

Enviro-SOA

Diensteplattform für das UIS Baden-Württemberg

H. Paoli
Forschungszentrum Informatik
Haid-und-Neu-Str. 10-14
76131 Karlsruhe

W. Heißler; R. Ebel; M. Schild
Landesanstalt für Umwelt, Messungen und Naturschutz Baden-Württemberg
Griesbachstr. 1
76185 Karlsruhe

1. EINFÜHRUNG	143
2. ZIELE.....	143
3. SZENARIEN	144
4. UMSETZUNG.....	145
4.1 BENUTZERFÜHRUNG	145
4.2 LAUFZEITUMGEBUNG	147
4.3 FORSCHUNGSASPEKTE	147
5. AUSBLICK.....	148
6. LITERATUR.....	148

1. Einführung

Durch die Entwicklung eines Diensteverzeichnisses wurde ein wichtiger Grundstein zur Schaffung einer dienstorientierten Architektur des UIS Baden-Württemberg gelegt /1/, /2/. Im Rahmen dieser Entwicklung wurden mehr als 100 Dienste beschrieben, welche nun den Fachanwendern zur Verfügung stehen, um deren tägliche Arbeit zu erleichtern. Das Dienstverzeichnis bietet außerdem eine Schnittstelle zu dem Geo-Metadaten-System „disy Prelude“, um auch die dort beschriebenen, geobasierten Dienste einheitlich in die Suche zu integrieren. Damit ist das Dienstverzeichnis ganz allgemein eine zentrale Plattform für die Recherche nach passenden Diensten.

Durch das hier beschriebene Projekt Enviro-SOA soll die Nutzbarkeit der einzeln und in Kombination eingesetzten Dienste für die Fachanwender erleichtert bzw. unterstützt werden.

2. Ziele

Mit dem hier beschriebenen Projekt wurde vorrangig das Ziel verfolgt, den Fachanwendern die einfache Nutzung der vorhandenen Dienste zu ermöglichen. Die Recherche nach Diensten, als ersten Schritt, ermöglicht bereits das Dienstverzeichnis, jedoch steht der Anwender nach dem Finden nützlicher Dienste vor dem Problem, wie diese Dienste konkret in Anspruch genommen werden können.

Im Wesentlichen müssen dazu vier Herausforderungen gemeistert werden:

- Finden der passenden Dienste,
- Angabe von Parameterwerten,
- Verknüpfung von Diensten,
- Ausführung der Dienste durch die Laufzeitumgebung.

Eine Eigenschaft der Dienste ist, dass sie Parameter haben können. Ein Dienst zur Bereitstellung von Abfalldaten kann z.B. Parameter für zu berücksichtigende geografische Regionen (Länder, Städte, Gemeinden etc.), Kalenderjahre oder Darstellungsart (Karte, Bericht, Tabelle etc.) anbieten. Die Anzahl und die Art der Parameter, welche ein Dienst anbietet, hängt letztlich davon ab, wie konkret oder generisch der Dienst ausgelegt ist. Ein generischer Dienst kann unter Umständen viele konkrete Dienste ersetzen und damit Entwicklungs- und Betriebskosten senken. Es kann aber auch sein, dass er für Fachanwender schwieriger einzusetzen ist, da an die Stelle impliziter Kenntnis nun explizite Optionen treten müssen. Die letztlich gewählte Granularität für die Erstellung der Dienste bewegt sich in diesem Spannungsfeld. Ein weiteres Ziel dieses Projekts besteht darin, stärker generische Dienste anbieten zu können, die bei Bedarf leicht und – ohne zusätzliche technische Kenntnis – durch die Fachanwender selbst konkretisiert und verwendet werden können.

Betrachtet man die Bedürfnisse der Anwender genauer, so wird klar, dass zur Bewältigung anstehender Aufgaben meist nicht nur ein einzelner Dienst benötigt wird, sondern vielmehr das sinnvolle Zusammenspiel mehrerer Dienste erforderlich ist. Dienste sollten also für die

Fachanwender aufgabenrelevant verknüpfbar sein. Für wiederkehrende Aufgaben sollten diese erstellten Verknüpfungen zudem dauerhaft zur Verfügung stehen. Letztlich sollten die Anwender also ihr eigenes, personalisiertes Berichtssystem ohne viel Aufwand aufbauen können. Gerade von flexiblen, dienstorientierten Architekturen werden solche Möglichkeiten erwartet. Der bereits vorhandene Dienstpool wird somit zum Baukasten für die Erstellung höherwertiger Dienste entsprechend den fachlichen Erfordernissen. Die höherwertigen Dienste sind aus anderen Diensten zusammengesetzt, stellen gleichzeitig aber auch neue Bausteine innerhalb des Baukastens dar. Die Fachanwender werden selbst zum Entwickler, ohne dass softwaretechnische Kenntnisse nötig sind. Für die Fachanwender ergibt sich damit der Vorteil, autarker zu werden und schneller Anforderungen umsetzen zu können. Gleichzeitig werden die Bereitsteller der Basisdienste, also der Dienste, die selbst nicht aus anderen Diensten zusammengesetzt sind, entlastet, da insgesamt weniger Basisdienste bereitgestellt und gepflegt werden müssen. Die höherwertigen Dienste werden durch die entsprechenden Fachanwender selbst verwaltet, wodurch auf natürliche Weise eine auch in der Praxis sinnvolle Arbeitsteilung entsteht.

Nicht zuletzt bedarf es auch einer einheitlichen Softwareplattform, die die Ausführung der Dienste technisch ermöglicht und den Anwendern die erwarteten Ergebnisse zurückliefert.

3. Szenarien

Zur Durchführung des Projekts wurde beschlossen, sich konkret an zwei Szenarien zu orientieren. Dies hat den Vorteil, frühzeitig die Bedürfnisse der Fachanwender anhand praktisch relevanter Fragestellungen zu erkennen und zu berücksichtigen. Als Szenarien wurden gemeinsam ausgewählt:

- Umweltthemendossier (am Bsp. Abfalldeponien),
- Umweltgemeindedossier.

Beim Abfall-Szenario steht die Frage nach Wechselwirkungen zwischen Abfalldeponien und umweltrelevanten Objekten wie z.B. Biotopen im Vordergrund. Ein Fachanwender, der sich für diese Fragestellung interessiert, steigt zunächst über das Thema „Deponien“ ein und wählt hierzu passende Dienste aus. Beim Thema „Biotope“ geht er analog vor. Die ausgewählten Dienste müssen mit den entsprechend notwendigen Parametern, z.B. den Kennungen der Deponien und Biotope, konfiguriert werden. Anschließend kann diese konfigurierte Dienstausswahl ausgeführt und/oder dauerhaft gespeichert werden.

Beim Szenario für das Umwelt-Gemeindedossier geht es darum, relevante Umweltinformationen gemeindebezogen in Form eines Dossiers bereitzustellen, z.B. um bestimmte Interessenvertreter zu informieren und bei anstehenden Entscheidungen mit Umweltbezug zu unterstützen. Ein zuständiger Fachanwender wird in diesem Fall zunächst über die betreffende Gemeinde einsteigen und relevante Dienste auswählen. Diese Auswahl kann bei Bedarf durch weitere nicht gemeindebezogene, aber dennoch für die Interessensgruppe relevante Dienste ergänzt werden. Wiederum müssen alle Dienste mit notwendigen Parametern wie z.B. Gemeindekennzahl, Datumswert, Darstellungsart usw. versorgt werden. Das dadurch erstellte Dossier wird dauerhaft gespeichert und bei Bedarf abgerufen.

Aus den betrachteten Szenarien lässt sich eine allgemeine Vorgehensweise ableiten, die in Abbildung 1 dargestellt ist.

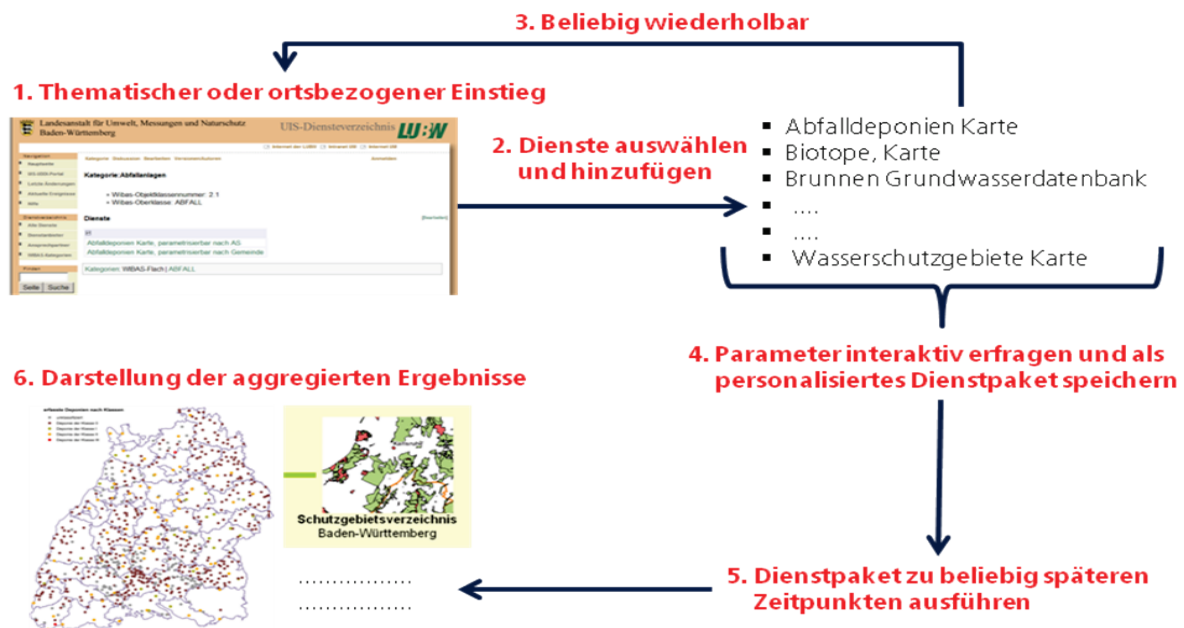


Abbildung 1: Allgemeine Vorgehensweise der Anwender zur Erstellung fachgerechter Berichte

4. Umsetzung

4.1 Benutzerführung

Um die vorgestellte, allgemeine Vorgehensweise technisch umsetzen zu können, bedarf es zunächst einmal einer geeigneten Benutzerführung. Hierzu gibt es grundsätzlich zwei Möglichkeiten: Zum einen ließe sich eine bereits vorhandene und etablierte Applikation um die neuen Anforderungen erweitern, zum anderen könnte eine neue Applikation erstellt werden. Betrachtet man den ersten Weg, so ergeben sich direkt zwei Kandidaten, die hierfür in Frage kämen, nämlich das Dienstverzeichnis selbst oder das Intranet-CMS. Für das Dienstverzeichnis spricht, dass hier bereits die Recherche nach Dienstbeschreibungen zur Verfügung steht und dieses System so auf natürliche Weise erweitert würde. Für das Intranet-CMS spricht, dass es gut bekannt und etabliert ist. Gegen das Dienstverzeichnis spricht, dass dieses System auf die Pflege von Dienstbeschreibungen optimiert wurde und daher der jetzt zu unterstützenden Anwendergruppe Funktionalitäten anbietet, die diese weder benötigen, noch in ihrer Tiefe (z.B. Bezug zum UDDI-Standard) verstehen wollen. Gegen das Intranet-CMS sprechen der wesentlich höhere technische Integrationsaufwand und dass die gebotene Funktionalität des CMS Anwender, die nur einzelne Dienste konfigurieren und ausführen wollen, eher verwirrt. Aus diesen Gründen wurde die Entscheidung getroffen, die Umsetzung anhand einer eigenen, webbasierten Anwendung durchzuführen.

Die Oberfläche soll dazu einfach und übersichtlich gestaltet, aber auch komfortabel sein. Dazu ist es notwendig, die Möglichkeiten des aktuellen Standes der Technik zu nutzen, der es erlaubt, webbasierte Anwendungen mit einem Komfort auszustatten, der bisher nur ech-

ten Desktopanwendungen vorbehalten war, wie z.B. Nutzung interaktiver und dynamischer Eingabeelemente, die per Maussteuerung bedienbar sind. Beispiele hierfür wären Listen, die interaktiv um weitere Elemente ergänzt werden können, oder Baumdarstellungen, bei denen sich Äste per Mausklick ein- und ausgeblendet lassen. Dabei sollte aber nicht jedes Mal die gesamte Seite vom Server neu geladen und vom Client neu gezeichnet werden, wodurch die Interaktivität herabgesetzt werden würde. Die dazu erforderliche Technologie wird mit AJAX (Asynchronous Javascript And XML) bezeichnet /3/ und wurde im Rahmen des Projekts durch die Verwendung des freien und als Open-Source verfügbaren Apache-Wicket-Frameworks realisiert /4/.

Das gewählte Layout der webbasierten Anwendung ist in Abbildung 2 zu sehen.

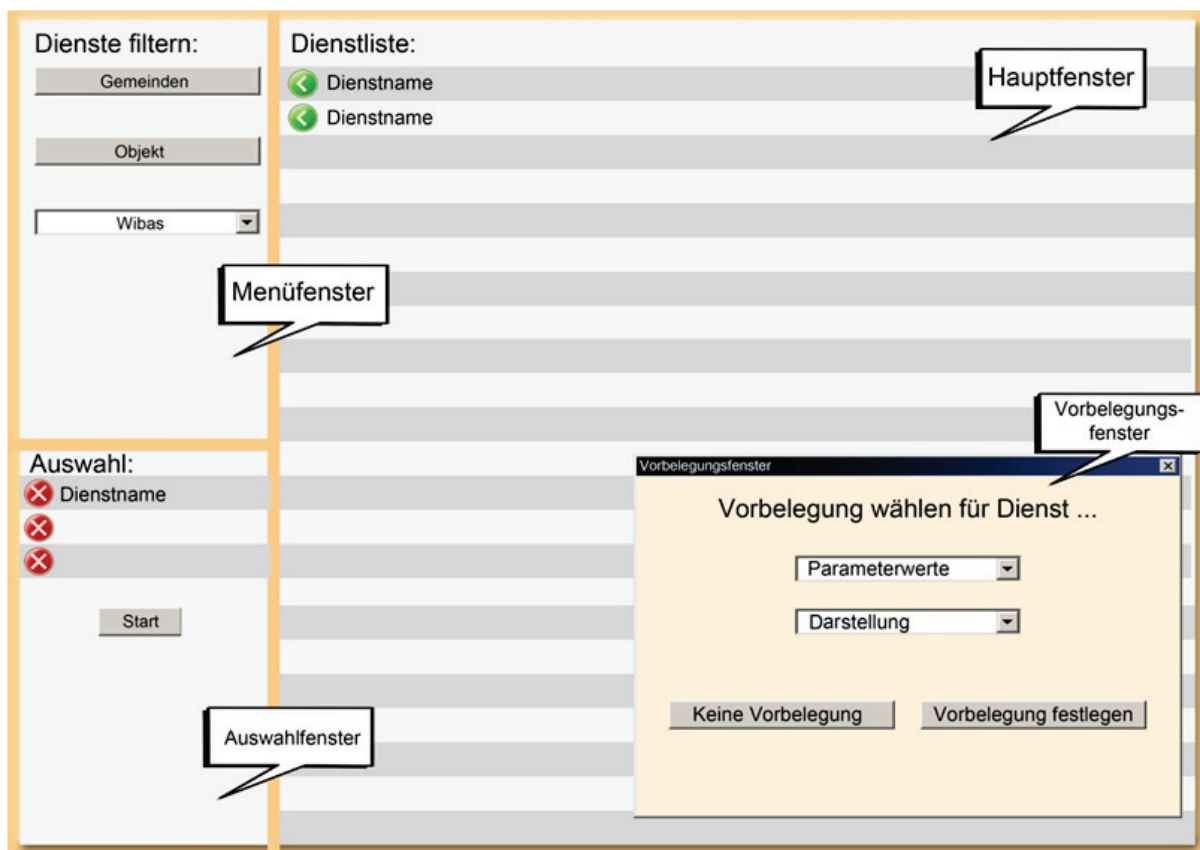


Abbildung 2: Layout der Benutzeroberfläche

Im Wesentlichen besteht die Oberfläche aus drei Bereichen (Fenstern). Ein Hauptfenster listet alle verfügbaren Dienste auf, aus denen der Anwender dann passende Dienste zur Konfiguration auswählt. Die Anzeige der Dienste im Hauptfenster kann dann durch ein Menüfenster gesteuert werden. Hier stehen vielfältige Filterungsmöglichkeiten zur Verfügung, um Dienste unter thematischen Gesichtspunkten anzuzeigen. Vom Anwender ausgewählte Dienste werden in einem Auswahlfenster zunächst gesammelt. Ist der Auswahlprozess abgeschlossen, so können die gesammelten Dienste durch ein Vorbelegungs-fenster, das der Reihe nach für alle Dienste erscheint, mit Parameterwerten versorgt werden. Es ist hier auch möglich, Parameter zunächst offen zu lassen, um diese bei der Ausführung dann später interaktiv eingeben zu können. Das ist hilfreich, falls zwar eine große Anzahl an Parametern für die Lösung einer Aufgabe bekannt und vorgegeben ist, aber einzelne Parameter von Aufruf zu Aufruf variieren können (z.B. Datumsangaben).

4.2 Laufzeitumgebung

Zur Ausführung und Verknüpfung der Dienste wird eine Laufzeitumgebung benötigt. Die Laufzeitumgebung verwendet das bereits entwickelte Dienstverzeichnis, um die Beschreibungen der vorhandenen Dienste abzurufen und somit im Hauptfenster anzeigen zu können. Die Dienstbeschreibungen sind aber auch notwendig, damit die Laufzeitumgebung erkennt, welche Parameter ein Dienst anbietet bzw. welche Werte dort erlaubt sind. Die Laufzeitumgebung kann mit diesen Informationen den Anwender bei der Konfiguration z.B. durch Wertelisten oder Vorbelegungen sinnvoll unterstützen. Des Weiteren muss die Laufzeitumgebung in der Lage sein, mehrere Dienste zu verknüpfen und in Form eines Berichts zusammenzufassen. Hierfür muss die Laufzeitumgebung mit unterschiedlichen Rückgabeformaten zurechtkommen (HTML, XML, PDF, Karten, Tabellen etc.). Es gibt zudem unterschiedliche Dienstypen wie Webservices, REST-Services oder einfach nur parametrisierbare Webanwendungen. Sollen alle diese Dienste ausführbar und miteinander verknüpfbar sein, dann muss die Laufzeitumgebung die technischen Voraussetzungen aller Dienstypen erfüllen und darf sich nicht auf einen einzigen Dienstyp wie z.B. Webservices beschränken.

Um diesen Anforderungen gerecht zu werden, bildet das Kernstück der Laufzeitumgebung eine BPEL-Engine. BPEL (Business Process Execution Language) ist ein Standard, der es zunächst einmal erlaubt, Webservices zu verknüpfen („orchestrieren“) und aufzurufen /5/, /6/. BPEL selbst ist eine XML-basierte Sprache, die auch gerne als „Programmiersprache im Großen“ aufgefasst wird. Durch die Interaktion mit den Anwendern werden durch die Benutzerführung BPEL-Programme im Hintergrund erzeugt, ohne dass die Anwender davon Notiz bekommen. Zur konkreten Realisierung wurde im Projekt die Apache ODE (Orchestration Director Engine) eingesetzt /7/, ein Produkt welches wiederum frei und Open-Source ist. Apache ODE unterstützt neben Webservices auch REST-Services und parametrisierbare Webanwendungen (als Sonderfall von REST-Services).

4.3 Forschungsaspekte

Durch die generierten BPEL-Programme entstehen letztlich neue Dienste, für die jedoch noch keine Dienstbeschreibung innerhalb des Dienstverzeichnisses vorliegt. Die rein technischen Aspekte der Dienstbeschreibung wie offene Parameter und deren syntaktische Eigenschaften lassen sich noch relativ einfach ermitteln. Schwieriger wird es, wenn die fachlichen Aspekte einer Dienstbeschreibung berücksichtigt werden sollen. So wird aus einem allgemeinen Abfalldatendienst durch die Bereitstellung einer Gemeindeganzahl und die Verknüpfung mit einer Karte eine grafische Abfalldatenanzeige erzeugt. Ein Anwender könnte im Anschluss eine entsprechende fachliche Beschreibung verfassen, aber nur mit zusätzlichem Aufwand. Würden dagegen die Beschreibungen der beteiligten Dienste maschinenverständlich abgelegt, könnte aus ihnen eine Dienstbeschreibung für zusammengesetzte Dienste automatisch hergeleitet und damit der Anwender entlastet werden. Im Rahmen des Projekts wurde dieser Ansatz, der noch offenes Forschungsgebiet ist, verfolgt. Dazu werden Dienstbeschreibungen der Basisdienste durch Verwendung semantischer Technologien annotiert (im „Semantic MediaWiki“). Bei der Verknüpfung von Dienstbeschreibungen kann damit über semantisches Schlussfolgern eine neue Dienstbeschreibung erzeugt und direkt im Dienstverzeichnis ohne weitere Benutzerinteraktion gespeichert werden.

5. Ausblick

Als Ergebnis des dargestellten Projekts steht im UIS BW eine Plattform zur Verfügung, welche die Suche, Verknüpfung und Ausführung von Diensten auf einfache Weise ermöglicht. Fachanwender werden dabei selbst zu Entwicklern, die höherwertige Dienste einfach, zeitnah und aufgabengerecht aus bereits vorhandenen Diensten erstellen können. Allerdings ließe sich die Ergebniszusammenführung der einzelnen Dienste noch weiter verbessern. Derzeit sind einfache Mechanismen zur Aggregation der Einzelergebnisse fest vorgegeben, die sich von den Anwendern nicht weiter beeinflussen lassen. Dies könnte mit einer Art Vorlagentechnologie erreicht werden. Eine Vorlage bestimmt dabei, welche Teile eines Dokuments durch ausgewählte Dienste erzeugt werden. Die Vorlagen würden extern gehalten und könnten jederzeit von den Anwendern angepasst werden. Ebenso ließen sich Standardvorlagen vorsehen, die bei Bedarf durch die Anwender selbst auf einfache Art und Weise angepasst bzw. personalisiert werden könnten.

Vorrangig erscheint allerdings eine Erprobung des Piloten bei diversen Anwendern, um seine Praxistauglichkeit sicher zu stellen und Anregungen für notwendige Weiterentwicklungen zu erhalten.

6. Literatur

- /1/ Paoli, H., Holtmann, C., Ebel, R. (2007): UIS-UDDI – Entwicklung eines Web Service-Verzeichnisses für das UIS Baden-Württemberg. In: Mayer-Föll, R., Keitel, A., Geiger, W.; Hrsg.: F+E-Vorhaben KEWA – Kooperative Entwicklung wirtschaftlicher Anwendungen für Umwelt, Verkehr und benachbarte Bereiche in neuen Verwaltungsstrukturen, Phase II 2006/07. Wissenschaftliche Berichte, Forschungszentrum Karlsruhe, FZKA 7350, S. 21-30.
- /2/ Paoli, H. et al. (2008): UIS-UDDI – Weiterentwicklung des Dienstverzeichnis für das Umweltinformationssystem Baden-Württemberg. In: Mayer-Föll, R., Keitel, A., Geiger, W.; Hrsg.: F+E-Vorhaben KEWA – Kooperative Entwicklung wirtschaftlicher Anwendungen für Umwelt, Verkehr und benachbarte Bereiche in neuen Verwaltungsstrukturen, Phase III 2007/08. Wissenschaftliche Berichte, Forschungszentrum Karlsruhe, FZKA 7420, S. 173-180.
- /3/ Garrett, J. J. (2005): Ajax: A New Approach to Web Applications, adaptive path, <http://adaptivepath.com/ideas/essays/archives/000385.php>, Abruf 09.06.2009
- /4/ Apache (2005): Apache Wicket, <http://wicket.apache.org/>, Abruf 09.06.2009
- /5/ OASIS (2002); OASIS Web Services Business Process Execution Language (WSBPEL) TC, http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsbpel, Abruf 09.06.2009
- /6/ Nitzsche, J., van Lessen, T., Karastoyanova, D., Leymann, F. (2007): BPEL for Semantic Web Services (BPEL4SWS). OTM Workshops (1) 2007: 179-188.
- /7/ Apache (2007): Apache ODE, <http://ode.apache.org/index.html>, Abruf 09.06.2009