muss es nun noch auf dem Web Server gespeichert werden. Dadurch wird sichergestellt, dass der anfragende Benutzer auch auf sein Dokument zugreifen kann, ohne hierfür den FME Rechner öffentlich zugänglich machen zu müssen:

```
string date = DateTime.Today.Day + "." + DateTime.Today.Month.ToString() + "." +
DateTime.Today.Year;
path = ConfigurationSettings.AppSettings["3DPDF_OUTPUT_DIR"] + date + "_" + id +
".pdf";
using (BinaryWriter writer =
new BinaryWriter(File.Open(path, FileMode.Create)))
{
       writer.Write(bArr_pdf);
}
```

Die Dateibezeichnung des fertigen PDF soll aus dem Datum sowie der Auftrags-ID bestehen. Daher wird ein String *„date“* erzeugt, welcher den aktuellen Tag, Monat sowie das Jahr enthält. Dieser String wird zusammen mit der an die Methode übergebenen ID als Dateiname verwendet. Mittels der *„BinaryWriter“* Klasse wird das PDF Dokument schließlich unter dem festgelegten Pfad abgespeichert.

Somit verbleibt noch das Senden einer Benachrichtigungs-Email an den Benutzer. Des geschieht innerhalb der *„sendEmail()“* Methode. Übergabeparameter sind die Auftrags-ID(*„id“*), sowie die E-Mail-Adresse des Benutzers(*„to“*):

```
public static void sendEmail(string to,int id)
        {
              …
        }
```

Aus dem *„System.Net.Mail“* Namensraum wird die Klasse *„MailMessage“* benötigt, ebenso die Klasse *„SmtpClient“*, welche den Server zum Versenden der Email repräsentiert:

```
MailMessage mail = new MailMessage();
SmtpClient smtp = new SmtpClient();
```

In mehreren Strings werden benötigte Informationen festgelegt, wie der anzuzeigende Absender, das Passwort des Mail Servers etc.:

```
string psaMailServer = ConfigurationSettings.AppSettings["3DPDF_MailServer"];
string from = ConfigurationSettings.AppSettings["3DPDF_MailFrom"];
string psaMailPWD = ConfigurationSettings.AppSettings["3DPDF_MailPassword"];
string msgBody = string.Empty;
```

Die hier benötigten Informationen werden aus der *„app.config“* Datei der Anwendung ausgelesen, um spätere Anpassungen einfacher vornehmen zu können. Anschließend wird dem *„MailMessage“* Objekt die benötigten Parameter übergeben:

```
mail.From = new MailAddress(from, "Absender");
mail.To.Add(to);
mail.Subject = "Ihr angefordertes 3DPDF";
mail.Body = "Unter folgendem Link können Sie ihr erstelltes PDF abrufen: " +
ConfigurationSettings.AppSettings["3DPDF_OUTPUT_DIR"] + "3DPDF_" + id + ".pdf";
```

```
mail.IsBodyHtml = true;
```

Der Absender (aus der *„app.config“*-Datei) und der Empfänger (aus der Datenbank) werden festgelegt, ebenso der Inhalt der E-Mail. Für Testzwecke werden hier lediglich Platzhalter eingetragen. Im späteren Betrieb kann hier entsprechender Text ergänzt werden. Um die so definierte Email nun versenden zu können, müssen auch dem *„SmtpClient“* Objekt die entsprechenden Parameter (Server und Zugangsdaten) übergeben werden:

```
smtp.Host = psaMailServer;
smtp.Credentials = new NetworkCredential(@from.Substring(0, @from.IndexOf("@")),
psaMailPWD);
smtp.Send(mail);
```

Mit der *„Send()“* Methode wird die E-Mail schließlich an den Empfänger versendet.

4.5.3 Anpassung an die WPS Schnittstelle

Die Methode zum Speichern des Datenbankeintrags (*„CREATE_PDF_JOB“*) soll nun in die WPS Schnittstelle von RipsWeb integriert werden, um aus den bestehenden Desktop- und Web-Anwendungen heraus auf die neuen Funktionalitäten zugreifen zu können.
Die WPS Schnittstelle bietet einen Zugriff auf die Geofunktionen der RipsWebservices.

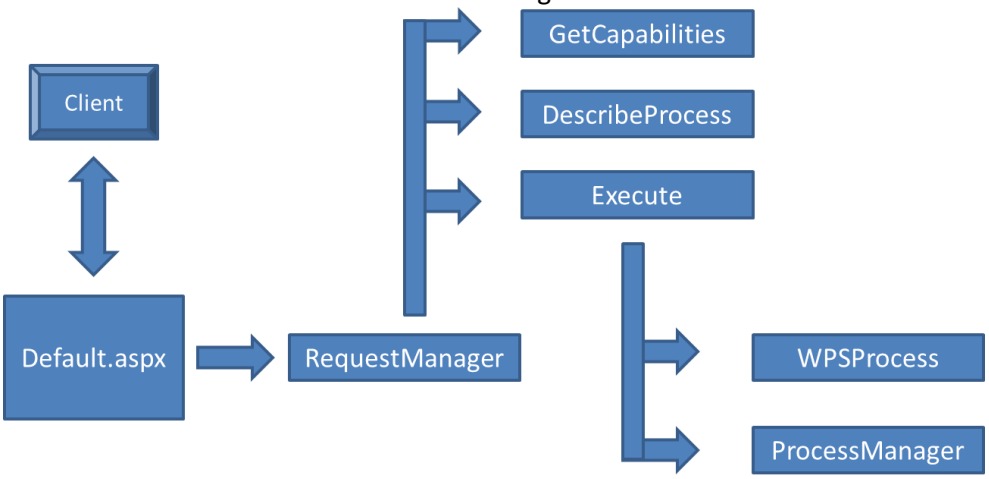

Abbildung 15: Struktur RipsWeb.WPS, nach: [Haberer_2011, S. 14]

Das Hinzufügen eines neuen Prozesses erfordert mehrere Schritte. Zunächst müssen die in XML Dokumenten hinterlegten Metainformationen ergänzt und angepasst werden. Das OGC konforme WPS Schema ist in der WPS Schnittstelle umgesetzt. Die Metainformationen zu den WPS Funktionen *„getCapabilities“*, *„describeProcess“* sowie *„Execute“* liegen in jeweils separaten Dokumenten vor. Entsprechend müssen beim Hinzufügen eines neuen Prozesses diese Informationen an allen Stellen ergänzt werden. Daher werden zunächst in den *“getCapabilities“* Dokumenten die entsprechenden Informationen eingetragen. Im *„getCapabilities“* Dokument werden sämtliche vom WPS Dienst angebotenen Dienste beschrieben (*„ProcessOfferings“*). Hier werden die vorhandenen Einträge um einen neuen Eintrag für den 3DPDF-Dienst ergänzt:

```
<wps:Process wps:processVersion="1.0.0">
      <ows:Identifier>PDF</ows:Identifier>
      <ows:Title>PDF</ows:Title>
      <ows:Abstract>Speichert einen 3DPDF Auftrag in die Datenbank</ows:Abstract>
</wps:Process>
```

Der sogenannte Identifier stellt hierbei die wichtigste Information dar, da hierrüber später die beiden anderen WPS Operationen *„describeProcess“* sowie *„Execute“* auf den Prozess zugreifen können. Titel und Beschreibung *(„Title“,“Abstract“)* enthalten Informationen, welche den Benutzer über die

zur Verfügung gestellten Funktionen informieren.
Da innerhalb der WPS Schnittstelle XML Dateien für mehrere verschiedene Webserver vorliegen, müssen diese Informationen überall dort ergänzt werden, wo die 3D PDF Funktionalität anschließend erwünscht ist.
Ist der Prozess in den XML Dateien eingetragen, so wird er zukünftig bei allen *„getCapabilities“* Anfragen an den WPS Server angezeigt. Im Rahmen des WPS Schema erfolgt die detaillierte Beschreibung eines Prozesses mit Hilfe einer *„describeProcess“* Anfrage. Die vom Server zu gebende Antwort wird in der Rips WPS Schnittstelle im Dokument *„DescribeProcess_Response.xml“* hinterlegt.
Innerhalb des Wurzelelements *„ProcessDescriptions“* werden hier die Beschreibungen der einzelnen Prozesse abgelegt. Für den PDF Dienst wird eine eigene Prozessbeschreibung (*„ProcessDescription“*) angelegt:

```
<ProcessDescription statusSupported="false" storeSupported="false"
wps:processVersion="1.0.0">
    <ows:Identifier>PDF</ows:Identifier>
    <ows:Title>3DPDF</ows:Title>
    <ows:Abstract>Speichert einen Datenbankeintrag fuer den asynchronen
3DPDFDienst</ows:Abstract>
```

Der Identifikator, der Titel sowie die Beschreibung sollten hierbei identisch zu jenen im *„getCapabilities“* Dokument sein. Neben den auch im *„getCapabilities“* enthaltenen Informationen liefert die *„describeProcess“* Methode auch Hinweise zu Ein- und Ausgabeparametern des Prozesses zurück. Dies geschieht innerhalb des `DataInputs`, respektive des `ProcessOutputs` Elements.
Die hier hinterlegten Parameter müssen jeweils einen eindeutigen Identifikator erhalten. Optional sind Titel und Beschreibung. Der Datentyp wiederrum muss angegeben werden. Beispielhaft soll hier der Parameter *„eMail“* dargestellt werden:

```
<Input minOccurs="1" maxOccurs="1">
        <ows:Identifier>eMail</ows:Identifier>
        <ows:Title>eMail Adresse</ows:Title>
        <ows:Abstract>Adresse, an welche das fertig gerechnete PDF geschickt werden
soll</ows:Abstract>
        <LiteralData>
          <ows:DataType>http://www.w3.org/2001/XMLSchema.xsd#~string</ows:DataType>
        </LiteralData>
      </Input>
```

Der Parameter muss genau einmal vorkommen (*„minOccurs“* und *„maxOccurs“* jeweils 1). Identifikator, Name und Beschreibung sind definiert. Der Datentyp ist String.
Analog werden Ausgabeparameter beschrieben. Die WPS Parameter bilden in diesem Fall genau die Parameter der *„CREATE_PDF_JOB“* Methode der Klasse *„_3DPDF“* ab. Somit wird sichergestellt, dass die Methode auch über WPS angesprochen und ausgeführt werden kann.
Im Rahmen des WPS Schemas wird ein Prozess mittels des *„Execute“* Befehls ausgeführt. In der RipsWeb WPS Schnittstelle wird für Testzwecke für jeden *„Execute“* Befehl ein eigenes XML Dokument angelegt, so auch für den PDF Dienst. Das XML Dokument wird mit Beispielwerten für die Eingabe- und Ausgabeparameter versehen. Dadurch kann ein Dienst direkt aus dem WPS Projekt heraus getestet werden, ohne dass ein externer Zugriff auf die WPS Schnittstelle erfolgen muss. Sind die XML Dateien ergänzt und angelegt kann die eigentliche Funktionalität in die WPS Schnittstelle integriert werden. Im Projekt *„RipsWeb.WPS.Infrastructure“* sowie in der *„app.config“* Datei werden Identifikatoren für den PDF Dienst ergänzt.
Die eigentliche Durchführung der WPS Operationen geschieht innerhalb der *„ProcessManager“* Klasse des WPS Projekts. Für jeden Prozess erfolgt die Anbindung an die *„RipsWeb.dll“* hier über eine entsprechende Methode. Der 3DPDF-Dienst wird in der Methode *„PDF()“* angebunden. Innerhalb der Methode werden die von der WPS Schnittstelle übergebenen Parameter in von der

„_3DPDF“ Klasse der *„RipsWeb.dll“* verwendbare Parameter umgewandelt und schließlich die Methode zum Erzeugen des PDF Dokuments aufgerufen.
Die meisten Operationen sind für die Umwandlung der Koordinaten erforderlich. Da die Eingabe über die WPS Schnittstelle die Bounding Box des PDF als GML Datentyp *„Polygon“* erfolgt, müssen aus diesem Polygon die minimalen und maximalen X- und Y-Werte ausgelesen und in ein String Array geschrieben werden. Über die vorhandene GML Parser Klasse werden aus dem übergebenen Polygon alle Koordinaten der Punkte extrahiert und in einem String Array gespeichert:

```
string[] coords = GetGeometry(process).Coordinates.Split('|');
```

Innerhalb dieses Array liegen nun alle Punkte des eingegebenen Polygons als X- und Y-Paare, getrennt durch einen Semikolon, vor. Unter Verwendung zweier *„ArrayList“* Elemente wird das Koordinatenarray anschließend durchlaufen. X-Werte werden in die erste, Y-Werte in die zweite *„ArrayList“* gespeichert:

```
System.Collections.ArrayList arrLstX = new System.Collections.ArrayList();
System.Collections.ArrayList arrLstY = new System.Collections.ArrayList();
foreach (string s in coords)
       {
              arrLstX.Add(s.Split(';')[0]);
              arrLstY.Add(s.Split(';')[1]);
       }
```

Die Verwendung von Array-Listen erlaubt das anschließende Sortieren dieser Listen. In den sortierten Listen stehen somit die kleinsten X- bzw. Y-Werte an erster Stelle. Die größten Werte an letzter Stelle. Somit kann durch Zugriff auf diese beiden Positionen der kleinste bzw. größte Wert für X und Y ermittelt werden. Die extrahierten Werte werden abschließend in einem String Array (*„xY“*) gespeichert und dienen danach als Eingabeparameter für die PDF Methode:

```
arrLstX.Sort();
string xMin = arrLstX[0].ToString();
string xMax = arrLstX[arrLstX.Count - 1].ToString();
arrLstY.Sort();
string yMin = arrLstY[0].ToString();
string yMax = arrLstY[arrLstX.Count - 1].ToString();
string[] xY = { xMin, xMax, yMin, yMax };
```

Die übrigen Parameter können direkt aus dem WPS Prozess extrahiert und in die benötigten Datenformate umgewandelt werden:

```
//Size X,Y of PDF:
int sizeX = Convert.ToInt32(process.DataInputs[1].Value);
int sizeY = Convert.ToInt32(process.DataInputs[2].Value);
//Email address:
string eMail = process.DataInputs[3].Value.ToString();
```

Somit sind alle Parameter verarbeitet und die Methode zum Erstellen des PDF kann aufgerufen werden. Als Rückgabewert liefert die Methode einen String mit einer entsprechenden Erfolgsmeldung zurück. Dieser String kann nun an die WPS Schnittstelle als Rückgabewert übergeben werden. Im Erfolgsfall wird dann die im String enthaltene Meldung an den Benutzer zurückgegeben, andernfalls die entsprechende Fehlermeldung:

```
string result = RipsWeb.ArcGisServer.Processing.DataServices._3DPDF.CREATE_PDF_JOB(xY,
sizeX, sizeY, eMail);
OutputDataType outputParam = process.ProcessOutputs.Output.First();
outputParam.OutputDataTypeData.Data.LiteralData.Text = result;
```

```
SetProcessStatusToProcessSucceeded(process);
```

Da die Aufgabe des WPS Dienstes lediglich das Erfassen von Aufträgen für die PDF Erstellung ist, muss die Methode zum eigentlichen Erstellen des PDF an anderer Stelle ausgeführt werden. Da im Rahmen der Konzeption festgelegt wurde, dass der Dienst asynchron umgesetzt werden soll, bietet es sich an, die bestehende Infrastruktur zum Abarbeiten von asynchronen Aufträgen zu verwenden.

4.5.4 Integration in den bestehenden asynchronen Windows Dienst

Um asynchrone Dienste im Rips Umfeld auszuführen, existiert bereits ein Windows Dienst, welcher dies ermöglicht. Zur besseren Integration soll die Funktionalität zum Erzeugen von 3D PDF ebenfalls mit Hilfe dieses Dienstes umgesetzt werden. Einen neuen, separaten Dienst zu erstellen wäre auch möglich, jedoch hätte ein weiterer, parallel laufender Dienst zusätzliche Systemressourcen beansprucht, ohne einen direkten Vorteil zu Bieten. Der bestehende Windows Dienst *„Rips.AsyncServices“* ermöglicht das Ausführen von Methoden der RipsWeb.dll alle 40 Sekunden, jede Stunde oder auch einmal am Tag. Um die Wartezeit des Benutzers möglichst gering zu halten, wird die PDF Funktionalität zu den alle 40 Sekunden ausgeführten Diensten hinzugefügt.
Bei einem Windows Dienst handelt es sich im Gegensatz zu einem regulären Windows Programm um eine ständig laufende, ausführbare Datei. Der Dienst muss auf dem Zielsystem installiert werden. Anschließend wird der Dienst gestartet, und führt seine Operationen in periodischen Abständen durch. Ein Windows Dienst verfügt über Methoden, welche beim Starten, Stoppen oder Pausieren des Dienstes ausgeführt werden. Innerhalb dieser Methoden können dann die zu diesem Ereignis erwünschten Methoden integriert werden. Beim Dienst *„Rips.AsyncServices“* ist die Umsetzung so gelöst, dass innerhalb der *„onStart()“* Methode des Dienstes eine Methode *„runAllThreads()“* gestartet wird. In dieser Methode wiederrum wird überprüft, ob für die für die unterschiedlichen Operationen angesetzte Zeit verstrichen ist, also 40 Sekunden, eine Stunde oder ein Tag. Ist dies der Fall so werden die entsprechenden Methoden *„run40SecThreads()“*, *„runHourlyThreads()“* oder *„runDailyThreads()“* aufgerufen. Anschließend wird der Thread für 40 Sekunden angehalten, und dann erneut die verstrichene Zeit überprüft.
Da die Datenbank relativ häufig auf vorhandene PDF Aufträge abgefragt werden soll, wird die PDF Erzeugung in die Methode *„run40SecThreads()“* integriert. Hierzu wird eine einfache Methode *„execute3DPDFService()“* erstellt. In der Methode wird nun lediglich überprüft, ob die PDF Erstellung in der *„app.config“* des Dienstes aktiviert wurde. Ist dies der Fall, so wird aus der RipsWeb.dll die vorher erstellte Methode *„CREATE_PDF_JOB()“* der Klasse *„_3DPDF“* gestartet, welche die entsprechenden Funktionalitäten enthält:

```
private void execute3DPDFService()
{
string runService = System.Configuration.ConfigurationSettings.AppSettings["RUN_3D-
Erstellung"];
if (runService.ToUpper() != "TRUE")
      return;
RipsWeb.ArcGisServer.Processing.DataServices._3DPDF o = new
RipsWeb.ArcGisServer.Processing.DataServices._3DPDF();
o.EXECUTE_PDF_JOB();
}
```

Der Aufruf dieser Methode erfolgt nun in der *„run40SecThreads()“* Methode. Dadurch wird sichergestellt, dass alle 40 Sekunden eine Abfrage an die Datenbank gesendet wird. Liegt dort ein neuer Auftrag vor, so wird das PDF erstellt und eine E-Mail an den Benutzer geschickt.

Damit der Windows Dienst richtig ausgeführt wird, muss er auf einem Server installiert werden. Anschließend muss die RipsWeb.dll, mitsamt der neuen Klasse *„_3DPDF“* in das Installationsverzeichnis des Dienstes kopiert werden. Die vorhandene Vorgängerversion der .dll-Datei wird ersetzt. So wird sichergestellt, dass die neuen Funktionalitäten dem Service zur Verfügung stehen. So lange der Dienst nun läuft, wird alle 40 Sekunden die Datenbank auf neue Aufträge abgefragt und diese dann bearbeitet.

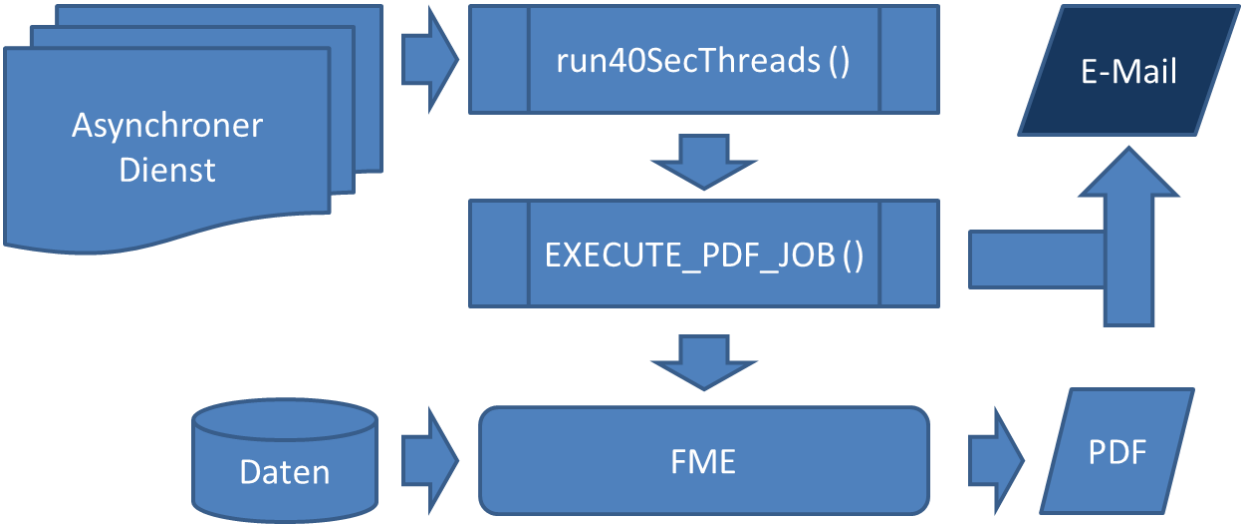

Abbildung 16: Asynchroner Windows Dienst

5 Erweiterung der Anwendung

Nachdem nun die grundsätzliche Funktionalität der Anwendung umgesetzt und die Machbarkeit gezeigt wurde, sollen im nächsten Schritt verschiedene Erweiterungen durchgeführt werden. Momentan werden innerhalb der PDF Erstellung feste Werte für die Geländeauflösung und die Überhöhung der 3D Darstellung verwendet. Dies soll durch dynamische Werte, in Abhängigkeit des dargestellten Gebiets, ersetzt werden. Weiterhin ist es für verschiedene Anwendungsfälle sinnvoll, andere Themen dreidimensional darzustellen. Schließlich soll die Visualisierung von Windenergieanlagen im Gelände umgesetzt werden.

5.1 Variable Überhöhung und DGM Auflösung

Um eine optimale dreidimensionale Darstellung, unabhängig vom gewählten Gebiet, zu erreichen, müssen DGM Auflösung und Geländeüberhöhung dynamisch umgesetzt werden. Hierbei ist insbesondere auch auf die Balance zwischen Genauigkeit und Dateigröße zu achten. Es wird angestrebt, die Größe der erzeugten PDF Dateien möglichst im Bereich von bis zu 1 Megabyte zu halten. Zu diesem Zweck wird mit verschiedenen Gebietsgrößen und Auflösungen die Auswirkung auf die Dateigröße überprüft. Zunächst werden feste Werte für bestimmte Maßstabsbereiche verwendet, wie aus folgender Tabelle ersichtlich:

Seitenlänge des PDF (Km)	DGM Auflösung (m)	Überhöhung	Dateigröße bei 1000x1000 Pixeln
<1	1	1	270kB
<10	1	1	775kB
<50	25	1.5	1054kB
<200	50	2	743kB
>200	100	3	1196kB

Tabelle 2: DGM Auflösung und Überhöhung

Baden-Württemberg hat in Nord-Süd Richtung eine Ausdehnung von etwa 250 km, in Ost-West Richtung etwa 200km. Daher bewegt sich auch das größte darzustellende Gebiet in diesem Bereich. Eine optimale DGM Auflösung stellt hier 100 Meter pro Pixel dar. Bei einer Dateigröße von 1000 mal 1000 Pixeln und einer Seitenlänge des darzustellenden Gebiets von 200 km ergibt sich eine Strecke von 200 Metern pro Pixel. Daher wäre prinzipiell eine DGM Auflösung von ebenfalls 200 Metern ausreichend. Da sich jedoch die Dateigröße bei der Verwendung eines 200 Meter Rasters nur unwesentlich ändert und die TIN Interpolation mit einem 100 Meter Raster bessere Ergebnisse liefert, wurde die DGM Auflösung für den Bereich von über 200 km Seitenlänge auf 100 Meter festgelegt. Um eine gute Geländedarstellung zu erreichen wurde eine dreifache Überhöhung des Geländes gewählt. Das verwendete DGM5 von Baden-Württemberg verfügt über Höhenwerte von 83 bis 1494 Metern. Der Höhenunterschied beträgt somit etwa 1400 Meter. Durch die dreifache Überhöhung ergibt sich ein wesentlich plastischeres Geländebild, ohne die Darstellung zu sehr zu verfälschen.

Im Bereich von 50 km bis 200 km Seitenlänge des PDF wird die DGM Auflösung auf 50 Meter festgelegt. Dadurch ergibt sich bei 50 km pro 1000 Pixel ein Höhenpunkt pro Pixel. Bei größeren Seitenlängen entsprechend mehr Punkte, wobei sich die Auswirkungen auf die Dateigröße in Grenzen halten. Grund hierfür ist, dass bei der TIN Erzeugung mittels FME aus einem gegebenen Raster nur die benötigten Punkte interpoliert werden. Dadurch kann die DGM Auflösung von 50 Metern für den ganzen Bereich von 50 km bis 200 km verwendet werden. Die Überhöhung wird reduziert auf den zweifachen Wert.
Für den Bereich von 10 km bis 50 km wird die DGM Auflösung auf 25 Meter festgelegt. Dies bedeutet bei einer Seitenlänge von 50 km, dass nur für jeden zweiten Pixel Höheninformationen vorliegen, was jedoch eine ausreichende Anzahl an Punkten für die Interpolation des TIN darstellt. Ab 25 km Seitenlänge liegt wiederrum für jeden Pixel ein Höhenwert vor. Reduziert sich die Seitenlänge weiter, so ergeben sich entsprechend mehrere Höhenwerte pro Pixel, aus denen der geeignete Wert interpoliert wird. Die Überhöhung wird weiter reduziert, auf das 1,5-fache der tatsächlichen Höhe.
Für PDF Dokumente mit Seitenlängen von weniger als 10 km werden pauschal Werte von 1 Meter Auflösung und einfacher Überhöhung verwendet. Da die Auflösung des DGM5 nur 5 Meter pro Pixel beträgt, stellt dies auch die maximale Auflösung dar. Durch die Angabe von 1 Meter wird jedoch sichergestellt, dass eine eventuelle Verwendung eines höheraufgelösten DGM in Zukunft möglich bleibt.
Generell stellt das Festlegen von exakten Werten für bestimmte Maßstabsbereiche einen Kompromiss dar. Ein Lösungsansatz wäre, die reale Wegstrecke durch die Anzahl der sie repräsentierenden Pixel zu teilen. Damit wäre sichergestellt, dass für jeden Pixel ein Höhenwert aus dem DGM vorliegt.
Um dies umzusetzen, wird der Klasse *„_3DPDF“* eine neue Methode hinzugefügt. Mit Hilfe dieser Methode soll die DGM Auflösung und die Überhöhung für beliebige Seitenlängen des 3D PDF ermittelt werden. Die benötigten Parameter sind somit die Bounding Box (zum Ermitteln der Seitenlängen) sowie die Anzahl der Pixel in X- und Y-Richtung des PDF:

```
public string[] surfaceTolerance(string[] xY,int MapX, int MapY)
{
	string [] surfUeber = new string [2];
	…
	return surfUeber;
}
```

Zunächst werden in der Methode die Seitenlängen in X und Y ermittelt. Der größte Wert von beiden wird in einer neuen Variablen gespeichert:

```
double xDif = Convert.ToDouble(xY[1]) - Convert.ToDouble(xY[0]);
double yDif = Convert.ToDouble(xY[3]) - Convert.ToDouble(xY[2]);
//max Dif (X or Y)
double mDif = 0;
if (xDif > yDif) mDif = xDif;
else mDif = yDif;
```

Ebenso wird der größte Pixelwert für X und Y ermittelt und gespeichert:

```
int pixel = 0;
if (MapX > MapY) pixel = MapX;
else pixel = MapY;
```

Nun wird die Seitenlänge durch die Pixelanzahl dividiert. Das Ergebnis stellt die erforderliche DGM

Auflösung dar. Durch die Konvertierung zu einem Integer Datentyp wird sichergestellt, dass das Ergebnis auf einen ganzzahligen Wert gerundet wird. Ebenso wird die Auflösung auf einen minimalen Wert von 1 festgelegt. Die DGM Auflösung wird anschließend an die erste Stelle des Rückgabe-Array geschrieben:

```
int dgm = (int)(mDif / pixel);
if (dgm < 1) dgm = 1;
surfUeber[0] = (dgm / 2).ToString();
```

An zweiter Stelle des Array wird nun noch die Überhöhung gespeichert. Die Überhöhung wird auf feste Werte, analog zu Tabelle 2, festgelegt. Von dreifacher Überhöhung bei Seitenlängen von über 200km, bis zu einfacher Überhöhung für Seitenlängen unter 10 km.
Das von der Methode zurückgegebene Array wird nun als Übergabeparameter für die FME Workbench verwendet. Hierzu muss die Workbench erweitert werden. Der Z-Wert des „*Scaler*“-Werkzeugs wird als Parameter „*Ueberh*“ definiert. Die Oberflächentoleranz des „*TIN*“-Werkzeugs entsprechend als Parameter „*Surf_Tol*“. Beim Kommandozeilenaufruf über PSEXEC werden die Parameter anschließend wie folgt übergeben:

```
"<Pfad zu FME>" "<Pfad zur Workbench>"
"--Surf_Tol" + surfTol[0] + "--Ueberh" + surfTol[1];
```

Somit werden bei jedem neu erstellten PDF geeignete Werte für Überhöhung und DGM Auflösung verwendet.

5.2 Visualisierung anderer Themen und von Luftbildern

Um bestimmte Themen oder auch orthographische Luftbilder als Textur für das Oberflächenmodell zu verwenden, müssen einige Anpassungen an der Anwendung vorgenommen werden. Zunächst muss eine neue FME Workbench angelegt werden, welche später gestartet werden kann. Die Workbench ist ähnlich aufgebaut wie die bereits bestehende. Aus dem erzeugten DGM wird ebenso mit Hilfe des „*TIN*“- Werkzeugs ein Oberflächenmodell erzeugt. Lediglich die auf der Oberfläche anzubringende Textur ändert sich. Diese wird je nach angefragtem Thema aus unterschiedlichen Quellen abgerufen. Gespeichert wird die Textur jedoch bei jedem Aufruf von FME unter dem gleichen Pfad. Dies vereinfacht die Durchführung und bedeutet, dass die gleiche Workbench, unabhängig vom Thema, verwendet werden kann.
Um auf neue Themen zugreifen zu können, werden als erstes die Pfade zum Abrufen der entsprechenden Bilddateien in der „*app.config*“ festgelegt. Dadurch kann später zur Laufzeit darauf zugegriffen werden.
Um festzuhalten, welches Thema vom Benutzer angefordert wurde muss als nächstes die Tabelle in der Datenbank erweitert werden. Es wird eine neue Spalte THEMA eingefügt, worin die angebotenen Themen Integer-codiert abgelegt werden. Die Speicherung in der Datenbank ermöglicht es später, beim Durchführen der eigentlichen PDF Erstellung, jedem Auftrag das richtige Thema zuzuordnen. Um dem vom Anwender gewählten Thema die korrekten Bilddateien laden sowie die entsprechend richtige Workbench starten zu können, werden neue Methoden innerhalb der Klasse „*_3DPDF*“ ergänzt. Innerhalb einer Methode „*getTheme()*“ erfolgt, abhängig vom in der Datenbank abgelegten Thema, der Aufruf der Methoden, welche die entsprechenden Karten- oder Bilddateien laden. Umgesetzt wird dies durch eine einfache Switch-Anweisung:

```
switch (theme)
```

```
{
        case "1":
        getMaps(xY, MapX, MapY);
        break;
        case "2":
        getOrtho(xY, MapX, MapY);
        break;
        case "3":
        getWind(xY, MapX, MapY);

        …
}
```

Innerhalb der so aufgerufenen Methoden werden nun die benötigten Dateien, das DGM sowie die Karten- oder Bilddateien, geladen. Die Methode zum Starten der FME Workbench („*startFME()*“) muss ebenfalls angepasst werden, damit die richtige Workbench gestartet wird. Hierbei ändern sich nur die Argumente für den Aufruf von PSEXEC, wie den Pfad zur Workbench oder eventuell benötigte Parameter. In einer Switch-Anweisung, ähnlich der obigen, werden die Argumente an einen String übergeben:

```
switch (theme)
            {
                case "1":
                    args = …
                    break;
                case "2":
                    args = …
                    break;
                case "3":
                    args = …
                    break;
            }
```

Der String mit den Argumenten wird dann an den PSEXEC Prozess als Argumente übergeben:

```
FME.StartInfo.Arguments = args;
```

Durch diesen Aufbau wird eine einfache Erweiterbarkeit erreicht. Sollen neue Themen hinzugefügt werden, so muss lediglich eine entsprechende Methode erstellt werden. Anschließend müssen die beiden Switch-Anweisungen um einen weiteren Fall erweitert werden. Prinzipiell ist es so möglich, beliebige Themen mittels eines 3D PDF darzustellen.

5.3 Windenergieanlagen

Als einen weiteren Anwendungsfall für den Einsatz von 3D PDF sollen Windenergieanlagen in einer dreidimensionalen Umgebung dargestellt werden. Zusätzlich benötigt werden für diesen Fall allerdings noch 3D Modelle der Anlagen. Im Rahmen der Umsetzung muss daher zunächst ein entsprechendes 3D Modell erstellt werden. Verwendet hierfür wird die Software Sketchup von Google. Sketchup ermöglicht das relativ einfache Erstellen von maßstabsgetreuen Modellen für den Einsatz in 3D Umgebungen. Im Rahmen dieser Arbeit wird zunächst ein generisches Modell einer Windenergieanlage erstellt, um die Machbarkeit aufzuzeigen. Das erstellte Modell repräsentiert eine Anlage mit einer Nabenhöhe von 70 Metern, was für die sich zurzeit in Betrieb befindlichen Anlagen einen häufig vorkommenden Wert darstellt.

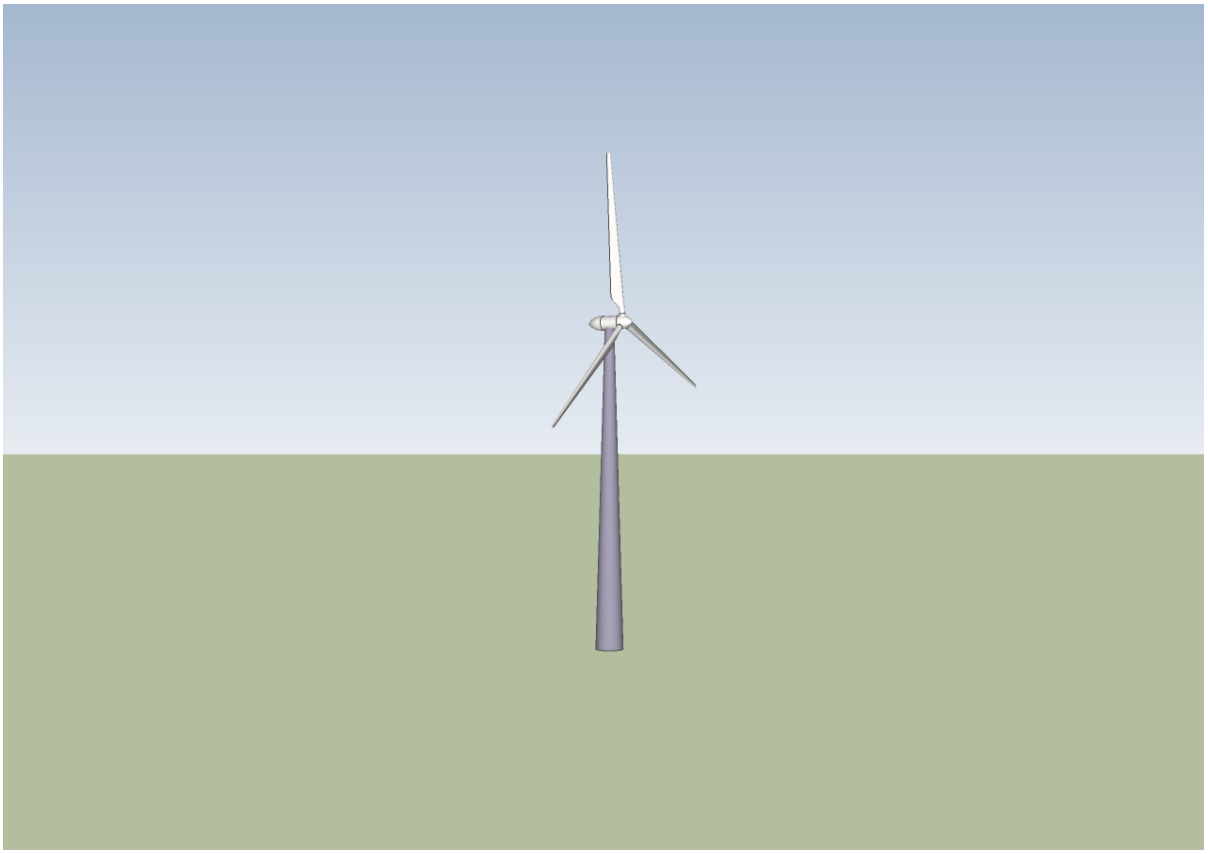

Abbildung 17: SketchUp Modell

Durch den Einsatz von FME ist es anschließend möglich, das erstellte Modell direkt in einer Workbench zu verwenden, und somit in einem 3D PDF darzustellen.
Die Darstellung der Anlagen soll an ihren tatsächlichen Standorten erfolgen, wobei diese bereits als Datensatz inklusive Koordinaten vorliegen. Um nun aus diesen verschiedenen Eingabedaten eine PDF Darstellung von Windenergieanlagen im Gelände umzusetzen, werden einige Werkzeuge von FME benötigt.
Zunächst muss aus den Eingangsdaten zu den Anlagenstandorten deren Koordinaten ermittelt werden. Anschließend wird aus dem übergebenen DGM eine Bounding Box erstellt. Mit dieser Bounding Box werden nun die Anlagenstandorte verschnitten. Ergebnis sind alle innerhalb des darzustellenden Gebietes vorhandenen Standorte.
Im nächsten Schritt muss das Sketchup Modell geladen und mit den ermittelten Standorten kombiniert werden. Hierfür wird mittels des „*Geometry Extractor*“ Werkzeugs die Geometrie des Modells ausgeschnitten. Diese Geometrie wird anschließend innerhalb einer FME Variable gespeichert („*Variable Setter*“).

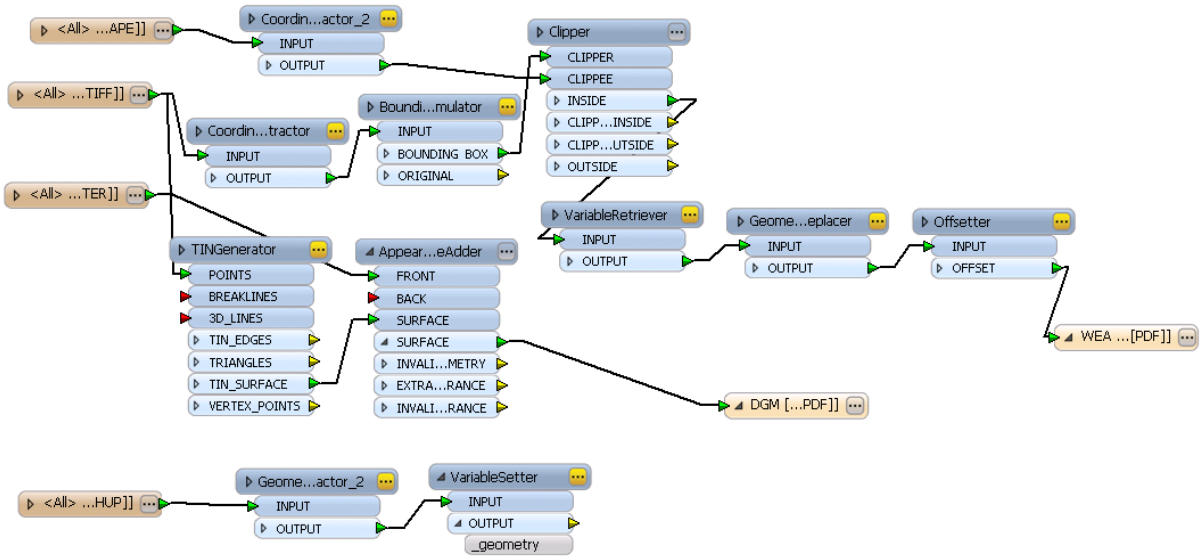

Abbildung 18: Windanlagen Workbench

Nun kann die Geometrie der ausgeschnittenen Anlagestandorte durch die in der Variablen gespeicherte Geometrie des Sketchup Modells ersetzt werden („*Variable Retriever*“, „*Geometry Replacer*“). Ergebnis ist das Modell einer Windenergieanlage pro vorhandene Koordinate. Um die Modelle nun korrekt im Raum platzieren zu können, wird mittels des „*Offsetter*“ Werkzeugs den Modellen wieder die ursprünglichen Werte für X, Y und Z zugewiesen.

Endergebnis sind somit Modelle von Windenergieanlagen an den korrekten Standorten. Das Hinzufügen von DGM und Textur der Oberfläche erfolgt auf gleiche Weise wie bei Dokumenten ohne zusätzliche 3D Modelle. Aus dem DGM wird ein TIN berechnet und dieses mit einer Textur versehen. Beim Schreiben des PDF Dokuments kombiniert FME nun die beiden Ebenen, Modelle und Oberfläche und die Anlagen werden korrekt im Gelände dargestellt.

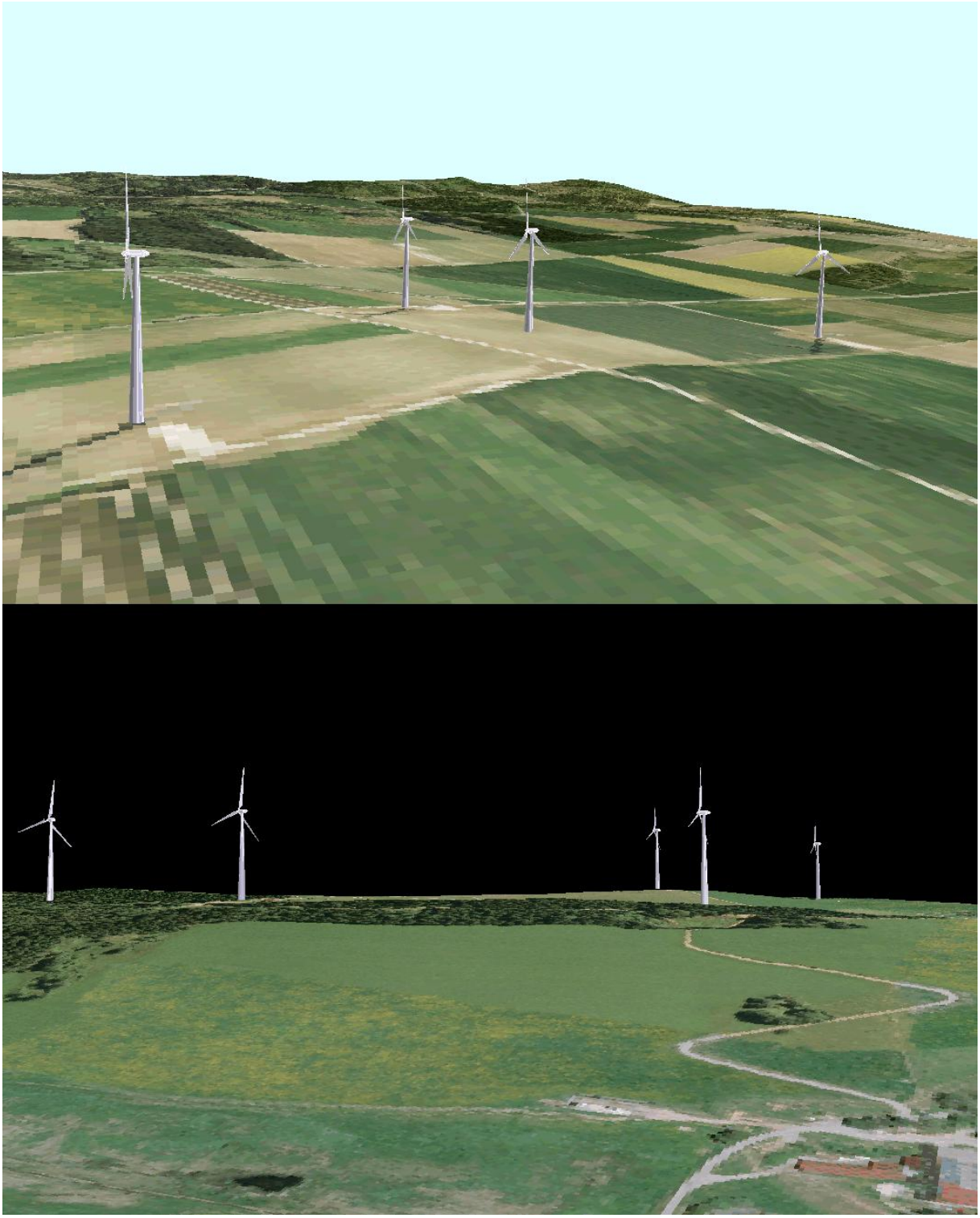

Abbildung 19: Windenergieanlagen im Gelände

5.4 Windpotential

Um die Verwendungsmöglichkeiten der Anwendung zu demonstrieren wird im Rahmen einer Windpotentialanalyse für den Landkreis Ludwigsburg eine 3D Darstellung erzeugt. Als Datengrundlage dient in diesem Fall die durchschnittliche Windgeschwindigkeit in 140 Meter über Grund, klassifiziert und farblich kodiert in 8 Klassen (Abb. 20). Diese Daten werden direkt aus der

Datenbank entnommen und stehen für die 3D Anwendung zur Verfügung.

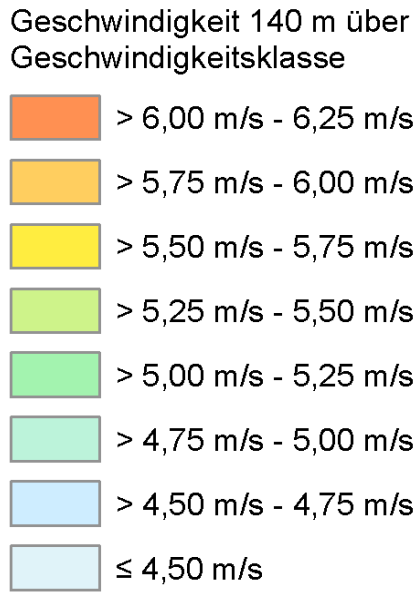

Abbildung 20: Windgeschwindigkeitsklassen

Als Textur für das Oberflächenmodell soll das gerechnete Relief von Baden-Württemberg dienen. Das digitale Geländemodell wird, wie in den anderen Anwendungsfällen, als ein TIN aus den vorhandenen höhenkodierten Rasterdaten berechnet. Zur besseren Orientierung werden weiterhin Gemeinden und Gewässer in die Darstellung aufgenommen, beides ebenfalls aus dem vorhandenen Datenbestand. Durch den modularen Aufbau der Anwendung ist es somit leicht möglich, beliebige Informationen dreidimensional zu visualisieren. Prinzipiell kann somit jeder an der LUBW vorliegende Datensatz in einem 3D PDF dargestellt werden.

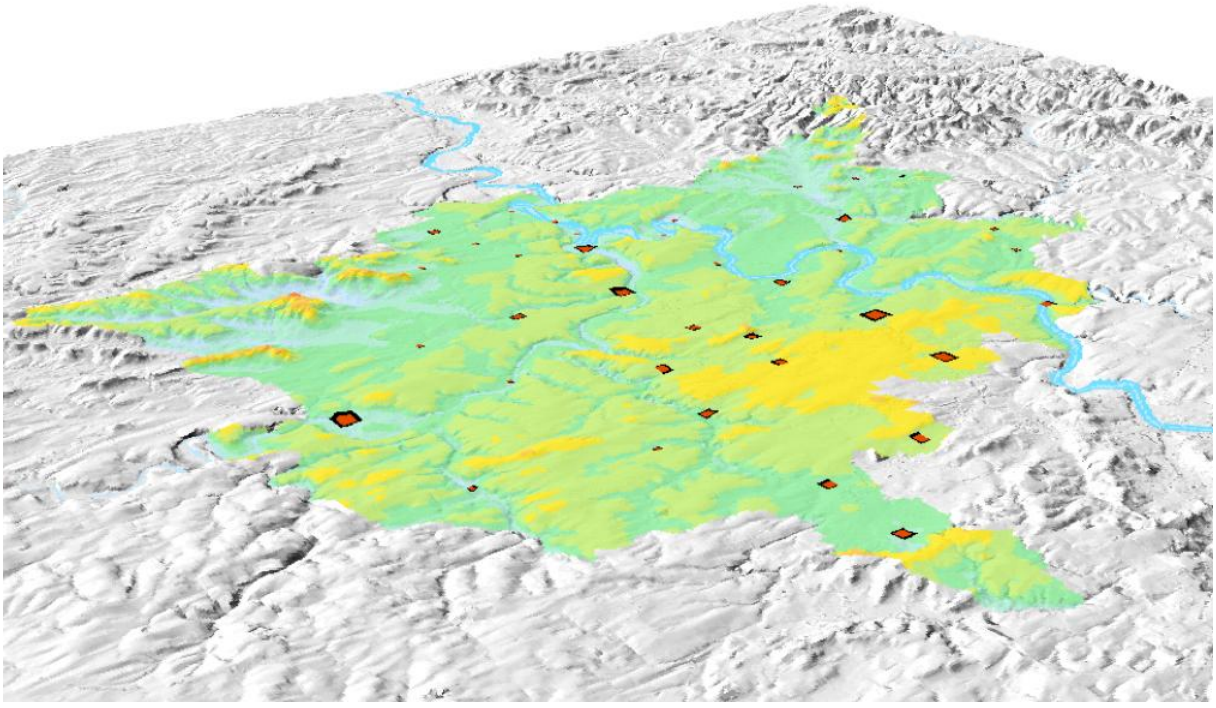

Abbildung 21: 3D Darstellung Windpotential Landkreis Ludwigsburg

6 Zusammenfassung und Ausblick

6.1 Zusammenfassung der Arbeit

Im ersten Kapitel wird zunächst ein kurzer Überblick über die Aufgaben und Ziele der Arbeit gegeben. Das Umweltinformationssystem Baden-Württemberg wird vorgestellt, ebenso das Räumliche Informations- und Planungssystem. Es folgt ein Überblick über die technischen Grundlagen, und die im Rahmen der Arbeit eingesetzten Technologien.

Die Vorteile des 3D-PDF Formats werden dargelegt. Neben der fast universellen Verfügbarkeit des Formats und der damit verbundenen großen Reichweite, ist auch durch die Software FME die Möglichkeit gegeben, ohne weitere Ausgaben für Lizenzkosten o.ä., solche Dokumente an der LUBW zu erstellen. Da die einzusetzenden Geodaten mit Hilfe des ArcGIS Server zur Verfügung gestellt werden, wird anschließend die Geoverarbeitung mit ArcGIS Server über die REST Schnittstelle näher betrachtet. REST, als Kommunikationsarchitektur für Hypermediasysteme, wird im nächsten Abschnitt behandelt. Die Bedingungen für eine REST konforme Kommunikation, nämlich Client-Server Aufbau, Zustandslosigkeit, Cache-Fähigkeit, Verwendung einer einheitlichen Schnittstelle, der Aufbau als Geschichtetes System sowie Code-on-Demand, werden erläutert. Die Implementierung der REST konformen Kommunikation mittels der REST API des ArcGIS Server wird gezeigt. Der Zugriff auf Vektor- und Rasterdaten, auf Geoprocessing Dienste und Routendienste erfolgt innerhalb der REST API über parametrisierte URLs. Der Aufbau solcher URLs, ausgehend vom Wurzelverzeichnis der ArcGIS Server Installation, wird dargestellt. Es schließt ein Abschnitt mit einer allgemeinen Übersicht über den Einsatz des ArcGIS Server und der Server API an. Die beim Einrichten eines Geoverarbeitungsdienstes auf einem Server zu beachtenden Schritte folgen. Es wird insbesondere auf die Einschränkungen bezüglich der einsetzbaren Datentypen eingegangen, sowie auf die verschiedenen Möglichkeiten, Werkzeuge zu veröffentlichen. Um auf die veröffentlichten Geoverarbeitungsdienste zugreifen zu können, bietet ESRI sogenannte Web APIs an. APIs existieren momentan für Javascript, Flex und Silverlight. Sie bieten die Möglichkeit, Dienste des ArcGIS Server in die entsprechende Webarchitektur einzubinden, und so über das Internet verfügbar zu machen. Als Beispiel für den Einsatz einer Web API wird nachfolgend Microsoft Silverlight und die entsprechende API von ESRI kurz vorgestellt. Die Integration und Anbindung von Geoprocessing Services über die API und die entsprechende REST Schnittstelle des Servers wird anschließend beschrieben. Den Abschluss des Kapitels „Technische Grundlagen“ bildet eine Übersicht über den WPS Standard der OGC. Die drei im WPS Standard geforderten Operationen für die Kommunikation mit Geoverarbeitungsservern sind „GetCapabilities“, „DescribeProcess“ und „Execute“. Die Beschreibung des Aufbaus WPS-konformer Anfragen und Antworten bildet den Hauptteil des Abschnitts.

In Kapitel drei werden anhand eines Anwendungsfalles die Möglichkeiten untersucht, in Zukunft GIS Funktionalitäten des ArcGIS Server zu nutzen. Ein einfaches Modell wird mit Hilfe des ArcGIS „*Model Builder*“ erstellt, und anschließend in eine Silverlight Anwendung integriert. Innerhalb der Anwendung sollen digitalisierte Radwege auf Kreisgrenzen beschränkt werden. Um dies zu gewährleisten, wird ein Modell erstellt, welches Linien mit Polygonen verschneidet. Dieses Modell wird anschließend mit Hilfe des ArcGIS Server veröffentlicht. Die Oberfläche der Anwendung wird mittels XAML entworfen, die Programmlogik im „Code-Behind“ der Webseite implementiert. Ergebnis der Arbeiten in Kapitel drei ist eine funktionsfähige Implementierung der neuen Zugriffsmöglichkeiten für das Geoprocessing mittels ArcGIS Server .

Kapitel vier beschäftigt sich mit der Umsetzung und Implementierung der 3D Anwendung. Ausgehend von den vorhandenen Grundlagen und Vorschlägen wird zunächst ein Konzept

entwickelt. Der fertige Dienst soll in das bestehende Rips Umfeld integriert werden. Daher muss zunächst über die WPS Schnittstelle die Anbindung der vorhandenen Anwendungen (GISterm etc.) ermöglicht werden. Weiterhin soll der Dienst asynchron ausgeführt werden, insbesondere bedingt durch das Vorhandensein nur einer Lizenz für die Software FME. Bei einer asynchronen Implementierung werden die vom Anwender getätigten Eingaben in einer Datenbanktabelle gespeichert. Über einen Windows Service wird dann in regelmäßigen Intervallen die Datenbank abgefragt. Liegen neue Einträge in der Datenbank vor, so liest der asynchrone Dienst die Einträge aus. Das Programm FME wird mit den in der Datenbank hinterlegten Parametern gestartet. Verläuft die Erstellung des PDF erfolgreich, so erhält der Anwender eine E-Mail mit einem Verweis auf das angeforderte PDF.

Für die PDF Erstellung wird im Programm FME eine sogenannte „*workbench*“ angelegt. Bei einer „*workbench*“ handelt es sich um eine Sammlung von Eingabedaten, Werkzeugen, Transformatoren und Ausgabedaten. Dadurch ermöglicht FME die Konvertierung von Geodaten in unterschiedliche Formate, ebenso wie das Bearbeiten und Transformieren dieser Daten. So ist es im Rahmen der 3D Anwendung möglich, aus DGM Rasterdaten ein TIN zu berechnen und dieses als Grundlage für das 3D-PDF zu benutzen.

Die benötigten Rasterdaten werden als Kartendienst (*„map service“*) bzw. Bilddienst (*„image service“*) durch einen ArcGIS Server zur Verfügung gestellt. Der Zugriff erfolgt über die REST Schnittstelle des Servers, über die der jeweilige Ausschnitt sowie das gewünschte Thema übermittelt werden können. Nach diesen vorbereitenden Arbeiten muss die Anwendung in das bestehende Rips Framework integriert werden. Zunächst wird die Datenbank erweitert, indem eine neue Tabelle angelegt wird. In dieser Tabelle werden zukünftig die Aufträge und die benötigten Daten gespeichert. Anschließend kann im Rips Framework eine neue Klasse angelegt werden, welche die Funktionalitäten zur PDF Erstellung zur Verfügung stellt. Ist die Klasse angelegt, so muss die Anbindung an die Rips WPS Schnittstelle implementiert werden. Über diese Schnittstelle erfolgt später die Anbindung der im Rips Umfeld vorhandenen Anwendungen. Der letzte, für die Umsetzung notwendige, Schritt ist die Integration in einen bestehenden Windows Dienst. Innerhalb dieses Dienstes sind im Rips periodisch durchzuführende Operationen hinterlegt. Da der Datenbankzugriff, und insbesondere auch die Ansteuerung von FME ebenfalls in Intervallen erfolgen sollen, werden die entsprechenden Operationen hier integriert.

Die somit fertiggestellte Anwendung wird im fünften Kapitel für verschiedene Anwendungsfälle erweitert und getestet. Als erstes wird die DGM Auflösung und die Geländeüberhöhung, welche als feste Werte in FME eingestellt waren, vom gewählten Maßstab abhängig gemacht. Damit kann zum einen gewährleistet werden, dass die Dateigröße sich immer im selben Rahmen bewegt. Zum anderen wird auch für jeden Maßstabsbereich ein ansprechendes und funktionales Geländebild generiert. Als weitere Anforderung soll die Visualisierung von beliebigen Themen innerhalb eines PDF umgesetzt werden. Dies wird erreicht, indem die gewünschten Daten mit Hilfe des ArcGIS Server zur Verfügung gestellt werden. Durch einfache Änderung in den Konfigurationsdateien der Anwendung kann dadurch jedes Thema visualisiert werden.

Eine wichtige Aufgabe für die Anwendung von 3D PDF soll die Visualisierung von Windenergieanlagen sein. Umgesetzt wird dies als Erweiterung der in Kapitel vier erstellten, generischen 3D Anwendung. 3D Modelle von Windenergieanlagen, beispielsweise aus Google Sketchup, werden zusätzlich eingelesen und maßstabsrichtig und georeferenziert im Gelände dargestellt.

Als Nachweis der Funktionalität der Anwendung wird im letzten Abschnitt des Kapitels das Thema Windpotential dreidimensional visualisiert. Der vorliegende Datensatz zu den durchschnittlichen

Windstärken in bestimmten Höhen wird für den Landkreis Ludwigsburg, zusammen mit der ebenfalls vorliegenden Relief-Schummerung, in einem 3D PDF dargestellt.

6.2 Ergebnis und Ausblick

Im Rahmen dieser Arbeit sollten Möglichkeiten untersucht werden, wie in Zukunft Funktionalitäten aus dem ArcGIS Umfeld in das Rips der LUBW integriert werden können. Darüber hinaus sollte das vorhandene Konzept für die dreidimensionale Visualisierung von Geodaten weiterentwickelt und in das Rips integriert werden.

Für das Geoprocessing können in Zukunft die von ESRI zur Verfügung gestellten APIs verwendet werden. Anhand eines Beispiels wurde gezeigt, wie unter Einsatz der API für Silverlight eine Geoprocessing Anwendung umgesetzt werden kann. Hierbei wurde auf die Erstellung von Geoprocessing Werkzeugen für den ArcGIS Server eingegangen, ebenso auf die Einbindung dieser Werkzeuge in die Anwendung.

Die Beispielanwendung für die dreidimensionale Visualisierung im 3D PDF Format wurde umgesetzt. Die Anwendung erlaubt es, beliebige Themen in beliebigen Maßstäben, bis hin zu landesweiten Darstellungen, zu veranschaulichen. Ein Schwerpunkt bildete hierbei die Anzeige von Windenergieanlagen, welche ebenso umgesetzt wurde. Die Anwendung wurde in das bestehende Rips Framework integriert und ist einsatzbereit.

Einige kleinere Änderungen an der Implementierung wären eventuell noch wünschenswert. Die Umsetzung als asynchroner Dienst mit dem Versand einer E-Mail wurde notwendig, da nur eine einzelne Lizenz für FME vorliegt. Eventuell ergibt sich hier in Zukunft eine Möglichkeit, den Anwendern ihr jeweiliges PDF direkt nach der Bearbeitung zur Verfügung zu stellen.

Darüber hinaus ergibt sich durch die beschränkte Funktionalität der WPS Schnittstelle der Umstand, dass der Benutzer für die Wahl des von ihm gewünschten Themas einen Wert in ein Textfeld eintragen muss (eine „1“ für Kartendarstellung, eine „2“ für Luftbilder etc.). Hier wäre eine komfortablere Benutzerführung, beispielsweise mittels eines Auswahlfeldes, wünschenswert. Dies scheitert momentan an der fehlenden Implementierung einer solchen Auswahl im GISterm/UDO Umfeld.

Für die fernere Zukunft bietet sich durch HTML ab der Version 5 eventuell die Möglichkeit, dreidimensionale Darstellungen direkt im Browser des Anwenders umzusetzen. In der Entwicklung befindet sich hier beispielsweise WebGL, welches über das HTML 5 „*canvas*“ Element und auf Grundlage der OpenGL 3D Spezifikation genau dies zum Ziel hat. Vorteil von WebGL ist insbesondere, dass keine externen Plug-Ins im Browser installiert werden müssen. Ein Nachteil ist, dass, durch die Ableitung von OpenGL, neben der Webentwicklung auch Grafikprogrammierung beherrscht werden muss. Mit XML3D, entwickelt am Deutschen Forschungszentrum für Künstliche Intelligenz, bietet sich hier eventuell die Möglichkeit, aus der Webentwicklung bekannte Konzepte auf die 3D-Entwicklung anzuwenden[Jochem_2012]. Es bleibt jedoch abzuwarten, in wie weit hier georeferenzierte Daten eingelesen und dargestellt werden können.

Literaturverzeichnis

[ADOBE_1] Adobe.*Technische Aspekte effektiver kommunizieren*. Abgerufen am 11. 2012 Januar von: http://www.adobe.com/de/manufacturing/solutions/work_instructions/

[ESRI_1] ESRI. *An overview of geoprocessing with ArcGIS Server*. Abgerufen am 4. Januar 2012 von: http://webhelp.esri.com/arcgisserver/9.3/java/index.htm#geoprocessing/an_ove-2102662086.htm

[ESRI_2] ESRI. *ArcGIS API for Silverlight -Overview*. Abgerufen am 6. Januar 2012 von: http://help.arcgis.com/en/webapi/silverlight/help/index.html

[ESRI_3] ESRI. *ArcGIS API for Silverlight- Geoprocessing task*. Abgerufen am 8. Januar 2012 von: http://help.arcgis.com/en/webapi/silverlight/help/index.html#/Geoprocessing_task/01660000000n000000/

[ESRI_4] ESRI. *ArcGIS API for Silverlight*- Discovering services. Abgerufen am 7. Januar 2012 von: http://help.arcgis.com/en/webapi/silverlight/help/index.html#/Discovering_services/01660000001w000000/

[ESRI_5] ESRI. *ArcGIS API for Silverlight-Download*. Abgerufen am 16. Januar 2012 von: http://help.arcgis.com/en/webapi/silverlight/

[ESRI_6] ESRI. *ArcGIS for Server*. Abgerufen am 4. Januar 2012 von: http://www.esri.com/software/arcgis/arcgisserver/index.html

[ESRI_7] ESRI. *ArcGIS Server REST API - Overview*. Abgerufen am 4. Januar 2012 von: http://resources.esri.com/help/9.3/arcgisserver/apis/rest/overview.html

[ESRI_8] ESRI. *Creating a server directory*. Abgerufen am 16. Januar 2012 von: http://webhelp.esri.com/arcgisserver/9.3/java/index.htm#create_svr_dir.htm

[ESRI_9] ESRI. *ModelBuilder*. Abgerufen am 13. Januar 2012 von: http://esri.de/products/arcgis/about/modelbuilder.html

[ESRI_10] ESRI. *Publishing resources*. Abgerufen am 16. Januar 2012 von: http://webhelp.esri.com/arcgisdesktop/9.2/index.cfm?TopicName=Publishing_resources

[ESRI_11] ESRI. *Using the Services Directory*. Abgerufen am 4. Januar 2012 von: http://resources.esri.com/help/9.3/arcgisserver/apis/rest/servicesdirectory.html

[ESRI_12] ESRI. *Web APIs*. Abgerufen am 4. Januar 2012 von: http://resources.arcgis.com/de/content/arcgisserver/web-apis

[ESRI_13] ESRI. *What is the future of the Web ADF?* Abgerufen am 4. Januar 2012 von: http://events.esri.com/uc/QandA/index.cfm?fuseaction=answer&conferenceId=2F6DC1A1-1422-2418-883C3868A9004888&questionId=3060

[Fielding_2000] Fielding, R. (2000). *Architectural Styles and the Design of Network-based Software Architectures. Doctoral dissertation.* Irvine, California: University of California.

[Haberer_2011] Haberer, S. (2011). *Dokumentation RipsWebServices und RipsWeb.WPS.* Karlsruhe: LUBW.

[Haddad_2009] Haddad, A. *Silverlight/ WPF Blog*. Abgerufen am 6. Januar 2012 von: http://blogs.esri.com/Dev/blogs/silverlightwpf/Default.aspx?p=3

[Jochem_2012] Jochem, R. (2012). *Neue 3D-Technologien für zukünftige Geoanwendungen im Netz*. gis.Trends + Markets 1/2012, S. 16-23.

[RIPS_1] Landesanstalt für Umwelt, Messungen und Naturschutz. *Allgemeine Informationen zu RIPS*. Abgerufen am 3. Januar 2012 von: http://www.lubw.baden-wuerttemberg.de/servlet/is/16154/

[UIS_2006] Mayer-Föll, R., & Kaufhold, G. (Hrsg.). (2006). *Rahmenkonzeption UIS.* Ulm: Universitätsverlag.

[RIPS_2006] Mayer-Föll, R., & Schulz, K.-P. (Hrsg.). (2006). *Konzeption RIPS.* Ulm: Universitätsverlag.

[MS_1] Microsoft. *Code-Behind and Partial Classes*. Abgerufen am 10. 1 2012 von: http://msdn.microsoft.com/en-us/library/cc221357%28v=vs.95%29.aspx

[MS_2] Microsoft. *Microsoft Silverlight Release History*. Abgerufen am 6. Januar 2012 von: http://www.microsoft.com/download/en/details.aspx?displaylang=en&id=12121

[MS_3] Microsoft. *Silverlight Overview*. Abgerufen am 6. Januar 2012 von: http://msdn.microsoft.com/en-us/library/bb404700.aspx

[WPS_1] Open Geospatial Consortium. (2007). *OpenGIS Web Processing Service.* (P. Schut, Editor). Abgerufen am 18. Januar 2012 von: http://portal.opengeospatial.org/files/?artifact_id=24151

[UM_1] Ministerium für Umwelt, Klima und Energiewirtschaft Baden-Württemberg. *RIPS - Geodatenmanagement*. Abgerufen am 3. Januar 2012 von: http://www.um.baden-wuerttemberg.de/servlet/is/71551/

[UM_2] Ministerium für Umwelt, Klima und Energiewirtschaft Baden-Württemberg. *Über das UIS*. Abgerufen am 3. Januar 2012 von: http://www.um.baden-wuerttemberg.de/servlet/is/58046/

[UM_3] Ministerium für Umwelt, Klima und Energiewirtschaft Baden-Württemberg. *Über das UIS BW- Systembeschreibungen*. Abgerufen am 3. Januar 2012 von: http://www.um.baden-wuerttemberg.de/servlet/is/59892/

[UM_4] Umweltministerium Baden-Württemberg. (2008). *Umweltinformationssystem Baden-Württemberg. Umfassende Umweltinformation in Dienste der Umweltvorsorge.* Stuttgart: Umweltministerium Baden-Württemberg.

[UM_5] Umweltministerium Baden-Württemberg. (2009). *Räumliches Informations- und Planungssystem. Eine Komponente des Umweltinformationssystems Baden-Württemberg.* Stuttgart: Umweltministerium Baden-Württemberg.

[WIKI_1] Wikipedia. *Representational state transfer*. Abgerufen am 4. Januar 2012 von: http://en.wikipedia.org/wiki/Representational_state_transfer

Anhang

A.1 Die Klasse _3DPDF

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Data;
using System.Net;
using System.Configuration;
using System.Net.Mail;
using System.IO;
using System.Diagnostics;
using System.Security;
using System.Security.Principal;
using System.Runtime.InteropServices;
using System.Threading;
using RipsWeb.Data;
using ESRI.ArcGIS.Geodatabase;



namespace RipsWeb.ArcGisServer.Processing.DataServices
{

    public class _3DPDF
    {
        #region "Impersonation"
        //Dlls to handle impersonation:
        [DllImport("advapi32.dll", SetLastError = true)]
        public static extern bool LogonUser(string lpszUsername, string lpszDomain,
        string lpszPassword, int dwLogonType,
        int dwLogonProvider, ref IntPtr phToken);
        [DllImport("kernel32.dll")]
        public static extern bool CloseHandle(IntPtr token);
        enum LogonType
        {
            Interactive = 2,
            Network = 3,
            Batch = 4,
            Service = 5,
            Unlock = 7,
            NetworkClearText = 8,
            NewCredentials = 9
        }
        enum LogonProvider
        {
            Default = 0,
            WinNT35 = 1,
            WinNT40 = 2,
            WinNT50 = 3
        }
        #endregion

        /// <summary>
        /// writes BB,PDF size and eMail address in DB; to be
        /// used later for PDF generation
```

```
        /// </summary>
        /// <param name="con">Database-Connection of type:
        ///  RipsWeb.Data.CommonConnection</param>
        /// <param name="BB">bounding box as String[] with xMin,xMax,yMin,yMax</param>
        /// <param name="sizeX">pixel count in X of pdf document</param>
        /// <param name="sizeY">pixel count in Y of pdf document</param>
        /// <param name="eMail">eMail address to send pdf to</param>
        public static string CREATE_PDF_JOB(string[] BB, int sizeX, int sizeY,
                                            string eMail, string theme)
        {
            RipsWeb.Data.CommonConnection pConnection =
            RipsWeb.Data.ApplicationConfiguredOracleConnection.
            getCommonConnection("3DPDF_dbSourceARCGISSERVER",
            "3DPDF_dbUserARCGISSERVER", "3DPF_dbPwdARCGISSERVER");

            try
            {
                // + Auftrag_Bearbeitet
                string strInsert = "INSERT INTO PS_3DPDF
               (AUFTRAG_ID,X_MIN,X_MAX,Y_MIN,Y_MAX,SIZE_X,
                SIZE_Y,EMAIL,AUFTRAG_BEARB,THEMA) VALUES (
                PS3DPDF_SEQ.nextval," +
                BB[0] + "," + BB[1] + "," + BB[2] + "," + BB[3] + "," +
                + sizeX + "," + sizeY + ",'" + eMail + "',1," + theme + ")";
                pConnection.openConnection();
                pConnection.executeScalar(strInsert);
                string strSelect = "SELECT AUFTRAG_ID,X_MIN,X_MAX,Y_MIN,Y_MAX,
                SIZE_X,SIZE_Y,EMAIL,THEMA,AUFTRAG_BEARB,ERSTELL_DATUM FROM PS_3DPDF";
                IDataReader dr = pConnection.getDataReader(strSelect);
                string result = "";
                while (dr.Read())
                {
                    result += dr.GetInt32(0) +" ";
                    result += dr.GetDouble(1) + " " + dr.GetDouble(2) + " " +
                              dr.GetDouble(3) + " " + dr.GetDouble(4) + " " +
                              dr.GetInt32(5) + " " + dr.GetInt32(6) + " " +
                              dr.GetString(7) + " " +dr.GetInt32(8) + " " +
                              dr.GetInt32(9) + " " + dr.GetDateTime(10) + "\r\n";
                }

                return result;
            }
            catch
            {
                throw;
            }
            finally
            {
                pConnection.closeConnection();
            }
        }
        public static void deleteDB()
        {
            RipsWeb.Data.CommonConnection pConnection =
            RipsWeb.Data.ApplicationConfiguredOracleConnection.
            getCommonConnection("3DPDF_dbSourceARCGISSERVER",
            "3DPDF_dbUserARCGISSERVER", "3DPF_dbPwdARCGISSERVER");
            try
            {
                string strDelete = "DELETE PS_3DPDF";
                pConnection.openConnection();
                pConnection.executeScalar(strDelete);
            }
```

```
            catch
            {
                throw;
            }
            finally
            {
                pConnection.closeConnection();
            }

        }
        public void EXECUTE_PDF_JOB()
        {
            //vars
            int id = 0;
            string[] xY = new string[4];
            string eMail = "";
            string theme = "";
            int MapX = 1000;
            int MapY = 1000;
            //sql
            string strSelectMin = "SELECT MIN(AUFTRAG_ID) FROM PS_3DPDF WHERE
                                   AUFTRAG_BEARB = 1 ";
            string strSelect = "SELECT
                                AUFTRAG_ID,X_MIN,X_MAX,Y_MIN,
                                Y_MAX,SIZE_X,SIZE_Y,EMAIL,THEMA FROM PS_3DPDF";
            RipsWeb.Data.CommonConnection pConnection =
            RipsWeb.Data.ApplicationConfiguredOracleConnection.
            getCommonConnection("3DPDF_dbSourceARCGISSERVER",
            "3DPDF_dbUserARCGISSERVER", "3DPF_dbPwdARCGISSERVER");
            try
            {
                //get Maps and DGM:
                pConnection.openConnection();
                IDataReader drMin = pConnection.getDataReader(strSelectMin);
                drMin.Read();
                if (!drMin.IsDBNull(0))
                {
                    id = drMin.GetInt32(0);
                    strSelect += " WHERE AUFTRAG_ID=" + id;
                    IDataReader dr = pConnection.getDataReader(strSelect);
                    while (dr.Read())
                    {
                        xY[0] = dr.GetDouble(1).ToString();
                        xY[1] = dr.GetDouble(2).ToString();
                        xY[2] = dr.GetDouble(3).ToString();
                        xY[3] = dr.GetDouble(4).ToString();
                        MapX = dr.GetInt32(5);
                        MapY = dr.GetInt32(6);
                        //Max Size for Map Service: 2000x2000:
                        if (MapX > 2000) MapX = 2000;
                        if (MapY > 2000) MapY = 2000;
                        eMail = dr.GetString(7);
                        theme = dr.GetInt32(8).ToString();
                    }
                    //getMaps(xY, MapX, MapY);
                    getTheme(theme, xY, MapX, MapY);
                    //Surface Tolerance
                    string[] surfTol = surfaceTolerance(xY,MapX,MapY);
                    //start FME:
                    startFME(id,surfTol,theme);
                    //send Email:
                    sendEmail(eMail,id);
                }
```

```
            }
            catch
            {
                throw;
            }
            finally
            {
                string strUpdate = "UPDATE PS_3DPDF SET AUFTRAG_BEARB = 2
                                    WHERE AUFTRAG_ID = " + id;
                pConnection.executeScalar(strUpdate);
                pConnection.closeConnection();
            }
        }
        /// <summary>
        /// saves a byte [] to the specified path
        /// uses impersonation to write to 'ripsfme'
        /// </summary>
        /// <param name="file">byte [] to save</param>
        /// <param name="path">path to save to</param>
        private void saveToDisc(byte[] file,string path)
        {
            IntPtr token = IntPtr.Zero;
            LogonUser("xxx",
                        "ripsfme",
                        "yyy",
                        (int)LogonType.NewCredentials,
                        (int)LogonProvider.WinNT50,
                        ref token);

            using (WindowsImpersonationContext context =
                   WindowsIdentity.Impersonate(token))
            {
                CloseHandle(token);
                using (BinaryWriter writer =
                new BinaryWriter(File.Open(path, FileMode.Create)))
                {
                    writer.Write(file);
                }
            }
        }
        /// <summary>
        /// starts FME.exe on 'ripsfme'
        /// writes returned PDF to local dir upon success
        /// </summary>
        public static void startFME(int id, string [] surfTol, string theme)
        {
            //PSEXEC arguments (which FME workbench to use)
            string args = "";
            switch (theme)
            {
                case "1":
                    args =  @"/AcceptEULA \\ripsfme -u ripsfme\xxx -p yyy" +
                            @" ""c:\programme\fme\fme.exe"" " +
                            @" ""d:\fme_3D_test\3dpdf_local.fmw "" " +
                            @"""--Surf_Tol"" " +
                            "\"" + surfTol[0] + "\" " +
                            @"""Ueberh"" " +
                            "\"" + surfTol[1] + "\" ";
                    break;
                case "2":
                    args =  @"/AcceptEULA \\ripsfme -u ripsfme\xxx -p yyy" +
                            @" ""c:\programme\fme\fme.exe"" " +
                            @" ""d:\fme_3D_test\3dpdf_ortho.fmw "" " +
```

```
                        @"""--Surf_Tol"" " +
                        "\"" + surfTol[0] + "\" " +
                        @"""Ueberh"" " +
                        "\"" + surfTol[1] + "\" ";
                    break;
                case "3":
                    args = @"/AcceptEULA \\ripsfme -u ripsfme\xxx -p yyy" +
                        @" ""c:\programme\fme\fme.exe"" " +
                        @" ""d:\fme_3D_test\3dpdf_wind.fmw "" " +
                        @"""--Surf_Tol"" " +
                        "\"" + surfTol[0] + "\" " +
                        @"""Ueberh"" " +
                        "\"" + surfTol[1] + "\" ";
                    break;
            }
            //PSEXEC process
            Process FME = new Process();
            FME.StartInfo.FileName =
            ConfigurationSettings.AppSettings["3DPDF_PSEXEC_DIR"];
            FME.StartInfo.Arguments = args;
            FME.StartInfo.UseShellExecute = false;
            FME.Start();
            while (!FME.HasExited)
            {
                FME.Refresh();
                Thread.Sleep(2000);
            }
            //FME Has Exited: can get PDF
            if (FME.ExitCode == 0)
            {
                //read from FME machine:
                byte[] bArr_pdf;
                string path = ConfigurationSettings.AppSettings["3DPDF_FME_DIR"]
                              +"3D_Test.pdf";
                FME.Close();
                using (BinaryReader reader =
                new BinaryReader(File.Open(path, FileMode.Open)))
                {
                    bArr_pdf = reader.ReadBytes((int)reader.BaseStream.Length);
                }
                //write to local 4_tage dir:
                string date = DateTime.Today.Day + "." +
                 DateTime.Today.Month.ToString() + "." + DateTime.Today.Year;
                path = ConfigurationSettings.AppSettings["3DPDF_OUTPUT_DIR"] + date
                       + "_" + id + ".pdf";
                using (BinaryWriter writer =
                new BinaryWriter(File.Open(path, FileMode.Create)))
                {
                    writer.Write(bArr_pdf);
                }
            }
            else //error
            {
                FME.Close();
            }

        }
        public static void sendEmail(string to,int id)
        {
            string psaMailServer =
             ConfigurationSettings.AppSettings["3DPDF_MailServer"];
            string from = ConfigurationSettings.AppSettings["3DPDF_MailFrom"];
```

```
            string psaMailPWD =
             ConfigurationSettings.AppSettings["3DPDF_MailPassword"];
            MailMessage mail = new MailMessage();
            string msgBody = string.Empty;
            mail.From = new MailAddress(from, "3DPDF");
            mail.To.Add(to);
            mail.Subject = "Ihr angefordertes 3DPDF";
            mail.Body = "Unter folgendem Link können Sie ihr erstelltes PDF abrufen: "
                      + ConfigurationSettings.AppSettings["3DPDF_OUTPUT_DIR"]
                      + "3DPDF_" + id + ".pdf";
            mail.IsBodyHtml = true;
            SmtpClient smtp = new SmtpClient();
            smtp.Host = psaMailServer;
            smtp.Credentials = new NetworkCredential(@from.Substring(0,
                                        @from.IndexOf("@")), psaMailPWD);
            smtp.Send(mail);
        }
        /// <summary>
        /// Returns surface tolerance for specific scales
        /// </summary>
        /// <param name="xY">Bounding Box</param>
        /// <returns>[0]=SurfaceTolerance;[1]=Ueberhoehung</returns>
        public string[] surfaceTolerance(string[] xY,int MapX, int MapY)
        {
            string [] surfUeber = new string [2];
            double xDif = Convert.ToDouble(xY[1]) - Convert.ToDouble(xY[0]);
            double yDif = Convert.ToDouble(xY[3]) - Convert.ToDouble(xY[2]);
            //max Dif (X or Y)
            double mDif = 0;
            if (xDif > yDif) mDif = xDif;
            else mDif = yDif;
            //DGM Tolerance (surfUeber[0])
            int pixel = 0;
            if (MapX > MapY) pixel = MapX;
            else pixel = MapY;
            int dgm = (int)(mDif / pixel);
            if (dgm < 1) dgm = 1;
            surfUeber[0] = (dgm / 2).ToString();

            //Ueberhöhung (surfUeber[1])
            if (xDif > 200000 | yDif > 200000)
            {
                surfUeber[1] = "3";
            }
            else if (xDif > 50000 | yDif > 50000)
            {
                surfUeber[1] = "2";
            }
            else if (xDif > 10000 | yDif > 10000)
            {
                surfUeber[1] = "1.5";
            }
            else
            {
                surfUeber[1] = "1";
            }

            return surfUeber;
        }
        public void getTheme(string theme, string [] xY, int MapX, int MapY)
        {
            switch (theme)
            {
```

```
                case "1":
                    getMaps(xY, MapX, MapY);
                    break;
                case "2":
                    getOrtho(xY, MapX, MapY);
                    break;
                case "3":
                    getWind(xY, MapX, MapY);
                    break;
            }
        }
        private void getMaps(string[] xY, int MapX = 1000, int MapY = 1000)
        {
            //REST Uri
            Uri redir;
            //Web Clients:
            var clientALK = new WebClient();
            var clientMap = new WebClient();
            var clientDGM = new WebClient();
            //Bounding Box Coordinates:
            string strXMin = xY[0].Replace(",", ".");
            string strXMax = xY[1].Replace(",", ".");
            string strYMin = xY[2].Replace(",", ".");
            string strYMax = xY[3].Replace(",", ".");
            //DGM Size (1/3 Map Size):
            string dgmX = (MapX / 3).ToString();
            string dgmY = (MapY / 3).ToString();
            //Map Size: // Maximum = 2000 px
            string strMapX = MapX.ToString();
            string strMapY = MapY.ToString();
            //Generate URIs, Read through Web Client
            //ALK
            redir = new Uri(ConfigurationSettings.AppSettings["3DPDF_ALK_URI"] +
            "?bbox=" + strXMin + "," + strYMin + "," + strXMax + "," + strYMax +
            "&size=" + strMapX + "," + strMapY +
            "&f=image&format=png");
            //Save in Byte []
            byte[] bALK = clientALK.DownloadData(redir);
            saveToDisc(bALK, ConfigurationSettings.AppSettings["3DPDF_FME_DIR"] +
                       "schrift.png");
            //Map
            redir = new Uri(ConfigurationSettings.AppSettings["3DPDF_HINTERGRUND_URI"]
            + "?bbox=" + strXMin + "," + strYMin + "," + strXMax + "," + strYMax +
            "&size=" + strMapX + "," + strMapY +
            "&f=image&format=png");
            //Save in Byte []
            byte[] bMap = clientMap.DownloadData(redir);
            saveToDisc(bMap, ConfigurationSettings.AppSettings["3DPDF_FME_DIR"] +
                       "map.png");
            //DGM
            redir = new Uri(ConfigurationSettings.AppSettings["3DPDF_DGM_URI"] +
            "?bbox=" + strXMin + "," + strYMin + "," + strXMax + "," + strYMax +
            "&size=" + dgmX + "," + dgmY +
            "&f=image&format=tiff" +
            "&pixelType=U32&interpolation=RSP_NearestNeighbor");
            //Save in Byte []
            byte[] bDGM = clientDGM.DownloadData(redir);
            saveToDisc(bDGM, ConfigurationSettings.AppSettings["3DPDF_FME_DIR"] +
                       "dgm.tiff");
        }
        private void getOrtho(string[] xY, int MapX = 1000, int MapY = 1000)
        {
            //REST Uri
```

```
            Uri redir;
            //Web Clients:
            var clientDGM = new WebClient();
            var clientOrtho = new WebClient();
            //Bounding Box Coordinates:
            string strXMin = xY[0].Replace(",", ".");
            string strXMax = xY[1].Replace(",", ".");
            string strYMin = xY[2].Replace(",", ".");
            string strYMax = xY[3].Replace(",", ".");
            //DGM Size (1/3 Map Size):
            string dgmX = (MapX / 3).ToString();
            string dgmY = (MapY / 3).ToString();
            //Map Size: // Maximum = 2000 px
            string strMapX = MapX.ToString();
            string strMapY = MapY.ToString();
            //Generate URIs, Read through Web Client
            //DGM
            redir = new Uri(ConfigurationSettings.AppSettings["3DPDF_DGM_URI"] +
            "?bbox=" + strXMin + "," + strYMin + "," + strXMax + "," + strYMax +
            "&size=" + dgmX + "," + dgmY +
            "&f=image&format=tiff" +
            "&pixelType=U32&interpolation=RSP_NearestNeighbor");
            //Save in Byte []
            byte[] bDGM = clientDGM.DownloadData(redir);
            saveToDisc(bDGM, ConfigurationSettings.AppSettings["3DPDF_FME_DIR"] +
                       "dgm.tiff");
            //Ortho
            redir = new Uri(ConfigurationSettings.AppSettings["3DPDF_Ortho_URI"] +
            "?bbox=" + strXMin + "," + strYMin + "," + strXMax + "," + strYMax +
            "&size=" + strMapX + "," + strMapY +
            "&f=image&format=png");
            //Save in Byte []
            byte[] bMap = clientOrtho.DownloadData(redir);
            saveToDisc(bMap, ConfigurationSettings.AppSettings["3DPDF_FME_DIR"] +
                       "ortho.png");
        }
        private void getWind(string[] xY, int MapX = 1000, int MapY = 1000)
        {
            //REST Uri
            Uri redir;
            //Web Clients:
            var clientDGM = new WebClient();
            var clientOrtho = new WebClient();
            //Bounding Box Coordinates:
            string strXMin = xY[0].Replace(",", ".");
            string strXMax = xY[1].Replace(",", ".");
            string strYMin = xY[2].Replace(",", ".");
            string strYMax = xY[3].Replace(",", ".");
            //DGM Size (1/3 Map Size):
            string dgmX = (MapX / 3).ToString();
            string dgmY = (MapY / 3).ToString();
            //Map Size: // Maximum = 2000 px
            string strMapX = MapX.ToString();
            string strMapY = MapY.ToString();
            //Generate URIs, Read through Web Client
            //getOrtho
            getOrtho(xY, MapX, MapY);
            //getWEA

        }

    }
}
```